







Multi-Class Imbalance Classification Based on Data Distribution and Adaptive Weights

Shuxian Li , Liyan Song , Xiaoyu Wu , *Member, IEEE*, Zheng Hu , Yiu-ming Cheung , *Fellow, IEEE*, and Xin Yao , *Fellow, IEEE*

Abstract—AdaBoost approaches have been used for multi-class imbalance classification with an imbalance ratio measured on class sizes. However, such ratio would assign each training sample of the same class with the same weight, thus failing to reflect the data distribution within a class. We propose to incorporate the density information of training samples into the class imbalance ratio so that samples of the same class could have different weights. As one could use the entire training set to calculate the imbalance and density factors, the weight of a training sample resulting from the two factors remains static throughout the training epochs. However, static weights could not reflect the up-to-date training status of base learners. To deal with this, we propose to design an adaptive weighting mechanism by making use of up-to-date training status to further alleviate the multi-class imbalance issue. Ultimately, we incorporate the class imbalance ratio, the density-based factor, and the adaptive weighting mechanism into a single variable, based on which the adaptive weights of all training samples are computed. Experimental studies are carried out to investigate the effectiveness

of the proposed approach and each of the three components in dealing with multi-class imbalance classification problem.

Index Terms—Multi-class imbalance classification, ensembles, AdaBoost, adaptive weight, data density.

I. INTRODUCTION

CLASS imbalance for which some are highly under-represented (over-represented) as minority (majority) classes compared to others is common in real-world applications of multi-class classifications such as human behavior recognition [1], video classification [2], and medical decision making [3], which has posed significant challenges to the management and use of the data sets. Unprocessed class imbalance in data often leads to misclassification, misleading, and even false results, especially on the minority classes. This issue becomes more obvious in applications such as medical decision making, when the predictive performance on the minority classes (e.g., rare diseases) is more important than that on the majority classes (normal). In particular, class imbalance with multiple classes in a data set poses even bigger challenges, which remain unsolved. Lately, the class imbalance problem has drawn increasing attention from both academia and industry in an effort to retain predictive performance [4].

Data sampling, cost-sensitive, and ensemble learning are three common strategies for dealing with the multi-class imbalance [4], [5], [6]. Sampling approaches encompass oversampling the minority class(es) or undersampling the majority class(es) to achieve a balanced distribution of samples [7], [8], [9]. They operate at the data-level and can enhance the predictive performance of various machine learning methods [7], [8], [9], while under some challenges such as overfitting for oversampling [7], the loss of important information for undersampling [7], [10], and the substantial computational costs [10], [11]. Cost-sensitive approaches involve assigning distinct misclassification costs or sample weights to various classes [7], [12]. These approaches are easy to implement and can be combined with various machine learning algorithms [4], [7]. However, deciding the cost matrix or sample weights in the training process is nontrivial. This challenge becomes even more pronounced when multiple classes are involved [4]; their values are determined in advance and as such remain static throughout the training process. Ensemble learning approaches involve combining multiple classifiers to enhance model accuracy, which have exhibited efficacy and flexibility [5], [13], [14],

Manuscript received 15 June 2022; revised 14 August 2023; accepted 19 March 2024. Date of publication 4 April 2024; date of current version 4 October 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62002148 and Grant 62250710682, in part by Guangdong Provincial Key Laboratory under Grant 2020B121201001, in part by the Program for Guangdong Introducing Innovative and Entrepreneurial Teams under Grant 2017ZT07X386, in part by the Research Institute of Trustworthy Autonomous Systems (RITAS), in part by the NSFC/Research Grants Council (RGC) Joint Research Scheme under Grant N_HKBU214/21, in part by the General Research Fund of RGC under Grant 12201321, Grant 12202622, and Grant 12201323, and in part by RGC Senior Research Fellow Scheme under Grant SRFS2324-2S02. Recommended for acceptance by Y. Shen. (*Corresponding authors: Yiu-ming Cheung; Xin Yao.*)

Shuxian Li is with the Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China, also with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China, and also with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: lishx@mail.sustech.edu.cn).

Liyan Song was with the Research Institute of Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China, also with the Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China. She is now with the Faculty of Computing, Harbin Institute of Technology, Harbin, Heilongjiang 150001, China (e-mail: songly@hit.edu.cn).

Xiaoyu Wu and Zheng Hu are with the RAMS Reliability Technology Lab, Huawei Technologies Company Ltd, Shenzhen, Guangdong 518129, China (e-mail: wuxiaoyu7@huawei.com; hu.zheng@huawei.com).

Yiu-ming Cheung is with the Department of Computer Science, Hong Kong Baptist University, Hong Kong, SAR, China (e-mail: ymc@comp.hkbu.edu.hk).

Xin Yao is with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong, SAR, China, and also with the School of Computer Science, University of Birmingham, B15 2TT Birmingham, U.K. (e-mail: xinyao@ln.edu.hk).

Digital Object Identifier 10.1109/TKDE.2024.3384961

[15]. Given that ensembles have demonstrated effectiveness in addressing multi-class imbalance problem [5], [15], [16], and within which AdaBoost possesses a solid theoretical foundation and is extensively employed in ensemble learning [5], [15], [16], [17], our approach follows this comparable research trajectory.

We note that existing strategies normally depend on the information of class sizes to deal with class imbalance. In addition to this specific knowledge of class imbalance, we consider obtaining other information to better reflect class imbalance that can be applied throughout the training epochs of the AdaBoost ensemble learning. Adopting class imbalance ratio will assign each individual training sample of the same class with the same weight value, failing to reflect the differences among training samples of the same class. In other words, the class imbalance ratio can only differentiate “between-class” training samples that belong to different classes. We propose the density related variable to capture the distribution of training samples of the same class to distinguish such training samples. Then, we incorporate this density information with the class imbalance ratio to further enhance the capability of learning approaches in dealing with class imbalance. In this way, training samples of the same class would gain different weights, potentially improving predictive performance of learning methods with class imbalance.

Since the imbalance and density factors are calculated based on the entire training set in advance, their values remain static across the entire training process. As a result, the weight of a training sample resulting from the two factors cannot be adapted to the most current training status, potentially hindering the class imbalance mechanism from better dealing with multi-class imbalance. More specifically, non-adapted weights cannot reflect the up-to-date training status of constructed base learners for this training sample, of which sample weight values that are suitable at early training epochs would be obsolete at latter epochs of the ensemble learning process. In this sense, assigning an “adaptive” sample weight by making use of up-to-date training status would further alleviate the multi-class imbalance issue. For this, we propose to quantify the up-to-date training status via the probability difference that training samples have been correctly and most incorrectly classified across training epochs so far, leading to the design of the adaptive margin.

We can then unify the between-class imbalance, the within-class density, and the adaptive margin into a single factor to determine the weights of training samples at each training epoch. In this way, the proposed sample weight would be able to both differentiate the between-class training samples and the within-class training samples of the same class adaptively throughout training epochs. Ultimately, the weights of training samples can be encoded into the AdaBoost ensemble framework, leading to a novel multi-class classification approach to tackle the multi-class imbalance problem. We name the approach as **AdaBoost Ensemble with Adaptive Distribution based Sample Weight (AdaBoost.AD)**. We will provide theoretical deduction to demonstrate why the proposed AdaBoost.AD learning algorithm would be valid.

Experimental studies on 12 multi-class and 8 binary-class imbalanced data sets are conducted to investigate to what extent

the proposed AdaBoost.AD can address the multi-class imbalance issue by comparing against state-of-the-art multi-class imbalance approaches. The effectiveness of each of the proposed three components is analyzed experimentally, demonstrating their impact on the performance of the proposed algorithm.

This paper contributes to the management of multi-class imbalance data sets when they are used for classification. The main contribution is summarized as follows.

- 1) We propose the adaptive distribution based sample weight that can incorporate the between-class imbalance ratio, the within-class density variable, and the adaptive margin altogether to deal with multi-class imbalance.
- 2) We integrate the proposed adaptive sample weight with the learning framework of AdaBoost ensembles, contributing to the proposed AdaBoost.AD. We further provide the theoretical support on the learning algorithm of AdaBoost.AD.
- 3) We demonstrate the effectiveness of the proposed AdaBoost.AD for dealing with multi-class imbalance; we also analyse each of the three components of the adaptive sample weight.

The remainder of this paper is organized as follows. Section II discusses related work of multi-class imbalance problem. How the within-class data distribution is defined and how it is incorporated with between-class data distribution are presented in Section III. Section IV presents the way to define the adaptive sample weights based on the training status throughout training epochs. How to incorporate the proposed sample weights with the AdaBoost ensemble framework and why the learning algorithm is valid are discussed in Section V. Experimental design and analysis of results are reported in Section VI. The paper is concluded in Section VII.

II. RELATED WORK

Methods dealing with the multi-class imbalance problem can generally be cast into three categories [4], [5], [6]: data sampling approaches [7], [8], [9], cost sensitive approaches [18], and ensemble approaches [5], [13], [14].

A. Data Sampling Methods for Multi-Class Imbalance

Methods of this category are data-level. The core idea is to increase samples of the minority classes or to reduce samples of the majority classes, thereby alleviating the imbalance issue among different classes. Then, learning models can be constructed to make prediction without alternations [7], [8], [9].

ROS (Random Over-Sampling) that randomly replicates minority samples and RUS (Random Under-Sampling) that randomly deletes majority samples are the most popular data sampling methods. They were originally designed to deal with binary-class imbalance problem yet can be easily extended to dealing with the multi-class imbalance problem [5], [7]. Later in 2013, Lin et al. proposed a more sophisticated over-sampling technique for dealing with multi-class imbalance that takes classification difficulty of each training sample into account [6].

Moreover, SMOTE (Synthetic Minority Over-sampling Technique) [8] is another popular over-sampling method that generates synthetic minority data. There have been many variants of SMOTE [9], [19], [20], [21]. ADASYN (Adaptive Synthetic Sampling Approach) is one of the most popular variant of SMOTE, which is better able to generate synthetic data around the minority samples that are more isolated from their class label in the neighborhood [9]. However, most sampling based methods cater for class imbalance by manipulating the between-class imbalance ratio which typically remain static throughout the iterative training process (if any).

Sampling methods normally need to compute the pair-wise distance, and euclidean distance is popularly adopted for the computation [8], [9], [19], [20], [21]. More recently, there are studies that employed other distance metrics to try to capture additional characteristics of the data space, and subsequently gained good ability in dealing with multi-class imbalance. These include the Mahalanobis distance based over-sampling [22], Helliger distance based over-sampling [23], adaptive Mahalanobis distance-based over-sampling [24], and entropy-based sample distance [10], [11].

There are also some synthetic image generation methods in the community of deep learning that can be potentially used to alleviate multi-class imbalance [11]. Yet, those approaches are image data specific and time-consuming.

Sampling methods are straightforward and uncomplicated to implement. They operate at the data-level and can enhance the predictive performance of various machine learning methods by mitigating the bias towards the majority classes. There are also some cons. Oversampling can induce overfitting by duplicating existing data [7], while undersampling may result in the loss of important information within the majority class, resulting in reduced model performance [7], [10]. Additionally, applying sampling approaches can be challenging in scenarios with multiple classes and substantial computational costs [10], [11].

B. Cost-Sensitive Methods for Multi-Class Imbalance

Methods of this category are algorithm-level, which normally deal with class imbalance inside the classification methods themselves by assigning distinct misclassification costs or sample weights to various classes. These approaches are easy to implement and can be combined with various machine learning algorithms once costs or sample weights are properly defined, dealing with multi-class imbalance directly [4], [7]. Usually, the sample weights of a minority class is usually larger than that of a majority class [7], [12].

Designing good strategies to set sample weights is essential for cost-sensitive methods to deal with multi-class imbalance. A simple way is to set up sample weights according to class sizes, so that learning algorithms would emphasize samples of higher weights [18]. Whereas this technique may fail to achieve good performance when the information of class sizes is too limited to depict informative data distribution. More advanced cost-sensitive techniques also exist in the literature, yet they

are either time-consuming [25] or designed for binary classifiers [26], [27].

So far, it is still a difficult issue to design good sample weights, which might necessitate domain expertise for appropriate setup, and this challenge becomes even more pronounced when multiple classes are involved [4].

In this paper, we will propose an advanced strategy to derive sample weights based on the between-class and within-class data distribution, and experimentally justify its effectiveness in dealing with multi-class imbalance.

C. Ensemble Methods for Multi-Class Imbalance

Ensemble approaches involve combining multiple classifiers to enhance model accuracy, which have shown to perform generally well in the multi-class classification problem, and thus are popularly chosen as the benchmark learning approach in dealing with multi-class imbalance [5], [16]. Although conventional ensemble methods were not deliberately designed to deal with class imbalance, they can be equipped easily with other techniques that cater for class imbalance such as sampling or cost-sensitive techniques [25], [28], [29], offering their flexibility. JanEnsemble [30] is a state-of-the-art ensemble approach specifically crafted for addressing multi-class imbalance problem. This method employs adaptive optimization techniques to combine a variety of base learners, leading to a more potent classifier. To mitigate the class imbalance issue, each individual base learner is trained using a synthesized balanced data set that contains samples from different classes.

AdaBoost is a popular ensemble learning framework that was originally designed for binary classification [31] and then generalized to multi-class classification [17]. Wang et al. further boosted AdaBoost to AdaBoost.NC by encouraging the diversity of training samples so that those with larger error and lower diversity were more likely to be retained for training next base learners [5], [16], [32].

AdaBoost-based methods are extensively employed in multi-class imbalance, which have demonstrated the effectiveness and robustness with a solid theoretical foundation, including such as SMOTEBoost [13], RUSBoost [14], AdaBoost.NC [5], LexiBoost and Dual-LexiBoost [15]. Among them, LexiBoost and Dual-LexiBoost are state-of-the-art and thus are chosen as our competing methods. They were designed as a two-stage lexicographic linear programming framework. Specifically, LexiBoost dealt with multi-class imbalance via solving a linear programming problem, whereas Dual-LexiBoost embedded the solver of linear programming into the learning process of the AdaBoost ensemble. Both LexiBoost and Dual-LexiBoost are capable of dealing with binary and multi-class imbalance problems. Although ensemble methods can entail a computational burden due to the involvement of multiple base learners, this concern is not so significant compared with their effectiveness in dealing with multi-class imbalance [5], [15], [16].

In this paper, we will propose an adaptive distribution based AdaBoost ensemble to deal with multi-class imbalance, which can also be used in the binary class scenario. Sections III, IV,

and **V** will discuss our sample weight based AdaBoost approach into detail.

III. DATA DISTRIBUTION BASED SAMPLE WEIGHT

As explained previously, we adopt the AdaBoost [17] as the multi-class learning framework. Suppose that we have a data set consisting of n training samples $\{(X_i, y_i)\}$ to build a set of base learners for dealing with a c -class classification problem, where $X_i \in \mathbb{R}^d$ denotes a d -dimensional input features of the i -th training sample, $y_i \in \{1, \dots, c\}$ denotes the true label of the i -th training sample, and $i = 1, \dots, n$. Assuming there are m training epochs in total for the AdaBoost ensemble, there will be m base learners $\{h_t(\cdot)\}$ with $t \in \{1, \dots, m\}$, each of which is associated to a model weight β_t and is constructed at the t -th training epoch [17].

Using the above notations, this section will first specify the way to depict multi-class imbalance based on between-class sizes and then present a (temporary) sample weight by encoding the within-class data density into the between-class imbalance ratio for dealing with multi-class imbalance.

A. Encoding Between-Class Data Distribution

Let $y \in \{1, \dots, c\}$ and $N(y)$ be the non-negative function that gives the number of training samples with label y . Following the conventional manner to deal with class imbalance [6], we define the multi-class imbalance ratio associated to a training sample (X, y) based on class sizes as

$$\gamma(X, y) = \frac{N(y)}{\min_{y'} N(y')} \geq 1, \quad (1)$$

where $\min_{y'} N(y')$ denotes the minimal class size, and $y' \in \{1, \dots, c\}$. Specifically, the smallest class is used as a “benchmark” based on which imbalance ratios of other classes would be computed. A smaller (larger) class in comparison to the benchmark class would obtain a smaller (larger) class imbalance value. Taking a three-class classification problem with the between-class data distribution of 10 : 5 : 2 as an example, training samples of the second class will have their imbalance ratio computed as $5/2 = 2.5$.

Training samples of larger (smaller) imbalance ratio indicate relatively adequate (inadequate) data information that can be provided to model training. Thus, one might need to associate lower (higher) weights to those training samples, as will be explained later in Section III-C. In practice, samples with different class labels would usually have different class imbalance ratios; whereas samples of the same class would have the same class imbalance ratio, demonstrating a “between-class” property of the data distribution.

Although class imbalance ratio could depict the between-class data distribution based on which one can design an approach to deal with multi-class imbalance, deriving sample weights only based on the imbalance ratio would result in the same weight for training samples of the same class, which does not take into account the difference among training samples of the same class. We will introduce a factor to depict the within-class data distribution of each class in the subsequent section.

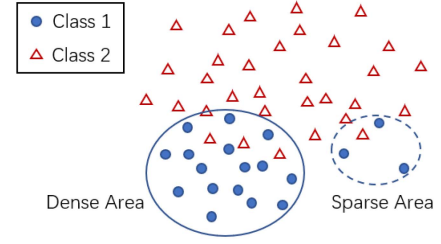


Fig. 1. Dense area vs sparse area of samples of class 1.

B. Encoding Within-Class Data Distribution

To specify the differences of data distribution among training samples of the same class, we will produce the density for each training sample within this class. Such density of a training sample would be decided based on the amount samples of the same class being located within the neighborhood of this data, being with a unified density scale that is especially decided for this class. As a result, the associated sample weights would reflect the information of not only the between-class but also within-class data distribution, potentially helping with the multi-class imbalance issue.

Fig. 1 illustrates a data scenario that samples of class 1 (blue dot) overlaps those of class 2 (red triangle) in two areas with varying within-class distribution properties: a “dense” area containing more samples of class 1 compared to a “sparse” area containing much less samples of class 1. Intuitively, incapability of dealing with samples of class 1 in the dense area would probably cause more significant drop in predictive performance of classification models compared to that in the sparse area. Therefore, training samples of class 1 in the dense (sparse) area should be over-fitted (under-fitted) while dealing with class imbalance to gain better overall predictive performance.

Given the i -th training sample (X_i, y_i) out of the total n , we use $\mathcal{N}(X_i)$ to denote the k -nearest neighbor set of X_i among the entire training samples, where k is a predefined parameter. Supposing that there are $k_i \leq k$ samples in the neighbor set $\mathcal{N}(X_i)$ that are with the same class as y_i , we name the subset of $\mathcal{N}(X_i)$ as the *within-class neighbor set* of (X_i, y_i) and denote it as $\mathcal{N}(X_i, y_i)$.¹ We use $\mathcal{N}(X_i, y_i)_j$ to denote the j -th within-class neighbor of X_i in the class neighbor set for $j = 1, \dots, k_i$, based on which the *general density* of (X_i, y_i) is defined as

$$\rho(X_i, y_i) = \begin{cases} \left(\frac{1}{k_i} \sum_{j=1}^{k_i} \frac{1}{d(X_i, \mathcal{N}(X_i, y_i)_j)} \right) / T, & \text{if } k_i > 0 \\ 0, & \text{if } k_i = 0, \end{cases} \quad (2)$$

where $d(\cdot, \cdot)$ denotes the distance of two training samples and T is a constant determined based on all training samples (defined later). We can see that the general density value of a training sample is determined based on the average distance of this sample away from its selective within-class neighbors and the within-class neighbors involved in this computation depends on how many training samples of this class are within the

¹Note the difference of $\mathcal{N}(X_i, y_i)$ from $\mathcal{N}(X_i)$, which needs to query the true label y_i for construction.

k -nearest neighbors. We adopt euclidean distance and set $k = 5$ throughout the experiments following previous study [10].

The constant T is defined as $\sum_{i=1}^n \rho(X_i, y_i)/n$, which can be considered as a normalizer for inducing a unit sample density across all training samples. It is for this reason that $\rho(X_i, y_i)$ is named as a general density. In this way, training samples from different data sets would gain a uniformed data scale, well prepared for the treatment later on. Specifically, training samples with larger (smaller) general density values than 1 (the benchmark unit) indicate them to be in a denser (sparser) area compared to the benchmark density.

General density of training samples is established at the scale of entire training set in order to unify the density scale of data sets with varying characteristics. To explicitly quantify the difference in terms of the within-class data distribution, we define the *inverse within-class density*² of training sample (X_i, y_i) as

$$\rho^{-1}(X_i, y_i) = \frac{\exp\{-\rho(X_i, y_i)\}}{Z_i}, \quad (3)$$

where $\rho(X_i, y_i)$ denotes the general density in (2) and Z_i represents the within-class normalizer at the level of each class (i.e., within-class), defined as

$$Z_i = \frac{1}{\|C_i\|} \sum_{X_j \in C_i} \exp\{-\rho(X_j, y_j)\}$$

where subset C_i contains training samples of the same label as y_i and $\|C_i\|$ is the size of the sub-training set. As a result, within-class density values for each class would have a unit inverse density.

Specifically, training samples of each class would derive a unified inverse within-class density, benchmarking around a unit value, and a smaller (larger) value indicates a desire for having a larger (smaller) sample weight for dealing with class imbalance.

C. Distribution Based Sample Weight

Training samples with smaller between-class imbalance and inverse within-class density values would be assigned with higher weights so that the training process would pay more attention to them, better dealing with class imbalance. This information can be incorporated into a single *data distribution* factor, being defined as

$$\alpha(X, y) = \exp\{-\gamma(X, y) \cdot \rho^{-1}(X, y)\}, \quad (4)$$

where $\gamma(\cdot)$ denotes the between-class imbalance ratio in (1) and $\rho^{-1}(\cdot)$ denotes the inverse within-class density in (3). We can see that $\alpha(\cdot)$ ranges in $(0, 1]$.

Supposing a new training epoch $t + 1$, the weight of training sample (X, y) in the conventional AdaBoost ensemble can be reformulated as

$$\omega^{(t+1)}(X, y) = \alpha(X, y) \cdot \omega^{(t)}(X, y) \cdot \exp\{-\beta_t \cdot \mathbb{1}_{[h_t(X)=y]}\}, \quad (5)$$

where $\alpha(\cdot)$ has been formulated in (4), $\omega^{(t)}(\cdot)$ denotes the sample weight that is iteratively decided at the previous training epoch t ,

²We opt for the ‘‘inverse’’ density to simplify the way to encode data distribution information into sample weights, as will be noted later.

β_t denotes the weight of the base learner $h_t(\cdot)$ that is constructed at training epoch t , and $\mathbb{1}_{[\text{condition}]}$ is the indicator function defined as

$$\mathbb{1}_{[\text{condition}]} := \begin{cases} 1 & \text{if the condition is true,} \\ 0 & \text{if otherwise.} \end{cases}$$

In particular, base learner $h_t(\cdot)$ and its weight β_t are obtained according to the conventional AdaBoost ensemble framework [17].

Therefore, we could encode the data distribution based sample weight directly into the ensemble learning framework of AdaBoost.

IV. MAKING SAMPLE WEIGHT ADAPTIVE

As the between-class imbalance ratio and the inverse within-class density of a training sample are calculated based on the entire training set in advance, their values remain static throughout the training epochs of the AdaBoost ensemble framework. As a result, the weight of a training sample resulting from the two factors as in (5) cannot be adapted to the most current training status, potentially hindering the class imbalance mechanism from working well.

More specifically, non-adapted weights cannot reflect the up-to-date training status how well base learners perform on this training sample, possibly causing sample weights that are suitable at early training epochs to become obsolete later on. Thus, assigning adaptive sample weights by making use of up-to-date training status would potentially further alleviate the multi-class imbalance problem.

This section will propose to quantify up-to-date training status via the probability difference that training samples have been correctly and the most incorrectly classified across training epochs so far, and then encode the training status into distribution based sample weights, contributing to the adaptive distribution based sample weight.

A. Tracing Up-to-Date Training Status

Given a sample (X, y) at training epoch t , we use $o^{(t)}(X, y')$ to denote the outputs of t base learners of AdaBoost that vote X to class $y' \in \{1, \dots, c\}$, which can be formulated as

$$o^{(t)}(X, y') = \sum_{\nu=1}^t \beta_\nu \cdot \mathbb{1}_{[h_\nu(X)=y']}, \quad (6)$$

where $\mathbb{1}_{[\cdot]}$ is the indicator function and $h_\nu(\cdot)$ denotes the base learner that is constructed earlier at the ν -th training epoch. With this in mind, the probability that X is predicted to class c can be quantified as

$$\mathbb{P}^{(t)}(X, y') = \frac{\exp\{o^{(t)}(X, y')\}}{\sum_{l=1}^c \exp\{o^{(t)}(X, l)\}}, \quad (7)$$

based on all the base learners that have been constructed so far via the softmax function.

Repeating the above calculation across all training samples for all label classes, we can attain a probability matrix as demonstrated in Table I, recording the up-to-date probabilities of each training sample being predicted to each class at the training epoch t .

TABLE I
PROBABILITY MATRIX AT TRAINING EPOCH t

Sample	Class 1	Class 2	...	Class c
X_1	$\mathbb{P}^{(t)}(X_1, 1)$	$\mathbb{P}^{(t)}(X_1, 2)$...	$\mathbb{P}^{(t)}(X_1, c)$
X_2	$\mathbb{P}^{(t)}(X_2, 1)$	$\mathbb{P}^{(t)}(X_2, 2)$...	$\mathbb{P}^{(t)}(X_2, c)$
...
X_n	$\mathbb{P}^{(t)}(X_n, 1)$	$\mathbb{P}^{(t)}(X_n, 2)$...	$\mathbb{P}^{(t)}(X_n, c)$

Given current training epoch t , we propose to quantify the up-to-date training status to be the difference between the probability that training samples have been correctly classified and that they are uppermost incorrectly classified throughout training epochs so far, for which we define as the *adaptive margin*. Specifically, based on the probability matrix of Table I, the adaptive margin of an arbitrary training sample (X, y) can be formulated as

$$\sigma^{(t)}(X, y) = \mathbb{P}^{(t)}(X, y) - \max_{y' \neq y} \mathbb{P}^{(t)}(X, y') + 1, \quad (8)$$

where y and $y' \in \{1, \dots, c\}$ denote data labels, $\mathbb{P}^{(t)}(X, y)$ represents the probability X can be correctly classified as computed in (7), and $\max_{y' \neq y} \mathbb{P}^{(t)}(X, y')$ represents the maximal probability X can be misclassified. We can know that $\sigma^{(t)}(\cdot)$ ranges from 0 to 2.

The special case of $\sigma^{(t)}(X_i) = 1$, corresponding to $\mathbb{P}^{(t)}(X_i, y_i) = \max_{y' \neq y_i} \mathbb{P}^{(t)}(X_i, y')$, showcases the highest uncertainty the up-to-date AdaBoost ensemble would confront for prediction. Training samples with $\sigma^{(t)}(\cdot) > 1$ mean that they can be correctly predicted by the up-to-date classifier; whereas those with $\sigma^{(t)}(\cdot) < 1$ mean that they would be wrongly classified. In this sense, $\sigma^{(t)}(X)$ can be used to indicate the difficulty training sample X would be correctly predicted, and smaller (larger) $\sigma^{(t)}(X)$ means a higher (lower) difficulty the AdaBoost ensemble can perform the prediction.

Altogether, we design the adaptive margin $\sigma^{(t)}(\cdot)$ to trace the up-to-date training status of base learners of the AdaBoost ensemble, based on which we would make sample weights adaptive in the subsequent section.

B. Adaptive Distribution Based Sample Weight

At a training epoch, samples with smaller adaptive margin would be assigned with higher weights so that the training process at this moment would pay more attention to them, better dealing with the up-to-date class imbalance issue. The adaptive margin factor can be further incorporated into static data distribution factor of (4), contributing to an *adaptive data distribution* factor as

$$\alpha^{(t)}(X, y) = \exp\{-\gamma(X, y) \cdot \rho^{-1}(X, y) \cdot \sigma^{(t)}(X, y)\} \quad (9)$$

where $\gamma(\cdot)$, $\rho^{-1}(\cdot)$, and $\sigma^{(t)}(\cdot)$ are the between-class imbalance ratio in (1), the inverse within-class density factor in (3), and the adaptive margin in (8), respectively. We can know that $\alpha^{(t)}(\cdot)$ ranges in $(0, 1]$.

Supposing a new training epoch $t + 1$, the weight of training sample (X, y) in the AdaBoost ensemble framework can be

defined as

$$\omega^{(t+1)}(X, y) = \alpha^{(t)}(X, y) \cdot \omega^{(t)}(X, y) \cdot \exp\{-\beta_t \cdot \mathbb{1}_{[h_t(X)=y]}\}, \quad (10)$$

where $\alpha^{(t)}(\cdot)$ has been defined in (9), and $\omega^{(t)}(X, y)$ denotes the adaptive sample weight that is iteratively decided at the previous training epoch t . Other notations are the same to the static sample weight in (5). The crucial difference between them is $\alpha^{(t)}(\cdot)$ that contribute the adaptive nature to the adaptive weight of (10).

Therefore, we can encode the adaptive data distribution based sample weight into the ensemble learning framework of AdaBoost as will be explained in Section V-A. It is also noteworthy that the non-adapted sample weights of (5) will not be employed in practice; rather it is a temporary outcome during the process of deriving the adaptive sample weight in this section.

V. ADABOOST ENSEMBLE WITH ADAPTIVE DISTRIBUTION BASED SAMPLE WEIGHT

This section aims to incorporate the adaptive distribution based sample weights into the AdaBoost ensemble framework (the backbone of the classification framework), leading to a novel multi-class imbalance learning approach. We name the approach as **AdaBoost Ensemble with Adaptive Distribution based Sample Weight (AdaBoost.AD)**. We will also present the learning algorithm of the proposed AdaBoost.AD in detail and provide theoretical support on why this learning algorithm to be valid.

A. Learning Algorithm of AdaBoost.AD

Algorithm 1 summarizes the learning algorithm of the proposed AdaBoost.AD, which generally follows the learning procedure of conventional AdaBoost [17]. The major differences between them are highlighted in blue (gray), among which the computation of model weights (step 6) and the update of sample weights (steps 1, 2, 7 and 8) are two important components.

For training epoch t , model weight β_t of the newly constructed base learner $h_t(\cdot)$ and sample weights $\omega^{(t+1)}(\cdot)$ that are about to be used at next training epoch are two important learnable variables in the learning algorithm of AdaBoost.AD. Learning sample weights has been presented in Section IV-B, so we will detail how to determine the model weight in this section. Since the update of the two variables are interlinked, we will reiterate the formulation of sample weights here for clarification.

Given sample weights $\omega^{(t)}(\cdot)$ that were decided at previous training epoch t , the model weight of the newly constructed base learner $h_t(\cdot)$ is determined as

$$\beta_t = \log \frac{\sum_{i=1}^n \omega^{(t)}(X_i, y_i) \cdot \mathbb{1}_{[h_t(X_i)=y_i]}}{\sum_{i=1}^n \omega^{(t)}(X_i, y_i) \cdot \mathbb{1}_{[h_t(X_i) \neq y_i]}}, \quad (11)$$

where $\mathbb{1}_{[\cdot]}$ is the indicator function. Similar to the conventional AdaBoost [17], the model weight of base learner of AdaBoost.AD is computed according to the ratio of the number of correct prediction over that of incorrect prediction. Normally, the denominator would be smaller than the numerator

Algorithm 1: Learning Algorithm of AdaBoost.AD. The Primary Enhancement of AdaBoost.AD in Comparison to the Conventional AdaBoost Includes Steps 1–2 and 6–8.

Inputs: (1) n training samples $\{(X_i, y_i)\}$ where $i = 1, \dots, n$ and $y_i \in \{1, \dots, c\}$, and (2) the number of base learner m .

Outputs: (1) base learners $\{h_t(\cdot)\}$ and (2) the associated model weights $\{\beta_t\}$ with $t = 1, \dots, m$.

- 1: Compute between-class imbalance $\gamma(\cdot)$ as (1).
- 2: Compute inverse within-class density $\rho^{-1}(\cdot)$ as (3).
- 3: Initial sample weights $\omega^{(1)}(X_i, y_i) = 1/n$.
- 4: **for** training epoch $t = 1$ until m **do**
- 5: Construct base learner $h_t(\cdot)$ based on training samples and up-to-date sample weights $\omega^{(t)}(\cdot)$ that was computed at previous training epoch.
- 6: Compute model weight β_t in (11) based on the latest sample weights $\omega^{(t)}(\cdot)$.
- 7: Compute or update adaptive margin $\sigma^{(t)}(\cdot)$ as (8).
- 8: Update sample weights $\omega^{(t+1)}(\cdot)$ as (12) by considering $\gamma(\cdot)$, $\rho^{-1}(\cdot)$, and $\sigma^{(t)}(\cdot)$ altogether.
- 9: **end for**

Prediction: Given a test sample X , the predicted label \hat{y} can be decided by $\hat{y} = \arg \max_{l \in \{1, \dots, c\}} \sum_{t=1}^m \beta_t \cdot \mathbb{1}_{[h_t(X)=l]}$.

so that we would usually have positive model weights, i.e., $\beta_t > 0$.

Next, we will prepare for the learning procedure at next training epoch with updated sample weights based on the new base learner $h_t(\cdot)$ and its derived model weight β_t . As has been formulated in (10), the weight of training sample (X_i, y_i) that is about to be used at next training epoch $t + 1$ is formulated as

$$\omega^{(t+1)}(X_i, y_i) = \boxed{\alpha^{(t)}(X_i, y_i)} \cdot \omega^{(t)}(X_i, y_i) \cdot \exp\{-\beta_t \cdot \mathbb{1}_{[h_t(X_i)=y_i]}\}. \quad (12)$$

We can see that the adaptive data distribution factor $\alpha^{(t)}(X, y)$ in (9) pioneers the advancement of the proposed AdaBoost.AD over the conventional AdaBoost. With the informative knowledge encoded, training samples associating to larger $\alpha^{(t)}(\cdot)$ would induce higher sample weights and become more important at the next training epoch, being potentially beneficial to alleviate the class imbalance issue with time adaptively. Moreover, the way we propose the learning algorithm of AdaBoost.AD follows that of the conventional AdaBoost ensemble, so that we can largely retain the learning framework of AdaBoost with the minimal yet crucial alterations.

Providing the trained AdaBoost.AD, the prediction of a test sample X can be performed as

$$\hat{y} = \arg \max_{l \in \{1, \dots, c\}} \sum_{t=1}^m \beta_t \cdot \mathbb{1}_{[h_t(X)=l]}, \quad (13)$$

being similar to the prediction mechanism of the conventional AdaBoost [17]. The effectiveness of the proposed AdaBoost.AD

TABLE II
ESSENTIAL NOTATIONS

Notation	Description
$\mathbb{1}_{[\cdot]}$	Indicator function
X_i	The feature vector of the i -th training sample
y_i	The true label of the i -th training sample
$h_t(\cdot)$	Base learners of the AdaBoost ensemble that is constructed at training epoch t
β_t	Model weight of base learner $h_t(\cdot)$
$\alpha^{(t)}(X_i, y_i)$	The adaptive data distribution factor of training sample (X_i, y_i) at training epoch t
$L^{(t)}(X_i, y_i)$	Loss of training sample (X_i, y_i) at training epoch t

in dealing with multi-class imbalance will be investigated experimentally in Section VI, by competing against state-of-the-art Boost-based ensembles.

B. Theoretical Analyses on the Learning Algorithm

In this section, we aim to provide theoretical supports on the validity of the proposed learning algorithm of AdaBoost.AD. Especially, we will validate our derivation of sample weight as in (12) and showcase the correct formulation of our model weight as in (11). We will pursue the theoretical analyses with the analogy of the learning procedure of the conventional AdaBoost ensemble for the purpose of simplicity and clarity. Essential notations can be found in Table II.

With respect to the conventional AdaBoost ensemble, at training epoch t , the loss of training sample (X_i, y_i) with respect to all base learners constructed so far can be formulated in the exponential form as

$$\mathcal{L}^{(t)}(X_i, y_i) = \exp\left(-\sum_{\nu=1}^t \beta_\nu \cdot g_\nu(X_i, y_i)\right),$$

where $g_\nu(X_i, y_i) = \mathbb{1}_{[h_\nu(X_i)=y_i]} - \frac{1}{2}$, and $h_\nu(\cdot)$ represents base learners of the AdaBoost ensemble that is constructed at training epoch ν . According to the property of exponential function, this sample loss can be rephrased as

$$\mathcal{L}^{(t)}(X_i, y_i) = \prod_{\nu=1}^t \exp(-\beta_\nu \cdot g_\nu(X_i, y_i)),$$

where $\exp(-\beta_\nu \cdot g_\nu(X_i, y_i))$ can represent the sample loss of (X_i, y_i) with respect to a single base learner $h_\nu(\cdot)$.

On the other hand, the adaptive data distribution factors $\{\alpha^{(\nu)}(X_i, y_i)\}$ in (9) encode the information that are about to facilitate the construction of base learners and the computation of associated model weights. Therefore, each $\alpha^{(\nu)}(X_i, y_i)$ should be aligned with the sample loss of each base learner, formulated as

$$\mathcal{L}^{(t)}(X_i, y_i) = \prod_{\nu=1}^t \boxed{\alpha^{(\nu-1)}(X_i, y_i)} \cdot \exp(-\beta_\nu \cdot g_\nu(X_i, y_i)), \quad (14)$$

contributing to the novel parts of the learning algorithm of the proposed AdaBoost.AD. For clarity, the advancement of

AdaBoost.AD over the conventional AdaBoost ensemble is highlighted in blue throughout this subsection.

We further decompose the $\mathcal{L}^{(t)}(X_i, y_i)$ into the multiplication of three components as $[\prod_{\nu=1}^t \alpha^{(\nu-1)}(X_i, y_i)] \cdot [\exp(-\sum_{\nu=1}^{t-1} \beta_\nu \cdot g_\nu(X_i, y_i))] \cdot \exp(-\beta_t \cdot g_t(X_i, y_i))$. Assembling to define $\omega^{(t)}(X_i, y_i)$ to be

$$\omega^{(t)}(X_i, y_i) = \left[\prod_{\nu=1}^t \alpha^{(\nu-1)}(X_i, y_i) \right] \cdot \left[\exp \left(- \sum_{\nu=1}^{t-1} \beta_\nu \cdot g_\nu(X_i, y_i) \right) \right]. \quad (15)$$

The sample loss in (14) can be rephrased as

$$\mathcal{L}^{(t)}(X_i, y_i) = \omega^{(t)}(X_i, y_i) \cdot \exp(-\beta_t \cdot g_t(X_i, y_i)). \quad (16)$$

By analogy to (15), the iterative formulation of sample weight of (X_i, y_i) at training epoch $t + 1$ can be deduced as

$$\begin{aligned} \omega^{(t+1)}(X_i, y_i) &= \left[\prod_{\nu=1}^{t+1} \alpha^{(\nu-1)}(X_i, y_i) \right] \cdot \left[\exp \left(- \sum_{\nu=1}^t \beta_\nu \cdot g_\nu(X_i, y_i) \right) \right] \\ &= \boxed{\alpha^{(t)}(X_i, y_i)} \cdot \omega^{(t)}(X_i, y_i) \cdot \exp \left(- \beta_t \cdot g_t(X_i, y_i) \right) \\ &= \boxed{\alpha^{(t)}(X_i, y_i)} \cdot \omega^{(t)}(X_i, y_i) \cdot \exp \left(- \beta_t \cdot \left(\mathbb{1}_{[h_t(X_i)=y_i]} - \frac{1}{2} \right) \right) \\ &= \boxed{\alpha^{(t)}(X_i, y_i)} \cdot \omega^{(t)}(X_i, y_i) \cdot \exp(-\beta_t \cdot \mathbb{1}_{[h_t(X_i)=y_i]}) \cdot c_t \end{aligned} \quad (17)$$

where $c_t = \exp(\beta_t/2)$ at the last step is a constant with respect to the training sample (X_i, y_i) and thus can be eliminated from the equation of sample weight in the end.

Therefore, the ultimate sample weight can be simplified, without loss of generalization, as

$$\omega^{(t+1)}(X_i, y_i) = \alpha^{(t)}(X_i, y_i) \cdot \omega^{(t)}(X_i, y_i) \cdot \exp(-\beta_t \cdot \mathbb{1}_{[h_t(X_i)=y_i]}),$$

providing theoretical support for our design of iterative sample weights $\omega^{(t+1)}(\cdot)$ based on the adaptive data distribution factor as in (10) and (12).

Next, we will derive how to compute the model weight of the newly constructed base learner based on the latest sample weights. Conventionally, the weight of a base learner should be deduced by minimizing the optimization function (across all training samples), formulated as

$$\sum_{i=1}^n \mathcal{L}^{(t)}(X_i, y_i) = \sum_{i=1}^n \omega^{(t)}(X_i, y_i) \cdot \exp(-\beta_t \cdot g_t(X_i, y_i)).$$

To minimize the objective, the partial derivatives in terms of each model weight β_t should be set to zero, contributing to the equations, for $t = 1, \dots, m$, that

$$\sum_{i=1}^n \omega^{(t)}(X_i, y_i) \cdot (-g_t(X_i, y_i)) \cdot \exp(-\beta_t \cdot g_t(X_i, y_i)) = 0.$$

Moreover, the entire training samples can be divided into two parts: those that can be correctly classified with the condition of “ $h_t(X_i) = y_i$ ” and those that are incorrectly classified with the condition of “ $h_t(X_i) \neq y_i$ ”. Substituting with $g_t(X_i, y_i) = \mathbb{1}_{[h_t(X_i)=y_i]} - \frac{1}{2}$, the previous equation can be rephrased as

$$\begin{aligned} &\sum_{h_t(X_i)=y_i} \omega^{(t)}(X_i, y_i) \cdot \exp \left(- \frac{\beta_t}{2} \right) \\ &= \sum_{h_t(X_i) \neq y_i} \omega^{(t)}(X_i, y_i) \cdot \exp \left(\frac{\beta_t}{2} \right). \end{aligned}$$

Extracting $\exp(\beta_t/2)$, assigning logarithm operator to both sides of the equation, and reallocating the indexes of summation, we can derive

$$\beta_t = \log \frac{\sum_{i=1}^n \omega^{(t)}(X_i, y_i) \cdot \mathbb{1}_{[h_t(X_i)=y_i]}}{\sum_{i=1}^n \omega^{(t)}(X_i, y_i) \cdot \mathbb{1}_{[h_t(X_i) \neq y_i]}}$$

Therefore, the above deduction provides theoretical support for our formulation of model weight β_t as in (11).

VI. EXPERIMENTAL STUDIES

This section aims to investigate the proposed AdaBoost.AD from two perspectives: the performance comparison against state-of-the-art multi-class imbalance approaches and the effect of the three components of the adaptive distribution based sample weight.

A. Experimental Setting

Experimental studies will be performed based on 12 open multi-class imbalanced data sets and 8 open binary-class imbalanced data sets, which are available from the Keel repository [33]. Table III summarizes the basic description of data sets investigated in this paper. Note that we have deleted the classes that have less than 10 samples in total. The reason is that we opt for 5-fold cross-validation (CV) [34] to evaluate predictive performance of learning approaches, where 4/5 samples are used for training and the rest 1/5 are used for evaluation. As a result, classes with less than 10 samples would have at most two test sample, potentially making performance evaluation on this class less reliable.

The proposed AdaBoost.AD (“AdaAD”) will be compared against two groups of multi-class ensemble methods. The first group consists of three classical AdaBoost ensemble methods, including AdaBoost.M1 [17], AdaBoost.M2 [17], and AdaBoost.NC [5]. They are equipped with the oversampling technique ADASYN [9] for dealing with class imbalance, leading to AdaBoost.M1 with ADASYN (“AdaM1+ADASYN”), AdaBoost.M2 with ADASYN (“AdaM2+ADASYN”), and AdaBoost.NC with ADASYN (“AdaNC9+ADASYN” with its model parameter $\lambda = 9$ following prior work [5]), respectively. Our preliminary experiments found general superiority of ADASYN when combining with the three classical AdaBoost ensembles in dealing with class imbalance in comparison to

TABLE III
STATISTICAL INFORMATION OF THE INVESTIGATED DATA SETS

Data set	#Data	#Fea	#Class	Class Distribution	IR
Balance	625	4	3	288:49:288	5.88
Contraceptive	1473	9	3	629:333:511	1.89
Ecoli	327	7	5	143:77:52:35:20	7.15
Glass	205	9	5	70:76:17:13:29	5.85
Hayesroth	132	4	3	51:51:30	1.7
Newthyroid	215	5	3	150:35:30	5
Pageblocks	537	10	3	492:33:12	41
Penbased	1100	16	10	115:114:114:106:114:106:105:115:105:106	1.10
Shuttle	2167	9	3	1706:338:123	13.87
Thyroid	720	21	3	17:37:666	39.18
Wine	178	13	3	59:71:48	1.48
Yeast	1479	8	9	463:429:244:163:51:44:35:30:20	23.15
Ecoli01	220	7	2	143:77	1.86
Ecoli1	336	7	2	77:259	3.36
Ecoli2	336	7	2	52:284	5.46
Ecoli3	336	7	2	35:301	8.6
Haberman	306	3	2	81:225	2.78
Newthyroid1	215	5	2	35:180	5.14
Newthyroid2	215	5	2	35:180	5.14
Yeast3	1484	8	2	163:1321	8.10

#Data denotes the total number of samples within this data set, #Fea denotes the number of features, #Class denotes the number of classes, Class Distribution illustrates the sample distribution across each class, and IR signifies the imbalance ratio computed as the ratio between the size of the largest class and the size of smallest class.

other techniques such as random oversampling, random under-sampling [7], hybrid sampling [35], and SMOTE [8].

The second group of competing methods consists of four state-of-the-art AdaBoost-based multi-class classifiers that can deal with class imbalance on their own, including two versions of LexiBoost[15], Dual-LexiBoost [15], and JanEnsemble [30]. Specifically, LexiBoost framed with AdaBoost.M1 and AdaBoost.M2 lead to “LexiBoostM1” and “LexiBoostM2” in the experimental studies. Except for JanEnsemble, decision tree is adopted as the base learner of the ensemble approaches and the total training epoch is set to 50, following the prior works [5], [36]. Our experimental configuration of JanEnsemble adheres to the source code provided by the authors [30], for which the base learners consisted of a variety of learning machines including k -Nearest Neighbor (kNN), Discriminant Analysis (DISCR), Support Vector Machines (SVM), Artificial Neural Networks (ANN), and Decision Trees (DT). The number of base learners is decided through automated adaptive optimization within the method.

Predictive performance is evaluated based on the 10 times 5-fold cross-validation (CV) [34]. G-mean [25] and Avg-AUC [37] are chosen as the performance metrics as they are popularly used and less sensitive to the class imbalance issue. For each performance metric, we will report the average performance metric “ \pm ” standard deviation. To gain a comprehensive performance analysis, we perform Friedman tests [38] for statistical comparisons of multiple methods across all data sets. The null hypothesis (H0) states that all models are equivalent in terms of predictive performance. The alternative hypothesis (H1) states that at least one pair of the methods differ. If the null hypothesis

is rejected, Holm-Bonferroni correction [39] will be conducted as the post-hoc test. For pair-wise comparisons on each data set, we conduct Wilcoxon signed rank tests [40] between the proposed AdaBoost.AD and each of the competing methods to determine whether there are significant performance differences. The second last rows of Tables IV and V show the results, in which “win” denotes the number of data sets that our AdaBoost.AD is significantly better than the corresponding competitor, “tie” denotes the number of data sets that no significant difference can be found between AdaBoost.AD and the competitor, and “lose” denotes the number of data sets that AdaBoost.AD is significantly worse than the competitor.

The experiments are run in MATLAB R2021a on Dell workstation with Intel(R) Core(TM) i7-8700 CPU. Experimental results of LexiBoostM1, LexiBoostM2, Dual-LexiBoost, and JanEnsemble were produced based on the implementation in MATLAB provided by the authors.

B. Performance Comparison

In this section, we will demonstrate the superiority of our proposed AdaBoost.AD against the six competing methods. Tables IV and V show the predictive performance of investigated methods in terms of G-mean and Avg-AUC, respectively. The proposed AdaBoost.AD is listed in the last column. Note that, like LexiBoostM1, LexiBoostM2, Dual-LexiBoost, and JanEnsemble, the proposed AdaBoost.AD does not need to be equipped with additional class imbalance technique since it is designed to tackle the multi-class imbalance problem on its own.

In terms of G-mean, Table IV shows that our AdaBoost.AD outperformed the other methods on 12 out of 20 data sets, thus illustrating its superiority of yielding good predictive performance. Friedman test at the significance level 0.05 rejects H0 with the p -value $3.885 \cdot 10^{-12}$, meaning that some of these methods perform significantly differently. Next, we conduct post-hoc tests to facilitate further comparisons, to determine whether our proposed method exhibits statistically significant differences from its competitors. AdaBoost.AD is chosen as the control method for consistently performing the best among all classifiers. Post-hoc tests show that our method has significant differences from all competing methods, as highlighted in yellow within the second last row of Table IV. Moreover, Friedman tests also provide rankings of these methods. Let r_i^j denote the rank of the i -th method on the j -th data set, and N denote the total number of data sets. The average rank of the i -th method is calculated as $R_i = \frac{1}{N} \sum_j r_i^j$, providing a reasonable idea of how well the method performs compared to others. The third last row of Table IV (“avgRank”) reports the average rank of each method across all data sets (R_i). The average rank of our proposed AdaBoost.AD is 1.8, being the best (lowest) value among the competing methods. This result indicates that our method consistently outperformed the others across data sets.

The second last row of Table IV reports the win-tie-lose counts of AdaBoost.AD in comparison to other methods. When compared against AdaM1+ADASYN, AdaM2+ADASYN, AdaNC9+ADASYN, LexiBoostM1, LexiBoostM2, Dual-LexiBoost, and JanEnsemble, our proposed AdaBoost.AD significantly outperforms each of them on 12, 13, 10, 19, 18, 15, and

TABLE IV
PREDICTIVE PERFORMANCE IN TERMS OF G-MEAN

Data set	AdaM1+ADASYN	AdaM2+ADASYN	AdaNC9+ADASYN	LexiBoostM1	LexiBoostM2	Dual-LexiBoost	JanEnsemble	AdaAD
balance	0.057±0.143	0.092±0.176	0.241±0.237	0.463±0.190	0.505±0.139	0.673±0.097	0.835±0.073	0.635±0.078
contraceptive	0.465±0.030	0.461±0.033	0.510±0.026	0.469±0.037	0.479±0.033	0.480±0.032	0.478±0.031	0.483±0.032
ecoli	0.785±0.063	0.769±0.080	0.764±0.091	0.700±0.172	0.705±0.086	0.731±0.080	0.803±0.055	0.768±0.073
glass	0.630±0.265	0.551±0.304	0.620±0.242	0.478±0.299	0.469±0.274	0.478±0.264	0.491±0.203	0.668±0.185
hayesroth	0.825±0.060	0.702±0.104	0.863±0.071	0.822±0.066	0.821±0.068	0.826±0.060	0.714±0.097	0.802±0.079
newthyroid	0.893±0.076	0.912±0.064	0.904±0.074	0.899±0.071	0.908±0.070	0.896±0.066	0.692±0.121	0.948±0.055
pageblocks	0.648±0.321	0.682±0.316	0.670±0.308	0.605±0.315	0.609±0.360	0.656±0.308	0.583±0.382	0.843±0.199
penbased	0.972±0.010	0.969±0.012	0.942±0.017	0.904±0.039	0.882±0.041	0.856±0.024	0.909±0.025	0.971±0.011
shuttle	0.999±0.003	0.999±0.003	0.998±0.003	0.996±0.005	0.996±0.005	0.996±0.003	0.994±0.006	0.998±0.003
thyroid	0.859±0.195	0.905±0.162	0.878±0.154	0.788±0.228	0.877±0.202	0.884±0.155	0.898±0.102	0.976±0.032
wine	0.933±0.042	0.943±0.047	0.930±0.063	0.926±0.044	0.925±0.046	0.928±0.057	0.938±0.035	0.967±0.028
yeast	0.183±0.238	0.093±0.189	0.208±0.249	0.114±0.195	0.052±0.129	0.231±0.218	0.152±0.207	0.305±0.214
ecoli01	0.956±0.029	0.972±0.025	0.972±0.022	0.971±0.022	0.966±0.028	0.977±0.022	0.974±0.027	0.978±0.020
ecoli1	0.873±0.052	0.869±0.049	0.879±0.052	0.793±0.066	0.814±0.069	0.858±0.056	0.853±0.048	0.880±0.047
ecoli2	0.870±0.088	0.892±0.064	0.872±0.067	0.813±0.091	0.844±0.073	0.839±0.069	0.875±0.097	0.899±0.058
ecoli3	0.770±0.154	0.814±0.091	0.782±0.119	0.710±0.151	0.717±0.150	0.735±0.129	0.771±0.145	0.844±0.089
haberman	0.519±0.090	0.526±0.081	0.599±0.081	0.520±0.080	0.529±0.076	0.560±0.085	0.551±0.093	0.585±0.077
newthyroid1	0.944±0.075	0.951±0.076	0.956±0.077	0.921±0.079	0.929±0.076	0.925±0.087	0.835±0.167	0.953±0.075
newthyroid2	0.959±0.056	0.957±0.048	0.957±0.061	0.936±0.067	0.940±0.066	0.940±0.050	0.837±0.136	0.974±0.054
yeast3	0.875±0.036	0.875±0.042	0.888±0.039	0.774±0.061	0.811±0.056	0.875±0.038	0.845±0.053	0.910±0.027
avgRank	4.525	3.925	3.3	6.65	6.1	4.65	5.05	1.8
win-tie-lose	12-7-1	13-7-0	10-8-2	19-1-0	18-2-0	15-4-1	15-3-2	control
correlation	-0.3069	-0.2354	-0.2775	-0.3746	-0.2836	-0.2016	-0.167	-0.097

Each entry is the mean±std of 10 times 5-fold CV. The last column corresponds to our proposed AdaBoost.AD (AdaAD). The best model on each data set is highlighted in bold. The third last row lists the average ranks (avgRank) of each model across data sets. Significant difference against AdaBoost.AD is highlighted in yellow (gray). The second last row lists numbers of win-tie-lose of AdaBoost.AD (AdaAD) versus other methods across all data sets. The last row lists the Spearman's correlation between performance of each method and imbalance ratios across data sets. The bold values denote the best average performance of models on the data set.

TABLE V
PREDICTIVE PERFORMANCE IN TERMS OF AVG-AUC

Data set	AdaM1+ADASYN	AdaM2+ADASYN	AdaNC9+ADASYN	LexiBoostM1	LexiBoostM2	Dual-LexiBoost	JanEnsemble	AdaAD
balance	0.666±0.019	0.671±0.021	0.710±0.026	0.692±0.042	0.690±0.045	0.772±0.055	0.884±0.044	0.738±0.047
contraceptive	0.608±0.021	0.606±0.022	0.639±0.020	0.611±0.026	0.619±0.022	0.614±0.024	0.622±0.023	0.614±0.025
ecoli	0.878±0.033	0.869±0.041	0.868±0.044	0.840±0.056	0.834±0.043	0.846±0.041	0.890±0.030	0.867±0.039
glass	0.845±0.044	0.829±0.044	0.825±0.049	0.778±0.067	0.769±0.047	0.768±0.045	0.748±0.061	0.835±0.041
hayesroth	0.880±0.038	0.792±0.070	0.906±0.047	0.877±0.044	0.880±0.043	0.883±0.036	0.805±0.061	0.868±0.050
newthyroid	0.926±0.049	0.939±0.041	0.933±0.047	0.929±0.048	0.935±0.047	0.926±0.045	0.817±0.058	0.963±0.037
pageblocks	0.832±0.090	0.847±0.097	0.840±0.087	0.805±0.093	0.823±0.109	0.838±0.085	0.830±0.101	0.907±0.076
penbased	0.985±0.005	0.983±0.006	0.969±0.009	0.949±0.021	0.937±0.022	0.923±0.013	0.953±0.012	0.984±0.006
shuttle	0.999±0.002	0.999±0.002	0.998±0.002	0.997±0.004	0.997±0.004	0.997±0.002	0.996±0.005	0.999±0.002
thyroid	0.921±0.068	0.945±0.069	0.926±0.065	0.878±0.103	0.932±0.077	0.930±0.065	0.934±0.062	0.983±0.022
wine	0.951±0.031	0.959±0.034	0.949±0.044	0.946±0.031	0.946±0.032	0.948±0.041	0.957±0.024	0.976±0.021
yeast	0.731±0.021	0.714±0.022	0.729±0.026	0.685±0.030	0.665±0.027	0.693±0.023	0.724±0.025	0.700±0.024
ecoli01	0.957±0.029	0.973±0.024	0.972±0.022	0.972±0.022	0.967±0.027	0.977±0.021	0.974±0.026	0.979±0.020
ecoli1	0.876±0.048	0.871±0.046	0.881±0.050	0.802±0.058	0.820±0.063	0.861±0.054	0.857±0.045	0.882±0.046
ecoli2	0.878±0.075	0.896±0.058	0.878±0.060	0.825±0.077	0.851±0.064	0.847±0.059	0.886±0.078	0.903±0.055
ecoli3	0.796±0.114	0.827±0.078	0.802±0.094	0.748±0.095	0.749±0.112	0.762±0.095	0.801±0.109	0.849±0.084
haberman	0.557±0.068	0.565±0.056	0.616±0.069	0.551±0.066	0.551±0.064	0.575±0.071	0.588±0.067	0.595±0.073
newthyroid1	0.948±0.065	0.954±0.067	0.959±0.067	0.926±0.069	0.934±0.066	0.931±0.075	0.862±0.123	0.956±0.066
newthyroid2	0.961±0.051	0.959±0.045	0.959±0.055	0.939±0.060	0.943±0.058	0.942±0.048	0.859±0.107	0.976±0.047
yeast3	0.880±0.033	0.880±0.038	0.891±0.036	0.789±0.051	0.822±0.048	0.879±0.035	0.854±0.046	0.911±0.026
avgRank	4.225	3.625	3.15	6.65	6.1	5.25	4.8	2.2
win-tie-lose	10-8-2	12-7-1	10-7-3	17-3-0	18-2-0	14-5-1	13-3-4	control
correlation	-0.2505	-0.1399	-0.252	-0.2557	-0.2339	-0.2038	-0.0654	-0.0797

Each entry is the mean±std of 10 times 5-fold CV. The last column corresponds to our proposed AdaBoost.AD (AdaAD). The best model on each data set is highlighted in bold. The third last row lists the average ranks (avgRank) of each model across data sets. Significant difference against AdaBoost.AD is highlighted in yellow (gray). The second last row lists numbers of win-tie-lose of AdaBoost.AD (AdaAD) versus other methods across all data sets. The last row lists the Spearman's correlation between performance of each method and imbalance ratios across data sets. The bold values denote the best average performance of models on the data set.

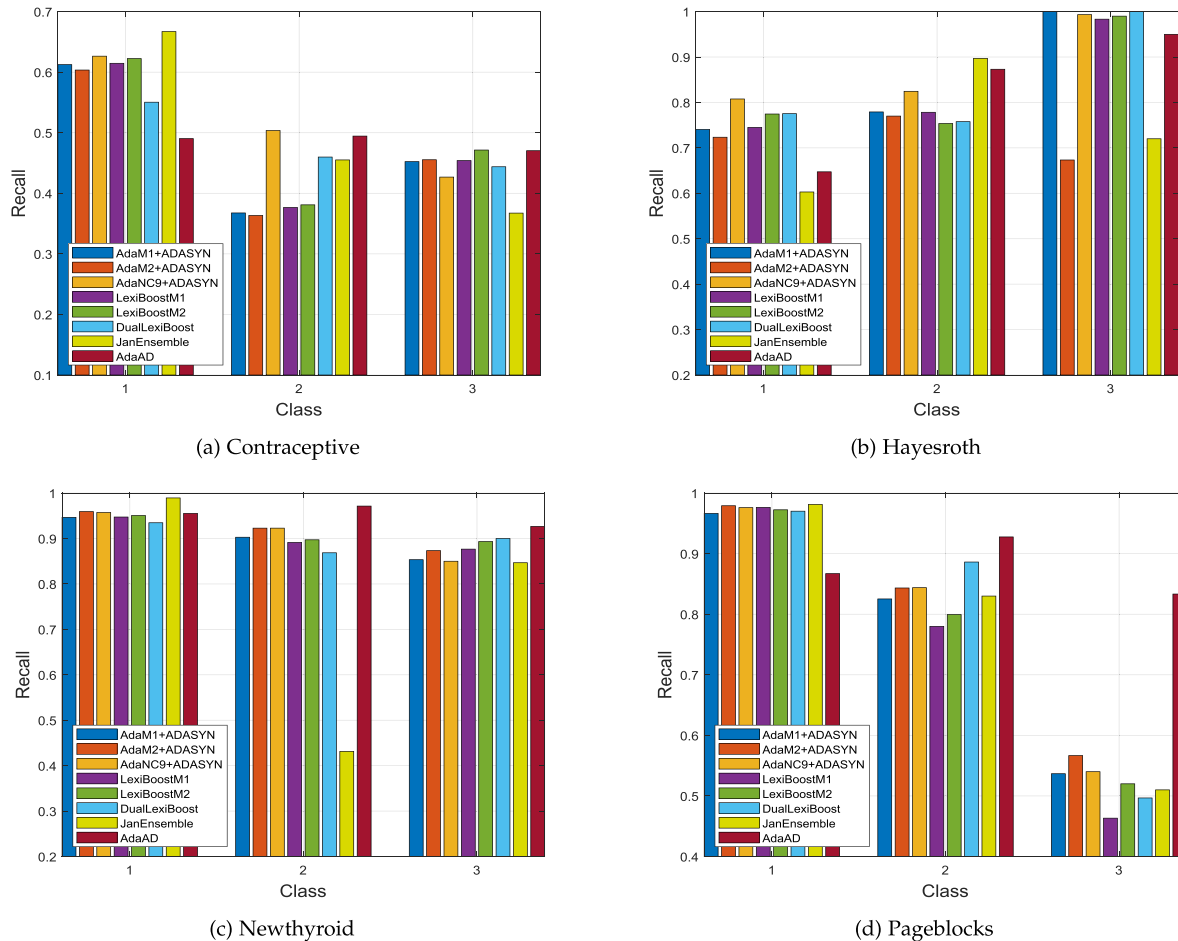


Fig. 2. Predictive performance in terms of recall on four example data sets. The x -axis represents the class index, and the y -axis represents the performance in terms of recall. We can refer to Table III for class sizes of each data set. For clear visualization, the scale of the y -axis is not unified.

15 data sets, and is significantly outperformed by each of them on only 1, 0, 2, 0, 0, 1, and 2 data set(s). Additionally, it achieves parity with each of them on 7, 7, 8, 1, 2, 4, and 3 data set(s).

In terms of Avg-AUC, Table V shows that our AdaBoost.AD outperformed the other methods on 11 out of 20 data sets. Friedman test at the significance level 0.05 rejects H_0 with the p -value $2.580 \cdot 10^{-11}$, meaning that some of these methods perform significantly differently. Next, we conduct post-hoc tests to facilitate further comparisons, to determine whether our method exhibits statistically significant differences from its competitors. AdaBoost.AD is chosen as the control method for consistently performing the best. Post-hoc tests exhibit that our method has significant superiority to all the competing methods except for AdaM2+ADASYN and AdaNC9+ADASYN, where statistical significance was not observed in terms of ranking despite the win-tie-lose results suggesting our method's advantage. Moreover, the average rank of our proposed AdaBoost.AD is 2.2, being the best (lowest) value among the competing methods. This result indicates that our method consistently outperformed the others across data sets.

The second last row of Table V reports the win-tie-lose counts of AdaBoost.AD in comparison to other methods. A closer inspection reveals that AdaBoost.AD significantly outperforms each of the competitors on 10, 12, 10, 17, 18, 14, and 13 data

sets, while being outperformed significantly by each of them on only 2, 1, 3, 0, 0, 1, and 4 data set(s). Additionally, it achieves parity with each of them on 8, 7, 7, 3, 2, 5, and 3 data sets.

Therefore, the proposed AdaBoost.AD consistently demonstrates superior predictive performance in terms of both G-mean and Avg-AUC across the majority of data sets. Nevertheless, there are several data sets where AdaBoost.AD is outperformed by the competitors. For example, AdaBoost.AD is inferior to AdaNC9 in terms of G-mean in two data sets (i.e., Contraceptive and Hayesroth).

Fig. 2(a) shows the recall in terms of each individual class for data set Contraceptive. We can see that the proposed AdaBoost.AD gets the lowest recall on class 1 in comparison to other methods, while retaining relatively better recall on class 2 and class 3. In consequence, the overall performance in terms of G-mean becomes relatively low. We can also see from Table III that class 2 and class 3 of Contraceptive are the minorities, so that they might be overemphasized in the learning process of AdaBoost.AD, sacrificing the predictive performance on class 1.

Fig. 2(b) shows the recall in terms of each individual class for data set Hayesroth. We can see that the proposed AdaBoost.AD gets the relatively lower recall on class 1, while gaining relatively better recall on class 2 and class 3. By referring to Table III, we

know that class 3 is the minority in contrast to class 1 and class 2 of having similar class sizes. For this phenomenon, it is conjectured that the proposed AdaBoost.AD tends to overemphasize the performance on class 3 (the minority) while sacrificing the performance on class 1 and class 2 altogether. It is also conjectured that the training samples of class 2 could be more difficult to gain correct predictions, and AdaBoost.AD would induce smaller adaptive margin values for them. In this sense, AdaBoost.AD will put more attention to training samples of class 2, gaining a relatively high recall on class 2.

In addition, we illustrate two data sets with three class labels for which AdaBoost.AD can significantly outperform the competitors in a large magnitude, namely Newthyroid and Pageblocks in Fig. 2(c) and (d), respectively. We can see that AdaBoost.AD usually outperform the competitors in terms of the recall of most classes. By referring to Table III, we know that class 2 and class 3 are the minorities in contrast to class 1 for Newthyroid and Pageblocks. The proposed AdaBoost.AD performs well to improve predictive performance on the minorities while having no significant impact on the majority class, as it is designed.

C. Effect of Each of the Three Components of Adaptive Distribution Based Sample Weight

In this section, we aim to study the effectiveness of each of the proposed three components of adaptive distribution based sample weight, including the between-class imbalance ratio, the inverse within-class density, and the adaptive margin factor.

To this end, we respectively remove each of the three components from the proposed sample weight of AdaBoost.AD, leading to AdaBoost.AD eliminating the between-class imbalance ratio (“AdaAD-imb”), AdaBoost.AD eliminating the inverse within-class density (“AdaAD-den”), and AdaBoost.AD eliminating the adaptive margin (“AdaAD-mrg”). The three variants would be compared against AdaBoost.AD. If the predictive performance declined significantly after eliminating the component, we can conclude that this eliminated component is crucial in dealing with multi-class imbalance.

In general, Table VI reports predictive performance in terms of G-mean. Friedman test at the significance level 0.05 rejects H0 with p -value $4.526 \cdot 10^{-5}$, and thus H1 is taken. Table VII reports predictive performance in terms of Avg-AUC. Friedman test at the significance level 0.05 rejects H0 with p -value 0.0012, and thus H1 is taken. Therefore, there is significant difference between AdaBoost.AD and the three degraded variants in terms of both G-mean and Avg-AUC. In the rest of this section, we will investigate the effect of each of the three components in more detail.

1) *Effect of the Between-Class Imbalance Ratio:* The effect of between-class imbalance ratio with respect to dealing with class imbalance is analyzed via the performance comparison between AdaBoost.AD vs AdaAD-imb. Specifically, the update of sample weights of AdaAD-imb at the training epoch $t + 1$ is formulated as

$$\omega_1^{(t+1)}(X, y) = \alpha_1^{(t)}(X, y) \cdot \omega_1^{(t)}(X, y) \cdot \exp\{-\beta_t \cdot \mathbb{1}_{[h_t(X)=y]}\},$$

TABLE VI
PERFORMANCE COMPARISON IN TERMS OF G-MEAN BETWEEN ADABOOST.AD AND ITS DEGRADED VARIANTS (ADAAD-IMB, ADAAD-DEN, AND ADAAD-MRG)

Data set	AdaAD-imb	AdaAD-den	AdaAD-mrg	AdaAD
Balance	0.205±0.218	0.636±0.071	0.657±0.122	0.635±0.078
Contraceptive	0.467±0.033	0.463±0.030	0.479±0.029	0.483±0.032
Ecoli	0.757±0.133	0.728±0.129	0.748±0.077	0.768±0.073
Glass	0.493±0.318	0.681±0.119	0.553±0.287	0.668±0.185
Hayesroth	0.800±0.070	0.747±0.074	0.705±0.136	0.802±0.079
Newthyroid	0.913±0.058	0.909±0.071	0.922±0.072	0.948±0.055
Pageblocks	0.570±0.351	0.882±0.086	0.725±0.210	0.843±0.199
Penbased	0.971±0.012	0.973±0.011	0.972±0.013	0.971±0.011
Shuttle	0.997±0.005	0.984±0.021	1.000±0.000	0.998±0.003
Thyroid	0.887±0.153	0.866±0.118	0.989±0.004	0.976±0.032
Wine	0.965±0.030	0.958±0.036	0.959±0.035	0.967±0.028
Yeast	0.009±0.065	0.298±0.181	0.287±0.134	0.305±0.214
Ecoli01	0.979±0.021	0.970±0.026	0.972±0.023	0.978±0.020
Ecoli1	0.846±0.056	0.875±0.047	0.867±0.055	0.880±0.047
Ecoli2	0.881±0.075	0.888±0.063	0.885±0.065	0.899±0.058
Ecoli3	0.675±0.142	0.848±0.076	0.810±0.099	0.844±0.089
Haberman	0.535±0.083	0.581±0.069	0.594±0.072	0.585±0.077
Newthyroid1	0.925±0.082	0.951±0.076	0.931±0.076	0.953±0.075
Newthyroid2	0.935±0.077	0.968±0.038	0.944±0.068	0.974±0.054
Yeast3	0.829±0.048	0.906±0.025	0.905±0.030	0.910±0.027
avgRank	3.35	2.60	2.50	1.55

Each entry is the mean±std of 10 times 5-fold CV. The best performed method on each data set is highlighted in bold. The bottom row lists the average ranks (avgRank) of each method across data sets. Significant difference against AdaBoost.AD is highlighted in yellow (gray). The bold values denote the best average performance of models on the data set.

where $\alpha_1^{(t)}(X, y) = \exp\{-\rho^{-1}(X, y) \cdot \sigma^{(t)}(X, y)\}$ denotes adaptive within-class density that is formulated by setting $\gamma(\cdot) = 1$ in (9). In this way, any positive impact of the imbalance ratio in the update of sample weights of AdaBoost.AD on predictive performance would be inhibited.

Table VI shows that the average rank of AdaAD-imb is 3.35 in terms of G-mean, being larger (worse) than the average rank of the proposed AdaBoost.AD. This indicates that after eliminating the between-class imbalance ratio from the sample weight of (12), the predictive performance declines significantly, demonstrating the effectiveness of the between-class imbalance ratio in dealing with class imbalance. Post-hoc tests of Friedman also show that AdaBoost.AD significantly outperforms AdaAD-imb with the p -value of $5.191 \cdot 10^{-6}$.

Similarly, we can see from Table VII that the average rank of AdaAD-imb is 3.15 in terms of Avg-AUC, being larger (worse) than the average rank of the proposed AdaBoost.AD. This further indicates that eliminating between-class imbalance ratio from the sample weight would cause a significant decline in predictive performance. Post-hoc tests of Friedman show significant inferiority of AdaAD-imb to AdaBoost.AD with p -value $1.193 \cdot 10^{-4}$.

Therefore, the between-class imbalance ratio of (1) plays an important role in dealing with class imbalance and should not be omitted.

2) *Effect of the Inverse Within-Class Density:* The effect of inverse within-class density with respect to dealing with class imbalance is analyzed via the performance comparison between AdaBoost.AD vs AdaAD-den. Specifically, the update of sample weights of AdaAD-den at the training epoch $t + 1$ is

TABLE VII
PERFORMANCE COMPARISON IN TERMS OF AVG-AUC BETWEEN
ADA BOOST.AD AND ITS DEGRADED VARIANTS (ADAAD-IMB, ADAAD-DEN,
AND ADAAD-MRG)

Data set	AdaAD-imb	AdaAD-den	AdaAD-mrg	AdaAD
Balance	0.671±0.023	0.739±0.043	0.756±0.054	0.738±0.047
Contraceptive	0.607±0.022	0.601±0.022	0.615±0.023	0.614±0.025
Ecoli	0.877±0.038	0.850±0.044	0.853±0.043	0.867±0.039
Glass	0.819±0.043	0.826±0.042	0.822±0.043	0.835±0.041
Hayesroth	0.863±0.045	0.822±0.053	0.812±0.080	0.868±0.050
Newthyroid	0.939±0.038	0.937±0.046	0.946±0.047	0.963±0.037
Pageblocks	0.813±0.084	0.918±0.059	0.830±0.084	0.907±0.076
Penbased	0.984±0.006	0.985±0.006	0.985±0.007	0.984±0.006
Shuttle	0.998±0.004	0.988±0.015	1.000±0.000	0.999±0.002
Thyroid	0.932±0.064	0.917±0.067	0.992±0.003	0.983±0.022
Wine	0.974±0.022	0.969±0.026	0.970±0.025	0.976±0.021
Yeast	0.710±0.019	0.680±0.023	0.673±0.026	0.700±0.024
Ecoli01	0.979±0.021	0.970±0.025	0.973±0.023	0.979±0.020
Ecoli1	0.852±0.051	0.877±0.046	0.870±0.052	0.882±0.046
Ecoli2	0.889±0.066	0.893±0.057	0.891±0.059	0.903±0.055
Ecoli3	0.729±0.095	0.852±0.074	0.821±0.085	0.849±0.084
Haberman	0.572±0.058	0.585±0.069	0.614±0.073	0.595±0.073
Newthyroid1	0.930±0.073	0.954±0.066	0.935±0.067	0.956±0.066
Newthyroid2	0.939±0.068	0.969±0.036	0.947±0.060	0.976±0.047
Yeast3	0.842±0.040	0.906±0.025	0.907±0.028	0.911±0.026
avgRank	3.15	2.75	2.45	1.65

Each entry is the mean±std of 10 times 5-fold CV. The best performed method on each data set is highlighted in bold. The bottom row lists the average ranks (avgRank) of each method across data sets. Significant difference against AdaBoost.AD is highlighted in yellow (gray). The bold values denote the best average performance of models on the data set.

formulated as

$$\omega_2^{(t+1)}(X, y) = \alpha_2^{(t)}(X, y) \cdot \omega_2^{(t)}(X, y) \cdot \exp\{-\beta_t \cdot \mathbb{1}_{[h_t(X)=y]}\},$$

where $\alpha_2^{(t)}(X, y) = \exp\{-\gamma(X, y) \cdot \sigma^{(t)}(X, y)\}$ denotes the adaptive between-class imbalance ratio that is formulated by setting $\rho^{-1}(\cdot) = 1$ in (9). In this way, any positive impact of the inverse within-class density in the update of sample weights of AdaBoost.AD on predictive performance would be inhibited.

Table VI shows that the average rank of AdaAD-den is 2.60, being larger (worse) than the average rank of AdaBoost.AD. This indicates that after eliminating the inverse within-class density from the sample weight of (12), the predictive performance declines significantly, demonstrating the effectiveness of the inverse within-class density in dealing with class imbalance. Post-hoc tests of Friedman also show that AdaBoost.AD significantly outperforms AdaAD-den with the p -value of $5.056 \cdot 10^{-3}$.

Similarly, we can see from Table VII that the average rank of AdaAD-den is 2.75, being larger (worse) than the average rank of the proposed AdaBoost.AD. This further indicates that eliminating inverse within-class density from the sample weight would cause a significant decline in predictive performance. Post-hoc tests of Friedman show significant inferiority of AdaAD-den to the proposed AdaBoost.AD with p -value $3.525 \cdot 10^{-3}$.

Therefore, the inverse within-class density factor of (3) plays an important role in dealing with class imbalance and should not be omitted.

3) *Effect of the Adaptive Margin Factor:* The effect of adaptive margin factor with respect to dealing with class imbalance is analyzed via the performance comparison between AdaBoost.AD vs AdaAD-mrg. Specifically, the update of sample weights of AdaAD-mrg at the training epoch $t + 1$ is formulated as

$$\omega_3^{(t+1)}(X, y) = \alpha_3(X, y) \cdot \omega_3^{(t)}(X, y) \cdot \exp\{-\beta_t \cdot \mathbb{1}_{[h_t(X)=y]}\},$$

where $\alpha_3(X, y) = \exp\{-\gamma(X, y) \cdot \rho^{-1}(X, y)\}$ denotes the static data distribution that is formulated by setting $\sigma^{(t)}(\cdot) = 1$ in (9). In this way, any positive impact of the adaptive margin in the update of sample weights of AdaBoost.AD on predictive performance would be inhibited.

Table VI shows that the average rank of AdaAD-mrg is 2.50, being larger (worse) than the average rank of AdaBoost.AD. This indicates that after eliminating the adaptive margin from the sample weight of (12), the predictive performance declines significantly, demonstrating the effectiveness of the adaptive margin in dealing with class imbalance. Post-hoc tests of Friedman also show that AdaBoost.AD significantly outperforms AdaAD-mrg with the p -value of $9.982 \cdot 10^{-3}$.

Similarly, we can see from Table VII that the average rank of AdaAD-mrg is 2.45, being larger (worse) than the average rank of the proposed AdaBoost.AD. This further indicates that eliminating the adaptive margin from the sample weight would cause a significant decline in predictive performance. Post-hoc tests of Friedman show significant inferiority of AdaAD-mrg to AdaBoost.AD with p -value $2.502 \cdot 10^{-2}$.

Therefore, the adaptive margin factor of (8) plays an important role in dealing with class imbalance and should not be omitted.

D. Impact of Imbalance Ratio on Performance

This section aims to investigate the influence of imbalance ratios on the performance of the investigated methods, including our proposed AdaBoost.AD, we employ Spearman's correlation [41] to establish a connection between the predictive performance of those methods and the imbalance ratios across data sets. The results of Spearman's correlation are interpreted as very weak (0.00 - 0.19), weak (0.20 - 0.39), moderate (0.40 - 0.59), strong (0.60 - 0.79), and very strong (0.80 - 1.00), following Evans [42].

The last row of Table IV shows that, in terms of G-mean, the correlations for AdaM1+ADASYN, AdaM2+ADASYN, AdaNC9+ADASYN, LexiBoostM1, LexiBoostM2, Dual-LexiBoost, JanEnsemble, and AdaBoost.AD are -0.3069 (weak), -0.2354 (weak), -0.2775 (weak), -0.3746 (weak), -0.2836 (weak), -0.2016 (weak), -0.167 (very weak), and -0.097 (very weak), respectively. This implies that both JanEnsemble and our proposed AdaBoost.AD display minimal (very weak) sensitivity to the imbalance ratios. This underscores the robustness of our proposed AdaBoost.AD, along with JanEnsemble, compared with other methods across a wide range of class imbalance ratios.

Similarly, the last row of Table V shows that, in terms of Avg-AUC, the correlations for AdaM1+ADASYN, AdaM2+ADASYN, AdaNC9+ADASYN, LexiBoostM1, LexiBoostM2, Dual-LexiBoost, JanEnsemble, and AdaBoost.AD

are -0.2505 (weak), -0.1399 (very weak) -0.252 (weak), -0.2557 (weak), -0.2339 (weak), -0.2038 (weak), -0.0654 (very weak), and -0.0797 (very weak), respectively. This implies that AdaM2+ADASYN, JanEnsemble, and AdaBoost.AD displace minimal (very weak) sensitivity to the imbalance ratios. This underscores the robustness of our AdaBoost.AD, along with AdaM2+ADASYN, JanEnsemble, compared with other methods across a wide range of class imbalance ratios.

Therefore, the proposed AdaBoost.AD consistently demonstrates minimal (very weak) sensitivity to a broad range of imbalance ratios when considering Spearman's correlation. This underscores its robust predictive capability across diverse imbalance ratios.

VII. CONCLUSION

We proposed an adaptive distribution based sample weight that can incorporate the between-class and the within-class data distribution into a single variable whilst tracing the up-to-date training status throughout the entire training process. The between-class imbalance ratio distinguishes training samples of different classes, dealing with class imbalance at a higher level of information describing data classes. The inverse within-class density further distinguishes the training samples of the same class based on the local data distribution information. The adaptive margin factor traces the up-to-date training status, having the potential to upgrade the between-class imbalance ratio and the inverse within-class density evolving with time.

Furthermore, we proposed to encode the adaptive distribution based sample weights into the AdaBoost ensemble approach, contributing to **AdaBoost Ensemble with Adaptive Distribution based Sample Weight (AdaBoost.AD)**. Theoretical support on the validity of the proposed learning algorithm of AdaBoost.AD is provided. Specifically, we validated our iterative update of the adaptive sample weight and demonstrated the correct formulation of the model weight.

Experimental results on 12 multi-class and 8 binary-class imbalanced data sets showed the superiority of the proposed AdaBoost.AD against state-of-the-art multi-class imbalance approaches in dealing with the class imbalance problem and achieving significantly better predictive performance. By eliminating each of the three components from the proposed adaptive distribution based sample weights of AdaBoost.AD, we induced three degraded variants of the proposed AdaBoost.AD, which are AdaAD-imb, AdaAD-den, and AdaAD-mrg, respectively. Experimental analyses showed the effectiveness of each of the components in dealing with class imbalance and showed that they should not be omitted.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Shounak Datta from Duke University, USA, for providing us the codes of LexiBoost and Dual-LexiBoost and helping us to implement them.

REFERENCES

- [1] B. A. M'hamed and B. Fergani, "A new multi-class wsvm classification to imbalanced human activity dataset," *J. Comput.*, vol. 9, no. 7, pp. 1560–1565, 2014.
- [2] S. Pouyanfar, S.-C. Chen, and M.-L. Shyu, "Deep spatio-temporal representation learning for multi-class imbalanced data classification," in *Proc. Int. Conf. Inf. Reuse Integration*, 2018, pp. 386–393.
- [3] L. Liang, T. Jin, and M. Huo, "Feature identification from imbalanced data sets for diagnosis of cardiac arrhythmia," in *Proc. Int. Symp. Comput. Intell. Des.*, 2018, pp. 52–55.
- [4] H. Guo, Y. Li, S. Jennifer, M. Gu, Y. Huang, and B. Gong, "Learning from class-imbalanced data: Review of methods and applications," *Expert Syst. Appl.*, vol. 73, pp. 220–239, 2017.
- [5] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 42, no. 4, pp. 1119–1130, Aug. 2012.
- [6] M. Lin, K. Tang, and X. Yao, "Dynamic sampling approach to training neural networks for multiclass imbalance classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 4, pp. 647–660, Apr. 2013.
- [7] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.
- [8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, 2002.
- [9] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2008, pp. 1322–1328.
- [10] L. Li, H. He, and J. Li, "Entropy-based sampling approaches for multi-class imbalanced problems," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 11, pp. 2159–2170, Nov. 2020.
- [11] S. S. Mullick, S. Datta, and S. Das, "Generative adversarial minority over-sampling," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1695–1704.
- [12] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2001, pp. 973–978.
- [13] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: Improving prediction of the minority class in boosting," in *Proc. Eur. Conf. Princ. Data Mining Knowl. Discov.*, Springer, 2003, pp. 107–119.
- [14] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUS-Boost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. - Part A, Syst. Hum.*, vol. 40, no. 1, pp. 185–197, Jan. 2010.
- [15] S. Datta, S. Nag, and S. Das, "Boosting with lexicographic programming: Addressing class imbalance without cost tuning," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 5, pp. 883–897, May 2020.
- [16] S. Wang, H. Chen, and X. Yao, "Negative correlation learning for classification ensembles," in *Proc. Int. Joint Conf. Neural Netw.*, 2010, pp. 1–8.
- [17] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, 1996, pp. 148–156.
- [18] R. Alejo, J. M. Sotoca, R. M. Valdovinos, and G. A. Casañ, "The multi-class imbalance problem: Cost functions with modular and non-modular neural networks," in *Proc. Int. Symp. Neural Netw.*, Springer, 2009, pp. 421–431.
- [19] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: A new over-sampling method in imbalanced data sets learning," in *Proc. Adv. Intell. Comput.*, Springer, 2005, pp. 878–887.
- [20] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Advances in Knowledge Discovery and Data Mining*. Berlin Germany: Springer, 2009, pp. 475–482.
- [21] S. Barua, M. M. Islam, X. Yao, and K. Murase, "MWMOTE—majority weighted minority oversampling technique for imbalanced data set learning," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 2, pp. 405–425, Feb. 2014.
- [22] L. Abdi and S. Hashemi, "To combat multi-class imbalanced problems by means of over-sampling techniques," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 1, pp. 238–251, Jan. 2016.
- [23] A. Kumari and U. Thakar, "Hellinger distance based oversampling method to solve multi-class imbalance problem," in *Proc. Int. Conf. Commun. Syst. Netw. Technol.*, 2017, pp. 137–141.
- [24] X. Yang, Q. Kuang, W. Zhang, and G. Zhang, "AMDO: An over-sampling technique for multi-class imbalanced problems," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 9, pp. 1672–1685, Sep. 2018.
- [25] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Proc. IEEE Int. Conf. Data Mining*, 2006, pp. 592–602.
- [26] B. Krawczyk, "Cost-sensitive one-versus-one ensemble for multi-class imbalanced data," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 2447–2452.

- [27] Z. Liu, D. Tang, J. Li, and R. Wang, "Objective cost-sensitive-boosting-WELM for handling multi class imbalance problem," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 1975–1982.
- [28] X.-Y. Liu and Q.-Q. Li, "Learning from combination of data chunks for multi-class imbalanced data," in *Proc. Int. Joint Conf. Neural Netw.*, 2014, pp. 1680–1687.
- [29] T. Alam, C. F. Ahmed, S. A. Zahin, M. A. H. Khan, and M. T. Islam, "An effective recursive technique for multi-class classification and regression for imbalanced data," *IEEE Access*, vol. 7, pp. 127 615–127 630, 2019.
- [30] Z. Jan, J. C. Munos, and A. Ali, "A novel method for creating an optimized ensemble classifier by introducing cluster size reduction and diversity," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3072–3081, Jul. 2022.
- [31] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*. Berlin, Germany: Springer, 1995, pp. 23–37.
- [32] S. Wang and X. Yao, "Relationships between diversity of classification ensembles and single-class performance measures," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 206–219, Jan. 2013.
- [33] J. Alcalá-Fdez et al., "Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Mult.-Valued Log. Soft Comput.*, vol. 17, no. 2/3, pp. 255–287, 2011.
- [34] F. Mosteller and J. W. Tukey, "Data analysis, including statistics," in *Handbook of Social Psychology*, vol. 2. Reading, MA, USA: Addison-Wesley, 1968, pp. 80–203.
- [35] C. Seiffert, T. M. Khoshgoftaar, and J. Van Hulse, "Hybrid sampling for imbalanced data," *Integr. Comput.-Aided Eng.*, vol. 16, no. 3, pp. 193–210, 2009.
- [36] P. Sobhani, H. Viktor, and S. Matwin, "Learning from imbalanced data using ensemble methods and cluster-based undersampling," in *Proc. Int. Workshop New Front. Mining Complex Patterns*, Springer, 2014, pp. 69–83.
- [37] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Mach. Learn.*, vol. 45, no. 2, pp. 171–186, 2001.
- [38] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
- [39] S. Holm, "A simple sequentially rejective multiple test procedure," *Scand. J. Statist.*, vol. 6, no. 2, pp. 65–70, 1979.
- [40] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in Statistics*. Berlin, Germany: Springer, 1992, pp. 196–202.
- [41] E. C. Fieller, H. O. Hartley, and E. S. Pearson, "Tests for rank correlation coefficients. I," *Biometrika*, vol. 44, no. 3/4, pp. 470–481, 1957.
- [42] J. D. Evans, *Straightforward Statistics for the Behavioral Sciences*. Pacific Grove, CA, USA: Thomson Brooks/Cole Publishing Co, 1996.



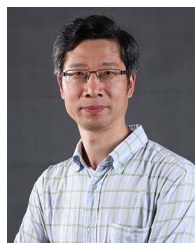
Xiaoyu Wu (Member, IEEE) received the PhD degree in communication engineering from the Beijing University of Posts and Telecommunications in 2018. She is currently an AI engineer with Huawei Technologies Company Ltd. Her research interests include Concept drifts, model adaptive learning, uncertainty Quantification, etc.



Zheng Hu received the PhD degree in computer science from Lyon University in Lyon France. He is the director of Reliability Technology Lab, HUAWEI, a leading global provider of information and communications technology (ICT) infrastructure and smart devices. He is also currently leading a corporate-level project, Trustworthy AI, in charge of the research and innovation of key technologies towards the reliable and safe AI system. Meanwhile his research also focuses on the software reliability, reliability theory and ad-hoc networks, etc. Before joint Huawei, he was the senior researcher in Orange Labs (France Telecom), working on the self-configuration network of smart home/smart building.



Yiu-ming Cheung (Fellow, IEEE) received the PhD degree from the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. He is currently a chair professor with the Department of Computer Science, Hong Kong Baptist University, Hong Kong. His current research interests include machine learning and visual computing, as well as their applications in data science, pattern recognition, multi-objective optimization, and information security. He is the editor-in-chief of *IEEE Transactions on Emerging Topics in Computational Intelligence*. Also, he serves as an associate editor for *IEEE Transactions on Cybernetics*, *Pattern Recognition*, *Knowledge and Information Systems*, and *Neurocomputing*, just to name a few. He is a fellow of AAAS, IET, BCS and AAIA.



Xin Yao (Fellow, IEEE) received the BSc degree from USTC in 1982, the MSc degree from the North China Institute of Computing Technologies in 1985, and the PhD from the University of Science and Technology of China (USTC) in 1990. He is the Tong Tin Sun chair professor of Machine Learning at Lingnan University, Hong Kong, and a part-time professor of Computer Science with the University of Birmingham, U.K. He was a distinguished lecturer of the IEEE Computational Intelligence Society (CIS). He served as the president (2014–15) of IEEE CIS and the editor-in-chief (2003–08) of *IEEE Transactions on Evolutionary Computation*. His major research interests include evolutionary computation, class imbalance learning, and their real-world applications. His work won the 2001 IEEE Donald G. Fink Prize Paper Award; 2010, 2016 and 2017 IEEE Transactions on Evolutionary Computation Outstanding Paper Awards; 2011 IEEE Transactions on Neural Networks Outstanding Paper Award; 2010 BT Gordon Radley Award for Best Author of Innovation (Finalist); and other best paper awards at conferences. He received a 2012 Royal Society Wolfson Research Merit Award, the 2013 IEEE CIS Evolutionary Computation Pioneer Award and the 2020 IEEE Frank Rosenblatt Award.



Shuxian Li received the bachelor's degree in mathematics from the Southern University of Science and Technology, China, in 2017. She is currently working toward the PhD degree in computer science with the Hong Kong Baptist University, Hong Kong SAR, China, under co-supervision with the Southern University of Science and Technology, China. Her research interests include machine learning, class imbalance learning, ensemble learning, and online learning.



Liyan Song received the bachelor's and master's degrees in mathematics from the Harbin Institute of Technology (China) and the PhD degree in computer science from the University of Birmingham (U.K.). He is an associate professor with the Faculty of Computing, Harbin Institute of Technology (China). Prior to that, she was a research assistant professor with the Department of Computer Science and Engineering, Southern University of Science and Technology (China). She was a research fellow with the School of Computer Science, University of Birmingham (U.K.).

Her main research interests are machine learning for predictive modeling in software engineering. She also has research interests in machine learning areas such as online learning, Bayesian learning, class imbalance learning and unsupervised learning.