RAMOBoost: Ranked Minority Oversampling in Boosting

Sheng Chen, *Student Member, IEEE,* Haibo He, *Member, IEEE,* and Edwardo A. Garcia

***Abstract*— In recent years, learning from imbalanced data has attracted growing attention from both academia and industry due to the explosive growth of applications that use and produce imbalanced data. However, because of the complex characteristics of imbalanced data, many real-world solutions struggle to provide robust efficiency in learning-based applications. In an effort to address this problem, this paper presents Ranked Minority Oversampling in Boosting (RAMOBoost), which is a RAMO technique based on the idea of adaptive synthetic data generation in an ensemble learning system. Briefly, RAMOBoost adaptively ranks minority class instances at each learning iteration according to a sampling probability distribution that is based on the underlying data distribution, and can adaptively shift the decision boundary toward difficult-to-learn minority and majority class instances by using a hypothesis assessment procedure. Simulation analysis on 19 real-world datasets assessed over various metrics—including overall accuracy, precision, recall, F-measure, G-mean, and receiver operation characteristic analysis—is used to illustrate the effectiveness of this method.**

***Index Terms*— Adaptive boosting, data mining, ensemble learning, imbalanced data.**

I. Introduction

LEARNING from imbalanced data (imbalanced learning) [1], [2] has become a critical and significant research issue in many of today's data-intensive applications, such as financial engineering, anomaly detection, biomedical data analysis, and many others. The amount and complexity of raw data that is captured to monitor, analyze, and support decision-making processes continue to grow at an incredible rate. Consequently, this enhances the capacity for computationally intelligent methods to play an essential role in applications involving large amounts of data. On the other hand, these opportunities also raise many new challenges for the research community in general [3]–[5].

Generally speaking, any dataset that exhibits an unequal distribution between its classes can be considered imbalanced. In real-world applications, datasets exhibiting severe imbalances are of great interest since they generally present significant difficulties for learning mechanisms. Typical imbalance ratios can range from 1:100 in fraud detection problems [6] to 1:100 000 in high-energy physics event classification [7]. However, imbalances of this form are just one aspect of the imbalanced learning problem. The imbalance learning problem generally manifests itself in two forms: *relative imbalances* and *absolute imbalances* [1], [8]. Absolute imbalances arise in datasets where minority examples are definitively scarce and underrepresented, whereas relative imbalances are indicative of datasets in which minority examples are well represented but remain severely outnumbered by majority class examples. Some studies have shown that the degradation of classification performance attributed to imbalanced data is not necessarily the result of relative imbalances but rather due to the lack of representative examples (absolute imbalances) [9], [10], [1], [11], [12]. In particular, for a given dataset that contains several sub-concepts, the distribution of minority examples over the minority class concepts may yield clusters with insufficient representative examples to form a classification rule [1]. This problem of concept data representation within a class is also known as the *within-class imbalance problem*, [10], [13], [14], and was verified to be more difficult to handle than datasets with only homogeneous concepts for each class [10], [12].

Logically, it would follow that solutions targeted at both relative and absolute imbalances would be more adept to handling a wide spectrum of imbalanced learning problems. To this end, this paper proposes RAMOBoost, which is a RAMO technique embedded with a boosting procedure to facilitate learning from imbalanced datasets. Based on an integration of oversampling and ensemble learning, RAMOBoost systematically generates synthetic instances by considering the class ratios of surrounding nearest neighbors of each minority class example in the underlying training data distribution. Unlike many existing approaches that use uniform sampling distributions, RAMOBoost adaptively adjusts the sampling weights of minority class examples according to their data distributions. Moreover, by integrating the ensemble learning methodology, RAMOBoost adopts an iterative learning procedure that assesses the hypothesis developed at each boosting iteration to adaptively shift the decision boundary to focus more on those difficult-to-learn instances of both the majority and the minority classes.

We organize the remainder of this paper as follows. In Section II, we present a brief review of the state-of-the-art techniques proposed in the community to address the imbalanced learning problem. In Section III, we discuss the motivation behind the RAMOBoost framework and present

Manuscript received February 24, 2009; revised November 2, 2009, March 16, 2010, August 1, 2010, and August 5, 2010; accepted August 5, 2010. Date of publication August 30, 2010; date of current version October 6, 2010.

S. Chen and E. A. Garcia are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030 USA (e-mail: schen5@stevens.edu; egarcia@stevens.edu).

H. He is with the Department of Electrical, Computer and Biomedical Engineering, University of Rhode Island, Kingston, RI 02881 USA (e-mail: he@ele.uri.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2010.2066988

the algorithm in detail. The computational complexity analysis of the proposed RAMOBoost algorithms is also presented in this section. In Section IV, simulation analysis on 19 real-world machine learning datasets are provided to illustrate the effectiveness of the proposed method. Details of experimental parameters and evaluation metrics are also presented in this section. Finally, a conclusion and brief discussion on future research directions are presented in Section V.

II. Related Works

A comprehensive review of the development of research in learning from imbalanced data, including the nature of the problem, the state-of-the-art approaches, the assessment metrics, and the major opportunities and challenges, has been presented in [1]. Interested readers can refer to that article for details. In this section, we provide a focused review of four major categories of research activity in imbalanced learning.

A. Sampling Methods

Building on the foundation of the random (simple) oversampling and undersampling techniques, researchers have developed advanced sampling methods to address the shortcomings of these basic techniques, such as overfitting and information loss. For example, the synthetic minority oversampling technique (SMOTE) algorithm [15] was proposed to search for the nearest neighbors of every minority instance and generates synthetic minority data by calculating linear interpolations between an original minority class instance and a randomly selected neighbor. Expanding on the SMOTE framework, the Borderline-SMOTE algorithm [16] locates those minority class examples that reside along the borders between majority and minority classes. Other popular approaches include the DataBoost-IM (imbalanced learning) [17], AdaBoost in conjunction with Over/Under-Sampling and Jittering of the data (JOUS-Boost approach) [18], and the integration of SMOTE with Tomek links and edited nearest neighbor [9]. Since many of the aforementioned sampling algorithms solely focus on relative imbalances, various approaches were proposed to explicitly address the within-class imbalance issue. Some of the important works include the cluster-based oversampling algorithm [11], support cluster machines [19], and others.

B. Cost-Sensitive Learning Methods

Cost-sensitive learning methods typically employ the use of cost matrices to estimate the costs of different classification errors. These techniques have shown great success when applied to imbalanced learning problems [1]. For example, in [20] an instance-weighting method was presented to induce cost-sensitive trees. In [21] and [22], the asymmetric AdaBoost method is proposed to handle face detection problems where the skewed class ratio can be quite high. The AdaCost method proposed in [23] combines cost-sensitive learning with boosting. By referring to the cost-sensitive matrix, AdaCost assigns different cost values to misclassified minority and majority examples by the trained hypothesis at each iteration loop. Other examples of cost-sensitive learning include the MetaCost framework [24], cost-sensitive neural network [25], cost-sensitive support vector machines (SVMs) [26], and others.

C. Kernel-Based Learning Methods

Kernel-based methods have recently become very popular across various fields including imbalanced learning [1]. In general, kernel-based methods facilitate learning by maximizing the separation margin between concepts in linearly separable feature spaces. For instance, the Kernel-based alignment algorithm was proposed in [27] and [28], in which imbalanced data is used as information prior to adjusting the kernel matrix in order to facilitate SVM learning for improved prediction accuracy. Another example of kernel-based learning presents a kernel classifier construction algorithm using orthogonal forward selection to optimize the generalization model for two-class imbalanced learning problems [29]. This is accomplished by using the regularized orthogonal weighted least squares method and a model selection criterion of maximal leave-one-out area under curve (AUC) of the receiver operating characteristic (ROC) graph.

D. Active Learning Methods

Active learning methods were originally developed for learning from datasets with unlabeled instances. Recently, active learning methods have found increased use in imbalanced learning applications [1]. For example, an SVM-based active learning approach for imbalanced datasets was proposed in [30] and [31]. This algorithm locates the "most informative" sample by evaluating a small fixed number of randomly selected examples instead of the entire dataset [31]. In [32], the stopping condition for active learning applications in word sense disambiguation (WSD) domains was investigated. To alleviate the complications introduced by within-class imbalances, the authors proposed a bootstrap-based oversampling technique to improve active learning performance for imbalanced WSD applications.

III. RAMOBoost Framework: Integration of Data Generation and Boosting Ensemble Learning

A. Preliminaries for RAMOBoost

Various techniques exist for quantizing the learning difficulty level of an example according to the distance between minority and majority cases. An intuitive method can be formulated as follows. First, locate the center (mean) of the majority examples in the feature space. Then, calculate the Euclidean distance between each minority example and the center, the set of minimal distanced minority examples up to some threshold, then, would represent the most informative examples. Fig. 1(a) illustrates this idea, in this figure, the stars and circles represent the minority and majority classes, respectively, and the triangle within the majority data represents the approximate center. The Euclidean distances between minority examples, S_1, S_2, S_3, S_4, and S_5 and the majority class center would be calculated, respectively. As efficient as it appears, this method exhibits a critical flaw, it assumes that there are no disjuncts (sub-concepts) within the majority class concept. Otherwise, there may exist several sub-concepts for the majority class and the task of computing the Euclidean distance against the centers becomes complicated and requires careful examination. We highlight this issue in Fig. 1(b). Here,

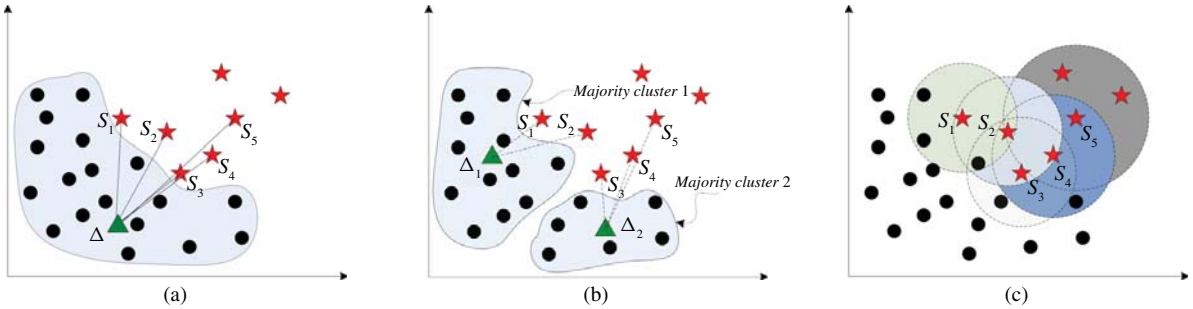

Fig. 1. Cluster and distance calculation based on Euclidean distance. (a) Intuitive approach to decide the informative data examples based on Euclidean distance. (b) Potential dilemma by applying this intuitive approach. (c) Proposed approach by using k nearest neighbors of each minority example and using this value to decide the weight to find the informative data examples.

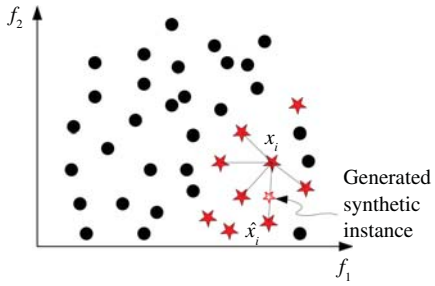

Fig. 2. Synthetic data generation based on the SMOTE algorithm.

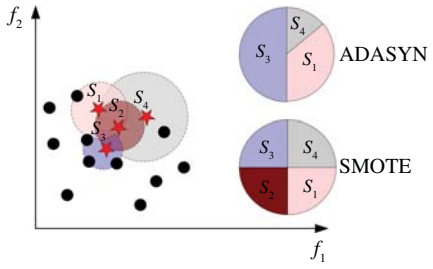

Fig. 3. Pie-chart representation of the synthetic instance proportions for SMOTE and ADASYN.

the majority class contains two clusters and each minority example has to locate its respective closest cluster center before calculating the distance to the majority center.

RAMOBoost targets this flaw in the following way. Instead of searching for the center(s) of the majority dataset, RAMOBoost determines the number of majority examples that are within the k nearest neighbors of each minority example and assigns this value as the information weight. We illustrate this idea in Fig. 1(c). Here, the highlighted areas surrounded by dashed circles represents the k nearest neighbor search area for each minority example, S_1, S_2, S_3, S_4, and S_5 ($k = 4$ in this case). Accordingly, there are 3, 1, 2, 1, and 0 majority examples that are within the four nearest neighbors of S_1, S_2, S_3, S_4, and S_5, respectively. This means the minority example S_1 is more likely to be located near the decision boundary (i.e., is a difficult-to-learn example), and, therefore, more synthetic samples should be generated for S_1 to force the final learning hypothesis to be more focused on the difficult learning regions. Our previous work adaptive synthetic (ADASYN) [33] also applies a similar idea to assess the difficulty level of minority data. The difference is that, instead of directly specifying the number of synthetic instances generated for each minority example as in ADASYN, RAMOBoost adopts this mechanism to determine the chance of each minority example for generating the synthetic instances.

Before presenting the details of the RAMOBoost algorithm, we briefly discuss the data generation mechanisms of SMOTE [15] and ADASYN [33], which motivated the work presented in this paper and can provide a better understanding of RAMOBoost.

In the SMOTE method, the k nearest neighbors of the minority data (k-NN, where k is a user-specified parameter) are identified for each minority example. The k-NN are defined as the set of k instances such that the Euclidean distance between the minority example under consideration and the examples of the minority set are of minimal length along the n-dimensional feature space. Synthetic samples are then created by randomly selecting one of the k nearest neighbors and multiplying the corresponding feature vector difference with a random number between [0, 1]

$$\boldsymbol{x_{new}} = \boldsymbol{x_i} + (\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x_i}) \times \Delta \tag{1}$$

where $\boldsymbol{x_i}$ is the minority example under consideration, $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$ is one of the k-nearest neighbors (minority class) for $\boldsymbol{x_i}$, and $\Delta \in [0, 1]$ is a random number. Therefore, the resulting synthetic instance, according to (1), is a point along the line segment joining the example under consideration $\boldsymbol{x_i}$ and the randomly selected k nearest neighbor $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$. Fig. 2 illustrates this procedure. In this way, SMOTE generates the same number of synthetic instances for each minority example (uniform distribution).

ADASYN also uses feature interpolation to generate synthetic instances. The difference from SMOTE is, instead of applying a uniform distribution for data generation like SMOTE, ADASYN uses a density distribution $\{\hat{r}_i\}$ as a criterion to automatically decide the number of synthetic samples that need to be generated for each minority example [33]. The density distribution criteria is defined as the normalized number of majority cases within the k-nearest neighbor of each minority example. Fig. 3 shows the proportion of synthetic instances generated by SMOTE and ADASYN based on three-nearest neighbors for minority examples S_1 through S_4. From Fig. 3, it is clear that: 1) SMOTE uniformly assigns the number of synthetic instances to be generated for S_1 to S_4, and 2) ADASYN uses a different distribution to determine

the number of synthetic instances for S_1 to S_4. ADASYN first calculates the number of majority cases within the three-nearest neighbors of S_1 to S_4 first, which is $\{2, 0, 3, 1\}$ in this example, and normalizes this into the density distribution $\{2/3, 0, 3/3, 1/3\} \Rightarrow \{1/3, 0, 1/2, 1/6\}$, which is used to bias the data generation process [33, Algorithm: ADASYN]. One issue worth noting is that ADASYN does not generate instances for minority examples with no majority cases in their k-nearest neighbors, like S_2 in Fig. 3.

B. RAMOBoost Learning Algorithm

The objective of RAMOBoost is twofold, to reduce the induction biases introduced from imbalanced data and to adaptively learn information from the data distribution. This is achieved in two respects: First, an adaptive weight adjustment procedure is embedded in RAMOBoost that shifts the decision boundary toward the difficult-to-learn examples from both the minority and majority classes. Second, a ranked sampling probability distribution is used to generate synthetic minority instances to balance the skewed distribution. Motivated by the SMOTE [15], SMOTEBoost [34], and ADASYN [33] algorithms, RAMOBoost facilitates imbalanced learning by iteratively developing an ensemble of hypotheses. However, unlike SMOTE, which samples minority examples indiscriminately and uniformly, RAMOBoost evaluates the potential learning contribution of each minority example and determines their sampling weights accordingly. This is achieved by calculating the distance of any single minority example from the set of nearest neighbors to determine how greatly it will benefit the learning process.

Before officially describing the RAMOBoost algorithm, we would like to explain several terms for improved readability. In lieu of directly operating on the original training dataset D, RAMOBoost follows the classic AdaBoost.M2 procedure to apply the boosting process on the misclassified dataset B. For the *n-class* classification problem, B is defined to be the set where each example in D is replicated $n-1$ times with a different class label other than the true one. $\{\hat{r_i}\}$ serves as the distribution function determining the probability that examples in B are chosen for generating the synthetic instances, it is calculated by mapping the number of majority cases in the k_1-nearest neighbors of each example in B into the range $(0,\ 1)$.

The RAMOBoost learning algorithm is presented presented in [Algorithm 1].

According to this description, the RAMOBoost algorithm includes two mechanisms to facilitate learning from imbalanced data. The first consists of Steps 2 to 5, where instances are adaptively generated according to their distributions. In this way, more synthetic instances are created for difficult-to-learn minority examples that are more likely to be misclassified compared to easy-to-learn minority examples. This is significantly different from the SMOTE algorithm where each minority example has equal weight and therefore the same numbers of synthetic instances are created for each minority example. The second mechanism, Steps 6 to 10 use the pseudo-loss of the current hypothesis h_t to update the sampling distribution D_t, which is used to sample the training dataset in the next iteration as shown in Step 1. Similar to the

Algorithm 1 $RAMOBoost(N, T, k_1, k_2, \alpha)$

Input:

1) Training dataset with m class examples $\langle(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m)\rangle$, where $\boldsymbol{x_i}$ $(i = 1, \ldots, m)$ is an instance of the n dimensional feature space $\boldsymbol{X}$ and $y_i \in Y = \{major, minor\}$ is the class identity label associated with instance $\boldsymbol{x_i}$.
2) N: number of synthetic data samples to be generated at each iteration
3) T: number of iterations; i.e., the number of base classifiers
4) k_1: number of nearest neighbors in adjusting the sampling probability of the minority examples
5) k_2: number of nearest neighbors used to generate the synthetic data instances
6) α: the scaling coefficient

Let $B = \{(i, y) : i \in \{1, \ldots, m\}, y \neq y_i\}$

Initialize: $D_1(i, y) = 1/|B|$ for $(i, y) \in B$ (for two class problems, $|B| = m$)

Do for $t = 1, 2, \ldots, T$.

1) Sample the mislabeled training data with D_t and get back the sampled dataset S_e of identical size. Slice S_e into the majority subset e_1 and the minority subset e_2 of size m_{lt} and m_{st}, respectively.
2) For each example $x_i \in e_2$, find its k_1 nearest neighbors in the dataset S_e according to the Euclidean distance in n-dimensional space and calculate r_i defined as
$$r_i = \frac{1}{1 + \exp(-\alpha \cdot \delta_i)}, \qquad i = 1, 2, \ldots, m_{st} \tag{2}$$
where δ_i is the number of majority cases in k_1 examples.
3) Normalize r_i according to
$$\hat{r_i} = \frac{r_i}{\sum_{i=1}^{m_{st}} r_i} \tag{3}$$
such that $\{\hat{r_i}\}$ is a distribution function: $\sum_{i=1}^{m_{st}} r_i = 1$. Define $\boldsymbol{d_t} = \{\hat{r_i}\}$.
4) Sample e_2 with $\boldsymbol{d_t}$ and get back a sampling minority dataset g_t, of size m_{st}.
5) For each example $\boldsymbol{x_i} \in g_t$, find its k_2 nearest neighbors in e_2 according to the Euclidean distance in n dimensional space and use linear interpolation to generate N synthetic data samples.
6) Provide the base classifier with sampling dataset S_e and the N synthetic data samples.
7) Get back a hypothesis h_t: $\boldsymbol{X} \times Y \rightarrow [0,\ 1]$.
8) Calculate the pseudo-loss of h_t
$$\varepsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i, y)\,(1 - h_t(\boldsymbol{x_i}, y_i) + h_t(\boldsymbol{x_i}, y)) \tag{4}$$
9) Set $\beta_t = \varepsilon_t / 1 - \varepsilon_t$
10) Update D_t
$$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \beta_t^{(1 + h_t(\boldsymbol{x_i}, y_i) - h_t(\boldsymbol{x_i}, y))} \tag{5}$$
where Z_t is a normalization constant.

End loop

output: The output hypothesis $h_{final}(\boldsymbol{x})$ is calculated as follows:

$$h_{final}(\boldsymbol{x}) = \arg\max_{y \in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t} \right) h_t(\boldsymbol{x}, y) \tag{6}$$

AdaBoost.M2 algorithm [35], [36], the pseudo-loss mechanism can adaptively shift the final hypothesis toward the decision boundary to facilitate the learning process.

The key components of the two RAMOBoost mechanisms are Steps 1 and 4 in the pseudo-code. By sampling the training dataset with the updated weight distribution function D_t at the tth iteration as shown in Step 1, RAMOBoost can progressively shift the decision boundary toward the difficult-to-learn examples. Meanwhile, in Step 4 RAMOBoost generates more synthetic instances for those difficult-to-learn minority examples by sampling the minority dataset e_2 with the distribution function obtained from manipulating the number of majority examples in the k nearest neighbors of each minority example through (2) and (3).

C. Analysis of the RAMOBoost Learning Methodology

1) Data Generation Mechanism: The proposed RAMOBoost algorithm shares some data generation aspects with SMOTE. A synthetic instance is generated by adding the feature vector of the minority example under consideration and a randomized real number in $[0, 1]$ multiplied by the difference between the feature vector of the minority example under consideration and one randomly chosen example within its k nearest neighbors from the minority sampling dataset (Step 5 in the pseudo-code), which can be formulated in the same way as (1). The difference between the two, however, is that RAMOBoost employs a systematic method to adaptively determine the number of synthetic instances created for each minority example in the sampling dataset according to their learnability, more synthetic instances will be generated for those difficult-to-learn examples. Therefore, the final hypothesis will be more focused on those difficult decision regions. Concretely, RAMOBoost generates synthetic instances by manipulating the number of majority cases in the k-nearest neighbors across the whole sampling dataset (Steps 2 and 3 in the pseudo-code), while SMOTE universally generates identical number of synthetic instances for each minority example.

Similar to RAMOBoost, ADASYN also aims to systematically generate synthetic minority instances according to the underlying data distribution instead of using a uniform sampling distribution. Consequently, ADASYN also has the ability to push the learning algorithm to be more focused on the difficult regions of the decision boundary [33]. However, ADASYN does this in an aggressive manner: almost all of the generated synthetic minority instances are very close to the decision boundary. This is because the data generation mechanism of ADASYN approves generation of synthetic data only when there exists at least one majority case in the k-nearest neighbor of the minority example under consideration. Specifically, the number of majority cases in the k-nearest neighbor of the minority example under consideration directly dictates the number of synthetic instances generated around it. In the extreme case, noisy examples in the minority class can have multiple synthetic instances generated, while examples that are relatively far away from the class boundary but are representative of the target concept of the minority class are not selected for synthetic data generation. In contrast, RAMOBoost employs a parameter-specified logistic function to firstly map δ, the number of majority cases within the k nearest neighbors of the minority example under consideration, to a real number r between 0 and 1 (2), which is then normalized to a distribution function (3) for determining the probability of each minority example. In this way, RAMOBoost considers all minority examples for synthetic generation, albeit at varied levels.

ADASYN additionally introduces complications at the decision boundary since almost all of the synthetic instances are located in the decision boundary region, which means that excessively more synthetic instances are probably generated for noisy examples with the minority class label. Overemphasizing the decision boundary region may magnify the influence of noise within the training dataset, thereby leading to performance depreciation. RAMOBoost, on the other hand, assigns high probabilities for minority examples close to the decision boundary, which means that synthetic instances are generated near the decision boundary on a relative basis as opposed to an absolute basis. As a result, the negative impact of noise is attenuated in RAMOBoost as compared to ADASYN.

In order to compare the data generation mechanism of RAMOBoost with that of SMOTE and ADASYN, we provide an example of a dataset with 2000 majority examples and 100 minority examples. Fig. 4(a) shows the original imbalanced data distribution, and Fig. 4(b)–(d) shows the post-SMOTE data distribution, post-ADASYN data distribution, and the post-RAMOBoost data distribution, respectively. In all of these figures, the x-marks, plus, and point shapes represent the original majority data, original minority data, and the generated synthetic data, respectively. Furthermore, for each figure we also illustrate the classification confusion matrix (in terms of instant counts) for performance assessment. Here we follow the suggestions of [15], [5], and [37] and use the minority class as the positive class and majority class as the negative class. The classifier used to make predictions on all datasets shown in Fig. 4 is classification and regression tree. Comparing the confusion matrix of each figure, we see that the proposed RAMOBoost method can improve classification performance. Specifically, the improvement of true negative (TN) counts for SMOTE with respect to the original dataset changes from 1992 to 1993, while for RAMOBoost it increases from 1992 to 1998. This is because in SMOTE the same numbers of instances are generated for each minority example, while in RAMOBoost the data generation process is adaptive according to the data distribution.

From Fig. 4(c) we also see that ADASYN is very aggressive in learning from the boundary since it generates synthetic data instances very close to the decision boundary. This may have two effects on the learning performance. It may increase

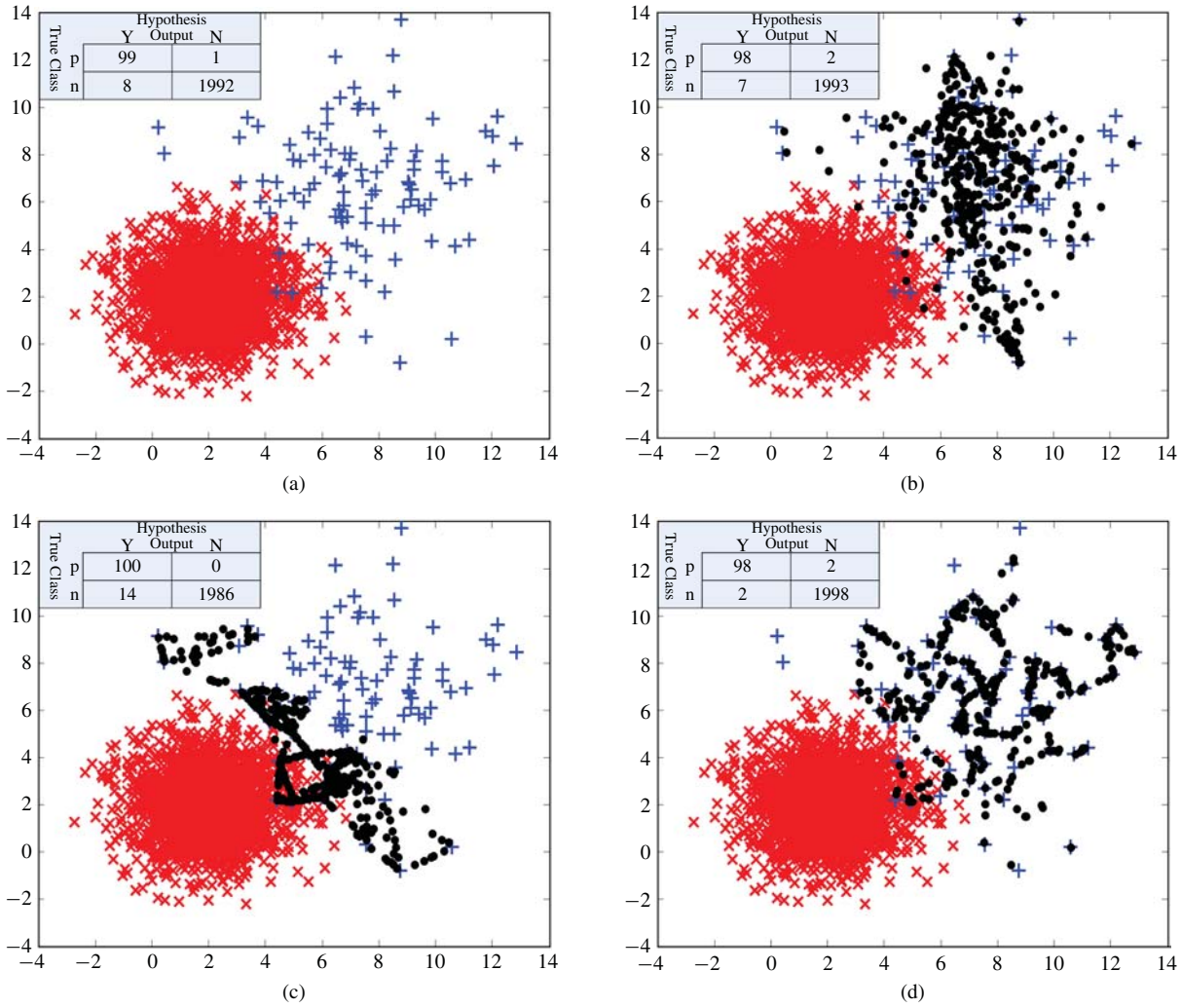

Fig. 4. Comparison of different synthetic data generation mechanisms. (a) Original imbalanced data distribution (2000 majority examples and 100 minority examples). (b) Data distribution after SMOTE method. (c) Data distribution after ADASYN method. (d) Data distribution after RAMOBoost method.

the classification accuracy of the minority data, as it provides a good representation of the minority data distribution close to the boundary (thereby improving the *Recall* performance, which will be discussed in detail in Section IV-C.1). However, it may also decrease the classification performance of the majority class, which in turn deteriorates the overall classification performance. One can observe from Fig. 4(c) that, although the classification accuracy for minority examples under the ADASYN technique is the best among all these methods (true positive (TP) $=$ 100, therefore *Recall* $=$ 1), the TN counts of ADASYN also decreases significantly (the lowest of all in this case with TN $=$ 1986). To this end, RAMOBoost can be considered to take advantage of both SMOTE and ADASYN to improve the overall learning performance. Our simulation analyses, which are based on various real-world datasets and various assessment metrics in Section IV-B, also confirm this.

2) The Boosting Procedure: Boosting has attracted significantly increased attention recently in the computational intelligence community [38], [39], [40], [41], [42]. In our proposed RAMOBoost approach, the boosting algorithm is essentially the same as the classic AdaBoost.M2. Rather than reducing the prediction error on the training dataset in each iteration loop in a stepwise manner, the boosting algorithm of AdaBoost.M2 can focus the weak learner on the labels that are hardest to discriminate by manipulating the pseudo-loss as defined in (4) [36]. In other words, the emphasis of AdaBoost.M2 is to make the incorrect class label as distinguishable as possible from the correct class label belonging to the example under consideration, this is why the mislabeled dataset is sampled in each iteration loop instead of the correct one.

Compared to AdaBoost.M1, AdaBoost.M2 enables the weak learner to make useful contributions to the accuracy of the final hypothesis even when the weak hypothesis does not predict the correct label with a probability greater than $1/2$ [36]. In this way, the iteration loop is not broken regardless of the performance of the trained hypothesis, which is not the case for other boosting algorithms including AdaBoost.M1. Given that it is generally very hard for a single weak learner to extract sufficient knowledge from the imbalanced dataset at one instance, the performance of RAMOBoost, and most ensemble-based algorithms, may suffer from an insufficient number of boosting iterations. Thus for an imbalanced dataset

of any size, whose target concept we assume is difficult to learn, it is our belief that the performance of RAMOBoost can be guaranteed satisfactory if it can iterate for enough epochs. This is our motivation for employing the AdaBoost.M2 algorithm in RAMOBoost.

D. Computational Complexity Analysis

In the training stage, the computational complexity of RAMOBoost arises from the construction of the hypothesis at each iteration loop as well as the boosting procedure. We assume the following in our analysis:

1) m–the dimension of the feature space;
2) n–the number of training examples;
3) p–the ratio of minority examples in the training dataset: $p \in \{0,\ 1\}$;
4) T–the number of boosting epochs.

The procedure of generating synthetic instances for the training dataset is initialized with the calculation of the probability of each minority example for generating synthetic instances. The time complexity can be decomposed into three steps: 1) calculating the Euclidean distance from the minority example under consideration to all the other examples in the training dataset–complexity $O(mn)$; 2) sorting all current Euclidean distance calculations in ascending order–complexity $O(n\log(n))$; 3) retrieving the first k_1 examples corresponding to the first k_1 items in the sorted Euclidean distance set–complexity $O(k_1)$. Thus, the time complexity for this step should be $O(mn + n\log(n) + k_1)$. In typical situations, k_1 and m are both significantly smaller than n, which simplifies the time complexity to approximately $O(n\log(n))$. The next step is to find the k_2 minority neighbors of each minority example for synthetic data generation. The time complexity of this calculation is the same with the first step, except that the Euclidean distance calculation is between the minority examples. Therefore, the time complexity is no greater than $O(n\log(n))$. Since there are altogether np minority examples in the training dataset, the total time complexity should be $O(pn^2\log(n))$, which can be simplified to $O(n^2\log(n))$. Lastly, since data generation is applied in each boosting iteration, the time complexity of synthetic data generation for the learning process of RAMOBoost is $O(n^2T\log(n))$.

For a neural network with multilayer perceptron (MLP), the time complexity for the boosting process excluding synthetic data generation is at worse $O(n^2T^2)$ [43]. Therefore, we summarize that the training process time complexity for RAMOBoost with MLP as a base classifier is $O(n^2T\log(n) + n^2T^2)$.

In the testing stage, the computational operation in each hypothesis is just a comparison operation, the time consumption for each of them is very small. Since the final hypothesis is a weighted combination of all trained hypothesis as shown in (6), the computational complexity of predicting the class label of an instance can be estimated as $O(T)$.

IV. Simulation and Discussion

In this section, we conduct various simulations of the proposed RAMOBoost method and compare its performance with SMOTEBoost, SMOTE, ADASYN, AdaCost, BorderlineSMOTE, and SMOTE-Tomek across different real-world datasets. The neural network with MLP is employed as the base learner. The MLP is configured as follows: The number of hidden layer neurons is set to be four, and the number of input neurons is equal to the number of features for each dataset. Similar to most of the existing imbalanced learning methods in literature, we also consider only two-class imbalanced problems in our current study. Therefore, the number of output neurons is set to two for all simulations. The sigmoid function is used as the activation function, and the inner training epochs is set to be 100 with a learning rate of 0.1.

Due to the concern that the scattered feature distribution of some datasets may hinder the neural network from converging fast enough for the parameter acceleration process before all datasets are presented to the comparative algorithms for learning, we first use the nonlinear normalization approach [44] to normalize the features of the datasets to reside in the interval [0, 1].

TABLE I
Summary of the Dataset Characteristics (Sorted in the Degree of Class Skew)

Dataset	# Feature	# Data	# Minority instances	# Majority instances	Imbalanced ratio
Sonar	60	208	97	111	0.47 : 0.53
Spambase	57	4601	1813	2788	0.39 : 0.61
Ionosphere	34	351	126	225	0.36 : 0.64
PID	8	768	268	500	0.35 : 0.65
Wine	13	178	59	119	0.33 : 0.67
German	24	1000	300	700	0.30 : 0.70
Phoneme	5	5404	1586	3818	0.29 : 0.71
Vehicle	18	846	199	647	0.24 : 0.76
Texture	40	5500	1000	4500	0.18 : 0.82
Segment	18	2310	330	1980	0.14 : 0.86
Page_Blocks	10	5473	560	4913	0.10 : 0.90
Satimage	36	6435	626	5809	0.10 : 0.90
Mf_Zernike	47	2000	200	1800	0.10 : 0.90
Vowel	10	990	90	900	0.09 : 0.91
Abalone	7	731	42	689	0.06 : 0.94
Glass	9	214	9	205	0.04 : 0.96
Yeast	8	483	20	463	0.04 : 0.96
Letter	16	20000	789	19211	0.04 : 0.96
Shuttle	9	43500	37	43463	0.001 : 0.999

A. Dataset Description

The performance of RAMOBoost is evaluated on 19 datasets from the UCI machine learning repository [45] and ELENA project [46]. These datasets vary in size and class distributions to ensure a thorough assessment of performance. Table I summarizes the characteristics of the datasets used in our simulation.

Since several of the original datasets are multiclass data, we modified those datasets following suggestions in literature to make them into two-class datasets. Table II shows the modifications that we used in this paper to create the minority and majority classes.

B. Assessment Metrics

Under the imbalanced learning scenario, the conventional assessment method of using a single criterion, such as overall

TABLE II
DESCRIPTION OF IMBALANCED DATASETS

Dataset	Minority class	Majority class
Sonar	Class 'R' (rock instances)	Class 'M' (metal cylinder instances)
Spambase	Spam email	Legitimate email
Ionosphere	'Bad radar' class	'Good radar' class
Pima-Indians-Diabetes (PID)	Positive class	Negative class
Wine	class '1'	Classes '2' and '3'
German	Customers with bad credit	Customers with good credit
Phoneme	Class of 'oral sounds (class 1)'	Class of 'nasal sounds (class 0)'
Vehicle	Class of 'Van'	Classes of 'OPEL', 'SAAS', and 'BUS'
Texture	Classes of '13' and '14'	Classes of '2', '3', '4', '6', '7', '8', '9', '10' and '12'
Segment	Class of 'brickface'	Classes of 'sky', 'foliage', 'cement', 'Window', 'path' and 'grass'
Page_blocks	Classes of 'horizontal line', 'graphic', 'Vertical line', and 'picture'	Class of 'text'
Satimage	Class of 'damp grey soil'	Classes of 'red soil', 'cotton crop', 'grey soil', 'Soil with vegetation stubble' and 'very damp grey soil'
Mf_Zernike	Class of digit '9'	Classes of digits '0', '1', '2', '3', '4', '5', '6', '7', and '8'
Vowel	Class 1	Classes of 2 to 11
Abalone	Class of '18'	Class of '9'
Glass	Class 6 ('tableware')	All other classes
Yeast	Class of 'POX'	Class of 'CYT'
Letter	Class of letter 'Z'	Classes of letters 'A'–'Y'
Shuttle	Class of 'Fpv Close'	Classes of 'Rad Flow', 'Fpv Open', 'High', 'Bypass', 'Bpv Close' and 'Bpv Open'

accuracy (OA), may not be able to provide a comprehensive assessment of the learning algorithm [34], [47], [17], [1], [48], [37], [49], [50]. Considering a simple case of a given dataset with 2% minority class examples and 98% majority class examples, a naïve approach of classifying every example to be the majority class can at best provide an OA of 98% over the entire dataset. However, in many real-world applications such as biomedical data analysis, such a classification performance would be unacceptable as it misclassifies all the minority cases, which generally are more important in such situations. As a result, the OA by itself may not be sufficient in evaluating the classification performance for imbalanced learning problems. In our simulations, we adopt five major assessment metrics related to the confusion matrix for analysis: OA, precision, recall, F-measure, and G-mean. The detailed discussions on these metrics and their applications for imbalanced learning can be found in [1].

In addition to these singular assessment metrics, we also adopted the ROC graph [47], [50] for evaluation in this paper. Briefly speaking, the ROC space is established by plotting the TPs rate tp_rate over false positives rate fp_rate. The ROC curves are normally formulated by adjusting the decision threshold to generate a series of points in the ROC space. In order to assess different classifiers' performance in this case, one generally uses the AUC as an evaluation criterion. A detailed discussion of ROC analysis and its assessment for classifier performances can be found in [47] and [50]. We would also like to note that there are other metrics that can potentially be used to assess the imbalanced learning performance. For instance, it was recently presented in [51] that *H-measure* could be a qualified alternative metric of AUC. The major motivation of *H-measure* is based on the fact that AUC is equivalent to averaging the misclassification loss over a cost ratio distribution dependent on the score distributions which are decided by the classifier itself rather than the target dataset [51]. This may introduce undesired subjectivity into performance evaluation. *H-measure* targets this flaw by decoupling the weight function for loss calculation from score distributions. For instance, it can apply a beta distribution to simulate the weight function, which ensures objectivity across all algorithms under comparison. The interested reader can refer to [51] for further details on *H-measure* and [1] for a critical review of the assessment metrics for imbalanced learning.

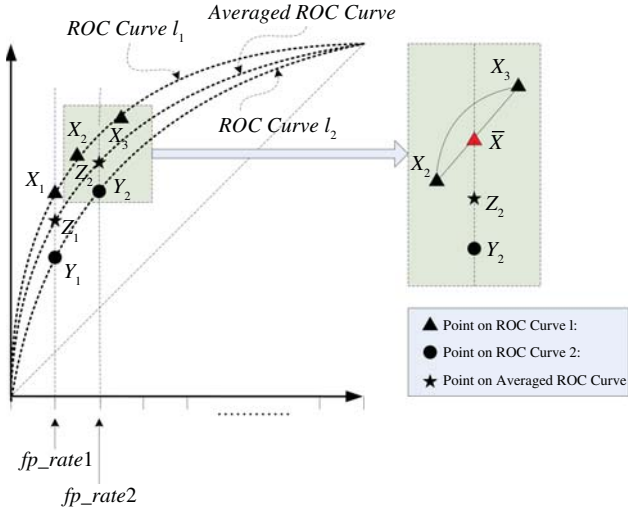

Fig. 5. Vertical averaging approach of ROC curves.

In order to reflect the ROC curve characteristics for all of the random runs, we adopt the vertical averaging approach in [47] to plot the averaged ROC curves. Our implementation of the vertical averaging method is illustrated in Fig. 5. Assume one would like to average two ROC curves: l_1 and l_2, each of which is formed by a series of points in the ROC space. The first step is to evenly divide the range of fp_rate into a set of intervals. Then at each interval, find the corresponding

TABLE III
EVALUATION METRICS AND PERFORMANCE COMPARISON

Dataset	Methods	OA	Precision	Recall	F-measure	G-mean	AUC
Sonar	RAMOBoost	**0.7798**	**0.7566**	0.7813	**0.7672**	**0.7796**	**0.86343**
	SMOTEBoost	0.7702	0.7459	0.7748	0.7579	0.7697	0.86176
	SMOTE	0.7606	0.7330	0.7687	0.7485	0.7605	0.84311
	ADASYN	0.5712	0.5184	**0.9815**	0.6780	0.4624	0.82382
	AdaCost	0.7721	0.7559	0.7644	0.7597	0.7711	0.76864
	BorderlineSMOTE	0.7606	0.7364	0.771	0.7494	0.7607	0.84205
	SMOTE-Tomek	0.7442	0.7379	0.8073	0.7144	0.7448	0.82378
Spambase	RAMOBoost	0.9448	**0.9244**	0.9387	**0.9315**	0.9438	0.98379
	SMOTEBoost	0.9435	0.9191	0.9418	0.9302	0.9432	0.98329
	SMOTE	0.9397	0.9194	0.9311	0.9251	0.9382	0.97942
	ADASYN	0.7746	0.6424	**0.9851**	0.7776	0.7904	0.96849
	AdaCost	**0.9472**	0.8974	0.9413	0.8588	**0.9462**	**0.98552**
	BorderlineSMOTE	0.9291	0.9028	0.936	0.8632	0.9302	0.97362
	SMOTE-Tomek	0.9376	0.9002	0.9384	0.8611	0.9377	0.97649
Ionosphere	RAMOBoost	**0.8411**	**0.8512**	0.6638	**0.744**	**0.7874**	**0.90138**
	SMOTEBoost	0.8251	0.8244	0.6346	0.7156	0.7662	0.88907
	SMOTE	0.8177	0.8026	0.6425	0.7106	0.7643	0.82093
	ADASYN	0.6749	0.5263	**0.7602**	0.6198	0.6912	0.79778
	AdaCost	0.8337	0.8237	0.6059	0.7352	0.7604	0.88186
	BorderlineSMOTE	0.8206	0.8466	0.6516	0.7078	0.7698	0.81265
	SMOTE-Tomek	0.8166	0.8494	0.6539	0.7110	0.7677	0.8265
PID	RAMOBoost	0.724	**0.5766**	0.7467	**0.6497**	**0.729**	0.79608
	SMOTEBoost	0.7229	0.5764	0.74	0.6466	0.7267	0.79825
	SMOTE	0.7214	0.5746	0.7511	0.6496	0.7281	0.80428
	ADASYN	0.5539	0.4357	**0.9709**	0.5994	0.5702	0.8144
	AdaCost	**0.7438**	0.2816	0.61	0.3849	0.7043	**0.81805**
	BorderlineSMOTE	0.7018	0.375	0.7656	0.5029	0.7154	0.7947
	SMOTE-Tomek	0.7039	0.3956	0.8102	0.5313	0.7248	0.81186
Wine	RAMOBoost	**0.9798**	**0.9525**	0.9885	**0.9696**	**0.9813**	**0.99940**
	SMOTEBoost	0.9787	0.9492	0.9885	0.9678	0.9805	0.99937
	SMOTE	0.9787	0.9505	0.9885	0.9684	0.9804	0.99908
	ADASYN	0.7933	0.6094	**1.0000**	0.7536	0.8352	0.99607
	AdaCost	0.9764	0.9319	0.9813	0.9648	0.9769	0.99905
	BorderlineSMOTE	0.9753	0.9419	0.9885	0.9681	0.9778	0.99796
	SMOTE-Tomek	0.9551	0.9467	0.9853	0.9696	0.9629	0.99753
German	RAMOBoost	0.7262	**0.5602**	0.5270	**0.5409**	**0.6547**	**0.74139**
	SMOTEBoost	0.7072	0.5258	0.5126	0.5176	0.6375	0.73357
	SMOTE	0.6850	0.4878	0.5570	0.5192	0.6420	0.71365
	ADASYN	0.4918	0.3651	**0.8762**	0.5143	0.5282	0.70182
	AdaCost	**0.7482**	0.3963	0.4797	0.5283	0.6446	0.71254
	BorderlineSMOTE	0.6846	0.4522	0.5754	0.5151	0.6492	0.7105
	SMOTE-Tomek	0.691	0.4777	0.6296	0.5148	0.6711	0.73469
Phoneme	RAMOBoost	0.7921	0.5914	0.9068	**0.7158**	**0.8222**	**0.90621**
	SMOTEBoost	0.8018	**0.6131**	0.8524	0.7128	0.8159	0.89472
	SMOTE	0.7860	0.5952	0.8248	0.6899	0.7942	0.87186
	ADASYN	0.7260	0.5137	**0.9513**	0.6671	0.7770	0.86497
	AdaCost	**0.8191**	0.2473	0.702	0.3657	0.7797	0.89395
	BorderlineSMOTE	0.7632	0.3308	0.8741	0.4799	0.7918	0.86103
	SMOTE-Tomek	0.7884	0.2985	0.8151	0.4369	0.7965	0.87136
Vehicle	RAMOBoost	0.9655	**0.9142**	0.9398	0.926	0.956	0.99487
	SMOTEBoost	**0.9667**	0.9137	0.946	**0.929**	0.9591	0.99446
	SMOTE	0.9589	0.891	0.9373	0.9132	0.9511	0.99314
	ADASYN	0.821	0.5665	**0.9927**	0.7206	0.8737	0.97517
	AdaCost	0.9652	0.9132	0.9575	0.371	0.9623	**0.99511**
	BorderlineSMOTE	0.961	0.9130	0.9652	0.3752	**0.9624**	0.99405
	SMOTE-Tomek	0.9482	0.9091	0.9361	0.3679	0.9436	0.98498
Texture	RAMOBoost	**0.9991**	**0.9986**	0.9966	**0.9976**	0.9981	**0.99999**
	SMOTEBoost	0.999	0.9976	**0.997**	0.9973	**0.9982**	0.99998
	SMOTE	0.9949	0.9853	0.9863	0.9858	0.9916	0.99920
	ADASYN	0.9156	0.6837	0.9950	0.8101	0.9453	0.99487
	AdaCost	0.9987	0.9798	0.9953	0.9946	0.9974	0.99991
	BorderlineSMOTE	0.9928	0.9783	0.9811	0.9917	0.9881	0.99856
	SMOTE-Tomek	0.9976	0.9793	0.9913	0.9937	0.9951	0.99967
Segment	RAMOBoost	**0.9966**	**0.9854**	0.9907	**0.9880**	**0.9941**	0.99976
	SMOTEBoost	0.9965	0.9853	0.9900	0.9876	0.9938	**0.99978**
	SMOTE	0.9958	0.9835	0.9863	0.9848	0.9918	0.99959
	ADASYN	0.9254	0.6253	**1.0000**	0.7980	0.9556	0.99903
	AdaCost	0.9965	0.9845	0.9913	0.9843	0.9943	0.99974
	BorderlineSMOTE	0.9954	0.984	0.9869	0.9822	0.9918	0.99950
	SMOTE-Tomek	0.9953	0.984	0.9863	0.9820	0.9915	0.99961

Dataset	Methods	OA	Precision	Recall	F-measure	G-mean	AUC
Page_Blocks	RAMOBoost	**0.9702**	0.8326	0.8928	**0.8614**	**0.9349**	**0.98899**
	SMOTEBoost	0.9696	**0.8340**	0.8825	0.8573	0.9297	0.98772
	SMOTE	0.9594	0.7781	0.8563	0.8140	0.9118	0.97993
	ADASYN	0.9251	0.5862	**0.9414**	0.7223	0.9322	0.97621
	AdaCost	0.9704	0.7912	0.8559	0.8469	0.9175	0.98861
	BorderlineSMOTE	0.9463	0.7853	0.8713	0.8171	0.912	0.97063
	SMOTE-Tomek	0.9576	0.7832	0.8627	0.8168	0.9139	0.97754
Satimage	RAMOBoost	0.9195	0.5671	0.7127	**0.6312**	0.819	**0.94860**
	SMOTEBoost	**0.923**	**0.5867**	0.6717	0.6276	0.7986	0.94678
	SMOTE	0.8977	0.4791	0.606	0.5327	0.7465	0.89748
	ADASYN	0.8422	0.3645	0.8431	0.5084	**0.8424**	0.92234
	AdaCost	0.9217	0.552	0.5426	0.371	0.7118	0.93255
	BorderlineSMOTE	0.8938	0.685	**0.9652**	0.3752	0.7598	0.90189
	SMOTE-Tomek	0.8957	0.701	0.9361	0.3679	0.773	0.90251
Mf_Zernike	RAMOBoost	0.8718	0.3608	0.369	0.3645	0.584	0.89452
	SMOTEBoost	0.8701	0.3544	0.364	0.3584	0.5798	0.89537
	SMOTE	0.8838	0.4409	0.592	0.5045	0.7356	0.8922
	ADASYN	0.8634	0.408	**0.809**	**0.5419**	**0.838**	0.90609
	AdaCost	0.8851	0.3604	0.446	0.3935	0.6441	**0.90656**
	BorderlineSMOTE	0.8827	0.3678	0.598	0.4217	0.7377	0.8924
	SMOTE-Tomek	**0.8877**	**0.3839**	0.745	0.4518	0.8199	0.90194
Vowel	RAMOBoost	**0.9988**	**0.9934**	**0.9931**	**0.9931**	**0.9962**	**0.99990**
	SMOTEBoost	0.9974	0.9842	0.9867	0.9853	0.9925	0.99988
	SMOTE	0.9794	0.8569	0.9379	0.893	0.9599	0.99615
	ADASYN	0.9101	0.5095	0.9488	0.6623	0.927	0.98512
	AdaCost	0.9913	0.903	0.9696	0.9651	0.9813	0.99906
	BorderlineSMOTE	0.9766	0.8710	0.9222	0.9591	0.9515	0.99552
	SMOTE-Tomek	0.9747	0.8890	0.9382	0.9624	0.9576	0.99344
Abalone	RAMOBoost	0.9405	0.4968	0.4889	0.4813	0.6808	**0.97609**
	SMOTEBoost	0.943	0.5181	**0.5348**	0.5173	0.7134	0.92271
	SMOTE	0.9477	**0.5886**	0.5328	0.5412	**0.7166**	0.92291
	ADASYN	0.9101	0.361	0.4838	0.3892	0.6684	0.89179
	AdaCost	**0.9521**	0.241	0.4003	0.455	0.6156	0.92395
	BorderlineSMOTE	0.9493	0.294	0.4855	**0.554**	0.686	0.90322
	SMOTE-Tomek	0.9441	0.261	0.4319	0.492	0.6433	0.9039
Glass	RAMOBoost	0.9748	0.6169	0.8464	0.7731	0.8610	0.99478
	SMOTEBoost	0.9748	0.6480	**0.9464**	0.7430	**0.9596**	0.99429
	SMOTE	0.9897	**0.8940**	0.9179	**0.8874**	0.9491	**0.99801**
	ADASYN	0.9421	0.4552	0.7986	0.4970	0.8555	0.97723
	AdaCost	**0.9907**	0.6377	0.9429	0.7722	0.9625	0.99741
	BorderlineSMOTE	0.9907	0.6368	0.9262	0.7040	0.9543	0.99757
	SMOTE-Tomek	0.9879	0.6359	0.9119	0.6988	0.9414	0.99736
Yeast	RAMOBoost	0.9581	0.467	0.4341	0.4418	0.6405	0.74512
	SMOTEBoost	0.9585	0.4941	0.4732	0.4687	0.6651	0.74878
	SMOTE	0.9722	**0.7557**	**0.5107**	**0.5761**	0.7030	0.81603
	ADASYN	0.9552	0.5276	0.4891	0.4758	0.6810	0.77902
	AdaCost	0.9718	0.479	0.4524	0.344	0.6593	0.7792
	BorderlineSMOTE	**0.9726**	0.492	0.4882	0.368	0.6812	0.8096
	SMOTE-Tomek	0.9768	0.420	0.5107	0.384	**0.7049**	**0.8241**
Letter	RAMOBoost	**0.9982**	**0.9882**	**0.9662**	**0.977**	**0.9827**	**0.99978**
	SMOTEBoost	0.9977	0.983	0.9591	0.9708	0.979	0.99977
	SMOTE	0.9921	0.9122	0.8853	0.8981	0.9391	0.99514
	ADASYN	0.9705	0.5841	0.9122	0.7109	0.942	0.9901
	AdaCost	0.9961	0.9836	0.9261	0.9705	0.9618	0.9989
	BorderlineSMOTE	0.9736	0.9264	0.9003	0.87	0.9375	0.9902
	SMOTE-Tomek	0.9925	0.9052	0.8863	0.867	0.9399	0.9943
Shuttle	RAMOBoost	**0.9999**	0.9495	0.9728	**0.9576**	0.9828	**0.9998**
	SMOTEBoost	**0.9999**	0.9442	0.9667	0.9565	0.9828	0.9997
	SMOTE	**0.9999**	**0.9719**	0.9444	0.9538	0.9716	0.9998
	ADASYN	0.9997	0.7984	**1**	0.8855	**0.9999**	0.9995
	AdaCost	**0.9999**	0.9410	0.9667	0.9521	0.9885	0.9998
	BorderlineSMOTE	**0.9999**	0.9400	0.9723	0.9520	0.9857	0.9998
	SMOTE-Tomek	**0.9999**	0.9430	0.9333	0.9320	0.9657	0.9998

tp_rate values of each ROC curve and average them. In Fig. 5, X_1 and Y_1 are the points from l_1 and l_2 corresponding to the interval fp_rate1. By averaging their tp_rate values, the corresponding ROC point Z_1 on the averaged ROC curve is obtained. However, there exist some ROC curves that do not have corresponding points on certain intervals. In this case, one can use the linear interpolation method to obtain the averaged ROC points. For instance, in Fig. 5, the point $\overline{X}$ (corresponding to fp_rate2) is calculated based on the linear interpolation of the two neighboring points X_2 and X_3. Once $\overline{X}$ is obtained, it can be averaged with Y_2 to get the corresponding Z_2 point on the averaged ROC curve. Our AUC results presented in this section are based on the average of all random runs according to the vertical averaging approach.

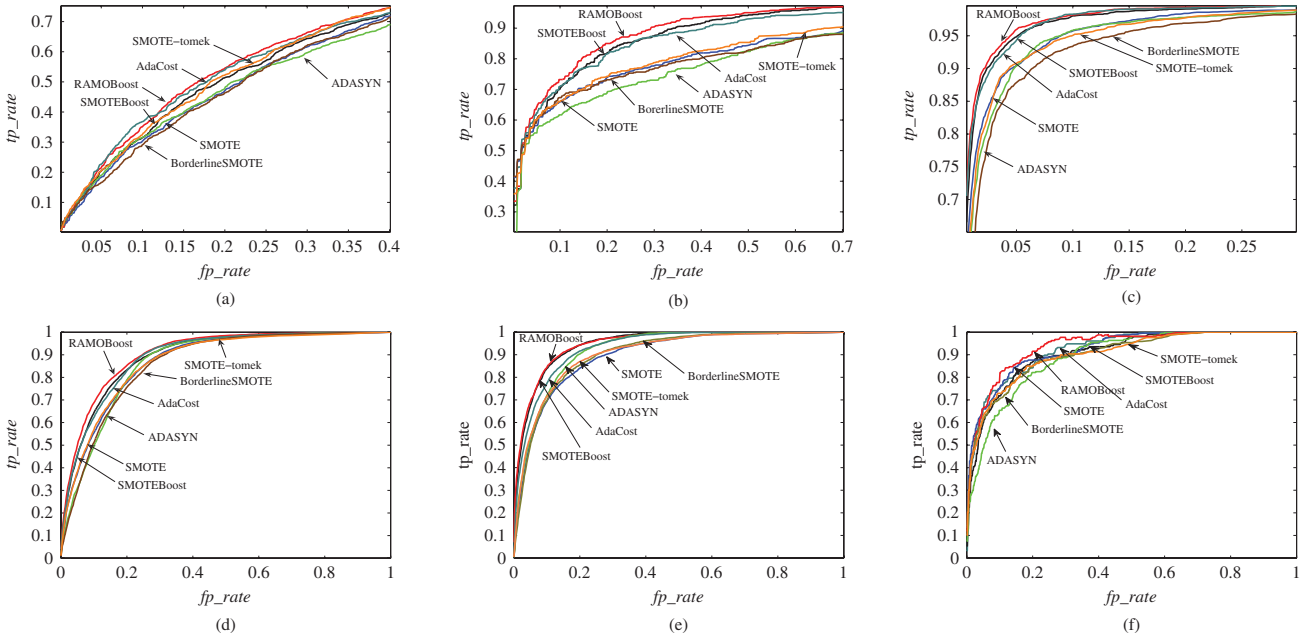

Fig. 6. Averaged ROC curves for RAMOBoost, SMOTEBoost, SMOTE, ADASYN, AdaCost, BorderlineSMOTE, and SMOTE-Tomek methods. (a) Averaged ROC curves for German dataset. (b) Averaged ROC curves for Ionosphere dataset. (c) Averaged ROC curves for Page_Blocks dataset. (d) Averaged ROC curves for Phoneme dataset. (e) Averaged ROC curves for Satimage dataset. (f) Averaged ROC curves for Abalone dataset.

TABLE IV
SIMULATION 1: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND SMOTEBOOST

Dataset	RAMOBoost	SMOTEBoost	Difference	Rank
Sonar	0.86343	0.86176	+0.00167	12
Spambase	0.98379	0.98329	+0.0005	9
Ionosphere	0.90138	0.88907	+0.01231	18
PID	0.79608	0.79825	−0.00217	14
Wine	0.9994	0.99937	+0.00005	5
German	0.74139	0.73357	+0.00782	16
Phoneme	0.90621	0.89472	+0.01149	17
Vehicle	0.99487	0.99446	+0.00041	7
Texture	0.99999	0.99998	+0.00001	1.5
Segment	0.99976	0.99978	−0.00002	3.5
Page_Blocks	0.98899	0.98772	+0.00127	11
Satimage	0.9486	0.94678	+0.00182	13
Mf_Zernike	0.89452	0.89537	−0.00085	10
Vowel	0.9999	0.99988	+0.00002	3.5
Abalone	0.97609	0.92271	+0.05338	19
Glass	0.99478	0.99429	+0.00049	8
Yeast	0.74512	0.74878	−0.00366	15
Letter	0.99978	0.99977	+0.00001	1.5
Shuttle	0.9998	0.9997	+0.0001	6
$R^+ = 147.5$ and $R^- = 42.5$				

TABLE V
SIMULATION 1: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND ADACOST

Dataset	RAMOBoost	AdaCost	Difference	Rank
Sonar	0.86343	0.82832	+0.03511	18
Spambase	0.98379	0.98552	−0.00173	9
Ionosphere	0.90138	0.88186	+0.01952	14
PID	0.79608	0.81805	−0.02197	15
Wine	0.9994	0.99905	+0.00035	5
German	0.74139	0.71254	+0.02885	16
Phoneme	0.90621	0.89395	+0.01226	12
Vehicle	0.99487	0.99511	−0.00024	4
Texture	0.99999	0.99991	+0.00008	3
Segment	0.99976	0.99974	0.00002	2
Page_Blocks	0.98899	0.98861	+0.00038	6
Satimage	0.9486	0.93255	+0.01605	13
Mf_Zernike	0.89452	0.90656	−0.01204	11
Vowel	0.9999	0.99906	+0.00084	7
Abalone	0.97609	0.92395	+0.05214	19
Glass	0.99478	0.99741	−0.00263	10
Yeast	0.74512	0.7792	−0.03408	17
Letter	0.99978	0.9989	0.00088	8
Shuttle	0.9998	0.9998	0.0000	1
$R^+ = 124$ and $R^- = 66$				

In order to evaluate the significance of the simulation results of the comparative algorithms, Wilcoxon signed-ranks test is used in this paper. Wilcoxon signed-ranks test is a nonparametric statistical procedure for comparing two samples that are paired, or related [52]. It assumes commensurability of differences, but only qualitatively, greater differences still count more, which is probably desired, but the absolute magnitudes are ignored. From a statistical point of view, the test is safer since it does not assume normal distributions. Also, outliers (exceptionally good/bad performances on a few datasets) have less effect on the Wilcoxon than on the t-test [53].

Suppose there are n objects to be observed by two algorithms, let us denote the difference value of the two algorithms' observation on the ith objects to be d_i, $i = 1, \ldots, n$. The differences are ranked according to their absolute values, ranks of the tied values are averaged. Let R^+ stand for the sum of the ranks of the objects on which the difference value of the two algorithms' observations are greater than zero, and R^- denote the sum of the opposite. Ranks of $d_i = 0$ are evenly split between R^+ and R^- [53]. Equations (7) and (8) conclude the calculations of R^+ and R^-

$$R^+ = \sum_{d_i>0} rank(|d_i|) + \frac{1}{2}\sum_{d_i=0} rank(|d_i|) \tag{7}$$

$$R^- = \sum_{d_i<0} rank(|d_i|) + \frac{1}{2}\sum_{d_i=0} rank(|d_i|). \tag{8}$$

TABLE VI

SIMULATION 1: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
R^+	147.5	154	161	124	165	144
R^-	42.5	36	29	66	25	46
T	**42.5**(+)	**36**(+)	**29**(+)	66(+)	**25**(+)	**46**(+)

TABLE VII

SIMULATION 2: AUC PERFORMANCE CHARACTERISTICS

Dataset	RAMO	SMOTE	ADASYN	BorderlineSMOTE	SMOTE-Tomek
Sonar	**0.846154**	0.838510	0.756069	0.836133	0.788572
Spambase	0.974339	**0.976614**	0.957916	0.970457	0.976122
Ionosphere	0.792544	**0.794108**	0.741437	0.792707	0.782712
PID	0.793827	0.792448	**0.798873**	0.784872	0.798328
Wine	**0.993323**	0.992665	0.974438	0.992554	0.989544
German	**0.737385**	0.718070	0.695423	0.721773	0.734250
Phoneme	**0.878892**	0.873555	0.866045	0.863607	0.871154
Vehicle	**0.990476**	0.990147	0.967718	0.986183	0.981260
Texture	0.998729	0.999154	0.989977	0.999343	**0.999382**
Segment	**0.999250**	0.999235	0.998590	0.999143	0.999232
Page_Blocks	**0.978171**	0.974781	0.972891	0.970749	0.959234
Satimage	**0.912619**	0.898285	0.894741	0.894326	0.904652
Mf_Zernike	0.883572	0.891459	**0.897191**	0.878563	0.895724
Vowel	**0.989597**	0.987338	0.984017	0.986068	0.986165
Abalone	0.866594	**0.868805**	0.831969	0.868212	0.858027
Glass	0.990220	**0.992166**	0.940170	0.991193	0.989247
Yeast	**0.846304**	0.818289	0.791463	0.830636	0.823307
Letter	**0.996011**	0.993136	0.993441	0.992036	0.993168
Shuttle	**0.999978**	0.999972	0.999956	0.999967	0.999978

If we set $T = \min\{R^+, R^-\}$, with a significance level of $\alpha = 0.05$ and the number of observed objects being n, the significance value N that T should be equal or less than for rejection of a null hypothesis can be retrieved by querying the critical value table, which can be accessed in [54]. In the rest of this section, Wilcoxon signed-ranks test is conducted between RAMOBoost and each of other comparative algorithms, i.e., RAMOBoost vs. SMOTEBoost, RAMOBoost vs. AdaCost, RAMO vs. SMOTE, etc. In all tables presenting the results of significance test, the "(+)" symbol signifies that RAMOBoost is quantitatively better than the comparative algorithm under consideration in terms of the specified assessment metric, and "(−)" denotes the opposite. Whenever there is a significance existing, we highlight the corresponding result by underscoring it.

C. Simulation Results

In our simulation, we use 20 boosting iterations ($T = 20$ in the algorithm) as suggested in [55] for ensemble learning. The number of synthetic data generated at each boosting iteration is set to 200% of the number of the minority instances [5]. The parameters k_1 and k_2 are set to be 5 and 10, respectively. The scaling coefficient α is set to 0.3, which was chosen using cross-validation techniques for optimizing RAMOBoost's performance. For SMOTEBoost, SMOTE, ADASYN, BorderlineSMOTE, and SMOTE-Tomek, the number of nearest neighbors is set to five. The cost factor C for AdaCost is set to three according to the suggestion of [23] (C should be an integer between 2 and 9).

Following the suggestion of [53], the significance test are conducted on the averaged AUC of all algorithms in a pairwise manner for all simulations introduced in the rest of this section.

TABLE VIII

SIMULATION 2: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

	RAMO vs.			
	SMOTE	ADASYN	BorderlineSMOTE	SMOTE-Tomek
R^+	135	177	170	156
R^-	55	13	20	34
T	55(+)	**13**(+)	**20**(+)	**34**(+)

1) Simulation 1: In this simulation, we apply all comparative algorithms to the 19 datasets described in Table I. The simulation results are based on the average of ten runs. At each run, we randomly select half of the dataset as training data and use the remaining half as testing data.

Fig. 6 gives several snapshots of the averaged ROC graphs of the RAMOBoost, SMOTEBoost, SMOTE, ADASYN, AdaCost, BorderlineSMOTE, and SMOTE-Tomek methods. Here Fig. 6(a)–(f) represents the results for the German, Ionosphere, Page_Blocks, Phoneme, Satimage, and Abalone datasets, respectively. This figure indicates that the RAMOBoost method is competitive when compared to other methods in ROC space.

Table III summarizes the performance of the comparative algorithms, in which the best performance of each algorithm across each evaluation criteria is highlighted. From Table III, we find that RAMOBoost can provide competitive simulation results on most of the datasets when compared to other comparative algorithms. Except for *Recall* performance, we

TABLE IX
SIMULATION 3: AUC PERFORMANCE CHARACTERISTICS

Dataset	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
Texture	0.99963	0.9999	0.99862	0.99315	**0.99995**	0.99887	0.99936
Segment	0.99932	0.9993	0.99937	0.99891	**0.99953**	0.99935	0.99938
Page_Blocks	**0.98484**	0.98107	0.97255	0.97386	0.98463	0.96692	0.96155
Satimage	**0.94081**	0.93506	0.90778	0.91162	0.92754	0.90355	0.90177
Mf_Zernike	0.88122	0.87958	0.88309	0.88905	**0.89816**	0.88033	0.89368
Vowel	**0.99685**	0.99637	0.98826	0.98661	0.99431	0.98816	0.98237
Abalone	**0.88867**	0.87861	0.87786	0.84324	0.87364	0.86522	0.86813
Glass	**0.99608**	0.99511	0.99316	0.95353	0.99411	0.98825	0.99416
Yeast	0.81676	0.80704	**0.82087**	0.81747	0.80492	0.8188	0.80832
Letter	**0.99922**	0.99911	0.99614	0.99247	0.99797	0.9916	0.99319

TABLE X
SIMULATION 3: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

	RAMOBoost *vs.*					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
R^+	52	45	48	40.5	50	48
R^-	3	10	7	14.5	5	7
T	**3**(+)	10(+)	**7**(+)	14.5(+)	**5**(+)	**7**(+)

TABLE XI
SIMULATION TIME (IN SECONDS) OF COMPARATIVE ALGORITHMS ACROSS ALL DATASETS

Dataset	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
Sonar	1.1995	1.2349	1.5778	1.4492	1.1368	1.272	1.4628
Spambase	15.3089	11.0303	16.7741	18.9078	7.0951	11.2494	132.8847
Ionosphere	1.6868	1.7125	1.6148	1.5352	1.4665	1.6891	2.267
PID	2.297	2.1587	2.6745	2.379	1.7895	1.9837	3.4942
Wine	1.4261	1.6415	1.4927	1.3863	1.2253	1.3867	1.5356
German	2.3884	2.6636	2.7584	3.4261	1.9259	2.8012	4.165
Phoneme	17.5034	14.8289	17.5991	24.3891	9.4682	17.1371	151.653
Vehicle	2.4206	2.2986	2.4739	3.1783	2.8343	3.1577	3.9203
Texture	9.2259	7.873	10.0554	12.7696	8.9655	16.2477	133.0821
Segment	3.7496	3.9507	3.9468	5.0766	3.4291	3.7959	19.1355
Page_Blocks	9.3199	7.2831	8.6075	12.721	6.453	6.9188	79.2096
Satimage	9.2517	11.4603	9.4158	16.1668	6.6197	7.7479	127.7468
Mf_Zernike	4.1083	3.5941	4.2358	5.3014	3.883	5.9433	26.9732
Vowel	2.5366	2.2451	2.2579	2.6727	1.9955	1.9551	3.8541
Abalone	1.6511	1.5876	1.8646	1.7697	1.6448	1.5338	2.0746
Glass	1.2615	1.3224	1.3394	1.3233	3.1117	1.1862	1.383
Yeast	1.6191	2.0146	1.8106	1.4243	2.062	2.0514	2.8833
Letter	40.5455	38.2847	21.596	105.2056	36.6272	20.9144	821.9379
Shuttle	38.8126	41.6861	62.0741	31.9209	43.2685	37.1499	790.5308

TABLE XII
SIMULATION OF TUNING THE OVERSAMPLING RATIO: AUC PERFORMANCE CHARATERISTICS

Oversampling ratio	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
100%	**0.93202**	0.91099	0.91676	0.89424	0.91908	0.90968	0.8867
200%	**0.9308**	0.90747	0.91655	0.89257	0.92351	0.90706	0.89791
300%	**0.93244**	0.91241	0.92108	0.88736	0.92208	0.90997	0.8977
400%	**0.93146**	0.90854	0.9221	0.88713	0.9227	0.90633	0.90664
500%	**0.92374**	0.90976	0.92317	0.87863	0.92206	0.91136	0.90995

see that ADASYN seems to provide a better *Recall* rate on most of these datasets. This is because ADASYN can learn very aggressively from the boundary since it generates synthetic data instances very close to the decision boundary [see Fig. 4(c)]. This means that ADASYN may push the algorithm to focus on the minority (positive) class data to improve the *Recall* criteria, while the overall performance may not improve significantly. In other words, if one algorithm classifies all testing data as "positive" (minority class), its "*Recall*" rate will be maximized even if the overall performance is low. The results in Table III shows that ADASYN performs better than other comparative algorithms in terms of *Recall*, which only stands for the number of correctly classified minority instances, but performs worse in all other assessment metrics, such as *F-measure* and *G-mean* which represent the algorithm's overall performance, on most of the datasets. These results confirm our discussions in Section III-C.1 regarding the different characteristics of these algorithms.

The significance test is applied on the simulation results to evaluate whether RAMOBoost can statistically outperform

TABLE XIII

SIMULATION OF TUNING THE OVERSAMPLING RATIO: SIGNIFICANCE TEST OF AUC BASED ON RANDOM RUNS BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

Oversampling Ratio	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
100%	**0**(+)	**5**(+)	**0**(+)	**0**(+)	**4**(+)	**0**(+)
200%	**0**(+)	**7**(+)	**1**(+)	**3**(+)	**4**(+)	**3**(+)
300%	**1**(+)	9(+)	**0**(+)	**7**(+)	**6**(+)	**0**(+)
400%	**0**(+)	**7**(+)	**0**(+)	**5**(+)	**2**(+)	**3**(+)
500%	**3**(+)	**8**(+)	**0**(+)	**7**(+)	**4**(+)	**4**(+)

TABLE XIV

SIMULATION OF TUNING THE IMBALANCED RATIO: POLICY OF COMBINATION OF CLASSES IN "ABALONE" DATASET

Index	Minority combination	Majority combination	# Minority	# Majority	Imbalanced ratio
I	$1 \oplus 2 \oplus 22 \oplus 24 \oplus 25 \oplus 26 \oplus 27 \oplus 28$	$8 \oplus 9 \oplus 10 \oplus 11$	15	2378	0.0063 : 0.9937
II	$\text{I} \oplus 23$	$8 \oplus 9 \oplus 10 \oplus 11$	24	2378	0.01 : 0.99
III	$\text{II} \oplus 21$	$8 \oplus 9 \oplus 10 \oplus 11$	38	2378	0.0157 : 0.9843
IV	$\text{III} \oplus 3$	$8 \oplus 9 \oplus 10 \oplus 11$	53	2378	0.0218 : 0.9782
V	$\text{IV} \oplus 20$	$8 \oplus 9 \oplus 10 \oplus 11$	79	2378	0.0322 : 0.9678
VI	$\text{V} \oplus 19$	$8 \oplus 9 \oplus 10 \oplus 11$	111	2378	0.0446 : 0.9554
VII	$\text{VI} \oplus 18 \oplus 4$	$8 \oplus 9 \oplus 10 \oplus 11$	210	2378	0.0811 : 0.9189
VIII	$\text{VII} \oplus 17 \oplus 15$	$8 \oplus 9 \oplus 10 \oplus 11$	371	2378	0.1350 : 0.8650
IX	$\text{VIII} \oplus 5$	$8 \oplus 9 \oplus 10 \oplus 11$	486	2378	0.1797 : 0.8303
X	$\text{IX} \oplus 6$	$8 \oplus 9 \oplus 10 \oplus 11$	745	2378	0.2386 : 0.7614

TABLE XV

SIMULATION OF TUNING THE IMBALANCED RATIO: AUC PERFORMANCE CHARACTERISTICS

Imbalanced ratio	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
0.0063 : 0.9937	**0.97887**	0.97558	0.94373	0.94163	0.96915	0.9166	0.90582
0.01 : 0.99	**0.90648**	0.90557	0.90495	0.90412	0.9049	0.84915	0.8784
0.0157 : 0.9843	0.91886	0.91913	0.92122	0.92361	**0.92921**	0.91407	0.89492
0.0218 : 0.9782	**0.95542**	0.95072	0.93376	0.93219	0.95144	0.89655	0.92929
0.0322 : 0.9678	**0.9584**	0.9523	0.94371	0.93093	0.95562	0.93095	0.93487
0.0446 : 0.9554	**0.94454**	0.93652	0.91817	0.92162	0.94109	0.86302	0.90876
0.0811 : 0.9189	0.95025	0.94594	0.93479	0.9348	**0.95068**	0.92926	0.91392
0.1350 : 0.8650	**0.91502**	0.90501	0.89994	0.87255	0.91194	0.89588	0.8867
0.1797 : 0.8303	**0.92274**	0.91745	0.91241	0.90206	0.91994	0.90475	0.90704
0.2386 : 0.7614	**0.89696**	0.884	0.87537	0.87725	0.89389	0.86997	0.86978

TABLE XVI

SIMULATION OF TUNING THE IMBALANCED RATIO: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
R^+	54	53	54	44	55	55
R^-	1	2	1	11	0	0
T	**1**(+)	**2**(+)	**1**(+)	11(+)	**0**(+)	**0**(+)

other comparative algorithms. Since there are 19 datasets, T should be less than or equal to 46 to reject a null hypothesis in the significance level of 0.05, according to the critical value table. Table IV shows the significance test result of averaged AUC for RAMOBoost vs. SMOTEBoost. One can conclude that RAMOBoost can statistically outperform SMOTEBoost ($T = \min\{R^+, R^-\} = 42.5 < 46$), it proves that although RAMOBoost shares the same boosting procedure and data generation technique with SMOTEBoost, the adaptive ranking mechanism for determining the number of synthetic instances for each minority example makes RAMOBoost perform better than SMOTEBoost. Table V shows the similar result with that of Table IV for RAMOBoost vs. AdaCost, from which, however, one can see that RAMOBoost cannot statistically outperform AdaCost ($T = \min\{R^+, R^-\} = 62$). For space consideration, the detailed statistical analysis for RAMOBoost against the remaining comparative algorithms is omitted. Instead, we provide in Table VI, the T-value of RAMOBoost against all comparative algorithms. One can see that RAMOBoost also statistically

outperforms SMOTE, ADASYN, BorderlineSMOTE, and SMOTE-Tomek.

We have also conducted the simulations of RAMOBoost on all datasets when the number of hidden layer neurons for the base classifier MLP is set to be ten. Our simulation results indicate that increasing the number of hidden layer neurons does not necessarily improve the learning performance in this case. We feel there might be several reasons for this, such as the potential overfitting issue. Furthermore, as suggested in [56], using a strong base classifier in the ensemble approach may not benefit the final learning performance due to increased bias of such classifiers. Due to space consideration, we refrain from providing the detailed results for all these experiments.

2) Simulation 2: In Section III-C.1, we investigated the data generation mechanism of RAMOBoost compared to that of SMOTE and ADASYN on a synthetic dataset shown in Fig. 4. One interesting question that arises is, if the data generation mechanism of RAMOBoost is extracted and wrapped up with other classifier in the way that SMOTE and ADASYN is used, which can be named as "RAMO," how will the learning performance of RAMO be when it is compared to other sampling approaches? To this end, we have conducted simulations for RAMO against SMOTE, ADASYN, BorderlineSMOTE, and SMOTE-Tomek on the 19 datasets described in Table I. For space considerations, we only provide simulation results of averaged AUC of the comparative algorithms in Table VII. The AUC value of the corresponding winning approach for each dataset is highlighted. Based on the results in Table VII, significance test is applied to evaluate whether RAMO can statistically outperform other existing approaches. Since the number of datasets used in Simulation 2 is the same as in Simulation 1, the significance value N is also 46. Table VIII demonstrates the significance test results, i.e., the T value of each comparative algorithms. One can see that RAMO can statistically outperform ADASYN, Borderline-SMOTE, and SMOTE-Tomek, but it cannot outperform SMOTE in this case.

3) Simulation 3: Another interesting simulation we conducted is to compare RAMOBoost with other comparative algorithms when both k_1 and k_2 are ten for RAMOBoost. We also configured the k value of SMOTEBoost, SMOTE, ADASYN, BorderlineSMOTE, and SMOTE-Tomek to be ten to provide a fair comparison. All other parameters remained the same. We compared the algorithms against the ten datasets with the largest skew ratio, since we are more interested in investigating how RAMOBoost performs with highly imbalanced datasets. In order to retain these severely imbalanced ratios, we adopted a different way of generating the training and testing datasets. Specifically, the training dataset was created by consolidating half of the randomly selected majority class examples and half of the randomly selected minority class examples. The remaining examples were used as testing dataset. One can easily verify that the training and testing datasets generated this way bear the same imbalance ratio as the original dataset. For space considerations, only the simulation results of AUC values are provided in Table IX, with the winning value across all comparative algorithms for each dataset highlighted. Based on the results in Table IX, significance test is applied to evaluate whether RAMOBoost can statistically outperform other existing approaches when both k_1 and k_2 are equal to ten. Since there are just ten datasets, T should be less than or equal to eight to reject a null hypothesis between two comparative algorithms. Table X presents the T value of RAMOBoost against other comparative algorithms, from which one can see that RAMOBoost can also statistically outperform SMOTEBoost, ADASYN, BorderlineSMOTE, and SMOTE-Tomek, but cannot outperform SMOTE and AdaCost in this case.

D. Computational Time for Simulation

Table XI shows the computational time in seconds of RAMOBoost on all datasets (based on the simulation environment of Intel Core Duo CPU L2500@ 1.83 GHz, 3 GB RAM, and MATLAB Version 7.3.0.267). From Table XI, one can see that the computational time cost of RAMOBoost is similar to that of the existing approaches. The runtime cost for SMOTE-Tomek is generally higher than other comparative algorithms especially when the size of the dataset is very large, i.e., "Letter" and "Shuttle." This is probably because SMOTE-Tomek iterates across the entire data space repeatedly until all Tomek links have been cleared.

E. Simulation Results on Tuning Parameters

To evaluate the robustness of RAMOBoost against other comparative algorithms in different parameter configurations and scenarios, simulations on tuning the minority oversampling ratios, the imbalanced ratio, and the class label noise and the attribute noise of the datasets have been conducted. For space consideration, we only present the results on the "Abalone" dataset. Again, MLP with the configuration described at the beginning of Section IV-C.1 is used as the base learner. The simulation results are also based on ten random runs. In each of these random runs, half of the original minority and majority datasets are randomly chosen and merged to be the training dataset and the remaining part is used as the testing dataset.

This experiment is motivated by [57], which suggested that the oversampling ratio could play a critical role for imbalanced learning problems. Here, we use the "Abalone" dataset described in Section IV-A as an example to show the performance by tuning the oversampling ratio. Specifically, the oversampling ratio for the minority class is increased progressively from 100% to 500% with an interval of 100%. Table XII displays the simulation results on ten random runs using the averaged AUC of the comparative algorithms for this dataset in which the best performance is highlighted. For the significance test, if we consider only the averaged AUC, since $T = R^- = 0$, RAMOBoost is undoubtedly significantly better than all comparative algorithms in all simulation scenarios. We also conducted the significance test based on the AUC of the random runs. In this case, N is equal to eight since the number of random runs is equal to ten. Table XIII shows the T value for the comparison between RAMOBoost and other comparative algorithms based on random runs.

The original "Abalone" dataset has 28 classes and 4177 examples, in which we employed only two classes to evaluate

TABLE XVII

SIMULATION OF TUNING THE IMBALANCED RATIO: SIGNIFICANCE TEST OF AUC BASED ON RANDOM RUNS BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

Imbalanced Ratio	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
0.0063 : 0.9937	**8**(+)	**1**(+)	**0**(+)	**6**(+)	**0**(+)	**0**(+)
0.01 : 0.99	**7**(+)	**8**(+)	**5**(+)	**7**(+)	**1**(+)	**5**(+)
0.0157 : 0.9843	9(-)	10(-)	**7**(-)	**8**(-)	**8**(+)	**2**(+)
0.0218 : 0.9782	**7**(+)	**2**(+)	**0**(+)	**7**(+)	**0**(+)	**1**(+)
0.0322 : 0.9678	**8**(+)	**3**(+)	**1**(+)	**9**(+)	**3**(+)	**1**(+)
0.0446 : 0.9554	**0**(+)	**1**(+)	**0**(+)	**5**(+)	**0**(+)	**0**(+)
0.0811 : 0.9189	**0**(+)	**0**(+)	**0**(+)	**8**(-)	**0**(+)	**0**(+)
0.1350 : 0.8650	**1**(+)	**0**(+)	**1**(+)	**5**(+)	**0**(+)	**0**(+)
0.1797 : 0.8303	**0**(+)	**0**(+)	**0**(+)	**0**(+)	**0**(+)	**0**(+)
0.2386 : 0.7614	**0**(+)	**0**(+)	**0**(+)	**0**(+)	**0**(+)	**0**(+)

TABLE XVIII

SIMULATION OF TUNING THE CLASS LABEL NOISE LEVEL: AUC PERFORMANCE CHARACTERISTICS

Noise level	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
5%	**0.91825**	0.90986	0.89977	0.87525	0.91615	0.90937	0.88927
10%	**0.90645**	0.90396	0.89337	0.85919	0.90435	0.89438	0.88483
15%	**0.8969**	0.88758	0.85463	0.83119	0.88407	0.88957	0.86429
20%	**0.91587**	0.88817	0.88142	0.88203	0.90667	0.90343	0.8914
25%	**0.90791**	0.89639	0.87178	0.87402	0.90094	0.89906	0.88178
30%	**0.91631**	0.89223	0.88137	0.88464	0.9128	0.90229	0.89378
35%	**0.90633**	0.87044	0.81876	0.77328	0.86926	0.89337	0.86083
40%	**0.8893**	0.87187	0.83236	0.822	0.88074	0.87376	0.83343
45%	0.88438	0.88015	0.8454	0.82038	0.88368	**0.90106**	0.8695
50%	**0.8787**	0.82759	0.81127	0.8032	0.86118	0.85911	0.86607

TABLE XIX

SIMULATION OF TUNING THE CLASS LABEL NOISE LEVEL: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
R^+	55	55	55	55	46	55
R^-	0	0	0	0	9	0
T	**0**(+)	**0**(+)	**0**(+)	**0**(+)	9(+)	**0**(+)

TABLE XX

SIMULATION OF TUNING THE CLASS LABEL NOISE LEVEL: SIGNIFICANCE TEST OF AUC BASED ON RANDOM RUNS BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

Noise level	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
5%	23(+)	**5**(+)	**0**(+)	22(+)	14(+)	**0**(+)
10%	27(+)	21(+)	**1**(+)	26(+)	15(+)	**3**(+)
15%	21(+)	**5**(+)	**1**(+)	18(+)	10(+)	**5**(+)
20%	11(+)	**7**(+)	**3**(+)	16(+)	**3**(+)	**0**(+)
25%	**8**(+)	**1**(+)	**0**(+)	14(+)	**5**(+)	**1**(+)
30%	**6**(+)	**2**(+)	**2**(+)	24(+)	20(+)	**6**(+)
35%	9(+)	**0**(+)	**0**(+)	**2**(+)	19(+)	**1**(+)
40%	23(+)	**3**(+)	**2**(+)	18(+)	15(+)	**0**(+)
45%	26(+)	**8**(+)	11(+)	27(+)	35(+)	**4**(+)
50%	**1**(+)	**0**(+)	**0**(+)	**8**(+)	13(+)	14(+)

the comparative algorithms on versatile datasets as described in Section IV-A. In order to obtain versatile imbalanced ratio, we manipulate the classes' combination of the original "Abalone" dataset to form minority class and majority class. Table XIV concludes the details for such combination and the corresponding imbalanced ratio. Table XV presents simulation results of ten random runs of experiments on tuning the imbalanced ratio, in which the best performance is highlighted. Based on the averaged AUC results in Table XV, Wilcoxon signed-ranks test tells us whether any significance exists between RAMOBoost and any of the comparative algorithms, which is shown in Table XVI. Using the "Abalone" dataset, we conducted significance tests on the AUC values of ten random runs. The results are given in Table XVII.

Noise in imbalanced datasets may exhibit unpredictably negative effects on the performance of learning algorithms. In order to systematically investigate the robustness of RAMOBoost, we manually introduce class label noise and attribute noise of different levels into the "Abalone" dataset and let RAMOBoost as well as other comparative algorithms learn

TABLE XXI
SIMULATION OF TUNING THE ATTRIBUTE NOISE LEVEL: AUC PERFORMANCE CHARACTERISTICS

Noise level	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
5%	**0.88085**	0.87754	0.85843	0.84749	0.86125	0.85856	0.843
10%	0.86914	0.86778	0.86167	0.83904	**0.87509**	0.84782	0.82937
15%	0.8617	0.86113	**0.8764**	0.84026	0.87397	0.86751	0.86461
20%	**0.82377**	0.80756	0.81314	0.78852	0.82155	0.81375	0.80046
25%	**0.82886**	0.81863	0.80462	0.77639	0.81464	0.82201	0.7792
30%	**0.81379**	0.80035	0.8078	0.76335	0.78836	0.79253	0.79726
35%	**0.81755**	0.79122	0.79077	0.78534	0.8061	0.78269	0.79851
40%	0.78242	0.78352	0.74457	0.7339	0.78625	0.77149	**0.78813**
45%	**0.78892**	0.75321	0.76008	0.74265	0.75714	0.76568	0.73578
50%	0.76642	0.66817	0.71511	0.73062	**0.78824**	0.71961	0.7449

TABLE XXII
SIMULATION OF TUNING THE CLASS LABEL NOISE LEVEL: SIGNIFICANCE TEST OF AVERAGED AUC BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
R^+	55	50	55	37	55	55
R^-	0	4	0	18	0	0
T	**0**(+)	**4**(+)	**0**(+)	18(+)	**0**(+)	**0**(+)

TABLE XXIII
SIMULATION OF TUNING THE CLASS LABEL NOISE LEVEL: SIGNIFICANCE TEST OF AUC BASED ON RANDOM RUNS BETWEEN RAMOBOOST AND EACH OF SMOTEBOOST, SMOTE, ADASYN, ADACOST, BORDERLINESMOTE, AND SMOTE-TOMEK[1]

Noise level	RAMOBoost vs.					
	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-Tomek
5%	21(+)	**7**(+)	**1**(+)	**5**(+)	**4**(+)	**1**(+)
10%	22.5(+)	26(+)	**2**(+)	27(-)	11(+)	**8**(+)
15%	27(+)	21(-)	13(+)	12(-)	24(-)	25(-)
20%	**7**(+)	15(+)	**6**(+)	21(+)	18(+)	11(+)
25%	18(+)	14(+)	**3**(+)	12(+)	19(+)	**3**(+)
30%	21(+)	27(+)	**6**(+)	13(+)	16(+)	**19**(+)
35%	10(+)	**8**(+)	10(+)	21(+)	**1**(+)	15(+)
40%	27(-)	18(+)	**6**(+)	27(-)	23(+)	22(-)
45%	**7**(+)	**2**(+)	**3**(+)	9(+)	11(+)	**5**(+)
50%	9(+)	9(+)	13(+)	**6**(-)	**7**(+)	17(+)

from it. For adding class label noise, we adopted the procedure of [58]. Specifically, given a pair of classes (X, Y) and a noise level x, an instance with its label X has an $x \cdot 100\%$ chance to be corrupted and mislabeled as Y, and so does an instance of class Y. Table XVIII shows the AUC value of RAMOBoost and other comparative learning algorithms under different class label noise levels. Tables XIX and XX show the significance test based on the averaged AUC and AUC values of ten random runs.

Attribute noise was manually added in accordance with the procedure in [59]. To corrupt each attribute A_i with a noise level $x \cdot 100\%$, the value of A_i is assigned a random value approximately $x \cdot 100\%$ of the time, with each alternative value being approximately equally likely to be selected. Table XXI shows the averaged AUC results of ten random runs. Tables XXII and XXIII present the significance test results based on the averaged AUC and the AUC of ten random runs.

All simulation results presented in this section illustrate the robustness of RAMOBoost when exposed to different internal (oversampling ratio) and external (imbalanced class ratio, noises) configurations. The significance tests also demonstrate the competitiveness of RAMOBoost against other comparative algorithms from a statistical point of view.

[1]RAMOBoost is significantly better than the comparative algorithm only if the corresponding table cell is highlighted and underscored with symbol "(+)"; symbol "(−)" represents the opposite.

V. CONCLUSION

In this paper, we presented the RAMOBoost method for imbalanced data classification problem. The key characteristics of RAMOBoost are adaptive learning and reduction of bias. This is accomplished by adaptively shifting the decision boundary toward those difficult examples in both minority and majority examples, and systematically creating minority synthetic instances based on the distribution function. Simulation results on 19 datasets across various assessment metrics, including OA, $Precision$, $Recall$, $F\text{-}measure$, $G\text{-}mean$, ROC graphs, and AUC, demonstrate the effectiveness and robustness of the proposed method.

As a new method for imbalanced learning problems, there are several interesting future research directions for RAMOBoost. For instance, our current study is focused on handling datasets with continuous features. It is possible to extend RAMOBoost to handle datasets with nominal features by adopting the techniques used in the SMOTE-N method [15].

Second, in this paper RAMOBoost is only evaluated on two-class imbalanced problems. It can be generalized to handle multiclass imbalanced learning problems to improve its applicability in practice. Third, in RAMOBoost the Euclidean distance is employed as the distance measure, however, there are other alternatives that are also eligible and worthy of trying and may show improved performance for the RAMOBoost framework. Finally, similar to many of the existing imbalanced learning algorithms, there are several parameters that need to be determined for RAMOBoost. We have shown some empirical results regarding this issue in this paper, and we also would like to note that a systematic and adaptive way to adjust those parameters could be a challenging (but important) issue for this method to be applied across different application domains. Our group is currently investigating all these issues. Motivated by our initial results in this paper, we believe that RAMOBoost may provide new insights for imbalanced learning problems and have the potential to be a powerful tool in many application domains.

REFERENCES

[1] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[2] F. Provost, "Learning with imbalanced data sets 101," in *Learning from Imbalanced Data Sets*, N. Japkowicz, Ed. Menlo Park, CA: AAAI Press, 2000.

[3] N. V. Chawla, N. Japkowicz, and A. Kołcz, "Editorial: Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 1–6, Jun. 2004.

[4] N. V. Chawla, N. Japkowicz, and A. Kołcz, "Uncertainty sampling for one-class classifiers," in *Proc. 12th Int. Conf. Mach. Learn., Workshop Learn. Imbalanced Data Sets II*, Washington D.C., Aug. 2003, pp. 81–88

[5] N. Japkowicz, "Learning from imbalanced data sets: A comparison of various strategies," in *Proc. Learn. Imbalanced Data Sets, Papers AAAI Workshop*, Menlo Park, CA, 2000, pp. 10–15.

[6] F. Provost and T. Fawcett, "Robust classification for imprecise environments," *Mach. Learn.*, vol. 42, no. 3, pp. 203–231, Mar. 2001.

[7] S. Clearwater and E. Stern, "A rule-learning program in high energy physics event classification," *Comput. Phys. Commun.*, vol. 67, no. 2, pp. 159–182, Dec. 1991.

[8] G. M. Weiss, "Mining with rarity: A unifying framework," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 7–19, Jun. 2004.

[9] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 20–29, Jun. 2004.

[10] N. V. Chawla, "C4.5 and imbalanced datasets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure," in *Proc. 12th Int. Conf. Mach. Learn., Workshop Learn. Imbalanced Data Sets II*, 2003, pp. 1–8.

[11] T. Jo and N. Japkowicz, "Class imbalances vs. small disjuncts," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 40–49, Jun. 2004.

[12] G. M. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *J. Artificial Intell. Res.*, vol. 19, no. 1, pp. 315–354, Jul. 2003.

[13] N. Japkowicz, "Class imbalances: Are we focusing on the right issue?" in *Proc. 12th Int. Conf. Mach. Learn., Workshop Learn. Imbalanced Data Sets II*, 2003, pp. 1–7.

[14] R. C. Prati, G. E. A. P. A. Batista, and M. C. Monard, "Class imbalances vs. class overlapping: An analysis of a learning system behavior," in *Proc. 3rd Mexican Int. Conf. Artificial Intell., Adv. Artificial Intell.*, 2004, pp. 312–321.

[15] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artificial Intell. Res.*, vol. 16, no. 1, pp. 321–357, Jan. 2002.

[16] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," in *Proc. Int. Conf. Intell. Comput., Adv. Intell. Comput.*, 2005, pp. 878–887.

[17] H. Guo and H. L. Viktor, "Learning from imbalanced data sets with boosting and data generation: The databoost-IM approach," *ACM SIGKDD Explorations Newslett.*, vol. 6, no. 1, pp. 30–39, Jun. 2004.

[18] D. Mease, A. J. Wyner, and A. Buja, "Boosted classification trees and class probability/quantile estimation," *J. Mach. Learn. Res.*, vol. 8, pp. 409–439, May 2007.

[19] J. Yuan, J. Li, and B. Zhang, "Learning concepts from large scale imbalanced data sets using support cluster machines," in *Proc. 14th Annu. ACM Int. Conf. Multimedia*, Santa Barbara, CA, 2006, pp. 441–450.

[20] K. M. Ting, "An instance-weighting method to induce cost-sensitive trees," *IEEE Trans. Knowl. Data Eng.*, vol. 14, no. 3, pp. 659–665, May–Jun. 2002.

[21] H. Masnadi-Shirazi and N. Vasconcelos, "Asymmetric boosting," in *Proc. Int. Conf. Mach. Learn.*, Corvallis, OR, 2007, pp. 609–619.

[22] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," in *Advances in Neural Information Processing System 14*. Cambridge, MA: MIT Press, 2001, pp. 1311–1318.

[23] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, "Adacost: Misclassification cost-sensitive boosting," in *Proc. 16th Int. Conf. Mach. Learn.*, 1999, pp. 97–105.

[24] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proc. 5th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 1999, pp. 155–164.

[25] X.-Y. Liu and Z.-H. Zhou, "Training cost-sensitive neural networks with methods addressing the class imbalance problem," *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 1, pp. 63–77, Jan. 2006.

[26] Y. H. Liu and Y. T. Chen, "Face recognition using total margin-based adaptive fuzzy support vector machines," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 178–192, Jan. 2007.

[27] G. Wu and E. Y. Chang, "KBA: Kernel boundary alignment considering imbalanced data distribution," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 786–795, Jun. 2005.

[28] G. Wu and E. Y. Chang, "Aligning boundary in kernel space for learning imbalanced dataset," in *Proc. 4th IEEE Int. Conf. Data Mining*, Brighton, U.K., 2004, pp. 265–272.

[29] X. Hong, S. Chen, and C. J. Harris, "A kernel-based two-class classifier for imbalanced data sets," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 28–41, Jan. 2007.

[30] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: Active learning in imbalanced data classification," in *Proc. 16th ACM Conf. Inform. Knowl. Manage.*, Lisbon, Portugal, 2007, pp. 127–136.

[31] S. Ertekin, J. Huang, and L. Giles, "Active learning for class imbalance problem," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Develop. Inform. Retrieval*, Amsterdam, The Netherlands, 2007, pp. 823–824.

[32] J. Zhu and E. Hovy, "Active learning for word sense disambiguation with methods for addressing the class imbalance problem," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Computat. Natural Lang. Learn.*, Prague, Czech Republic, 2007, pp. 783–790.

[33] H. He, Y. Bai, E. A. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. Int. Joint Conf. Neural Netw.*, Jun. 2008, pp. 1322–1328.

[34] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTEboost: Improving prediction of the minority class in boosting," in *Proc. Principles Knowl. Discovery Databases*, Cavtat-Dubrovnik, Croatia, 2003, pp. 107–119.

[35] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, 1996, pp. 148–156.

[36] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, Aug. 1997.

[37] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: One-sided selection," in *Proc. 14th Int. Conf. Mach. Learn.*, Nashville, TN, 1997, pp. 179–186.

[38] J. C. F. Caballero, F. J. Martinze, C. Hervas, and P. A. Gutierrez, "Sensitivity vs. accuracy in multiclass problems using memetic Pareto evolutionary neural networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 750–770, May 2010.

[39] N. Garcia-Pedrajas, "Constructing ensembles of classifiers by means of weighted instance selection," *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 258–277, Feb. 2009.

[40] M. D. Muhlbaier, A. Topalis, and R. Polikar, "Learn^{++}.NC: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 152–168, Jan. 2009.

[41] C. Shen and H. Li, "Boosting through optimization of margin distributions," *IEEE Trans. Neural Netw.*, vol. 21, no. 4, pp. 659–666, Apr. 2010.
[42] P. Sun and X. Yao, "Sparse approximation through boosting for learning large scale kernel machines," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 883–894, Jun. 2010.
[43] W. Hu, W. Hu, and S. Maybank, "Adaboost-based algorithm for network intrusion detection," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 38, no. 2, pp. 577–583, Apr. 2008.
[44] H. He and X. Shen, "A ranked subspace learning method for gene expression data classification," in *Proc. Int. Conf. Artificial Intell.*, 2007, pp. 358–364.
[45] A. Asuncion and D. J. Newman. *UCI Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml/datasets.html
[46] *Elena Project* [Online]. Available FTP: ftp.dice.ucl.ac.be/pub/neuralnets/ELENA/databases
[47] T. Fawcett, "ROC graphs: Notes and practical considerations for data mining researchers," HP Lab., Palo Alto, CA, Tech. Rep. HPL-2003-4, 2003.
[48] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Mach. Learn.*, vol. 30, nos. 2–3, pp. 195–215, Feb.–Mar. 1998.
[49] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *Proc. 20th Int. Conf. Mach. Learn., Workshop Learn. Imbalanced Data Sets II*, Washington D.C., 2003, pp. 1–8.
[50] F. Provost and T. Fawcett, "Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions," in *Proc. 3rd Int. Conf. Knowl. Discovery Data Mining*, Newport Beach, CA, 1997, pp. 43–48.
[51] D. J. Hand, "Measuring classifier performance: A coherent alternative to the area under the ROC curve," *Mach. Learn.*, vol. 77, no. 1, pp. 103–123, Oct. 2009.
[52] G. W. Corder and D. I. Foreman, *Nonparametric Statistics for Non-Statisticians: A Step-by-Step Approach*. New York: Wiley, 2009.
[53] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, no. 7, pp. 1–30, Dec. 2006.
[54] *Critical Value Table of Wilcoxon Signed-Ranks Test* [Online]. Available: http://www.euronet.nl/users/warnar/demostatistiek/tables/WILCOXON-TABEL.htm
[55] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artificial Intell. Res.*, vol. 11, pp. 169–198, Aug. 1999.
[56] L. Breiman, "Arcing classifiers," *Ann. Statist.*, vol. 26, no. 3, pp. 801–824, 1998.
[57] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, "Automatically countering imbalance and its empirical relationship to cost," *Data Mining Knowl. Discovery*, vol. 17, no. 2, pp. 225–252, Oct. 2008.
[58] D. Anyfantis, M. Karagiannopoulos, S. Kotsiantis, and P. Pintelas, "Robustness of learning techniques in handling class noise in imbalanced datasets," in *Proc. IFIP Int. Federation Inform. Process.*, vol. 247. 2007, pp. 21–28.
[59] X. Zhu, X. Wu, and Y. Yang, "Error detection and impact-sensitive instance ranking in noisy datasets," in *American Association for Artificial Intelligence*. Cambridge, MA: MIT Press, 2004, pp. 378–383.

Sheng Chen (S'06) received the B.S. and M.S. degrees in control science and engineering from Huazhong University of Science and Technology, Wuhan, China, in 2004 and 2007, respectively. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ.

His current research interests include machine learning, data mining, and computational intelligent systems.

Haibo He (M'06) received the B.S. and M.S. degrees in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 1999 and 2002, respectively, and the Ph.D. degree in electrical engineering from Ohio University, Athens, in 2006.

He is currently an Assistant Professor at the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston. From 2006 to 2009, he was an Assistant Professor at the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ. His current research interests include self-adaptive systems, machine learning, data mining, computational intelligence, and applications in critical engineering fields such as smart grid and sensor networks, very large scale integration and field-programmable gate array design.

Dr. He has served regularly on the organizing committees of numerous international conferences, and also served as a Guest Editor for several journals, including *Applied Mathematics and Computation*, *Soft Computing*, and *Journal of Experimental & Theoretical Artificial Intelligence*. He is currently the Editor of the IEEE Computational Intelligence Society Electronic Letter, an Editorial Board Member of *Cognitive Computation*, and an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS and IEEE TRANSACTIONS ON SMART GRID.

Edwardo A. Garcia received the B.S. degree in mathematics from New York University, New York, and the B.E. degree in computer engineering from Stevens Institute of Technology, Hoboken, NJ, both in 2008.

He is currently with the Stevens Institute of Technology. His current research interests include machine learning, biologically inspired intelligence, cognitive neuroscience, data mining for medical diagnostics, and mathematical methods for functional-magnetic resonance imaging.