Dynamic Sampling Approach to Training Neural Networks for Multiclass Imbalance Classification

Minlong Lin, Ke Tang, *Senior Member, IEEE*, and Xin Yao, *Fellow, IEEE*

***Abstract*— Class imbalance learning tackles supervised learning problems where some classes have significantly more examples than others. Most of the existing research focused only on binary-class cases. In this paper, we study multiclass imbalance problems and propose a dynamic sampling method (DyS) for multilayer perceptrons (MLP). In DyS, for each epoch of the training process, every example is fed to the current MLP and then the probability of it being selected for training the MLP is estimated. DyS dynamically selects informative data to train the MLP. In order to evaluate DyS and understand its strength and weakness, comprehensive experimental studies have been carried out. Results on 20 multiclass imbalanced data sets show that DyS can outperform the compared methods, including pre-sample methods, active learning methods, cost-sensitive methods, and boosting-type methods.**

***Index Terms*— Cost-sensitive learning, dynamic sampling, multiclass imbalance learning, multilayer perceptrons.**

I. Introduction

IN MANY classification problems, the class distribution is imbalanced, i.e., there are many more examples of some classes than others. Imbalanced class distribution may occur in many domains, such as network intrusion detection [1], financial engineering [2], and medical diagnostics [3]. In such cases, the classifiers tend to produce high classification accuracies on the majority classes but poor classification accuracies on the minority ones.

Class imbalance problem has been studied by many researchers [4]–[6]. So far, pre-sampling methods appear to be the most commonly used approach for this kind of problems. Generally speaking, a pre-sampling method makes the training set balanced, either by oversampling the minority class or by undersampling the majority class. To be more specific, random oversampling (ROS) method randomly duplicates the examples of the minority class. However, ROS may also introduce redundancy and cause over-fitting [7]. Instead of randomly duplicating examples, synthetic minority oversampling technique (SMOTE) [8] and its improved versions, such as Borderline-SMOTE [9] and adaptive synthetic sampling (Adasyn) [10], generate synthetic examples for the minority class. In this way, the decision region of the minority class becomes more general [8]. On the contrary, undersampling methods, such as random undersampling (RUS) and one-sided selection (OSS) [11] prune the examples of the majority class. One main drawback of this type of methods is that the information contained in the pruned examples might be lost and thus deteriorate the performance of classifier [7].

Besides pre-sampling methods, cost-sensitive methods [12] are also regarded as important approaches to class imbalance problems. The main idea is that, to avoid the minority class being overlooked, a higher misclassification cost should be assigned to it than to the majority class. In this way, a class imbalance problem can be formulated as a cost sensitive learning problem and solved by an existing method [13].

In the above mentioned methods, both the sampling process and assigning costs to different classes are usually conducted before and independent of the training process. After such a step, the training set (or the misclassification costs) remains unchanged during the training process. Such a pretreatment procedure may lead to a few drawbacks. First, the redundant examples due to oversampling will waste the training time or cause over-fitting. Second, any lost information due to undersampling will never be recovered and learned. Finally, since it is difficult to set a proper cost matrix for a class imbalance problem [14], an inappropriate cost matrix may mislead the training process. To avoid these drawbacks, it is desirable to integrate the pretreatment process (i.e., sampling process or cost matrix setting) and training process. Boosting and active learning methods, though not directly motivated by class imbalance problems, actually integrate the sampling and training procedure. Hence, both they have been extended to tackle class imbalance problems. Boosting-type methods, such as SMOTEBoost [15], DataBoost-IM [16], and RAMOBoost [17], are based on boosting and data generation. All these methods add new examples to the training set and hence can be viewed as oversampling methods. On the other hand, active learning methods for class imbalance problems, such as SVM-based active learning [18], [19], select the examples from the decision border to train the learner.

Manuscript received February 16, 2012; revised November 3, 2012; accepted November 4, 2012. Date of publication February 4, 2013; date of current version February 13, 2013. This work was supported in part by the 973 Program of China under Grant 2011CB707006, the National Natural Science Foundation of China under Grant 61175065, Grant U0835002, and Grant 61028009, the National Natural Science Foundation of Anhui Province under Grant 1108085J16, and the European Union Seventh Framework Programme under Grant 247619 and Grant 270428.

M. Lin and K. Tang are with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: sunnyboy@mail.ustc.edu.cn; ketang@ustc.edu.cn).

X. Yao is with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China, and also with CERCIA, School of Computer Science, University of Birmingham, Edgbaston, B15 2TT, U.K. (e-mail: x.yao@cs.bham.ac.uk).

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/TNNLS.2012.2228231

Hence, this type of methods can be viewed as undersampling methods. In the literature, empirical studies have shown the advantages of boosting-type and active learning methods over the pretreatment methods [17], [18], and hence support the (natural) consideration that sampling should be integrated with the training process.

Most of the above methods were mainly developed for binary-class problems, while their efficacy on multiclass problems have not been well investigated. On one hand, it is nontrivial to extend boosting-type and active learning methods to multiclass problems. On the other hand, sampling and cost sensitive methods are readily applicable for both binary-class and multiclass problems [14]. However, it is not easy to pre-define the sampling ratio or misclassification cost for each class. In fact, empirical study has shown that most sampling methods are ineffective on multiclass problems and often cause negative effect on the problems with large number of classes [13]. Besides, although progress has been made on multiclass classification problem in some recent studies [20], [21], the class imbalance issue was not addressed there. Therefore, it is desirable to design a method, which can directly address multiclass imbalance problems.

Motivated by the above reasons, this paper presents a method with the following two characteristics: it should integrate sampling and training processes, and it should be able to directly deal with multiclass problems. Multilayer perceptrons (MLPs) have strong abilities to learn complex classification boundary and can be directly used for multiclass problems. By using a sequential training model, the sampling process can be entirely integrated to the training process. Therefore, MLP is adopted as our basic model and a data selection method is developed for it to deal with multiclass imbalance problem. To be more specific, a novel heuristic is proposed to determine whether a training example should be learned (i.e., be used to update the weights of an MLP) in a training epoch. The heuristic makes the decision based on the current status of the MLP, and thus does not solely rely on the initial setting of sampling ratio. Furthermore, since the status of MLP changes as the learning progresses, an example not learned in one epoch may still be beneficial in a later epoch. Making use of the sequential training mode, no example will be discarded throughout the whole training procedure, and the information loss caused by a pre-sampling procedure can be avoided. Since such a scheme can be viewed as dynamically choosing training examples for each epoch, the proposed method is named dynamic sampling (DyS) method for MLPs. Extensive empirical studies on 20 multiclass imbalanced data sets from the UCI machine learning repository [22] have shown the advantages of our approach over the existing methods.

The rest of this paper is organized as follows. In Section II, we will briefly review some pre-sampling methods, cost-sensitive methods, boosting-type methods, and active learning methods for class imbalance learning. Then, we will describe our method, i.e., DyS, in Section III. The experimental study will be presented in Section IV. Finally, in Section V, we conclude this paper and discuss future work.

II. Related Work

A. Pre-Sampling Methods

Pre-sampling methods focus on dealing with the training set before training. The aim is to make the training set balanced or less imbalanced by adding examples to or removing examples from the training set before training the classifier.

ROS and RUS are two basic sampling methods [6], [23] in which examples are randomly duplicated ROS or removed RUS to make the training set balanced. ROS and RUS can also be employed to a training set simultaneously. For multiclass problems, ROS and RUS can be simply implemented [24]. In ROS, all of the classes except the largest one will be duplicated to make the number of examples in every class the same as that of the largest class. In RUS, all of the classes except the smallest one will be pruned to make the number of examples in every class the same as that of the smallest class.

ROS and RUS do not consider the relationship between examples. Therefore, data redundancy or information loss may easily occur [7]. Various sampling methods with some heuristic techniques have been proposed. SMOTE [8] is an oversampling method, which generates synthetic data for the minority class. It can provide more related minority class examples and make the decision regions larger. However, SMOTE generated the same number of synthetic examples for each minority example and this strategy may cause data overlapping [7]. Some methods, which can overcome this limitation of SMOTE have been proposed, such as borderline-SMOTE [9] and Adasyn [10]. borderline-SMOTE only oversamples the borderline examples of the minority class. Adasyn adapts the number of synthetic examples for every minority example according to the distributions.

The oversampling methods try to overcome the property of imbalanced class distribution by adding examples to the training set. However, the duplicating or generating of examples may make the training set noisier and cause over-fitting. Furthermore, adding training examples will also increase the training time. To overcome these drawbacks, data cleaning techniques were proposed. Tomek link [25] is a useful definition for cleaning data. It can be used to clean up data after an oversampling method, such as the ROS, SMOTE, and Adasyn. Tomek link has also been employed in undersampling methods, such as OSS [11], which is based on the 1-NN classification method and Tomek links. In OSS, all the minority class examples and the misclassified (by 1-NN) majority class examples are selected, and then the majority class examples that participate in Tomek links are removed.

B. Cost-Sensitive Methods

Cost-sensitive methods for class imbalance problems always follow two steps [23]: set a cost matrix for the class imbalance problem to formulate the problem as a cost-sensitive problem, and then, employ a method to solve the cost-sensitive problem.

Many cost-sensitive methods have been proposed, such as cost-sensitive boosting [14], [26], meta cost [27], rescaling training data [12], [28], and modified back propagation (BP) algorithm for MLPs [29].

Cost-sensitive boosting is a modification of AdaBoost [30]. The misclassification cost is introduced into AdaBoost for updating the distribution in each iteration. MetaCost is based on the bagging algorithm [31] and example relabeling. Zhou and Liu [28] proposed a rescaling method named for multiclass cost-sensitive learning. Kukar and Kononenko [29] proposed a cost-sensitive method based on MLPs by introducing a cost-sensitive modification to the BP algorithm.

Although the studies on cost-sensitive problems considered both binary-class and multiclass cases, there are only a few methods presented for setting a proper cost matrix for a class imbalance problem. Japkowicz and Stephen [5] modified the misclassification cost to compensate for the imbalance ratio of the classes in their experiments on binary-class problems. Sun *et al.* [26] presented a method to test different misclassification cost ratios of the majority class to the minority class. It was an enumeration method and time consuming. In another research [14], which addressed the multiclass cases, Sun *et al.* introduced genetic algorithm (GA) [32] to search for misclassification costs of different classes. As stated in [14], it was also time consuming. The difficulty of setting a cost matrix for a class imbalance problem has made cost-sensitive methods not as practical as one would hope for class imbalance problems.

C. Other Methods

In the above reviewed methods, sampling process or setting cost matrix is executed independently before the training process, and the pre treatments are not changed during the training process. Another type of methods, which integrates sampling and training processes, has been proposed.

Boosting-type methods, including SMOTEBoost [15], DataBoost-IM [16], and RAMOBoost [17], borrow the idea of boosting [30], [33], and dynamically change the training data set during the training process. In SMOTEBoost, SMOTE is used to generate synthetic examples from every minority example before training every weak learner. In DataBoost-IM, the difficult-to-learn examples (no matter majority or minority class) are considered for generating synthetic examples before training every weak learner. In RAMOBoost, only the difficult minority examples are considered for generating synthetic examples before training every weak learner. With the strategy of boosting, this kind of methods can select training examples at the training stage. Besides, as discussed in [34], diversity in ensemble can have positive impact on class imbalance learning. Therefore, as ensemble-based methods, these methods have shown good performance in class imbalance learning. However, ensemble-based methods are usually more complex and time consuming than a single classifier.

Active learning has also been employed to design methods for class imbalance problems. Active learning was first proposed for addressing the problem that randomly selecting training examples may lead to large and highly redundant training sets [35]. Generally speaking, active learning makes the learning process more effective by querying for training data. The goals of querying include: to obtain as informative as possible examples, to remove redundant examples, and to achieve better generalization ability using fewer training data. Ertekin *et al.* [18], [19] proposed an SVM-based active learning method for class imbalance problems. They proposed an efficient way to select informative examples from a smaller pool of examples. The informative examples are the ones, which are close to the hyperplane.

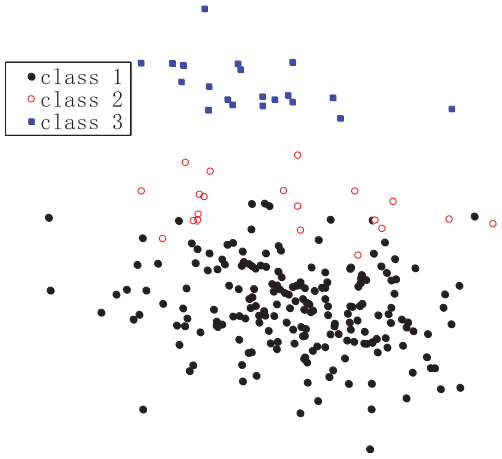


Fig. 1. Example of a three-class problem. Classes 2 and 3 are both minority classes. However, class 2 is close to class 1 while class 3 is distant from class 1. Therefore, a good sampling technique should be capable of focusing more on class 2 in this situation.

Fernández-Navarro *et al.* [36] proposed an over-sampling method, namely the dynamic SMOTE radial basis function (DSRBF), to dynamically generate examples during the training process. In DSRBF, a memetic algorithm was proposed to optimize the radial basis functions neural networks (RBFNN). In different generations of the memetic algorithm, new examples of the classes with the worst classification accuracy were generated.

Bhowan *et al.* proposed an ensemble-based method using genetic programming for class imbalance problems [37]. In [37], the accuracies of majority and minority classes were viewed as two learning objects and a multiobjective genetic programming approach was proposed to maximize both majority and minority accuracies.

III. Dynamic Sampling

To achieve good performance for class imbalance problems while maintain simplicity, two issues need to be considered, i.e., the class distribution and the difficulty of correctly classifying an example. When considering the class distribution, the classes with fewer examples should be emphasized more. When considering the second issue, the examples that are more difficult to be correctly classified should be emphasized more. Fig. 1 demonstrates such a scenario, where classes 2 and 3 are the minority classes. In comparison, it is more difficult to distinguish class 2 from class 1, and thus a good sampling technique should be capable of focusing more on class 2. Although some existing sampling techniques, such as borderline-SMOTE and Tomek Link method, can be viewed as attempting to identify difficult examples, it is interesting to note that a previous comprehensive study revealed that these methods are actually not as effective as the very simple

oversampling approach even on binary-class problems [6]. Moreover, the existing sampling methods are predominately used independently of learning algorithms. In this manner, the information loss or redundancy caused by sampling techniques may lead to a classifier with unsatisfactory performance.

A. Main Idea

Our basic idea is to design a simple DyS that dynamically selects examples for training during the training process. We do not pre-delete any example to prevent information loss, and we dynamically select examples for training to avoid redundant information and to make the best use of the training data. Specifically, in the sequential mode for training MLP, the training examples are used to update the MLP one by one. For a training example (x_i, t_i), where x_i is the input, t_i is the target, and the output from the MLP is denoted as $y_i = f(x_i)$, the MLP is updated by

$$f_{k+1} \leftarrow T(f_k, x_i, t_i) \tag{1}$$

where T is the updating process, which modifies the connection weights of the MLP. In the normal training mode, all of the examples in the training set will pass though the MLP and be used to update the MLP by (1). The stage that every training example passes through the MLP is called an epoch. The following may occur in this kind of training mode.

1) If the data set is imbalanced, f may be biased toward the majority classes and ignore the minority classes.
2) The examples that make little progress of f will waste the training time.
3) If the data set is redundant, over learning may occur and it will decrease the generalization performance.

Therefore, not every example that passes through the MLP should be used to update the MLP. To this end, a filter process is considered. During the training process, a probability will be estimated for every example and then the probability will be used to decide whether the current example be used to update the MLP. According to this main idea, the general steps in each epoch can be described as follows.

Step1: Randomly fetch an example x from the training set.
Step2: Estimate the probability p that the example should be used for training.
Step3: Generate a uniform random real number μ between 0 and 1.
Step4: If $\mu < p$, then use x to update the MLP by (1).
Step5: Repeat steps 1–4 until there is no example in the training set.

The main issue of DyS is the estimation of the probability p in step 2 and the detail will be described next.

B. Algorithm

In a problem with classes, we set output nodes for the MLP and for an example belonging to class c, we set the target output of the example as $t = \{t_i | t_c = 1,\ t_{j|j\neq c} = 0\}$. The real output of the example is denoted as $y = \{y_i | i = 1, 2, \ldots, m\}$ so the node with the highest output designates the class. Both the hidden node function and the output node function are set as the logistic function $\varphi(x) = 1/(1 + \exp(-x))$ so that $y_i \in (0, 1)$.

For an example belonging to class c, the confidence of the current MLP to correctly classify the example can be defined as

$$\delta = y_c - \max_{i\neq c}\{y_i\}. \tag{2}$$

$\delta < 0$ indicates the example is misclassified by the current MLP. A larger value of δ indicates the example has been learned better and the necessity of updating the MLP by the example is lower. Therefore, p should be negatively correlated to δ.

In addition to the confidence, the imbalance should be considered. The example belonging to a minority class should have higher probability to be used to update the MLP than that of a majority class. Let $r_c = n_c/n$ be the ratio of class c, where n_c is the number of examples belonging to class c and n is the number of all the examples. For concerning more minority class examples, p should be negatively correlated to r_c. To this end, the confidence is redefined as

$$\Delta = \delta \cdot r_c / z \tag{3}$$

where z is a normalization factor and we set $z = \min_i\{r_i\}$ so that the margin for the class with the lowest number of examples is equal to (2).

$\Delta < 0$ indicates that the example is misclassified and the example need to be learned. So $p = 1$ if $\Delta < 0$. Otherwise, $\Delta \in [0, \max_j\{r_j\}/\min_j\{r_j\}]$, and the probability should be negatively correlated to Δ, so the probability can be heuristically estimated as $p = \exp\{-\Delta\}$. Therefore, the probability that an example belonging to class c will be used to update the MLP is estimated as

$$p = \begin{cases} 1, & \text{if } \delta \leq 0 \\ \exp\left(-\delta \cdot r_c / \min_i\{r_i\}\right), & \text{otherwise.} \end{cases} \tag{4}$$

In the selection mechanism by p.

1) The examples that are misclassified will be selected to update the MLP.
2) For the examples that are correctly classified, the examples of minority classes are emphasized more than those of majority classes.
3) For the examples that are correctly classified, the examples which are classified more ambiguous [i.e., the value of δ in (2) is smaller] are emphasized more.

The selection mechanism can allay the affection of class imbalance and pay more attentions to the examples that are difficult to classify. However, at the very beginning of training process (e.g., the MLP is just randomly initialized), it is usually the case that the MLP misclassifies abundant of examples from both majority and minority classes. According to the first condition of (4), the MLP will be updated by more examples from majority classes than by those from minority classes. That means, the selection mechanism may still bias toward majority classes in this situation.

To avoid the bias, an oversampling process is implemented at the beginning of every epoch. At the beginning of the first epoch, the examples of all classes except the largest classes are

duplicated to make the data set balanced, i.e., the number of examples in every class is equal to that in the largest class. As the training process goes on, the duplicate ratio is attenuated. Denote the duplicate ratio for class c at the first epoch as α_c, the duplicate ratio for class is heuristically attenuated to $\alpha_c/\text{In } ep$ at the end of the epth epoch, where $ep > 2$.

The pseudocode of DyS is presented in Algorithm 1. In the pseudocode, S denotes the training set, epoch denotes the number of epochs, m denotes the number of classes in S, n denotes the number of examples in S and $r_c = n_c/n$, where n_c is the number of examples belonging to class c. Besides, the following defines some functions used in the pseudocode.

Duplicate (S, $\boldsymbol{\rho}$): Randomly duplicate the examples in S by the ratio $\boldsymbol{\rho} = \{\rho_1, \rho_2, \ldots, \rho_m$, the number of duplicated examples for class c is $\lfloor n_c \times \rho_c \rfloor$. Return the balanced training set.

MLP (x): Return the output of x from the MLP. The output is denoted as $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$.

Update MLP ($\boldsymbol{x}, \boldsymbol{t}$): Use ($\boldsymbol{x}, \boldsymbol{t}$) to update the MLP.

Uniform (): Return a uniform random real number between 0 and 1.

Unlike the pre-sampling and cost-sensitive methods, DyS integrates the sampling and training processes as a whole. Different from boosting-type methods, which sequentially train a number of base classifiers, DyS continuously updates the weights of a single MLP. In comparison to active learning methods, DyS makes use of the label and all the features of an example to determine whether the example should be learned, while the label information is not used in a standard active learning setting. In comparison with DSRBF, both DyS and DSRBF integrate the sampling and the training processes. The difference is, in DyS, all of the examples (regardless of whether in or not in the same class) are differently treated when resampling the examples, while in DSRBF, the examples of the same class are equally treated when resampling the examples.

IV. Experimental Study

To evaluate the performance of DyS, comprehensive experiments on 20 multiclass imbalanced data sets have been conducted. The aim of these experiments is to verify whether DyS really offers some advantages in terms of overall performance and its effect to the learning behavior. Following this consideration, DyS was first compared with a number of state-of-the-art approaches. After that, the impact of DyS on the training process is analyzed. In addition, the detailed performance on each class is also presented to further demonstrate the efficacy of DyS.

A. Compared Methods

First of all, DyS was compared with some pre-sampling methods. Since a recent study [6] has shown that more elaborated methods do not seem to perform any better than very simple methods, such as ROS and RUS, ROS and RUS were used for our comparison. ROS and RUS can also be employed simultaneously to make the data set balanced and the number of examples was the same as the original training set. This method was also employed in our experiments and will be referred to as random fixed-size sampling (RFS) hereafter. Besides, no-sampling (NoS), in which the original training set without any pre-sampling process was used to provide a baseline for our comparisons.

In addition to the pre-sampling methods, cost-sensitive learning provides another approach to class imbalance problems [14], [28]. Besides, the idea of DyS is closely related to that of active learning, boosting-type algorithms, and DSRBF [36]. Hence, DyS is also compared to these approaches in our experiments. Specifically, since there is no existing active learning method for training MLP in class imbalance situation, we designed an MLP-based active learning method for this purpose. For comparison with cost-sensitive learning, three representative cost-sensitive learning methods, namely minimization of misclassification costs (MMC) [29], $\text{Rescale}_{\text{new}}$ [28], and AdaC2.M1 [14] were chosen. Then, as a recently boosting-type method for class imbalance problems, RAMOBoost was generalized to multiclass problems and included in the comparison. Finally, as a method, which was specifically targeting at multiclass imbalance problems, DSRBF was also implemented by replacing RBFNNs with MLPs as its basic classifier for the comparison.

B. Metrics

In class imbalance problems, the overall classification accuracy is not a good metric for measuring the performance of classifiers. Take a binary-class case with the class distribution of 1:99 as an example, if a classifier classifies all the examples to the majority class, the classification accuracy will be very high (99%). However, this classifier is not useful in practice.

Instead of the classification accuracy, the area under the ROC curve (AUC) [38] has been widely used to evaluate the performance of a classifier [39]. Originally, AUC is only applicable to binary-class cases. Hand and till extended AUC to multiclass problems and proposed a measure, which they called M for multiclass classification problems [40]. This measure is referred as MAUC in our paper. Since MAUC inherits many merits of AUC, it has been adopted frequently when addressing classification problems with multiple classes [41]. We will also use MAUC in our study, which facilitates comparisons with relevant work as well. Furthermore, to evaluate the classification performance in detail, the geometric means of the classification accuracy of every class (G-mean) will be employed as another performance metric in our experimental study. G-mean is defined as [14]

$$G\text{-mean} = \left(\Pi_{i=1}^{m} \frac{tr_i}{n_i}\right)^{\frac{1}{m}} \tag{5}$$

where m is the number of classes, n_i is the number of examples in class i, and tr_i is the number of correctly classified examples in class i.

C. Experimental Setup

The proposed algorithm and others for comparison have been evaluated on 20 multiclass imbalanced data sets from

TABLE I
BASIC CHARACTERISTICS OF THE DATA SETS (#F: THE NUMBER OF FEATURES, #C: THE NUMBER OF CLASSES)

Name	#F	#C	Class Distribution
Abalone	8	18	15: 57: 115: 259: 391: 568: 689: 634: 487: 267: 203: 126: 103: 67: 58: 42: 32: 26
Arrhythmia	259	7	245: 44: 15: 15: 25: 50: 22
Balance-scale	4	3	49: 288: 288
Car	6	4	1210: 384: 65: 69
Chess	6	18	2796: 27: 78: 246: 81: 198:471: 592: 683: 1433: 1712: 1985: 2854: 3597: 4194: 4553: 2166: 390
Contraceptive	9	3	629: 333: 511
Dermatology	34	6	112: 61: 72: 49: 52: 20
Ecoli	6	5	143: 77: 35: 20: 52
Glass	9	4	70: 76: 17: 29
New-thyroid	5	3	150: 35: 30
Nursery	8	4	4266: 4320: 328: 4044
Page-blocks	10	5	4913: 329: 28: 88: 115
Satellite	36	6	1533: 703: 1358: 626: 707: 1508
Soybean	35	17	20: 20: 20: 88: 44: 20: 20: 92: 20: 20: 20: 44: 20: 91: 91: 15: 16
Splice	60	3	767: 768: 1655
Thyroid-allhypo	27	3	3481: 194: 95
Thyroid-allrep	27	4	3648: 38: 52: 34
Thyroid-ann	21	3	166: 368: 6666
Letter-1	16	26	10: 766: 736: 805: 768: 775: 773: 734: 755: 747: 739: 761: 792: 783: 753: 803: 783: 758: 748: 796: 813: 764: 752: 787: 786: 734
Letter-2	16	26	10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 10: 734

TABLE II
PARAMETERS FOR MLP (#HID.: THE NUMBER OF HIDDEN NODES, # EP.: THE NUMBER OF EPOCHS)

Data Set	#Hid.	#Ep.
Abalone	20	500
Arrhythmia	5	100
Balance-scale	15	500
Car	20	200
Chess	20	200
Contraceptive	15	200
Dermatology	2	1000
Ecoli	5	200
Glass	10	2000
New-thyroid	4	200
Nursery	20	100
Page-blocks	20	100
Satellite	15	100
Soybean	10	100
Splice	5	100
Thyroid-allhypo	10	200
Thyroid-allrep	10	100
Thyroid-ann	20	100
Letter-1	10	1000
Letter-2	10	1000

the UCI machine learning repository [22]. The basic characteristics of the data sets are presented in Table I, including the number of features (#F), the number of classes (#C), and the class distribution. In the first 18 datasets, the numbers of classes are not very large. However, it will be interesting to see the performance of DyS on data sets with a great many classes. Since there is no publicly available imbalanced dataset with a great many classes, a multiclass (26 classes) dataset with balanced class distribution, i.e., letter-recognition, was used to form two imbalanced data sets by randomly removing examples of some classes. The basic characteristics of the two data sets are also presented in Table I (*Letter-1* and *Letter-2*).

BP was employed for training MLP. Table II presents the parameters for the MLP in our experimental studies for every method, including the number of hidden nodes (#Hid.) and the number of training epochs (#Ep.). Since the emphasis of DyS is on the sampling strategy during training, we did not pay much attention to optimizing the parameters for the MLP and the same parameters were used for all the methods. The parameters are set heuristically by experience. The learning rate for the MLP was set to 0.1. An independent five-fold cross validation was implemented using the BP algorithm without any sampling method to adapt the other parameters and achieve a general setting for those parameters. They have not been optimized. During the training process, effective training means that the connection weights of MLP were updated. In order to make a fair comparison, we made the number of effective trainings the same for every method. In particular, the number of effective trainings for every method was #EP. $\times$ n, where n was the number of examples in the training set.

Twenty times five-fold stratified cross validation was executed for all the methods except AdaC2.M1, where the results from [14] were used directly. The data partitions of different methods in every cross validation were kept the same. It means that each single execution of all of the methods is based on the same training-testing data sets. With this kind of setting, Wilcoxon signed-rank test [42] can be employed for the comparison on every data set.

D. Experimental Results

1) Comparison With Basic Pre-Sampling Methods: The average and standard deviation of the MAUC and G-mean values are presented in Table III. Wilcoxon signed-rank test with the level of significance $\alpha = 0.05$ was employed for the comparison. In the columns for other methods, the values with underline (or bold) denote that DyS performed significantly better (or worse) than them on those data sets, and the values with normal type denote that there are no significant differences. The last row of each table presents the number of "win-draw-lose" of DyS versus others over all data sets.

We can observe from Table III(a) that when comparing by the values of MAUC, DyS significantly outperforms other methods on most data sets. On a few data sets, there is no

TABLE III

AVERAGE AND STANDARD DEVIATION OF MAUC AND G-MEAN VALUES OF DYS AND OTHER PRE-SAMPLING METHODS BASED ON 20 TIMES FIVE-FOLD CROSS-VALIDATION. WILCOXON SIGNED RANK TEST WITH THE LEVEL OF SIGNIFICANCE WAS EMPLOYED FOR THE COMPARISON. IN THE COLUMNS FOR OTHER METHODS, THE VALUES WITH UNDERLINE (OR BOLD) DENOTE THAT DYS PERFORMED SIGNIFICANTLY BETTER (OR WORSE) THAN THEM ON THOSE DATA SETS AND THE VALUES WITH NORMAL TYPE DENOTE THAT THERE ARE NO SIGNIFICANT DIFFERENCES. THE LAST LINE OF EACH TABLE PRESENTS THE NUMBER OF "WIN-DRAW-LOSE" OF DYS VERSUS OTHERS OVER ALL DATA SETS

(a) MAUC

Data Set	DyS	ROS	RUS	RFS	NoS
Abalone	0.8135 ± 0.0057	0.8063 ± 0.0092	0.7424 ± 0.0158	0.8053 ± 0.0084	0.8037 ± 0.0050
Arrhythmia	0.8395 ± 0.0308	**0.8469 ± 0.0308**	0.8175 ± 0.0390	0.8395 ± 0.0283	0.6901 ± 0.0249
Balance-scale	0.9742 ± 0.0114	0.9699 ± 0.0151	0.9653 ± 0.0154	0.9711 ± 0.0147	0.9265 ± 0.0276
Car	0.9890 ± 0.0062	0.9656 ± 0.0122	0.9471 ± 0.0180	0.9669 ± 0.0141	0.9824 ± 0.0063
Chess	0.9052 ± 0.0079	0.8799 ± 0.0095	0.7682 ± 0.0155	0.8798 ± 0.0099	0.8316 ± 0.0164
Contraceptive	0.7268 ± 0.0202	**0.7294 ± 0.0212**	0.7231 ± 0.0208	0.7281 ± 0.0218	**0.7290 ± 0.0212**
Dermatology	0.9828 ± 0.0150	0.9701 ± 0.0245	0.9582 ± 0.0215	0.9672 ± 0.0263	0.9175 ± 0.0149
Ecoli	0.9598 ± 0.0157	0.9578 ± 0.0175	0.9534 ± 0.0180	0.9575 ± 0.0183	0.9063 ± 0.0229
Glass	0.8726 ± 0.0463	0.8785 ± 0.0474	0.8354 ± 0.0521	0.8667 ± 0.0479	0.8201 ± 0.0404
New-thyroid	0.9985 ± 0.0027	0.9985 ± 0.0032	0.9980 ± 0.0043	0.9986 ± 0.0028	0.9813 ± 0.0201
Nursery	0.9973 ± 0.0017	**0.9988 ± 0.0005**	0.9929 ± 0.0021	**0.9988 ± 0.0005**	0.9204 ± 0.0019
Page-blocks	0.9717 ± 0.0081	**0.9809 ± 0.0059**	0.9639 ± 0.0129	**0.9808 ± 0.0057**	0.8306 ± 0.0256
Satellite	0.9829 ± 0.0020	0.9806 ± 0.0021	0.9801 ± 0.0021	0.9805 ± 0.0024	0.9797 ± 0.0025
Soybean	0.9979 ± 0.0013	0.9948 ± 0.0039	0.9906 ± 0.0052	0.9942 ± 0.0026	0.9362 ± 0.0309
Splice	0.9472 ± 0.0095	0.9437 ± 0.0080	0.9371 ± 0.0074	0.9386 ± 0.0083	0.9408 ± 0.0086
Thyroid-allhypo	0.9770 ± 0.0097	0.9780 ± 0.0102	0.9482 ± 0.0160	0.9778 ± 0.0099	0.8049 ± 0.0178
Thyroid-allrep	0.9219 ± 0.0246	0.9191 ± 0.0296	0.9123 ± 0.0285	0.9185 ± 0.0287	0.7359 ± 0.0053
Thyroid-ann	0.9822 ± 0.0065	**0.9873 ± 0.0047**	0.9764 ± 0.0081	**0.9872 ± 0.0049**	0.7907 ± 0.0150
Letter-1	0.9632 ± 0.0042	0.9262 ± 0.0067	0.8878 ± 0.0108	0.9264 ± 0.0071	0.9045 ± 0.0126
Letter-2	0.9083 ± 0.0225	0.9047 ± 0.0228	0.9001 ± 0.0250	0.9021 ± 0.0240	0.8183 ± 0.0316
No. of "win-draw-lose" of DyS versus others		11-4-5	19-1-0	12-5-3	19-0-1

(b) G-mean

Data Set	DyS	ROS	RUS	RFS	NoS
Abalone	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0018 ± 0.0180	0.0000 ± 0.0000	0.0000 ± 0.0000
Arrhythmia	0.1328 ± 0.2398	0.1035 ± 0.1948	0.2117 ± 0.2579	0.1200 ± 0.2117	0.0000 ± 0.0000
Balance-scale	0.9479 ± 0.0336	0.9131 ± 0.0388	0.9029 ± 0.0373	0.9162 ± 0.0368	0.7505 ± 0.0925
Car	0.9408 ± 0.0420	0.8654 ± 0.0351	0.8233 ± 0.0468	0.8684 ± 0.0384	0.8668 ± 0.0509
Chess	0.4517 ± 0.0701	0.4600 ± 0.0209	0.2705 ± 0.0153	0.4536 ± 0.0198	0.0032 ± 0.0324
Contraceptive	0.5184 ± 0.0375	**0.5445 ± 0.0287**	**0.5434 ± 0.0294**	**0.5448 ± 0.0280**	0.5018 ± 0.0383
Dermatology	0.9265 ± 0.0418	0.9002 ± 0.1645	0.8363 ± 0.1421	0.8900 ± 0.1674	0.7353 ± 0.2802
Ecoli	0.8198 ± 0.0582	0.8154 ± 0.0554	0.7916 ± 0.0608	0.8138 ± 0.0599	0.0120 ± 0.0847
Glass	0.5789 ± 0.2276	**0.6713 ± 0.1509**	0.5943 ± 0.1655	**0.6395 ± 0.1648**	0.0000 ± 0.0000
New-thyroid	0.9639 ± 0.0322	0.9640 ± 0.0398	0.9598 ± 0.0420	0.9634 ± 0.0327	0.8462 ± 0.0840
Nursery	0.9979 ± 0.0019	0.9828 ± 0.0039	0.9506 ± 0.0083	0.9821 ± 0.0043	0.0000 ± 0.0000
Page-blocks	0.9158 ± 0.0264	0.9175 ± 0.0188	0.8651 ± 0.0303	0.9152 ± 0.0189	0.0000 ± 0.0000
Satellite	0.8656 ± 0.0149	0.8634 ± 0.0117	0.8604 ± 0.0120	0.8621 ± 0.0123	0.8544 ± 0.0131
Soybean	0.9539 ± 0.0234	0.9437 ± 0.0240	0.9090 ± 0.0289	0.9339 ± 0.0227	0.0000 ± 0.0000
Splice	0.8370 ± 0.0224	0.8322 ± 0.0188	0.8245 ± 0.0163	0.8238 ± 0.0197	0.8078 ± 0.0217
Thyroid-allhypo	0.9207 ± 0.0312	0.9161 ± 0.0313	0.8566 ± 0.0378	0.9222 ± 0.0317	0.1360 ± 0.1500
Thyroid-allrep	0.7095 ± 0.0779	**0.7279 ± 0.0847**	**0.7315 ± 0.0866**	**0.7267 ± 0.0783**	0.0000 ± 0.0000
Thyroid-ann	0.9455 ± 0.0288	0.9095 ± 0.0452	0.9160 ± 0.0286	0.9453 ± 0.0206	0.0089 ± 0.0438
Letter-1	0.7784 ± 0.0798	0.7120 ± 0.1268	0.4841 ± 0.0904	0.6891 ± 0.1756	0.0514 ± 0.1882
Letter-2	0.0068 ± 0.0683	0.0072 ± 0.0719	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
No. of "win-draw-lose" of DyS versus others		8-9-3	13-5-2	8-9-3	18-2-0

significant difference between DyS and other methods. And on another few data sets, DyS performs worse than other methods, especially when comparing with ROS and RFS. DyS performs worse than ROS on five data sets and worse than RFS on three data sets. DyS is able to perform better in general because of the simple framework of its training process. It can select the

Algorithm 1 PseudoCode for the Dynamic Sampling Algorithm

```
Dynamic-Sampling(S, epoch)
  r_max = max_i{r_i} ;
  r_min = min_i{r_i};
  for i = 1 to m
  ρ_i = α_i = r_max/r_i − 1;
  end for
  for ep = 1 to epoch
    S' = Duplicate(S, ρ);
    while S' ≠ ∅
      (x, t) = a random example from S';
      S' = S' − {(x, t)};
      c = the class label that (x, t) belongs to;
      y = MLP(x);
      δ = y_c − max_{i≠c}{y_i}
      if δ < 0 then
        p = 1;
      else
        p = exp(−δ · r_c/r_min);
      end if
      r = uniform();
      if r < p then
        updateMLP(x, t);
      end if
    end while
    if ep > 2 then
      for i = 1 to m
        ρ_i = α_i / ln ep;
      end for
    end if
  end for
end Dynamic-Sampling
```

Algorithm 2 PseudoCode for the MLP-Based Active Learning for Multiclass Imbalance Problems

```
ActiveLearning(S, ϵ)
  S=StratifiedSelect(TS, k);
  R=TS − S;
  Use S to train the MLP;
  while R ≠ ∅ or stop condition is not satisfied
    for l=1 to |R|
      (x, t) =the lth example of R;
      y = MLP(x);
      y_max = max{y_i};
      y'_max = max{{y_i} − y_max};
      if y_max − y'_max < ϵ then
        S=S + {(x, t)};
        R=R − {(x, t)};
      end if
    end for
    Use S to train the MLP;
  end while
end ActiveLearning
```

informative examples during the training process and adapt to the situation. That means, as the training process progresses, the strength that an example is informative can be changed according to the current knowledge that the MLP has learned.

From Table III(b), we can observe that when comparing by the values of G-mean, DyS also performs better than or comparable to others on most data sets. Some G-mean values of all the methods are very small (<0.5) on some of the data sets, such as *Abalone*, *Arrhythmia*, *Chess*, and *Letter-2*. It indicates that none of the mentioned methods can construct an effective classifier on these data sets. Since G-mean is the geometric mean of the classification accuracy of every class, poor accuracy of even one class will lead to poor G-mean. Poor value of G-mean indicates that the classifier cannot effectively classify at least one class, which makes the classifier less useful in practice.

In order to see the statistical difference over all the data sets, Wilcoxon signed-rank test with the level of significance $\alpha = 0.05$ was employed to make a statistical test over all the MAUC and G-mean values of all the executions on all data sets. The result shows that DyS significantly outperforms ROS, RUS, RFS, and NoS.

2) Comparison With Active Learning: Ertekin *et al.* [18] proposed a SVM-based active learning method for class imbalance problems in which they focused on binary-class cases only. They selected informative examples according to the distance to the decision hyperplane. The closer an example is to the hyperplane, the more informative the example is. In order to make a comparison of DyS and active learning methods, we will introduce this idea into MLP to propose an MLP-based active learning method for class imbalance problems in multiclass cases.

As presented before, we set the number of the output neurons as the number of the classes, and the neuron with the highest output of all neurons designates the class. Then the difference between the highest output and the second highest output can be used to judge that whether an example is close to the boundary or not.

The pseudocode for MLP-based active learning for class imbalance problems is presented in Algorithm 2. In the pseudocode, TS denotes the training set and is a parameter that defines the threshold of an example be in the boundary or not. The function stratified select (S, k) means that select k examples from S randomly with the ratio of each class in the selected set the same to S and return the selected set. The function MLP(x) returns the output of from the MLP and the output is denoted as $y = (y_1, y_2, \ldots, y_m)$.

In the experiments, k was set to 50% of the number of the training examples and $\in$ was set to 0.01. The results are presented in the "AL" column of Table IV. Wilcoxon signed rank test with the level of significance $\alpha = 0.05$ was employed for the comparison. We can observe from Table IV that, DyS significantly outperforms active learning in terms of MAUC on all datasets and in terms of G-mean on all data sets but *Abalone* and *Letter-2*, on which the G-mean values of DyS and active learning are both 0. The performance of active learning does not seem to be good for multiclass imbalance problems. It is

TABLE IV

AVERAGE AND STANDARD DEVIATION OF MAUC AND G-MEAN VALUES OF DYS, ACTIVE LEARNING METHOD (AL), MMC, RESCALE, RAMOBOOST, AND DSMLP BASED ON 20 TIMES FIVE-FOLD CROSS-VALIDATION. WILCOXON SIGNED RANK TEST WITH THE LEVEL OF SIGNIFICANCE WAS EMPLOYED FOR THE COMPARISON. IN THE COLUMNS FOR OTHER METHODS, THE VALUES WITH UNDERLINE (OR BOLD) DENOTE THAT DYS PERFORMED SIGNIFICANTLY BETTER (OR WORSE) THAN THEM ON THOSE DATA SETS AND THE VALUES WITH NORMAL TYPE DENOTE THAT THERE ARE NO SIGNIFICANT DIFFERENCES. THE LAST LINE OF EACH TABLE PRESENTS THE NUMBER OF "WIN-DRAW-LOSE" OF DYS VERSUS OTHERS OVER ALL DATA SETS

(a) MAUC

Data Set	DyS	AL	MMC	Rescale	RAMOBoost	DSMLP
Abalone	0.8135 ± 0.0057	0.8026 ± 0.0056	0.6626 ± 0.0118	0.7582 ± 0.0041	0.7973 ± 0.0103	0.8091 ± 0.0056
Arrhythmia	0.8395 ± 0.0308	0.6935 ± 0.0312	0.6623 ± 0.0481	0.6994 ± 0.0494	0.8478 ± 0.0289	0.8047 ± 0.0313
Balance-scale	0.9742 ± 0.0114	0.9323 ± 0.0319	0.7965 ± 0.0193	0.9561 ± 0.0159	0.9623 ± 0.0182	0.9682 ± 0.0125
Car	0.9890 ± 0.0062	0.9799 ± 0.0078	0.7763 ± 0.0146	0.9348 ± 0.0106	0.9833 ± 0.0077	0.9865 ± 0.0057
Chess	0.9052 ± 0.0079	0.8357 ± 0.1929	0.5468 ± 0.0097	0.7871 ± 0.0063	**0.9228 ± 0.0042**	0.8485 ± 0.0130
Contraceptive	0.7268 ± 0.0202	0.7204 ± 0.0199	0.7119 ± 0.0203	0.7283 ± 0.0206	0.7176 ± 0.0203	0.7276 ± 0.0207
Dermatology	0.9828 ± 0.0150	0.9189 ± 0.0184	0.8443 ± 0.0439	0.9334 ± 0.0325	**0.9944 ± 0.0042**	0.9825 ± 0.0109
Ecoli	0.9598 ± 0.0157	0.9103 ± 0.0252	0.7622 ± 0.0248	0.9004 ± 0.0153	0.9588 ± 0.0157	0.9566 ± 0.0180
Glass	0.8726 ± 0.0463	0.8085 ± 0.0488	0.7825 ± 0.0403	0.8410 ± 0.0412	0.8808 ± 0.0416	0.8753 ± 0.0490
New-thyroid	0.9985 ± 0.0027	0.9633 ± 0.0408	0.8504 ± 0.0292	0.9932 ± 0.0140	0.9983 ± 0.0031	0.9981 ± 0.0036
Nursery	0.9973 ± 0.0017	0.9211 ± 0.0017	0.8211 ± 0.0068	0.9869 ± 0.0019	**0.9997 ± 0.0003**	0.9178 ± 0.0010
Page-blocks	0.9717 ± 0.0081	0.8248 ± 0.0344	0.6967 ± 0.0484	0.8368 ± 0.0210	**0.9825 ± 0.0059**	0.9631 ± 0.0102
Satellite	0.9829 ± 0.0020	0.9789 ± 0.0028	0.9767 ± 0.0022	0.9790 ± 0.0023	0.9818 ± 0.0018	0.9783 ± 0.0029
Soybean	0.9979 ± 0.0013	0.9463 ± 0.0312	0.7593 ± 0.0174	0.9221 ± 0.0209	0.9980 ± 0.0013	0.9411 ± 0.0259
Splice	0.9472 ± 0.0095	0.9242 ± 0.0090	0.9426 ± 0.0075	0.9455 ± 0.0079	0.9262 ± 0.0080	0.9369 ± 0.0105
Thyroid-allhypo	0.9770 ± 0.0097	0.8086 ± 0.0236	0.8211 ± 0.0062	0.8988 ± 0.0119	0.9593 ± 0.0136	0.9692 ± 0.0132
Thyroid-allrep	0.9219 ± 0.0246	0.7356 ± 0.0056	0.6316 ± 0.0576	0.8434 ± 0.0336	0.9157 ± 0.0209	0.9122 ± 0.0230
Thyroid-ann	0.9822 ± 0.0065	0.7888 ± 0.0170	0.8152 ± 0.0065	0.8729 ± 0.0075	0.9822 ± 0.0059	**0.9840 ± 0.0061**
Letter-1	0.9632 ± 0.0042	0.9101 ± 0.0121	0.5318 ± 0.0061	0.8551 ± 0.0136	**0.9823 ± 0.0049**	0.9262 ± 0.0083
Letter-2	0.9083 ± 0.0225	0.8707 ± 0.0246	0.8691 ± 0.0237	0.8864 ± 0.0224	**0.9249 ± 0.0205**	0.8643 ± 0.0232
No. of "win-draw-lose" of DyS versus others		20-0-0	20-0-0	18-2-0	8-6-6	15-4-1

(b) G-mean

Data Set	DyS	AL	MMC	Rescale	RAMOBoost	DSMLP
Abalone	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0018 ± 0.0178	0.0000 ± 0.0000
Arrhythmia	0.1328 ± 0.2398	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0702 ± 0.1835	0.0342 ± 0.1263
Balance-scale	0.9479 ± 0.0336	0.7804 ± 0.1056	0.4115 ± 0.1164	0.9161 ± 0.0352	0.9231 ± 0.0525	0.9014 ± 0.0483
Car	0.9408 ± 0.0420	0.8437 ± 0.0601	0.0000 ± 0.0000	0.8017 ± 0.0260	0.9001 ± 0.0432	0.9034 ± 0.0372
Chess	0.4517 ± 0.0701	0.0173 ± 0.0761	0.0000 ± 0.0000	0.0000 ± 0.0000	**0.5035 ± 0.0181**	0.3878 ± 0.0722
Contraceptive	0.5184 ± 0.0375	0.5017 ± 0.0429	0.4835 ± 0.0306	**0.5402 ± 0.0275**	**0.5336 ± 0.0311**	**0.5399 ± 0.0279**
Dermatology	0.9265 ± 0.0418	0.7034 ± 0.3040	0.0168 ± 0.1006	0.7065 ± 0.3443	**0.9572 ± 0.0237**	0.8916 ± 0.1636
Ecoli	0.8198 ± 0.0582	0.0463 ± 0.1706	0.0000 ± 0.0000	0.5934 ± 0.1615	0.8136 ± 0.0625	0.8100 ± 0.0558
Glass	0.5789 ± 0.2276	0.0896 ± 0.2154	0.0065 ± 0.0458	**0.5953 ± 0.1292**	0.5360 ± 0.2811	0.6029 ± 0.2402
New-thyroid	0.9639 ± 0.0322	0.7826 ± 0.2173	0.0000 ± 0.0000	0.9586 ± 0.0357	0.9685 ± 0.0332	**0.9551 ± 0.0449**
Nursery	0.9979 ± 0.0019	0.0000 ± 0.0000	0.1807 ± 0.1640	0.9146 ± 0.0103	0.9942 ± 0.0044	0.0000 ± 0.0000
Page-blocks	0.9158 ± 0.0264	0.0000 ± 0.0000	0.0000 ± 0.0000	0.5299 ± 0.0468	0.9139 ± 0.0381	0.7729 ± 0.0992
Satellite	0.8656 ± 0.0149	0.8338 ± 0.0294	0.8230 ± 0.0200	0.8517 ± 0.0119	0.8676 ± 0.0140	0.8491 ± 0.0146
Soybean	0.9539 ± 0.0234	0.0437 ± 0.1915	0.0000 ± 0.0000	0.0000 ± 0.0000	0.9439 ± 0.0269	0.0000 ± 0.0000
Splice	0.8370 ± 0.0224	0.7882 ± 0.0206	0.8311 ± 0.0149	0.8359 ± 0.0197	0.8030 ± 0.0178	0.8012 ± 0.0692
Thyroid-allhypo	0.9207 ± 0.0312	0.2220 ± 0.1707	0.0000 ± 0.0000	0.7674 ± 0.0246	0.8208 ± 0.0450	0.8788 ± 0.0421
Thyroid-allrep	0.7095 ± 0.0779	0.0000 ± 0.0000	0.0000 ± 0.0000	0.3091 ± 0.2712	0.5855 ± 0.1146	0.6743 ± 0.1124
Thyroid-ann	0.9455 ± 0.0288	0.0331 ± 0.0833	0.0000 ± 0.0000	0.7597 ± 0.0255	0.9056 ± 0.0246	0.9145 ± 0.0386
Letter-1	0.7784 ± 0.0798	0.0732 ± 0.2208	0.0000 ± 0.0000	0.0000 ± 0.0000	**0.8064 ± 0.1659**	0.6730 ± 0.2003
Letter-2	0.0068 ± 0.0683	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0000 ± 0.0000
No. of "win-draw-lose" of DyS versus others		18-2-0	18-2-0	14-4-2	9-7-4	14-4-2

difficult to set the parameter to find an approximate boundary for the whole training set. The function stratified select in the pseudocode was a stochastic selection method without considering the imbalance of multiple data sets. The operators of active learning framework, such as the selection of the initial training set and the criterion to select the examples from the

boundary, could be improved in multiclass cases. This will be studied in depth in our future work.

3) Comparison With Cost-Sensitive Methods: Kukar and Kononenko [29] proposed some methods by modifying BP for cost-sensitive problems and they concluded that minimization of misclassification costs (MMC) was best of the proposed methods in [29]. Therefore, we employ MMC to compare with DyS. The cost matrix was formulated as the one used in [28]

$$C_{ij} = \begin{cases} \frac{1}{n_i}, & i \neq j \\ 0, & i = j \end{cases} \tag{6}$$

where C_{ij} denotes the cost of misclassifying the example of class i to class j, n_i is the number of examples in class i.

The results are presented in the "MMC" column of Table IV. The observations are similar to the comparison with active learning. DyS significantly outperformed MMC in terms of MAUC on all datasets and in terms of G-mean on all data sets but *Abalone*, on which the G-mean values of DyS and MMC are both 0. The G-mean values of MMC are 0 on most data sets, which means that the examples of at least one class were totally misclassified on those data sets. To see the classification condition on each class, the classification accuracy of each class was calculated for MMC and the average values over the 20 times five-fold cross validation on every data set were presented in Table V. Comparing the class distribution of every data set in Table I, we can observe that the classification accuracy of a smaller class is usually higher than that of a larger class. Take *Splice* for example. The class distribution is 767: 768: 1655 and the classification accuracy of each class is 0.8612, 0.9082, and 0.7354, respectively. On most data sets, the classification accuracies of some larger classes are 0 and those of some smaller classes are 1. We can conclude that MMC has made the MLP biased too much to the minority classes when there are multiple classes. The reason may be the definition of cost matrix in (6). As presented in [29], MMC could perform well, i.e., get a small misclassification cost value on a specific cost matrix. However, when applying MMC to a pure class imbalance problem, a proper cost matrix should be set up for the specific problem [5], [14], [28]. In practice, it is very hard to define a cost matrix precisely.

The following makes a comparison between DyS and another cost-sensitive method. Zhou and Liu proposed a cost-sensitive learning method, which was named for multiclass problems [28] and the proposed method was also employed for pure class imbalance problems. We compared the method in our experimental study (referred as Rescale) and MLP was used as the basic classifier. The results are presented in the "rescale" column of Table IV. We can observe from Table IV that, when comparing in terms of MAUC, DyS significantly outperforms rescale on 18 data sets and ties with rescale on the remaining two data sets. When comparing in terms of G-mean, DyS significantly outperforms rescale on 14 data sets and is significantly outperformed by rescale on four data sets and ties with Rescale on the remaining two data sets [14].

The following presents a brief comparison between DyS and AdaC2.M1. The results that were presented in [14] will be used directly. In [14], the misclassification costs were set by a GA, which was very time consuming. In [14], three data sets were used, i.e., *Car*, *New-thyroid*, and *Nursery*. The data sets were divided as 4:1 for training and testing sets, and the process was repeated 10 times to get an average performance. In our experimental setup, 20 times five-fold cross validation was executed and the ratio of the size of training set to testing set was also 4:1. The results are given in Table VI. Since MAUC values were not presented in [14], only G-mean values are used for the comparison here. We can observe from Table VI that DyS has higher G-mean values on all three data sets.

4) Comparison With RAMOBoost: RAMOBoost [17] was a recently proposed boosting-type method for class imbalance

TABLE V
AVERAGE OF CLASSIFICATION ACCURACY OF EACH CLASS OVER 20 TIMES FIVE-FOLD CROSS VALIDATION USING THE MMC ALGORITHM

Data Set	The Classification Accuracy of Each Class
Abalone	1, 0.0085, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.005, 0.992
Arrhythmia	0, 0, 0.9367, 1, 0, 0, 0.121
Balance-scale	1, 0.2863, 0.303
Car	0, 0.0001, 0.95 0.8871
Chess	0, 0.9623, 0.0588, 0, 0.6686, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Contraceptive	0.309, 0.7818, 0.4744
Dermatology	0.0113, 0.0929, 0.9348, 0.9939, 1, 1
Ecoli	0, 0, 1, 1, 0
Glass	0.0036, 0.0607, 0.9883, 0.9367
New-thyroid	0, 1, 1
Nursery	0.0075, 0.9997, 1, 0.9684
Page-blocks	0, 0, 1, 0.0044, 0
Satellite	0.957, 0.9419, 0.7169, 0.8994, 0.8932, 0.6057
Soybean	0.795, 0.99 0.9775, 0, 0, 0.825, 0.96 0, 0.5625, 0.0575, 0.415, 0, 0.255, 0, 0, 1, 0.98
Splice	0.8612, 0.9082, 0.7354
Thyroid-allhypo	0, 0.9139, 0.9595
Thyroid-allrep	0, 0.3748, 0.0712, 0.794
Thyroid-ann	0.9615, 0.9328, 0
Letter-1	1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
Letter-2	0.79, 0.11, 0.425, 0.725, 0.425, 0.11, 0.07, 0.245, 0.785, 0.61, 0.485, 0.79, 0.88, 0.35, 0.1, 0.955, 0.185, 0.335, 0.545, 0.77, 0.8, 0.855, 0.86, 0.345, 0.05, 0

TABLE VI
COMPARISON OF DyS AND AdaC2.M1 ON G-MEAN VALUES

Data Set	DyS	AdaC2.M1
Car	0.9408 ± 0.0420	0.9146
New-thyroid	0.9639 ± 0.0322	0.9014
Nursery	0.9979 ± 0.0019	0.9745

The results of AdaC2.M1 are achieved directly from [14].

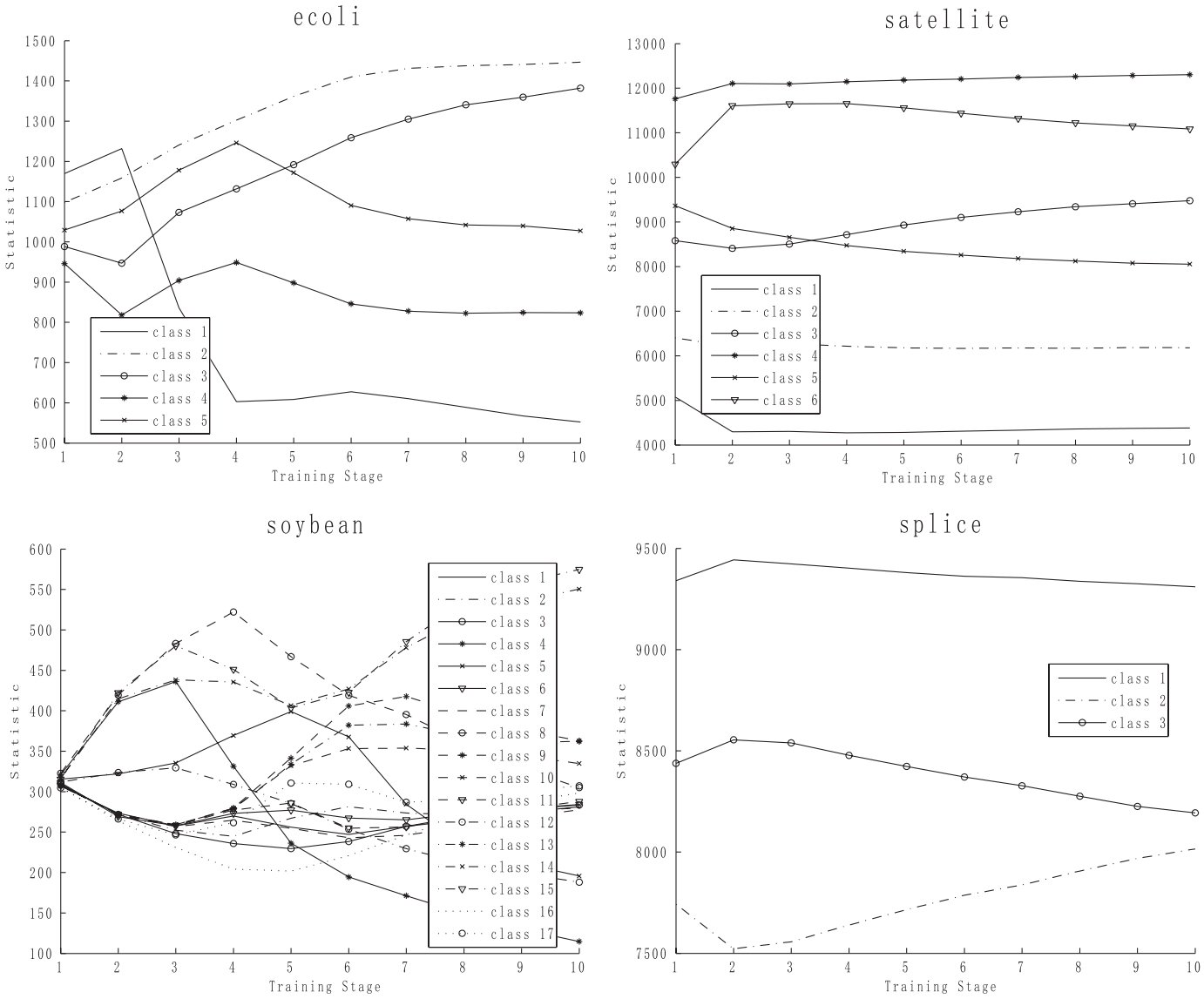


Fig. 2. Statistic of every class being selected for updating the MLP at different training stages on *Ecoli*, *Satellite*, *Soybean*, and *Splice*. The statistic is defined as follows. The training procedure is divided into ten stages. Since each update of the MLP is associated with an example, we recorded, for each class, the number of its examples that led to an update of MLP during each training stage. This value is regarded as a statistic.

problems. In our comparative study, RAMOBoost was generalized to multiclass imbalance problems. For multiclass problems, every class except for the largest one is regarded as the minority class during the data generation procedure and for every minority class, synthetic examples are generated to make the number of every class the same as the largest one.

An MLP with the same parameters as those given in Table II was used as the base classifier of RAMOBoost. The number of classifiers was set to 5. The results are presented in the "RAMOBoost" column of Table IV. We can observe from Table IV that DyS significantly outperforms RAMOBoost on eight data sets, is significantly outperformed by RAMOBoost on six data sets and ties with RAMOBoost on the remaining data sets, when comparing in terms of MAUC. When comparing in terms of G-mean, DyS significantly outperforms RAMOBoost on nine data sets, is significantly outperformed by RAMOBoost on only four data sets and ties with RAMOBoost on the remaining data sets. RAMOBoost is an ensemble-based method, so the model is more complex than DyS, which is a simple method with a single classifier. The results indicate that, with its selection operator, DyS is capable of achieving good performance with a single classifier.

5) Comparison With DSRBF: DSRBF was a recently proposed method for multiclass imbalance problems. The sampling strategy of DSRBF is a bit similar to DyS. Therefore, it will be interesting to see the difference in performance between the two methods. However, in DSRBF and RBFNN was used as the basic classifier while in DyS and MLP was used. To make an impartial comparison, in this comparative study, DSRBF was re-implemented by replacing RBFNN with MLP as its basic classifier. The parameters of MLP were the same as those given in Table II. According to the modification, we can see the performance of DSRBF when the sampling strategy was employed into MLP and the sampling strategies

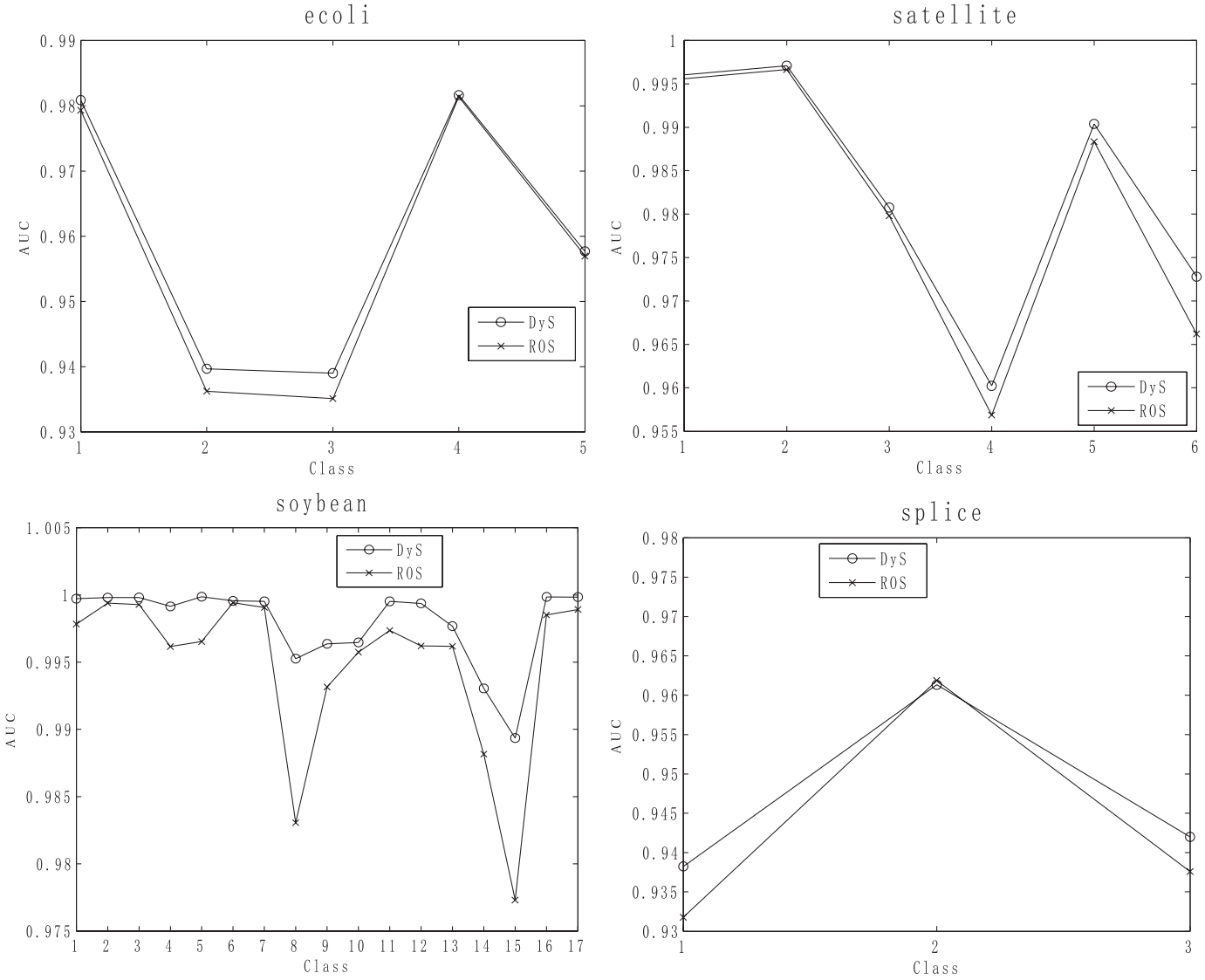


Fig. 3. AUC of each class on *Ecoli*, *Satellite*, *Soybean*, and *Splice*.

of DyS and DSRBF could be impartially compared. Since the basic classifier was replaced with MLP, the modified methed was named DSMLP.

The results are presented in "DSMLP" column of Table IV. We can observe from Table IV that DyS significantly outperforms DSMLP on 15 data sets, is significantly outperformed by DSMLP on one data set only and ties with DSMLP on the remaining data sets, when comparing in terms of MAUC. When comparing in terms of G-mean, DyS significantly outperforms DSMLP on 14 data sets, is significantly outperformed by DSMLP on two data sets and ties with DSMLP on the remaining data sets. From these observations, it can be concluded that DyS outperforms DSMLP in general. In other words, the sampling strategy of DyS can outperform that of DSRBF when using MLPs as their basic classifiers.

E. Further Analysis

From the results presented above, we can conclude that DyS can outperform other methods on most of data sets in spite of its simplicity. However, in comparison with ROS, the performance improvements are relatively small on some data sets, such as *Ecoli*, *Satellite*, *Soybean,* and *Splice*. Since an ROS procedure is also involved in DyS, further analysis is carried out to identify the source of difference between the two approaches.

To understand the impact of DyS on the learning process, the training procedure was divided into ten stages. Since each update of the MLP is associated with an example, we recorded for each class the number of its examples that led to an update of MLP during each training stage. This value can be regarded as a "statistic" that measures the focus of the training process on each class. Fig. 2 plots this statistic on *Ecoli*, *Satellite*, *Soybean*, and *Splice*. It can be observed that, as learning progresses, the number of examples of each class that were used for updating MLP changed dynamically. This observation indicates that the importance of each class may indeed change during training, and simple pre-sampling technique is unable to handle this situation. Besides, some small classes are gradually

emphasized more. Such a scenario can only be credited to DyS since the oversampling ratio is attenuated. To summarize, the above observations demonstrate that DyS and ROS can lead to substantially different learning behaviors.

In addition, to see different performance on each class separately, AUC of each class is calculated. AUC of class is defined as

$$\mathrm{AUC}_i = \frac{1}{c-1}\sum_{j\neq i} A(i, j) \tag{7}$$

where $A(i, j)$ is the AUC value of class i and class j [40]. Fig. 3 presents the AUC value of each class on *Ecoli*, *Satellite*, *Soybean*, and *Splice*. They are the average of all the 100 values of 20 times five-fold cross validation. Compared with ROS, we can observe that DyS typically resulted in larger improvements on some small and potentially difficult classes, i.e., the classes corresponding to low AUC than other classes. Take *Soybean* as an example, the AUC values of class 8 and class 15 are obviously lower than others, but these two values are well improved when using DyS. In some cases, DyS degraded AUC values on some classes (see the results on *Splice*). However, the AUC values on all classes usually became more balanced. Since AUC can be viewed as an indicator of the difficulty of a class being correctly classified, Fig. 3 indicates that DyS can focus more on difficult classes, which either led to improvements on all classes, or prevented the final classifier from biasing too much to a specific class, typically a large one.

V. Conclusion

This paper investigated a novel training algorithm of MLPs for multiclass imbalance problems. A simple yet effective dynamic sampling method, DyS, has been proposed. In each epoch of the training process, a probability was estimated for every example fed to the MLP. It is more likely that an example will be used to update the MLP if the estimated probability is higher. In this way, DyS manages to dynamically select the training data to be used in each epoch. Extensive experiments on 20 multiclass imbalanced data sets from the UCI machine learning repository showed that DyS can outperform other relevant methods in most cases, including pre-sample methods, active learning methods, cost-sensitive methods, and boosting-type methods.

In cases that learning was conducted in an iterative manner, it is likely that the difficulty and importance of examples/classes change along with the learning process. A further analysis indicates that DyS is able to partially address this problem since it managed to dynamically adjust the focus of learning during the learning process. It is also found that the superior overall performance of DyS is due to the enhancement of performance on all classes, or a more balanced performance on them.

Although an MLP is used as the base learner in this paper, the general idea of DyS can be extended to other learning algorithms that work in an iterative mode. For example, since the general framework of DyS is similar to boosting, this framework can also be introduced to ensemble learning methods. Although our analysis has partially revealed the reasons why DyS outperformed the other compared algorithms, the link between DyS and problem/data set of interest still deserves a deeper investigation. This will be another direction of our future work. Finally, more analysis of the impact of different parameter values on the performance of DyS on different data sets will be needed in the future. It would be interesting to develop a theoretical foundation to explain why such a simply method like DyS could perform so well in practice.

References

[1] J. Zhang and M. Zulkernine, "Network intrusion detection using random forests," in *Proc. 3rd Annu. Conf. Privacy, Secur. Trust*, 2005, pp. 53–61.

[2] G. Wang, "Asymmetric random subspace method for imbalanced credit risk evaluation," in *Proc. Softw. Eng. Knowl. Eng., Theory Pract.*, 2012, pp. 1047–1053.

[3] M. A. Mazurowski, P. A. Habas, J. M. Zurada, J. Y. Lo, J. A. Baker, and G. D. Tourassi, "Training neural network classifiers for medical decision making: The effects of imbalanced datasets on classification performance," *Neural Netw.*, vol. 21, nos. 2–3, pp. 427–436, 2008.

[4] T. M. Khoshgoftaar, J. V. Hulse, and A. Napolitano, "Supervised neural network modeling: An empirical investigation into learning from imbalanced data with labeling errors," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 813–830, May 2010.

[5] N. Japkowicz and S. Stephen, "The class imbalance problem: A systematic study," *Intell. Data Anal.*, vol. 6, no. 5, pp. 429–449, 2002.

[6] J. V. Hulse, T. M. Khoshgoftaar, and A. Napolitano, "Experimental perspectives on learning from imbalanced data," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 935–942.

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

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

[9] H. Han, W. Y. Wang, and B. H. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *Adv. Intell. Comput.*, vol. 2, no. 5, pp. 878–887, 2005.

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

[11] M. Kubat and S. Matwin, "Addressing the curse of imbalanced data sets: One-sided sampling," in *Proc. Int. Conf. Mach. Learn.*, 1997, pp. 179–186.

[12] C. Elkan, "The foundations of cost-sensitive learning," in *Proc. Int. Joint Conf. Artif. Intell.*, 2001, pp. 973–978.

[13] Z. H. Zhou and X. Y. Liu, "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.

[14] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Proc. Int. Conf. Data Min.*, 2006, pp. 592–602.

[15] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "SMOTE-Boost: Improving prediction of the minority class in boosting," in *Proc. Knowl. Dis. Databas. Conf.*, 2003, pp. 107–119.

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

[17] S. Chen, H. He, and E. A. Garcia, "RAMOBoost: Ranked minority over-sampling in boosting," *IEEE Trans. Neural Netw.*, vol. 21, no. 10, pp. 1624–1642, Oct. 2010.

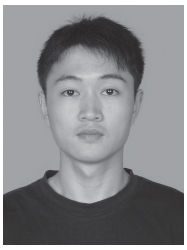
[18] S. Ertekin, J. Huang, L. Bottou, and L. Giles, "Learning on the border: Active learning in imbalanced data classification," in *Proc. 16th ACM Conf. Inf. Knowl. Manag.*, 2007, pp. 127–136.

[19] S. Ertekin, J. Huang, and C. L. Giles, "Active learning for class imbalance problem," in *Proc. 30th Annu. Int. ACM SIGIR Conf. Res. Devel. Inf. Retrieval*, 2007, pp. 823–824.

[20] T. Mu, J. Jiang, Y. Wang, and J. Y. Goulermas, "Adaptive data embedding framework for multiclass classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1291–1303, Aug. 2012.

[21] S. Xiang, F. Nie, G. Meng, C. Pan, and C. Zhang, "Discriminative least squares regression for multiclass classification and feature selection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 13, no. 11, pp. 1738–1754, Nov. 2012.

[22] A. Asuncion and D. J. Newman, *UCI Machine Learning Repository*. (2007) [Online]. Available: http://www.ics.uci.edu/∼learn/MLRepository.html
[23] N. V. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: Special issue on learning from imbalanced data sets," *ACM SIGKDD Expl. Newslett.*, vol. 6, no. 1, pp. 1–6, 2004.
[24] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Trans. Syst., Man Cybern. B*, vol. 42, no. 4, pp. 1119–1130, Apr. 2012.
[25] I. Tomek, "Two modifications of CNN," *IEEE Trans. Syst., Man Cybern.*, vol. 6, no. 6, pp. 769–772, Jun. 1976.
[26] Y. Sun, M. S. Kamel, A. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recogn.*, vol. 40, no. 12, pp. 3358–3378, 2007.
[27] P. Domingos, "Metacost: A general method for making classifiers cost-sensitive," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 1999, pp. 155–164.
[28] Z. Zhou and X. Liu, "On multiclass cost-sensitive learning," in *Proc. Nat. Conf. Artif. Intell.*, 2006, pp. 567–572.
[29] M. Kukar and I. Kononenko, "Cost-sensitive learning with neural networks," in *Proc. Eur. Conf. Artif. Intell.*, 1998, pp. 445–449.
[30] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Mach. Learn.*, 1996, pp. 148–156.
[31] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
[32] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press, 1992.
[33] R. E. Schapire, *The Boosting Approach to Machine Learning: An Overview*. New York: Springer Verlag, 2003, pp. 149–172.
[34] S. Wang and X. Yao, "Relationships between diversity of classification ensembles and single-class performance measures," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 206–219, Jan. 2013.
[35] D. A. Cohn, "Neural network exploration using optimal experiment design," *Neural Netw.*, vol. 9, no. 6, pp. 1071–1083, 1996.
[36] F. Fernández-Navarro, C. Hervás-Martínez, and P. A. Gutiérrez, "A dynamic over-sampling procedure based on sensitivity for multiclass problems," *Pattern Recogn.*, vol. 44, no. 8, pp. 1821–1833, 2011.
[37] U. Bhowan, M. Johnston, M. Zhang, and X. Yao. (2012, May). Evolving diverse ensembles using genetic programming for classification with unbalanced data. *IEEE Trans. Evol. Comput.* [Online]. Available: www.ieeexplore.ieee.org.10.1109/TEVC.2012.2199119
[38] J. A. Hanley and B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve," *Radiology*, vol. 143, no. 1, pp. 29–36, 1982.
[39] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recogn.*, vol. 30, no. 7, pp. 1145–1159, 1997.
[40] D. J. Hand and R. J. Till, "A simple generalisation of the area under the ROC curve for multiple class classification problems," *Mach. Learn.*, vol. 45, no. 2, pp. 171–186, 2001.
[41] M. Robnik-Šikonja, "Improving random forests," *Mach. Learn.*, vol. 10, no. 1, pp. 359–370, 2004.
[42] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometr. Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.



Minlong Lin received the B.Sc. and Ph.D. degrees from the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2007 and 2012, respectively.

He is currently a Research Associate with Sogou Company, where he is involved in research on data analysis and machine leaning. His current research interests include neural networks, machine learning, and pattern recognition.

Ke Tang (S'05–M'07–SM'12) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2002 and 2007, respectively.

He joined the Nature Inspired Computation and Applications Laboratory (NICAL), School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 2007, as an Associate Professor, and was promoted to a Professor in 2011. He is an Honorary Senior Research Fellow with the School of Computer Science, University of Birmingham, Birmingham, U.K. He has authored or co-authored more than 50 refereed papers in journals and conferences. His current research interests include evolutionary computation, machine learning, and real-world applications.

Dr. Tang is an Associate Editor of the IEEE Computational Intelligence Magazine and was the Program Co-Chair of CEC2010 Barcelona.

Xin Yao (M'91–SM'96–F'03) received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, China, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, and the Ph.D. degree from USTC in 1982, 1985, and 1990, respectively.

He was an Associate Lecturer and Lecturer of simulated annealing and evolutionary algorithms at USTC from 1985 to 1990. In 1990, he was a Post-Doctoral Fellow with the Computer Sciences Laboratory, Australian National University, Canberra, Australia, where he was involved in research on simulated annealing and evolutionary algorithms. In 1991, he was with the Knowledge-Based Systems Group, Division of Building, Construction and Engineering, Commonwealth Scientific and Industrial Research Organization, Melbourne, Australia, where he was involved in research on an industrial project on automatic inspection of sewage pipes. In 1992, he was a Lecturer with the School of Computer Science, University College, University of New South Wales, Australian Defense Force Academy, Sydney, Australia, where he was promoted to a Senior Lecturer and Associate Professor. Since 1999, he has been a Professor (Chair) of computer science with the University of Birmingham, Birmingham, U.K. He is currently the Director of the Center of Excellence for Research, Computational Intelligence and Applications, School of Computer Science, University of Birmingham, and also a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, USTC. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. He has authored or co-authored more than 300 referenced papers. His current research interests include evolutionary artificial neural networks, automatic modularization of machine learning systems, evolutionary optimization, constraint-handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner's dilemma, data mining, and real-world applications.

Dr. Yao was the recipient of the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. research on simulated annealing and evolutionary algorithms in 1989 and the 2001 IEEE Donald G. Fink Prize Paper Award for his research on evolutionary artificial neural networks. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2003 to 2008, an Associate Editor or an Editorial Board Member of 12 other journals, and an Editor of the World Scientific Book Series on Advances in Natural Computation.