Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 3358–3378
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

# Cost-sensitive boosting for classification of imbalanced data

Yanmin Sun[a,*], Mohamed S. Kamel[a], Andrew K.C. Wong[b], Yang Wang[c]

[a]*Electrical and Computer Engineering Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[b]*Systems Design Engineering Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1*
[c]*Pattern Discovery Technologies Inc., Waterloo, Ontario, Canada, N2L 5Z4*

Received 3 September 2006; received in revised form 10 February 2007; accepted 17 April 2007

**Abstract**

Classification of data with imbalanced class distribution has posed a significant drawback of the performance attainable by most standard classifier learning algorithms, which assume a relatively balanced class distribution and equal misclassification costs. The significant difficulty and frequent occurrence of the class imbalance problem indicate the need for extra research efforts. The objective of this paper is to investigate meta-techniques applicable to most classifier learning algorithms, with the aim to advance the classification of imbalanced data. The AdaBoost algorithm is reported as a successful meta-technique for improving classification accuracy. The insight gained from a comprehensive analysis of the AdaBoost algorithm in terms of its advantages and shortcomings in tacking the class imbalance problem leads to the exploration of three cost-sensitive boosting algorithms, which are developed by introducing cost items into the learning framework of AdaBoost. Further analysis shows that one of the proposed algorithms tallies with the stagewise additive modelling in statistics to minimize the cost exponential loss. These boosting algorithms are also studied with respect to their weighting strategies towards different types of samples, and their effectiveness in identifying rare cases through experiments on several real world medical data sets, where the class imbalance problem prevails.


*Keywords:* Classification; Class imbalance problem; AdaBoost; Cost-sensitive learning

## 1. Introduction

Classification is an important task of knowledge discovery in databases (KDD) and data mining. Classification modelling is to learn a function from training data, which makes as few errors as possible when being applied to data previously unseen. A range of classification modelling algorithms, such as decision tree, neural network, Bayesian network, nearest neighbor, support vector machines, and the newly reported associative classification, have been well developed and successfully applied to many application domains. However, reports from both academia and industry indicate that imbalanced class distribution of a data set has posed a serious difficulty to most classifier learning algorithms, which assume a relatively balanced distribution [1–4].

The imbalanced class distribution is characterized as having many more instances of some classes than others. Particularly for a bi-class application, the imbalanced problem is one in which one class is represented by a large of samples, while the other one is represented by only a few. Standard classifiers generally perform poorly on imbalanced data sets because they are designed to generalize from training data and output the simplest hypothesis that best fits the data. The simplest hypothesis, however, pays less attention to rare cases in an imbalanced data set. Therefore, classification rules that predict the small class tend to be fewer and weaker than those that predict the prevalent class. Consequently, test samples belonging to the small class are misclassified more often than those belonging to the prevalent class. In most applications, even though the degree of imbalance varies from one application to another, the correct classification of samples in the rare class often has a greater value than the contrary case. For example, in a disease diagnostic problem where the disease cases are usually quit rare as compared with normal populations, the recognition goal is to detect people with diseases. A favorable classification model is one that provides a higher identification rate on the disease category. Imbalanced or skewed class distribution problem is therefore also referred to as small or rare class problem.

* Corresponding author. Tel.: +1 519 8884567x33746.
*E-mail address:* y8sun@engmail.uwaterloo.ca (Y. Sun).

0031-3203/$30.00 
doi:10.1016/j.patcog.2007.04.009

Research on the class imbalance problem is critical in data mining and machine learning. Two observations account for this point: (1) the class imbalance problem is pervasive in a large number of domains of great importance in data mining community. In addition to the example of the rare medical diagnosis [5], other reported applications include detection of oil spills in satellite radar images [3], the detection of fraudulent calls [2], risk management [6], modern manufacturing plants [4], text classification [7], and etc.; and (2) most popular classification modelling systems are reported to be inadequate when encountering the class imbalance problem. These classification systems involve decision trees [1,8–10], support vector machines [9,11–13], neural networks [9], Bayesian network [6], nearest neighbor [8,14] and the newly reported associative classification approaches [15,16].

The significant difficulty of the class imbalance problem and its frequent occurrence in practical applications of machine learning and data mining have attracted a lot of research interests. Two workshops on this problem were held in 2000 [17] and 2003 [18] at the AAAI and ICML conferences, respectively. A number of research papers dedicated to this problem can be found in Ref. [19] and other publications. Research efforts address on three aspects of the class imbalance problem: (1) the nature of the class imbalance problem (i.e., "in what domains do class imbalances most hinder the performance of a standard classifier?" [9]); (2) the possible solutions in tackling the class imbalance problem; and (3) the proper measures for evaluating classification performance in the presence of the class imbalance problem. Within these reported works, most efforts concentrate on the second issue. Published solutions to the class imbalance problem can be categorized as data level and algorithm level approaches [1]. At the data level, the objective is to re-balance the class distribution by resampling the data space, including oversampling instances of the small class and undersampling instances of the prevalent class. Sometimes this can involve a combination of the two techniques [20–22]. At the algorithm level, solutions try to adapt existing classifier learning algorithms to bias towards the small class, such as cost-sensitive learning [23] and recognition-based learning [24]. Obvious shortcomings with the resampling (data level) approaches are: (1) the optimal class distribution of a training data is usually unknown; (2) an ineffective resampling strategy may risk losing information of the prevalent class when being undersampled and overfitting the small class when being oversampled; and (3) extra learning cost for analyzing and processing data is unavoidable in most cases. Solutions at the algorithm level being either classifier learning algorithm-dependent or application-dependent are shown to be effective if applied in a certain context. These factors indicate the need for additional research efforts to advance the classification of imbalanced data.

The objective of this paper is to investigate meta-techniques applicable to most classifier learning algorithms to advance the classification of imbalanced data. Some ensemble methods have emerged as meta-techniques for improving the generalization performance of existing learning algorithms. Specially, AdaBoost [25–28] is reported as the most successful boosting algorithm with a promise of improving classification accuracies of a "weak" learning algorithm. However, the promise of accuracy improvement is trivial in the context of the class imbalance problem, where accuracy is less meaningful. Since AdaBoost is an accuracy-oriented algorithm, its learning strategy may bias towards the prevalent class as it contributes more to the overall classification accuracy. Consequently, the identification performance on the small class is not always satisfactory by applying AdaBoost.

In this paper, the AdaBoost algorithm is adapted for advancing the classification of imbalanced data. Three cost-sensitive boosting algorithms are developed by introducing cost items into the learning framework of AdaBoost. The cost items are used to denote the uneven identification importance between classes, such that the boosting strategies can intentionally bias the learning towards the class associated with higher identification importance and eventually improve the performance on it. For each proposed boosting algorithm, its weight update parameter is deduced taking the cost items into consideration. This step is necessary in maintaining good efficiency of a boosting algorithm. From a statistician's point of view, the breakthrough of AdaBoost occurred in Ref. [29], where it has been shown that AdaBoost is equivalent to forward stagewise additive modelling using exponential loss function [30]. Our study shows that one of our proposals tallies with the stagewise additive modelling to minimize the cost exponential loss. These boosting algorithms are further investigated with respect to their weighting strategies towards different types of samples, and their effectiveness in identifying rare cases through experiments on several real world medical data sets, where the class imbalance problem prevails.

The rest of the paper is organized as follows. Following the introduction, Section 2 presents a comprehensive study on the class imbalance problem, including the nature of the problem, reported solutions, and proper measures for evaluating classification performance in the presence of the class imbalance problem. Section 3 provides a thorough discussion on the AdaBoost algorithm to gain insight into advantages and challenges of the boosting approach in tackling the class imbalance problem. Section 4 describes our proposed cost-sensitive boosting algorithms and illustrates the forward stagewise additive modelling using exponential loss function. Section 5 analyzes and compares their weighting strategies of these boosting algorithms. Section 6 reports our experimental results. Section 7 highlights the conclusions and states some points for future research.

## 2. Class imbalance problem

### 2.1. Nature of the problem

In a data set with the class imbalance problem, the most obvious characteristic is the skewed data distribution between classes. However, theoretical and experimental studies presented in Refs. [8–10,31–33] indicate that the skewed data distribution is not the only parameter that influences the modelling of a capable classifier in identifying rare events. Other influential facts include small sample size, separability and the existence of within-class sub-concepts.

- *Imbalanced class distribution*: The imbalance degree of a class distribution can be denoted by the ratio of the sample size of the small class to that of the prevalent class. In practical applications, the ratio can be as drastic as 1:100, 1:1000, or even larger [1]. In Ref. [33], research was conducted to explore the relationship between the class distribution of a training data set and the classification performances of decision trees. Their study indicates that a relatively balanced distribution usually attains a better result. However, at what imbalance degree the class distribution deteriorates the classification performance cannot be stated explicitly, since other factors such as sample size and separability also affect performance. In some applications, a ratio as low as 1:35 can make some methods inadequate for building a good model, but in some other cases, 1:10 is tough to deal with [32].
- *Small sample size*: Given a fixed imbalance degree, the sample size plays a crucial role in determining the "goodness" of a classification model. In the case that the sample size is limited, uncovering regularities inherent in small class is unreliable. Experimental observations reported in Ref. [9] indicate that as the size of the training set increases, the large error rate caused by the imbalanced class distribution decreases. This observation is quite understandable. When more data can be used, relatively more information about the small class benefits the classification modelling, which becomes able to distinguish rare samples from the majority. Hence, the authors of Ref. [9] suggest that the imbalanced class distribution may not be a hindrance to classification by providing a large enough data set, assuming that the data set is available and the learning time required for a sizeable data set is acceptable.
- *Separability*: The difficulty in separating the small class from the prevalent class is the key issue of the small class problem. Assuming that there exist highly discriminative patterns among each class, then not very sophisticated rules are required to distinguish class objects. However, if patterns among each class are overlapping at different levels in some feature space, discriminative rules are hard to induce. Experiments conducted in Ref. [34] vary the degree of overlap between classes. It is then concluded that "the class imbalance distribution, by itself, does not seem to be a problem, but when allied to highly overlapped classes, it can significantly decrease the number of minority class examples correctly classified". A similar claim based on experiments is also reported in Ref. [9] as "Linearly separable domains do not sensitive to any amount of imbalance. As a matter of fact, as the degree of concept complexity increases, so does the system's sensitivity to imbalance."
- *Within-class concepts*: In many classification problems, a single class is composed of various sub-clusters, or sub-concepts. Samples of a class are collected from different sub-concepts. These sub-concepts do not always contain the same number of examples. This phenomena is referred to as *within-class imbalance*, corresponding to the imbalanced class distribution between classes [31]. The presence of within-class sub-concepts worsens the imbalance distribution problem (no matter between or within class) in two aspects: (1) the presence of within-class sub-concepts increases the learning concept complexity of the data set; and (2) the presence of within-class sub-concepts is implicit in most cases.

### 2.2. Reported research solutions

A number of solutions to the class imbalance problem are reported in the literature. These solutions are developed at both the data and algorithmic levels. At the data level, the objective is to re-balance the class distribution by resampling the data space. At the algorithm level, solutions try to adapt existing classifier learning algorithms to strengthen learning with regards to the small class. Cost-sensitive learning solutions incorporating both the data and algorithmic level approaches assume higher misclassification costs with samples in the rare class and seek to minimize the high cost errors. Several boosting algorithms are also reported as meta-techniques to tackle the class imbalance problem. These boosting approaches will be discussed in details in Section 4.6.

#### 2.2.1. Data-level approaches

Solutions at the data level include many different forms of resampling, such as randomly oversampling the small class, randomly undersampling the prevalent class, informatively oversampling the small class (in which no new samples are created, but the choice of samples to resample is targeted rather than random), informatively undersampling the prevalent class (the choice of samples to eliminate is targeted), oversampling the small class by generating new synthetic data, and combinations of the above techniques [1,20,21,35].

Even though resampling is an often-used method in dealing with the class imbalance problem, the matter at issue is what is or how to decide the optimal class distribution given a data set. A thorough experimental study on the effect of a training set's class distribution on a classifier's performance is conducted in Ref. [33]. The general conclusion is that, with respect to the classification performance on the small class, a balanced class distribution (class size ratio is 1:1) performs relatively well but is not necessarily optimal. Optimal class distributions differ from data to data.

In addition to the class distribution issue, how to effectively resample the training data is another issue. Random sampling is simple but not sufficient in many cases. For example, if the class imbalance problem of a data set is represented by within-class concepts, random oversampling may over-duplicate samples on some parts and less so on others. A more favorable resampling process should be, first, detecting the sub-concepts constituting the class; then, oversampling each concept, respectively, to balance the overall distribution. However, such an informative resampling process increases the cost for data analysis. Informatively undersampling the prevalent class attempting to make the selective samples more representative poses another problem: what is the criterion in selecting samples? For example, if samples are measured by some distance

measurements, those majority class samples which are relatively far away from the minority class samples may represent more majority class features, and those which are relatively close to the minority class samples may be crucial in deciding the class boundary with some classifier learning algorithms. Which part should be more focused on when selecting quality samples? These issues cannot be settled systematically. A number of techniques are reported, but each of them may be effective if applied in a certain context.

### 2.2.2. Algorithm-level approaches

Generally, a common strategy to deal with the class imbalance problem is to choose an appropriate inductive bias. For decision trees, one approach is to adjust the probabilistic estimate at the tree leaf [36,37]; another approach is to develop new pruning techniques [37]. For SVMs, proposals such as using different penalty constants for different classes [38], or adjusting the class boundary based on kernel-alignment ideal [13], are reported. For association rule mining, multiple minimum supports for different classes are specified to reflect their varied frequencies in the database [39]. To develop an algorithmic solution, one needs knowledge of both the corresponding classifier learning algorithm and the application domain, especially a thorough comprehension on why the learning algorithm fails when the class distribution of available data is uneven.

In recognition-based one-class learning, a system is modelled with only examples of the target class in the absence of the counter examples. This approach does not try to partition the hypothesis space with boundaries that separate positive and negative examples, but it attempts to make boundaries which surround the target concept. For classification purposes, it measures the amount of similarity between a query object and the target class, where a threshold on the similarity value is introduced. Two classifier learning algorithms are studied in the context of the one-class learning approach: neural network training [24] and SVMs [40]. Under certain conditions such as multi-modal domains, the one-class approach is reported to be superior to discriminative (two-class learning) approaches [24]. The threshold in this approach represents the boundary between the two classes. A too strict threshold means that positive data will be sifted, while a too loose threshold will include considerable negative samples. Hence, to set up an effective threshold is crucial with this approach. Moreover, many machine learning algorithms such as decision trees, Naïve Bayes and associative classification, do not function unless the training data includes examples from different classes.

### 2.2.3. Cost-sensitive learning

Cost-sensitive classification considers the varying costs of different misclassification types. A cost matrix encodes the penalty of classifying samples from one class as another. Let $C(i, j)$ denote the cost of predicting an instance from class $i$ as class $j$. With this notation, $C(+, -)$ is the cost of misclassifying a positive (rare class) instance as the negative (prevalent class) instance and $C(-, +)$ is the cost of the contrary case. In dealing with the class imbalance problem, the recognition importance of positive instances is higher than that of negative instances. Hence, the cost of misclassifying a positive instance outweighs the cost of misclassifying a negative one (i.e., $C(+, -) > C(-, +)$); making a correct classification usually presents 0 penalty (i.e., $C(+, +) = C(-, -) = 0$). The cost-sensitive learning process then seeks to minimize the number of high cost errors and the total misclassification cost.

A cost-sensitive classification technique takes the cost matrix into consideration during model building and generates a model that has the lowest cost. Reported works in cost-sensitive learning fall into three main categories:

- *Weighting the data space*: The distribution of the training set is modified with regards to misclassification costs, such that the modified distribution is biased towards the costly classes. This approach can be explained by the *translation theorem* derived in Ref. [41]. Against the normal space without considering the cost item, let us call a data space with domain $X \times Y \times C$ as the *cost-space*, where $X$ is the input space, $Y$ is the output space and $C$ is the cost associated with mislabelling that example. If we have examples drawn from a distribution $D$ in the cost-space, then we can have another distribution $\hat{D}$ in the normal space that

$$\hat{D}(X, Y) \equiv \frac{C}{E_{X,Y,C\sim D}[C]} D(X, Y, C) \tag{2.1}$$

  where $E_{X,Y,C\sim D}[C]$ is the expectation of cost values. According to the translation theorem, those optimal error rate classifiers for $\hat{D}$ will be optimal cost minimizers for $D$. Hence, when we update sample weights integrating the cost items, choosing a hypothesis to minimize the rate of errors under $\hat{D}$ is equivalent to choosing the hypothesis to minimize the expected cost under $D$.
- *Making a specific classifier learning algorithm cost-sensitive*: For example, in the context of decision tree induction, the tree-building strategies are adapted to minimize the misclassification costs. The cost information is used to: (1) choose the best attribute to split the data [4,42]; and (2) determine whether a subtree should be pruned [43].
- *Using Bayes risk theory to assign each sample to its lowest risk class*: For example, a typical decision tree for a binary classification problem assigns the class label of a leaf node depending on the majority class of the training samples that reach the node. A cost-sensitive algorithm assigns the class label to the node that minimizes the classification cost [37,44].

Methods in the first group, converting sample-dependent costs into sample weights, are also known as *cost-sensitive learning by example weighting* [45]. The weighted training samples are then applied to standard learning algorithms. This approach is at the data level without changing the underlying learning algorithms. Methods in the second and third groups, adapting the existing learning algorithms, are at the algorithm level. Cost-sensitive learning assumes that a cost matrix is

Table 1
Confusion matrix

| | Predicted as positive | Predicted as negative |
|---|---|---|
| Actually positive | True positives (TP) | False negatives (FN) |
| Actually negative | False positive (FP) | True negatives (TN) |

known for different types of errors or samples. Given a data set, however, the cost matrix is often unavailable.

### 2.3. Evaluation measures

Evaluation measures play a crucial role in both assessing the classification performance and guiding the classifier modelling. Traditionally, accuracy is the most commonly used measure for these purposes. However, for classification with the class imbalance problem, accuracy is no longer a proper measure since the rare class has very little impact on accuracy as compared to the prevalent class [10,46]. For example, in a problem where a rare class is represented by only 1% of the training data, a simple strategy can be to predict the prevalent class label for every example. It can achieve a high accuracy of 99%. However, this measurement is meaningless to some applications where the learning concern is the identification of the rare cases.

In the bi-class scenario, one class with very few training samples but high identification importance is referred to as the positive class; the other as the negative class. Samples can be categorized into four groups after a classification process as denoted in the confusion matrix presented in Table 1.

Several measures can be derived using the confusion matrix:

- True Positive Rate: $TP_{rate} = \dfrac{TP}{TP + FN}$.
- True Negative Rate: $TN_{rate} = \dfrac{TN}{TN + FP}$.
- False Positive Rate: $FP_{rate} = \dfrac{FP}{TN + FP}$.
- False Negative Rate: $FN_{rate} = \dfrac{FN}{TP + FN}$.
- Positive Predictive Value: $PP_{value} = \dfrac{TP}{TP + FP}$.
- Negative Predictive Value: $NP_{value} = \dfrac{TN}{TN + FN}$.

Clearly neither of these measures are adequate by themselves. For different evaluation criteria, several measures are devised.

#### 2.3.1. F-measure

If only the performance of the positive class is considered, two measures are important: True Positive Rate ($TP_{rate}$) and Positive Predictive Value ($PP_{value}$). In information retrieval, True Positive Rate is defined as *recall* denoting the percentage of retrieved objects that are relevant:

$$\underline{R}ecall = TP_{rate} = \frac{TP}{TP + FN}. \tag{2.2}$$

Positive Predictive Value is defined as *precision* denoting the percentage of relevant objects that are identified for retrieval:

$$\underline{P}recision = PP_{value} = \frac{TP}{TP + FP}. \tag{2.3}$$

$F$-measure ($F$) is suggested in Ref. [47] to integrate these two measures as an average

$$F\text{-}measure = \frac{2RP}{R + P}. \tag{2.4}$$

In principle, $F$-measure represents a harmonic mean between recall and precision [48]:

$$F\text{-}measure = \frac{2}{1/R + 1/P} \tag{2.5}$$

The harmonic mean of two numbers tends to be closer to the smaller of the two. Hence, a high $F$-measure value ensures that both recall and precision are reasonably high.

#### 2.3.2. G-mean

When the performance of both classes is concerned, both True Positive Rate ($TP_{rate}$) and True Negative Rate ($TN_{rate}$) are expected to be high simultaneously. Kubat et al. [3] suggested the $G$-mean defined as

$$G\text{-}mean = \sqrt{TP_{rate} \cdot TN_{rate}}. \tag{2.6}$$

$G$-mean measures the balanced performance of a learning algorithm between these two classes. The comparison among harmonic, geometric, and arithmetic means are illustrated in Ref. [48] by way of an example. Suppose that there are two positive numbers 1 and 5. Their arithmetic mean is 3, their geometric mean is 2.236, and their harmonic mean is 1.667. The harmonic mean is the closest to the smaller value and the geometric mean is closer than the arithmetic mean to the smaller number.

#### 2.3.3. ROC analysis

Some classifiers, such as Bayesian network inference or some neural networks, assign a probabilistic score to its prediction. Class prediction can be changed by varying the score threshold. Each threshold value generates a pair of measurements of ($FP_{rate}$, $TP_{rate}$). By linking these measurements with the False Positive Rate ($FP_{rate}$) on the $X$-axis and the True Positive Rate ($TP_{rate}$) on the $Y$-axis, a ROC graph is plotted. The ideal model is one that obtains 1 True Positive Rate and 0 False Positive Rate (i.e., $TP_{rate} = 1$ and $FP_{rate} = 0$). A model that makes a random guess should reside along the line connecting the points ($TP_{rate} = 0$, $FP_{rate} = 0$), where every instance is predicted as a negative class, and ($TP_{rate} = 1$, $FP_{rate} = 1$), where every instance is predicted as a positive class. A ROC graph depicts relative trade-offs between benefits (true positives) and costs (false positives) across a range of thresholds of a classification model. A ROC curve gives a good summary of the performance of a classification model. To compare several classification models by comparing ROC curves, it is hard to claim a winner unless one curve clearly dominates the others over the entire space [49]. The area under a ROC curve

(AUC) provides a single measure of a classifier's performance for evaluating which model is better on average. It has been shown in Ref. [50] that there is a clear similarity between AUC and well-known Wilcoxon statistics.

## 3. Why boosting?

### 3.1. Ensemble learning

The basic idea of classifier ensemble learning is to construct multiple classifiers from the original data and then aggregate their predictions when classifying unknown samples. The main motivation for combining classifiers in redundant ensembles is to improve their generalization ability: each component classifier is known to make errors with the assumption that it has been trained on a limited set of data, however, the patterns that are misclassified by the different classifiers are not necessarily the same [54]. The effect of combining redundant ensembles is also studied in terms of the statistical concepts of bias and variance. Given a classifier, bias–variance decomposition distinguishes among the *bias error*, the *variance error* and the *intrinsic error*. The bias can be characterized as a measure of its ability to generalize correctly to a test set, while the variance can be similarly characterized as a measure of the extent to which the classifier's prediction is sensitive to the data on which it was trained. The variance is then associated with overfitting: if a method overfits the data, the predictions for a single instance will vary between samples [55]. The improvement in performance arising from ensemble combinations is usually the result of a reduction in variance. This occurs because the usual effect of ensemble averaging is to reduce the variance of a set of classifiers. Bagging [51], Random forest [52] and AdaBoost [29,53] are all reported ensemble learning methods to be successful in variance reduction.

AdaBoost is also observed to be capable of bias reduction. As *stumps* (single-split trees with only two terminal nodes typically have low variance but high bias) are used as the base learner, Bagging performs very poorly and AdaBoost improves the base classification significantly [29,53]. The AdaBoost algorithm weighs each sample reflecting its importance and places the most weights on those examples which are most often misclassified by the preceding classifiers. Such a focus may cause the learner to produce an ensemble function that differs significantly from the single learning algorithm.

With an imbalanced data set, small class samples occurring infrequently, models that describe the rare classes have to be highly specialized. Standard learning methods pay less attention to the rare samples as they try to extract the regularities from the data set. Such a model performs poorly on the rare class due to the introduced bias error. AdaBoost attempts to reduce the bias error as it focuses on misclassified examples [25]. The sample weighting strategy of AdaBoost is equivalent to resampling the data space combining both up-sampling and down-sampling. It hence belongs to data-level solutions, which are applicable to most classification systems without changing their learning methods. Therefore, the advantages of AdaBoost for learning imbalanced data are:

1. a boosting algorithm is applicable to most classification systems;
2. resampling the data space automatically eliminates the extra learning cost for exploring the optimal class distribution and the representative samples;
3. resampling the data space through weighting each sample results in little information loss as compared with eliminating some samples from the data set;
4. combining multiple classifications has little risk of model overfitting; and
5. AdaBoost is capable of reducing the bias error of a certain classification learning method.

These positive features make the boosting approach an attractive technique in tackling the class imbalance problem. Given a data set with imbalanced class distribution, misclassified samples are often in the minority class. When the AdaBoost algorithm is applied, samples in the minority class may receive more weights; and that the successive learning will focus on the minority class. Intuitively, the AdaBoost algorithm might improve the classification performance on the small class. However, experimental results reported in Refs. [46,56,57] show that the improved identification performances on the small class are not always guaranteed or satisfactory. The straightforward reason is that AdaBoost is accuracy-oriented: its weighting strategy may bias towards the prevalent class since it contributes more to the overall classification accuracy. Hence, the learning issue becomes how to adapt the AdaBoost algorithm to incline its boosting strategy towards the interested class.

Random forest is also adapted for classification of imbalanced data in Ref. [58]. As Random forest is specially designed only for decision tree classifiers, we leave it without further consideration.

### 3.2. AdaBoost algorithm

AdaBoost algorithm reported in Refs. [26,27] takes as input a training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ where each $x_i$ is an $n$-tuple of attribute values belonging to a certain domain or instance space $X$, and $y_i$ is a label in a label set $Y$. In the context of bi-class applications, we can express $Y = \{-1, +1\}$. The Pseudocode for AdaBoost is given in Fig. 1.

It has been shown in Ref. [27] that the training error of the final classifier is bounded as

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leqslant \prod_t Z_t, \tag{3.3}$$

where

$$Z_t = \sum_i D^t(i)\ \exp(-\alpha_t y_i h_t(x_i)) \tag{3.4}$$

$$= \sum_i D^t(i) \left( \frac{1 + y_i h_t(x_i)}{2} e^{-\alpha} + \frac{1 - y_i h_t(x_i)}{2} e^{\alpha} \right) \tag{3.5}$$

Given: $(x_1, y_1), \cdots, (x_m, y_m)$ where $x_i \in X$, $y_i \in Y = \{-1, +1\}$

Initialize $D^1(i) = 1/m$.

For $t = 1, \cdots, T$:

1. Train base learner $h_t \to Y$ using distribution $D^t$
2. Choose weight updating parameter: $\alpha_t$
3. Update and normalize sample weights:

$$D^{t+1}(i) = \frac{D^t(i) exp(-\alpha_t h_t(x_i) y_i)}{Z_t} \tag{3.1}$$

Where, $Z_t$ is a normalization factor.

Output the final classifier:

$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x)) \tag{3.2}$$

Fig. 1. AdaBoost algorithm.

minimizing $Z_t$ on each round, $\alpha_t$ is induced as

$$\alpha_t = \frac{1}{2} \log \left( \frac{\sum_{i, y_i = h_t(x_i)} D^t(i)}{\sum_{i, y_i \neq h_t(x_i)} D^t(i)} \right). \tag{3.6}$$

The sample weight updating goal of AdaBoost is to decrease the weight of training samples which are correctly classified and increase the weights of the opposite part. Therefore, $\alpha_t$ should be a positive value, which demands the training error should be less than randomly guessing (0.5) based on the current data distribution, that is

$$\sum_{i, y_i = h_t(x_i)} D^t(i) > \sum_{i, y_i \neq h_t(x_i)} D^t(i). \tag{3.7}$$

### 3.3. Forward stagewise additive modelling

It has been shown that AdaBoost is equivalent to forward stagewise additive modelling using exponential loss function. The exponential loss function is related to the Bernoulli likelihood [29]. From this point, the rather obscure work of the computational learning is well explored in a likelihood method of standard statistical practice [30]. The exponential loss function is defined as

$$L(y, f(x)) = \exp(-yf(x)), \tag{3.8}$$

where

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{3.9}$$

so that $H(x) = \text{sign}(f(x))$. Such on each round, one must solve

$$(\alpha_t, h_t) = \arg\min_{\alpha, h} \sum_i \exp[-y_i (f_{t-1}(x_i) + \alpha h(x_i))] \tag{3.10}$$

$$= \arg\min_{\alpha, h} \sum_i D^t(i) \exp(-\alpha y_i h(x_i)), \tag{3.11}$$

where $D^t(i) = \exp(-y_i f_{m-1}(x_i))$. The solution to Eq. (3.11) is then obtained in two steps. First, for any value of $\alpha > 0$,

$$h_t = \arg\min_h \sum_i D^t(i) I[y_i \neq h_t(x_i)], \tag{3.12}$$

where for any predicate $\pi$, $I[\pi]$ equals 1 if $\pi$ holds, 0 otherwise. Therefore, $h_t$ is the classifier that minimizes the weighted error rate based on the current data distribution. Once the classifier is fixed, the second step is to decide the value of $\alpha$ to minimize right side of Eq. (3.11). This job is the same with the learning objective of AdaBoost (Eq. (3.4)). Then $\alpha$ can be fixed as stated in Eq. (3.6). The approximation is then updated as

$$f_t(x) = f_{t-1}(x) + \alpha_t h_t(x) \tag{3.13}$$

which causes the weights for next iteration will be

$$D^{t+1}(i) = D^t(i) \cdot \exp(-\alpha_t y_i h_t(x_i)). \tag{3.14}$$

It has been proved in Refs. [27,29], after each update to the weights, the weighted sample distributions between the misclassified part and the correctly classified part are even, i.e., $\sum_{i, h_t(x_i) = y_i} D^{t+1}(i) = \sum_{i, h_t(x_i) \neq y_i} D^{t+1}(i)$. This makes the new weighted problem maximally difficult for next iteration.

## 4. Cost-sensitive boosting algorithms

The weighting strategy of AdaBoost is to increase weights of misclassified samples and decrease weights of correctly classified samples until the weighted sample distributions between misclassified samples and correctly classified samples are even on each round. This weighting strategy distinguishes samples on their classification outputs: correctly classified or misclassified. However, it treats samples of different types (classes) equally: weights of misclassified samples from different classes are increased by an identical ratio, and weights of correctly

classified samples from different classes are decreased by another identical ratio. The learning objective in dealing with the imbalance class problem is to improve the identification performance on the small class. This learning objective expects that the weighting strategy of a boosting algorithm will preserve a considerable weighted sample size of the small class. A desirable boosting strategy is one which is able to distinguish different types of samples, and boost more weights on those samples associated with higher identification importance.

To denote the different identification importance among samples, each sample is associated with a cost item: the higher the value, the higher the importance of correctly identifying this sample. Let $\{(x_1, y_1, C_1), \ldots, (x_m, y_m, C_m)\}$ be a sequence of training samples, where each $x_i$ is an $n$-tuple of attribute values; $y_i$ is a class label in $Y = \{-1, +1\}$; and $C_i \subset [0, +\infty)$ is an associated cost item. For an imbalanced data set, samples with class label $y = -1$ are much more than samples with class label $y = +1$. As the learning objective is to improve the identification performance on the small class, the cost values associated with samples of the small class can be set higher than those associated with samples of the prevalent class. Keeping the same learning framework of AdaBoost, the cost items can be fed into the weight update formula of AdaBoost (Eq. (3.1)) to bias its weighting strategy. There are three ways to introduce cost items into the weight update formula of AdaBoost: inside the exponent, outside the exponent, and both inside and outside the exponent. Three modifications of Eq. (3.1) then become

- Modification I:

$$D^{t+1}(i) = \frac{D^t(i)\exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t}. \tag{4.1}$$

- Modification II:

$$D^{t+1}(i) = \frac{C_i D^t(i)\exp(-\alpha_t h_t(x_i) y_i)}{Z_t}. \tag{4.2}$$

- Modification III:

$$D^{t+1}(i) = \frac{C_i D^t(i)\exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t}. \tag{4.3}$$

Each modification can be taken as a new boosting algorithm denoted as AdaC1, AdaC2 and AdaC3, respectively. As these algorithms use cost items, they can also be regarded as cost-sensitive boosting algorithms. For the AdaBoost algorithm, the selection of the weight update parameter is crucial in converting a weak learning algorithm into a strong one [29]. When the cost items are introduced into the weight updating formula of the AdaBoost algorithm, the updated data distribution is affected by the cost items. Without re-inducing the weight update parameter, which takes the cost items into consideration for each cost-sensitive boosting algorithm, the boosting efficiency is not guaranteed. With the AdaBoost algorithm, the weight update parameter $\alpha$ is calculated to minimize the overall training error of the combined classifier. Using the same inference method, we induce the weight update parameter $\alpha$ for each algorithm.

### 4.1. AdaC1

Unravelling the weight update rule of Eq. (4.1), we obtain

$$D^{t+1}(i) = \frac{\exp(-\sum_t \alpha_t C_i y_i h_t(x_i))}{m\prod_t Z_t} = \frac{\exp(-C_i y_i f(x_i))}{m\prod_t Z_t}, \tag{4.4}$$

where

$$Z_t = \sum_i D^t(i)\exp(-\alpha_t C_i y_i h_t(x_i)) \tag{4.5}$$

and

$$f(x_i) = \sum_t \alpha_t h_t(x_i). \tag{4.6}$$

The over all training error is bounded as

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leqslant \frac{1}{m}\sum_i \exp(-C_i y_i f(x_i)) \tag{4.7}$$

$$= \sum_i \left(\prod_t Z_t\right) D^{t+1}(i) = \prod_t Z_t. \tag{4.8}$$

Thus, the learning objective on each boosting iteration is to find $\alpha_t$ so as to minimize $Z_t$ (Eq. (4.5)). According to Ref. [27], once $c_i y_i h_t(x_i) \in [-1\ \ +1]$, the following inequality holds:

$$\sum_i D^t(i)\exp(-\alpha C_i y_i h(x_i))$$

$$\leqslant \sum_i D^t(i)\left(\frac{1+C_i y_i h_t(x_i)}{2}e^{-\alpha} + \frac{1-C_i y_i h_t(x_i)}{2}e^{\alpha}\right). \tag{4.9}$$

By zeroing the first derivative of the right-hand side of inequality (4.9), $\alpha_t$ can be determined as

$$\alpha_t = \frac{1}{2}\log\frac{1 + \sum_{i, y_i = h_t(x_i)} C_i D^t(i) - \sum_{i, y_i \neq h_t(x_i)} C_i D^t(i)}{1 - \sum_{i, y_i = h_t(x_i)} C_i D^t(i) + \sum_{i, y_i \neq h_t(x_i)} C_i D^t(i)}. \tag{4.10}$$

To ensure that the selected value of $\alpha_t$ is positive, the following condition should hold:

$$\sum_{i, y_i = h_t(x_i)} C_i D^t(i) > \sum_{i, y_i \neq h_t(x_i)} C_i D^t(i). \tag{4.11}$$

### 4.2. AdaC2

Unravelling the weight update rule of Eq. (4.2), we obtain

$$D^{t+1}(i) = \frac{C_i^t \exp(-\sum_t \alpha_t y_i h_t(x_i))}{m\prod_t Z_t} = \frac{C_i^t \exp(-y_i f(x_i))}{m\prod_t Z_t}, \tag{4.12}$$

where $f(x_i)$ is the same as defined in Eq. (4.6) and

$$Z_t = \sum_i C_i D^t(i)\exp(-\alpha_t y_i h_t(x_i)). \tag{4.13}$$

Then, the training error of the final classifier is bounded as

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leqslant \frac{1}{m}\sum_i \exp(-y_i f(x_i)) = \prod_t Z_t \sum_i \frac{C_i D^t(i)}{C_i^{(t+1)}}, \quad (4.14)$$

where $C_i^{(t+1)}$ denotes the $(t+1)$th power of $C_i$. There exists a constant $\gamma$ such that $\forall i, \gamma < C_i^{(t+1)}$. Then

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leqslant \prod_t Z_t \sum_i \frac{C_i D^t(i)}{C_i^{(t+1)}} \leqslant \frac{1}{\gamma}\prod_t Z_t. \quad (4.15)$$

Since $\gamma$ is a constant, the learning objective at each boosting iteration is to find $\alpha_t$ so as to minimize $Z_t$ (Eq. (4.13)). Using the same inferring method as in AdaC1, $\alpha_t$ is uniquely selected as

$$\alpha_t = \frac{1}{2}\log\frac{\sum_{i,y_i=h_t(x_i)} C_i D^t(i)}{\sum_{i,y_i\neq h_t(x_i)} C_i D^t(i)}. \quad (4.16)$$

To ensure that the selected value of $\alpha_t$ is positive, the following condition should hold:

$$\sum_{i,y_i=h_t(x_i)} C_i D^t(i) > \sum_{i,y_i\neq h_t(x_i)} C_i D^t(i). \quad (4.17)$$

### 4.3. AdaC3

The weight update formula (Eq. (4.3)) of AdaC3 is a combination of that of AdaC1 and AdaC2 (with the cost items being both inside and outside the exponential function). Then the training error bound of AdaC3 can be expressed as

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leqslant \frac{1}{\gamma}\prod_t Z_t, \quad (4.18)$$

where $\gamma$ is a constant and $\forall i, \gamma < C_i^{(t+1)}$, and

$$Z_t = \sum_i C_i D^t(i)\exp(-\alpha_t C_i y_i h_t(x_i)). \quad (4.19)$$

Since $\gamma$ is a constant, the learning objective at each boosting iteration is to find $\alpha_t$ so as to minimize $Z_t$ (Eq. (4.19)). Restricting $C_i y_i h_t(x_i) \in [-1\ \ +1]$, $\alpha_t$ can be determined as

$$\alpha_t = \frac{1}{2}\log\frac{\sum_i C_i D^t(i) + \sum_{i,y_i=h_t(x_i)} C_i^2 D^t(i) - \sum_{i,y_i\neq h_t(x_i)} C_i^2 D^t(i)}{\sum_i C_i D^t(i) - \sum_{i,y_i=h_t(x_i)} C_i^2 D^t(i) + \sum_{i,y_i\neq h_t(x_i)} C_i^2 D^t(i)}. \quad (4.20)$$

To ensure that the selected value of $\alpha_t$ is positive, the following condition should hold:

$$\sum_{i,y_i=h_t(x_i)} C_i^2 D^t(i) > \sum_{i,y_i\neq h_t(x_i)} C_i^2 D^t(i). \quad (4.21)$$

### 4.4. Cost-sensitive exponential loss and AdaC2

AdaC2 tallies with the stagewise additive modelling, where steepest descent search is carried on to minimize the overall cost loss under the exponential function. By integrating a cost item $C$ into Eq. (3.8), the cost-sensitive exponential loss function becomes

$$C \cdot L(y, f(x)) = C \cdot \exp(-yf(x)). \quad (4.22)$$

The goal is to learn a classifier which minimizes the expected cost loss under the exponential function. On each iteration, $h_t$ and $\alpha_t$ are learned separately to solve

$$(\alpha_t, h_t) = \arg\min_{\alpha,h}\sum_i C_i \cdot \exp[-y_i(f_{t-1}(x_i) + \alpha h(x_i))] \quad (4.23)$$

$$= \arg\min_{\alpha,h}\sum_i C_i \cdot D^t(i)\exp(-\alpha y_i h(x_i)), \quad (4.24)$$

where $D^t(i) = \exp(-y_i f_{m-1}(x_i))$. The solution to Eq. (4.24) is obtained in two steps. First, for any value of $\alpha > 0$, $h_t$ is the one which minimizes the cost error, which is

$$h_t = \arg\min_h \sum_i C_i \cdot D^t(i) I[y_i \neq h_t(x_i)]. \quad (4.25)$$

Standard classification learning algorithms minimize the error rate instead of the expected cost. Here, translation theorem (2.1) can be applied to solve this problem. If we weight each sample by its cost item, we obtain a distribution in the normal space

$$\hat{D}^t(i) = C_i \cdot D^t(i). \quad (4.26)$$

Then, those optimal error rate classifiers for $\hat{D}$ will be optimal cost minimizers for $D$. Thus, $h_t$ can be fixed to minimize the error rate for $\hat{D}^t$ which is equivalent to minimizing cost error for $D^t$. Once the classifier is fixed, the second step is to decide the value of $\alpha$ to minimize the right side of Eq. (4.24). This job shares with the learning objective of AdaC2 (Eq. (4.13)). $\alpha$ is then fixed as stated in Eq. (4.16). The approximation is then updated as

$$f_t(x) = f_{t-1}(x) + \alpha_t h_t(x) \quad (4.27)$$

which causes the weights for the next iteration to be

$$D^{t+1}(i) = D^t(i) \cdot \exp(-\alpha_t y_i h_t(x_i)). \quad (4.28)$$

To minimize the cost-sensitive exponential loss (Eq. (4.22)), the learning objective on each round is to minimize the expected cost (Eq. (4.25)). By applying the translation theorem, each sample is reweighted by its cost factor. Therefore, each sample weight for learning of the next iteration is updated as

$$\hat{D}^{t+1}(i) = C_i \cdot D^t(i) \cdot \exp(-\alpha_t y_i h_t(x_i)). \quad (4.29)$$

Using the method in Refs. [27,29], it can be proved that for the next iteration we will have

$$\sum_{i,h_t(x_i)=y_i} \hat{D}^{t+1}(i) = \sum_{i,h_t(x_i)\neq y_i} \hat{D}^{t+1}(i). \quad (4.30)$$

This makes the learning of the next iteration maximally difficult.

### 4.5. Cost factors

For cost-sensitive boosting algorithms, the cost items are used to characterize the identification importance of different samples. The cost value of a sample may depend on the nature of the particular case [59]. For example, in detection of fraud, the cost of missing a particular case of fraud will depend on the amount of money involved in that particular case [2]. Similarly, the cost of a certain kind of mistaken medical diagnosis may be conditional on the particular patient who is misdiagnosed [59]. In the case that the misclassification costs or learning importance for samples in one class are the same, a unique number can be set up for each class. For a bi-class imbalanced data set, there will be two cost items: $C_P$ denoting the learning importance (misclassification cost) of the positive (small) class and $C_N$ denoting that of the negative (prevalent) class. Since the purpose of the cost-sensitive boosting is to boost a larger class size on the positive class, $C_P$ should be set greater than $C_N$. With a higher cost value on the positive class, a considerable weighted sample size of the positive class is boosted to strengthen learning. Consequently, more relevant samples can be identified.

Referring to the confusion matrix Table 1, the recall value (Eq. (2.2)) measures the percentage of retrieved objects that are relevant. A higher positive recall value is more favorable for a bi-class imbalanced data set based on the fact that misclassifying a positive sample as a negative one will cost much more than the reverse. There are some econometric applications, like credit card fraud detection, misclassifying a valuable customer as a fraud may cost much more than the opposite case in the current climate of intense competition. The cost of misclassifying a negative case is regarded as higher than misclassifying a positive sample [60]. For this kind of application, we still associate a higher cost value with the positive class. By applying the cost-sensitive boosting algorithm, many more relevant samples are included to generate a "denser" data set for further analysis, and so a conclusive decision.

Given a data set, the cost setup is usually unknown. For a binary application, the cost values can be decided using empirical methods. Suppose the learning objective is to improve the identification performance on the positive class. This learning objective expects a higher $F$-measure value (Eq. (2.4)) on the positive class. As stated previously, with a higher cost value of the positive class than that of negative class, more weights are expected to be boosted on the positive class, and the recall value of the positive class is improved. However, if weights are overboosted on the positive class, more irrelevant samples will be included simultaneously. The precision value (Eq. (2.3)), measuring the percentage of relevant objects in the set to all objects returned by a search, decreases. Hence, there is a trade-off between recall and precision values: when recall value increases, precision value decreases. To get a better $F$-measure value, weights boosted on the positive class should be fair in order to balance the recall and precision values. Therefore, cost values can be tested by evaluating the $F$-measure value iteratively. The situation is similar if the learning objective is to balance the classification performance evaluated by $G$-mean (Eq. (2.6)).

As stated in Ref. [61], given a set of cost setups, the decisions are unchanged if each one in the set is multiplied by a positive constant. This scaling corresponds to changing the accounting unit of costs. Hence, it is the ratio between $C_P$ and $C_N$ that denotes the deviation of the learning importance between two classes. Therefore, the job of searching for an effective cost setup for applying cost-sensitive boosting algorithms is actually to obtain a proper ratio between $C_P$ and $C_N$, for a better performance according to the learning objective.

### 4.6. Other related algorithms

There are some other reported boosting algorithms for classification of imbalanced data in literature. These boosting algorithms can be categorized into two groups: the first group represents those that can be applied to most classifier learning algorithms directly, such as AdaCost [56], CSB1 and CSB2 [57], and RareBoost [46]; the second group includes those that are based on a combination of the data synthesis algorithm and the boosting procedure, such as SMOTEBoost [62], and DataBoost-IM [63]. Synthesizing data may be application-dependent and hence involves extra learning cost. We only consider boosting algorithms that can be applied directly to most classification learning algorithms. Among this group, AdaCost [56], CSB1 and CSB2 [57] employ cost items to bias the boosting towards the small class, and RareBoost [46] has been developed to directly address samples of the four types as tabulated in Table 1.

#### 4.6.1. AdaCost

In AdaCost, Eq. (3.1) is replaced by

$$D^{t+1}(i) = D^t(i) \cdot \exp(-\alpha_t y_i h_t(x_i) \beta_{\mathrm{sgn}(h_t(x_i), y_i)}), \tag{4.31}$$

where $\beta$ is called a *cost adjustment function*. The requirement for this function is as follows: for an instance with a higher cost factor, the function increases its weight "more" if the instance is misclassified, but decreases its weight "less" otherwise. In Ref. [56], the authors provide their recommended setting as: $\beta_+ = -0.5C_n + 0.5$ and $\beta_- = 0.5C_n + 0.5$, where $C_n$ is the cost of misclassifying the $n$th example.

In AdaCost, $\alpha_t$ is calculated as

$$\alpha_t = \frac{1}{2} \log \frac{1 + r_t}{1 - r_t}, \tag{4.32}$$

where

$$r_t = \sum_i D^t(i) \exp(-\alpha_t y_i h_t(x_i) \beta_{\mathrm{sgn}(h_t(x_i), y_i)}). \tag{4.33}$$

AdaCost is a variation of AdaC1 by introducing a cost adjustment function instead of a cost item inside the exponential func-

tion. However, the selection of the cost adjustment function is somehow ad hoc, and when the cost factors are set equally for both positive and negative class, the AdaCost algorithm will not reduce to the AdaBoost algorithm.

#### 4.6.2. CSB1 and CSB2

CSB1 modifies weight update formula of AdaBoost (Eq. (3.1)) to

$$D^{t+1}(i) = \frac{D^t(i)C_{\mathrm{sgn}(h_t(x_i),y_i)}\exp(-y_i h_t(x_i))}{Z_t}. \tag{4.34}$$

And CSB2 changes it to

$$D^{t+1}(i) = \frac{D^t(i)C_{\mathrm{sgn}(h_t(x_i),y_i)}\exp(-\alpha_t y_i h_t(x_i))}{Z_t}, \tag{4.35}$$

where $\mathrm{sgn}(h_t(x_i), y_i)$ denotes "+" if $h_t(x_i)$ equals to $y_i$ ($x_i$ is correctly classified), "–" otherwise. The parameters $C_+$ and $C_-$ are set as $C_+ = 1$ and $C_- = cost(y_i, h_t(x_i)) \geqslant 1$, where $cost(i, j)$ is the cost of misclassifying a sample of class $i$ to class $j$. For the weight updating of the next iteration, CSB1 does not use any $\alpha_t$ factor (or $\alpha_t = 1$) and CSB2 uses the same $\alpha_t$ as computed by AdaBoost. Even though the weight update formula of CSB2 is similar to AdaC2, CSB2 does not reference the weight update parameter $\alpha$ by taking the cost set up into consideration. Affected by the cost factors, the boosting efficiency of AdaBoost varies by both CSB1 and CSB2.

#### 4.6.3. RareBoost

RareBoost scales False Positive examples in proportion to how well they are distinguished from True Positive examples, and scales False Positive examples in proportion to how well they are distinguished from True Negative examples (refer to Table 1). In their algorithm, the weight update factor $\alpha_t^p$ for positive predictions at the $t$th iteration is calculated as

$$\alpha_t^p = \frac{1}{2}\ln\frac{TP_t}{FP_t}, \tag{4.36}$$

where $TP_t$ and $FP_t$ denote the weight summation over all True Positive examples and False Positive examples, respectively. The weight update factor $\alpha_t^n$ for negative predictions at the $t$th iteration is calculated as

$$\alpha_t^n = \frac{1}{2}\ln\frac{TN_t}{FN_t} \tag{4.37}$$

where $TN_t$ and $FN_t$ denote the weight summation over all True Negative examples and False Negative examples, respectively. Weights are then updated separately using different factors respecting positive predictions and negative predictions.

This weighting strategy decreases the weights of True Positives (TP) and True Negatives (TN), and increases the weights of False Positives (FP) and False Negatives (FN) only if $TP > FP$ and $TN > FN$. The constraint of $TP > FP$ is equivalent to that the precision measure (Eq. (2.3)) of the positive class should be greater than 0.5. In the presence of the class imbalance problem, the small class is always associated with both poor recall and precision values. Hence, such a constraint is a strong condition. Without this condition being satisfied, the algorithm will collapse. We therefore discard this algorithm without further consideration.

## 5. Resampling effects

In this section, we will show how each boosting algorithm updates weights corresponding to the four types of examples tabulated in Table 1. Here we are not trying to figure out how the weight of a specific training sample will change over all the iterations, given that its role among TP and FN or FP and TN will switch from iteration to iteration. Our interest is on how the weight updating mechanism of each boosting algorithm treats the four groups of samples differently. In this paper, our study concentrates on AdaBoost [26,27], AdaCost [56], CSB2 [57], and the proposed three boosting algorithms, AdaC1, AdaC2, and AdaC3. For cost-sensitive boosting algorithms, we use $C_P$ to denote the misclassification cost of the positive class and $C_N$ for that of the negative class. This study is inspired by the method used in Ref. [46].

Referring to Eqs. (3.1), (4.1)–(4.3), (4.31), and (4.35), AdaBoost, AdaC1, AdaC2, AdaC3, AdaCost and CSB2 update sample weights on these four groups from the $t$th iteration to the $(t + 1)$th iteration as summarized in Table 2.

For the AdaBoost algorithm, weights of False Negatives and False Positives are improved equally; weights of True Positives and True Negatives are decreased equally with $\alpha_t$ being a positive number. The learning objective in dealing with the imbalance class problem is to obtain a satisfactory identification performance on the positive (small) class. This learning objective expects that the weighting strategy of a boosting algorithm preserves a considerable weighted sample size of the small class. A desirable weight updating rule is to increase the weights of False Negatives more than those of False Positives, but decrease the weights of True Positives more conservatively than those of True Negatives [56]. The resampling strategies of each cost-sensitive boosting algorithm are:

- AdaC1: False Negatives get more weight increase than False Positives with $e^{C_P\cdot\alpha^t} > e^{C_N\cdot\alpha^t}$; True Positives lose more weights than True Negatives with $1/e^{C_P\cdot\alpha_t} < 1/e^{C_N\cdot\alpha_t}$.
- AdaC2: False Negatives receive a greater weight increase than False Positives; True Positives lose less weights than True Negatives given $C_P > C_N$.

Table 2
Weighting strategies

| | | |
|---|---|---|
| AdaBoost | $TP_{t+1} = TP_t/e^{\alpha_t}$ | $FP_{t+1} = FP_t \cdot e^{\alpha^t}$ |
| | $TN_{t+1} = TN_t/e^{\alpha_t}$ | $FN_{t+1} = FN_t \cdot e^{\alpha^t}$ |
| AdaC1 | $TP_{t+1} = TP_t/e^{C_P\cdot\alpha_t}$ | $FP_{t+1} = FP_t \cdot e^{C_N\cdot\alpha^t}$ |
| | $TN_{t+1} = TN_t/e^{C_N\cdot\alpha_t}$ | $FN_{t+1} = FN_t \cdot e^{C_P\cdot\alpha^t}$ |
| AdaC2 | $TP_{t+1} = C_P \cdot TP_t/e^{\alpha_t}$ | $FP_{t+1} = C_N \cdot FP_t \cdot e^{\alpha^t}$ |
| | $TN_{t+1} = C_N \cdot TN_t/e^{\alpha_t}$ | $FN_{t+1} = C_P \cdot FN_t \cdot e^{\alpha^t}$ |
| AdaC3 | $TP_{t+1} = C_P \cdot TP_t/e^{C_P\cdot\alpha_t}$ | $FP_{t+1} = C_N \cdot FP_t \cdot e^{C_N\cdot\alpha^t}$ |
| | $TN_{t+1} = C_N \cdot TN_t/e^{C_N\cdot\alpha_t}$ | $FN_{t+1} = C_P \cdot FN_t \cdot e^{C_P\cdot\alpha^t}$ |
| AdaCost | $TP_{t+1} = TP_t/e^{\beta_+^+\cdot\alpha_t}$ | $FP_{t+1} = FP_t \cdot e^{\beta_-^-\cdot\alpha^t}$ |
| | $TN_{t+1} = TN_t/e^{\beta_+^-\cdot\alpha_t}$ | $FN_{t+1} = FN_t \cdot e^{\beta_-^+\cdot\alpha^t}$ |
| CSB2 | $TP_{t+1} = TP_t/e^{\alpha_t}$ | $FP_{t+1} = C_N \cdot FP_t \cdot e^{\alpha^t}$ |
| | $TN_{t+1} = TN_t/e^{\alpha_t}$ | $FN_{t+1} = C_P \cdot FN_t \cdot e^{\alpha^t}$ |

Table 3
Resampling effects

| An ideal weighting strategy | True predictions | False predictions |
|---|---|---|
| | Decreases the weights of True Positives more conservatively than those of True Negatives | Increases the weights of False Negatives more than those of False Positives |
| AdaBoost | × | × |
| AdaC1 | × | ∨ |
| AdaC2 | ∨ | ∨ |
| AdaC3 | Uncertain | ∨ |
| AdaCost | ∨ | ∨ |
| CSB2 | × | ∨ |

- AdaC3: Sample weights are updated by the combinational results of AdaC1 and AdaC2. As both AdaC1 and AdaC2 increase more weights on False Negatives than False Positives, AdaC3 furthers this effect. On the correctly classified part, AdaC1 decreases weights of True Positives more than those of True Negatives, while AdaC2 preserves more weights on True Positives than True Negatives. Due to the complicated situations of training error and cost setups, it is difficult to decide when AdaC3 preserves more weights or decreases more weights on True Positives.
- AdaCost: False Negatives receive a greater weight increase than False Positives and True Positives loss less weights than True Negatives by using a cost adjustment function. The recommended cost adjustment function is: $\beta_+ = -0.5C_n + 0.5$ and $\beta_- = 0.5C_n + 0.5$, where $C_n$ is the misclassification cost of the $n$th example, $\beta_+$ ($\beta_-$) denotes the output in case of the sample correctly classified (misclassified). Since the cost factors for instances in the positive class is set greater than those for instances in the negative class, $\beta_+^+ < \beta_+^-$ and $\beta_-^+ > \beta_-^-$, where $\beta_+^+$ ($\beta_-^+$) denotes the outputs for correctly classified (misclassified) samples in the positive class, $\beta_+^-$ ($\beta_-^-$) denotes the outputs for correctly classified (misclassified) samples in the negative class.
- CSB2: Weights of True Positives and True Negatives are decreased equally; False Negatives get more boosted weights than False Positives.

Their resampling effects regarding the boosting objective for learning imbalanced data are summarized in Table 3. In brief, all cost-sensitive boosting algorithms increase the weights of False Negatives more than those of False Positives. On the true predictions (True Positive and True Negative), their weighting strategies are different. AdaC2 and AdaCost are preferable by preserving more weights on the True Positives; AdaC1 and CSB2 are on the opposite; and AdaC3 is uncertain as it is a combinational result of AdaC1 and AdaC2.

## 6. Experiments

In this section, we set up experiments to investigate boosting algorithms—AdaBoost, AdaC1, AdaC2, AdaC3, AdaCost and CSB2—in terms of their capabilities in dealing with the class imbalance problem.

To test and compare these boosting algorithms, two kinds of classification systems are specially selected as the base classifiers: one is the well-known and widely used decision tree classification system C4.5 and another one is an associative classification system. C4.5 remains a popular classifier for research on the class imbalance problem [1]. Association mining is reported sensitive to the class imbalance problem as significant associations among rarely occurring events are prone to being missed [10]. Three associative classification systems are studied in Ref. [64]. In this paper, we test boosting algorithms on the high-order pattern and weight-of-evidence rule based classifier (HPWR). Interested readers should refer to Refs. [16,64,65] for more details on the algorithms.

Given a data set with imbalanced class distribution, classification performance regarding the small class is usually poor. In dealing with this observation, the learning objective can be either to achieve high recognition success of the small class, or to balance identify ability between two classes. For the former, the classification performance is evaluated by $F$-measure; for the latter, the classification performance is evaluated by $G$-mean. The ROC analysis method is used to compare classification models or select possibly optimal models given various learning parameters. The ROC analysis method, however, needs a classifier to yield a score representing the degree to which an example pertains to a class. For decision trees, the class distributions at each leaf is usually used as the score [34]. For HPWR, no such score is provided. For consistent results, these boosting algorithms are evaluated by $F$-measure and $G$-mean. The experimental results evaluated by these two measures expose similar features of these boosting algorithms. Due to space limitation, we only present results evaluated by $F$-measure. Interested readers can find other results evaluated by $G$-mean in Ref. [66].

In each boosting algorithm, the parameter $T$ governing the number of classifiers generated is set as 10 for these experiments. The iteration round of boosting can be terminated through one of the three conditions: (1) the iteration round reaches the prefixed number, like 10 in our experiments; (2) the condition with respect to each boosting algorithm (i.e., Eqs. (3.7), (4.11), (4.17), or (4.21), respectively) does not hold anymore; and (3) the training error of the current classifier is

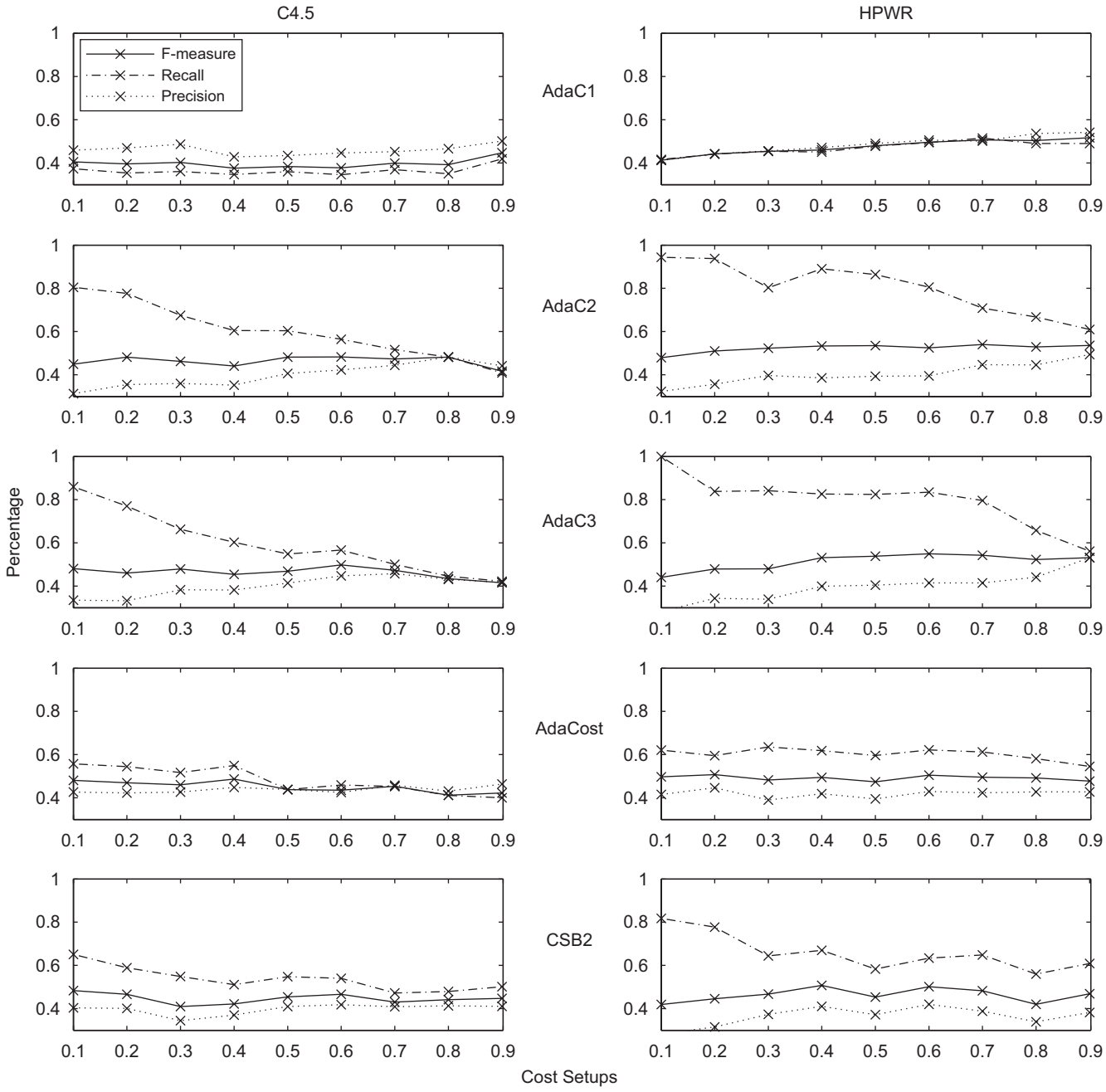


Fig. 2. *F*-measure, recall and precision values of the positive class respecting to the cost setups of the negative class by applying AdaC1, AdaC2, AdaC3, AdaCost and CSB2 to the base learners C4.5 and HPWR on the Cancer data.

0 while the sample weights do not change, and the classifiers in the succeeding iterations remain unchanged.

### 6.1. Data sets

We use four medical diagnosis data sets taken from the UCI Machine Learning Database [5] for the test. All data sets have two output labels: one denotes the disease category which is treated as the positive class, and another represents the normal category which is treated as the negative class. These data sets are selected as the class imbalance problem inherent in the data hinders the learning from building an effective classification model to distinguish diseased people from the normal population. These four data sets are: Breast cancer data (Cancer), Hepatitis data (Hepatitis), Pima Indian's diabetes database (Pima), and Sick-euthyroid data (Sick).

1. *Breast cancer data*: This breast cancer domain was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. Each instance is described by 9 attributes, 3 of which are linear and 6 are nominal. There are 286 instances in this data set, 9 instances with missing

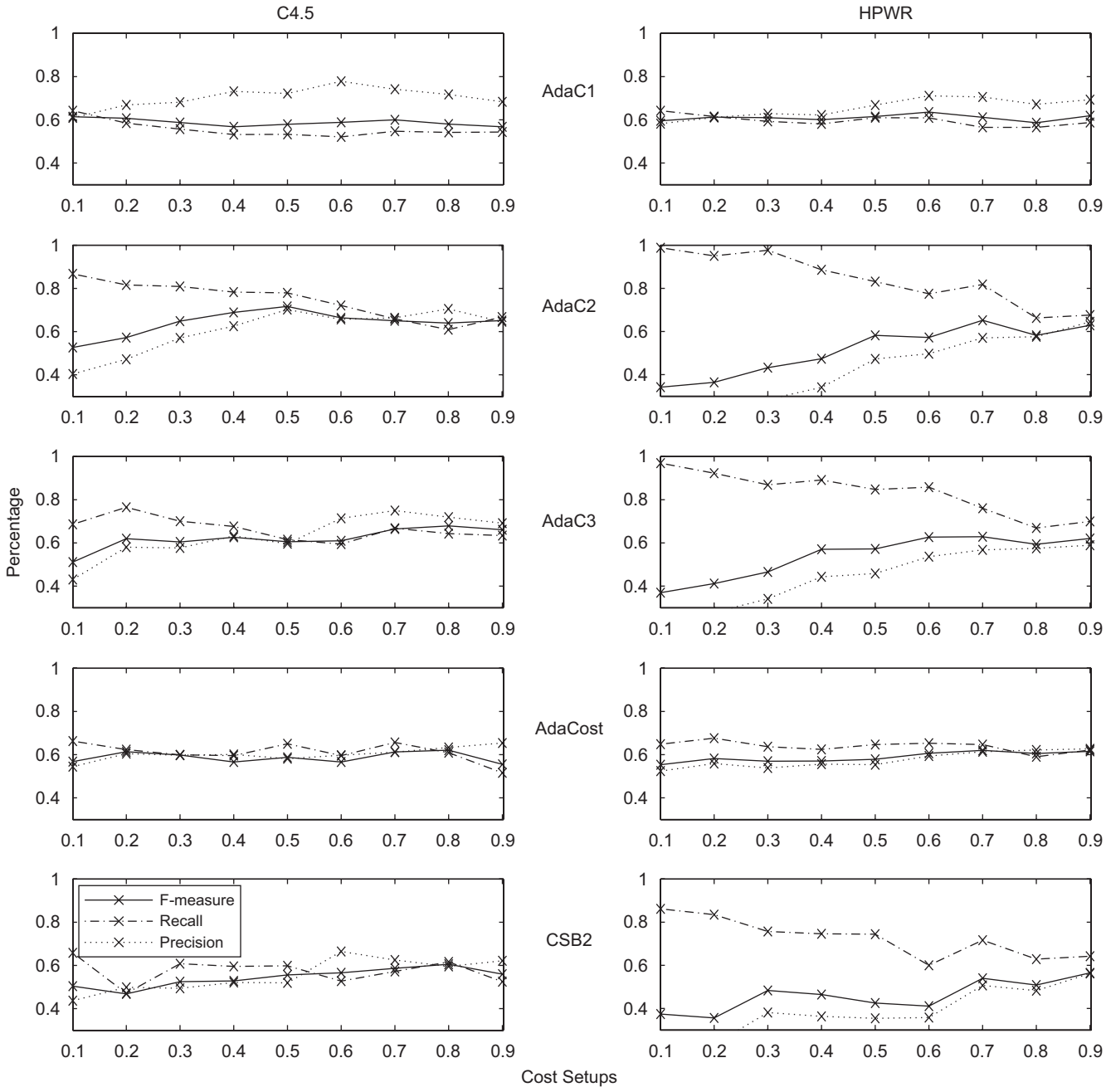


Fig. 3. *F*-measure, recall and precision values of the positive class respecting to the cost setups of the negative class by applying AdaC1, AdaC2, AdaC3, AdaCost and CSB2 to the base learners C4.5 and HPWR on the Hepatitis data.

values. Class distributions are 29.7% of recurrence-events (positive class) and 70.3% of no-recurrence-events (negative class).

2. *Hepatitis data*: This is a small collection of hepatitis domain data with only 155 instances in the whole data set. Each instance is described by 19 attributes with only one being continuously valued. The data set is composed of 32 positive instances (20.65%) in class "DIE" and 123 negative instances (79.35%) in class "LIVE".
3. *Pima Indian's diabetes database*: The diagnostic variable investigated is whether the patient shows signs of diabetes. In this data, each instance is described by 8 continuously valued attributes. There are 768 instances, 500 instances being negative and 268 being positive. Therefore, the two classes are non-evenly distributed with 34.9% of positive instances and 65.1% of negative instances, respectively.
4. *Sick-euthyroid data*: The goal of this data set is to predict the disease of thyroid domains. The data were collected with 25 attributes, 7 being continuous and 18 being Boolean values. The data set contains 3163 instances, with 9.26% of the instances being euthyroid and the remaining 90.74% being negative. There are several instances with missing attribute values.

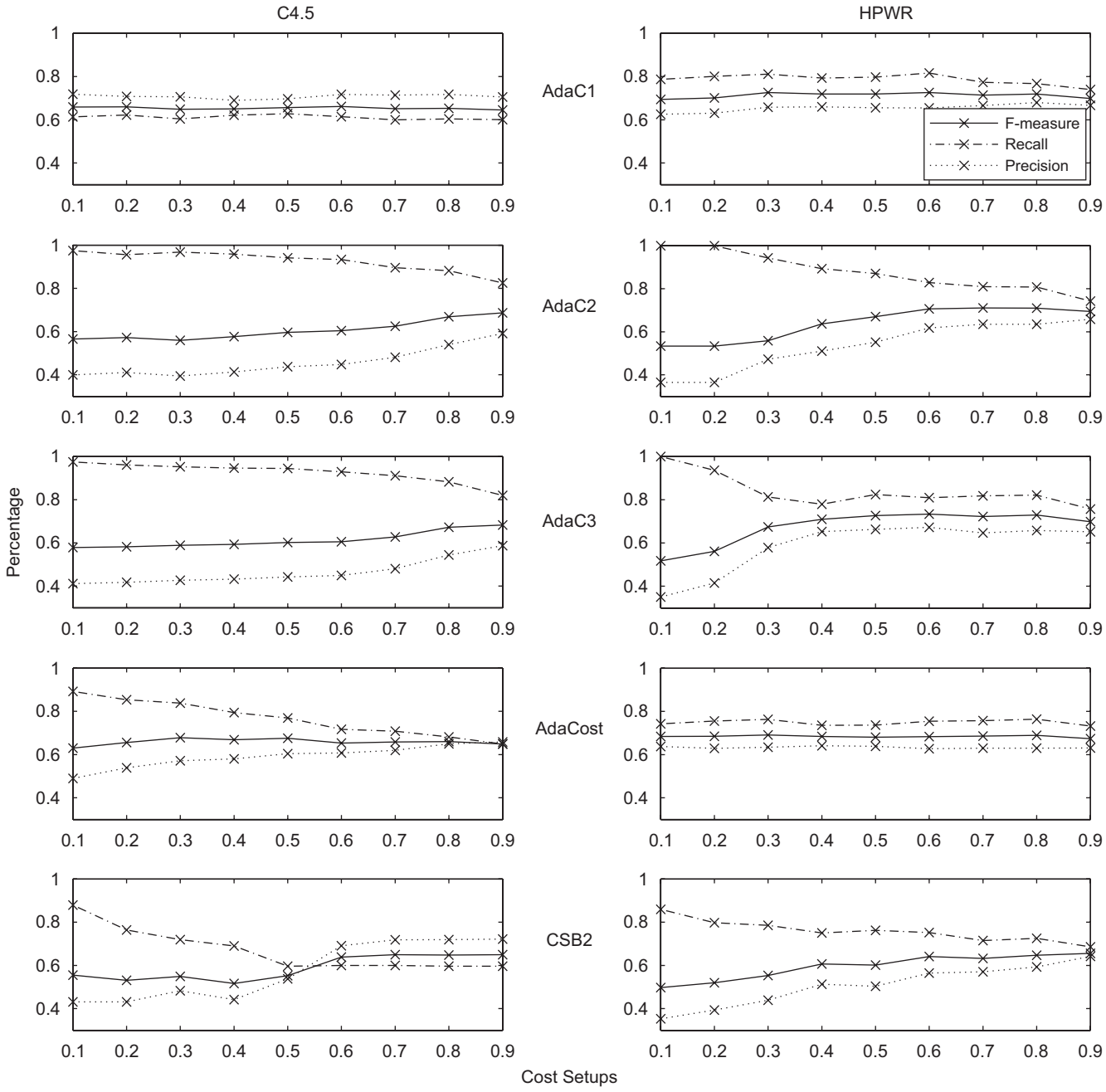


Fig. 4. $F$-measure, recall and precision values of the positive class respecting to the cost setups of the negative class by applying AdaC1, AdaC2, AdaC3, AdaCost and CSB2 to the base learners C4.5 and HPWR on the Pima data.

There are some missing attribute values in data set Cancer, Hepatitis and Sick. C4.5 handles missing attribute values and HPWR treats missing attribute values as having an unknown value "?". Each data set is randomly divided into two disjointed parts: 80% for training and the remaining 20% for testing. This process is repeated 10 times to obtain an average performance.

### 6.2. Cost setups

In our experiments, the misclassification costs for samples in the same category are set with the same value: $C_P$ denotes the misclassification cost of the positive class and $C_N$ represents that of the negative class. With these cost-sensitive boosting algorithms, cost items are used to boost more weights towards the positive (small) class. The larger the cost ratio of the positive class to the negative class, the more the weights are expected to boost on the positive class. The ratio between $C_P$ and $C_N$ denotes the deviation of the learning importance between the two classes. As values of $C_P$ and $C_N$ are not available for these data sets, a range of values is tested. Considering the constraint stated in Eq. (4.9) and (4.20) (i.e., cost values should be restricted in $[-1\ +1]$), we fix the cost item of the positive

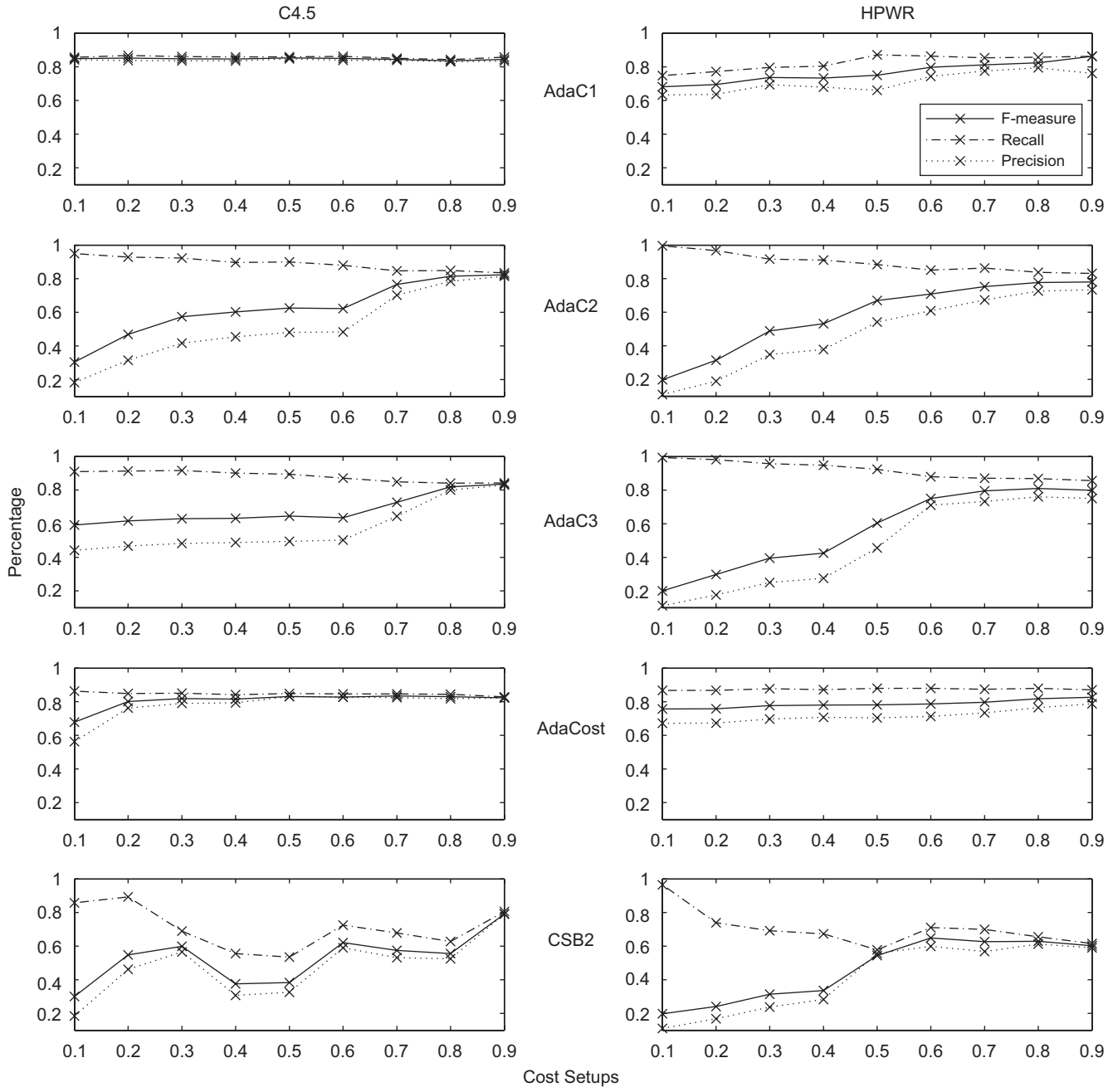


Fig. 5. $F$-measure, recall and precision values of the positive class respecting to the cost setups of the negative class by applying AdaC1, AdaC2, AdaC3, AdaCost and CSB2 to the base learners C4.5 and HPWR on the Sick data.

class to 1 and change the cost item of the negative class from 0.1 to 0.9. That is, a set of cost settings of [1.0 : 0.1, 1.0 : 0.2, 1.0 : 0.3, 1.0 : 0.4, 1.0 : 0.5, 1.0 : 0.6, 1.0 : 0.7, 1.0 : 0.8, 1.0 : 0.9] is going to be tested. The cost ratio of the positive class to the negative class is growing smaller as the cost item of the negative class changes from 0.1 to 0.9. When these two items are set equally as $C_P = C_N = 1$, the proposed three boosting algorithms AdaC1, AdaC2 and AdaC3 reduce to the original AdaBoost algorithm. For CSB2, the requirements for the cost setup are: if a sample is correctly classified, $C_P = C_N = 1$; otherwise, $C_P > C_N \geqslant 1$. Hence, we fix the cost setting for False Negatives as 1 and use the cost settings of $C_N$ for True Positives, True Negatives and False Positives. Then the weights of true predictions are updated from the $t$th iteration to the $(t+1)$th iteration by $TP_{t+1} = C_N \cdot TP_t / e^{\alpha_t}$ and $TN_{t+1} = C_N \cdot TN_t / e^{\alpha_t}$.

### 6.3. $F$-measure evaluation

The cost setup is one aspect that influences the weights boosted towards each class. Another factor that decides the weight distributions is the resampling strategy of each boost-

Table 4
$F$-measure comparisons

| | | | Base | AdaBoost | AdaC1 | AdaC2 | AdaC3 | AdaCost | CSB2 |
|---|---|---|---|---|---|---|---|---|---|
| | | Cost | | | 1:0.9 | 1:0.6 | 1:0.6 | 1:0.4 | 1:0.1 |
| | | *F* | 39.51 | 42.60 | 44.77 | *48.22* | ***49.81*** | *48.72* | *48.31* |
| | C4.5 | *R* | 34.53 | 39.25 | 41.86 | 56.45 | 56.65 | 54.92 | 65.10 |
| Cancer | | *P* | 47.92 | 48.36 | 50.18 | 42.28 | 44.76 | 44.81 | 40.37 |
| data | | Cost | | | 1:0.9 | 1:0.7 | 1:0.6 | 1:0.2 | 1:0.4 |
| | | *F* | 41.44 | 46.44 | *50.70* | *53.98* | ***54.97*** | *50.75* | *50.73* |
| | HPWR | *R* | 44.17 | 44.68 | 49.06 | 70.80 | 83.45 | 59.50 | 66.98 |
| | Od = 6 | *P* | 40.91 | 49.58 | 54.18 | 44.61 | 41.44 | 44.64 | 41.04 |
| | | Cost | | | 1:0.1 | 1:0.5 | 1:0.8 | 1:0.8 | 1:0.8 |
| | | *F* | 50.89 | 55.87 | *61.39* | ***71.63*** | *70.13* | *62.05* | *60.55* |
| | C4.5 | *R* | 53.84 | 53.98 | 64.20 | 77.89 | 66.78 | 60.85 | 61.60 |
| Hepa | | *P* | 50.08 | 69.56 | 60.86 | 70.16 | 74.94 | 63.29 | 59.53 |
| data | | Cost | | | 1:0.6 | 1:0.7 | 1:0.7 | 1:0.7 | 1:0.9 |
| | | *F* | 56.45 | 61.86 | *63.59* | ***65.19*** | *62.88* | 61.93 | 58.51 |
| | HPWR | *R* | 68.68 | 59.23 | 60.86 | 81.73 | 75.99 | 64.67 | 64.23 |
| | Od = 3 | *P* | 52.20 | 66.93 | 71.11 | 57.02 | 56.77 | 61.29 | 56.20 |
| | | Cost | | | 1:0.6 | 1:0.9 | 1:0.9 | 1:0.3 | 1:0.7 |
| | | *F* | 64.98 | 65.61 | 66.17 | **68.69** | 68.29 | 67.77 | 64.98 |
| | C4.5 | *R* | 59.57 | 60.42 | 61.40 | 82.52 | 81.99 | 83.75 | 59.96 |
| Pima | | *P* | 72.26 | 72.55 | 71.75 | 59.15 | 58.69 | 57.11 | 71.81 |
| data | | Cost | | | 1:0.3 | 1:0.7 | 1:0.6 | 1:0.3 | 1:0.9 |
| | | *F* | 67.98 | 68.17 | *72.59* | 71.03 | ***73.36*** | 69.05 | 65.58 |
| | HPWR | *R* | 70.97 | 68.85 | 81.15 | 80.94 | 80.89 | 76.33 | 68.55 |
| | Od = 4 | *P* | 65.52 | 67.77 | 65.83 | 63.49 | 67.25 | 63.40 | 64.22 |
| | | Cost | | | 1:0.5 | 1:0.9 | 1:0.9 | 1:0.7 | 1:0.9 |
| | | *F* | 84.79 | 84.04 | **85.31** | 82.38 | 83.53 | 83.46 | 79.18 |
| | C4.5 | *R* | 83.87 | 82.96 | 84.75 | 83.46 | 84.26 | 84.74 | 80.57 |
| Sick | | *P* | 85.73 | 85.19 | 86.03 | 81.46 | 82.97 | 82.45 | 79.15 |
| data | | Cost | | | 1:0.9 | 1:0.9 | 1:0.8 | 1:0.9 | 1:0.6 |
| | | *F* | 69.22 | *79.77* | ***86.36*** | *78.05* | *81.04* | *82.67* | 64.88 |
| | HPWR | *R* | 70.89 | 80.99 | 86.36 | 83.19 | 86.85 | 87.11 | 71.05 |
| | Od = 3 | *P* | 68.30 | 79.08 | 76.15 | 73.44 | 76.01 | 78.78 | 59.96 |

ing algorithm. A thorough study on the resampling effects of these boosting algorithms (Section 5) indicates their distinctive boosting emphasis with respect to the four types of examples tabulated in Table 1. In this part of the experiments, we explore: (1) how these boosting schemes affect the recall and precision values of the positive class as the cost ratio is changing; and (2) whether or not these boosting algorithms are able to improve the recognition performance on the positive class. For the first issue, we plot the $F$-measure, recall and precision values corresponding to the cost setups of the negative class to illustrate the trend. For the second issue, we tabulate the best $F$-measure values on the positive classes attainable by these boosting algorithms, within the cost setups on the experimental data sets.

Figs. 2–5 show the trade-offs between recall and precision. Each figure corresponds to one data set. In each figure, each sub-graph plots $F$-measure, recall and precision values of the positive class with respect to the cost setups when applying one boosting algorithm out of AdaC1, AdaC2, AdaC3, AdaCost and CSB2 to one base classifier, left side C4.5 and right side HPWR. From these plots, some general views we obtained are:

1. Except AdaC1, the other algorithms are able to achieve higher recall values than precision values with the recall line lying above the $F$-measure line and the precision line below the $F$-measure line in most setups. AdaC1 cannot always obtain higher recall values than precision values. In the plots of C4.5 applied to Cancer, Hepatitis and Pima data and in the plots of HPWR applied to Cancer and Hepatitis data, recall values are lower than precision values with all cost setups;
2. AdaC2 and AdaC3 are sensitive to the cost setups. When the cost item of the negative class is set with a small value denoting a large cost ratio of positive class to negative class, AdaC2 and AdaC3 can achieve very high recall values, but very low precision values as well; there is an obvious trend with plots of AdaC2 and AdaC3 in that the recall lines fall and precision lines climb when the cost setup of the negative class is changing from smaller to larger values. Comparatively, AdaC1 and AdaCost are less sensitive to the cost setups. Their recall lines and precision lines stay relatively

flat when the cost setup changes. CSB2 produces values oscillating slightly as the cost setup changes.
3. Comparing AdaCost with AdaC1, recall values of AdaCost are higher than those of AdaC1 in most cases.

These observations are consistent with the analysis of the resampling effects of these boosting algorithms. AdaC1, AdaC2 and AdaC3 all boost more weights on False Negatives than those on False Positives; on the correctly classified part, AdaC1 decreases weights of True Positives more than those of True Negatives, AdaC2 preserves more weights of True Positives than those of True Negatives. Therefore, AdaC1 conserves more weights on the negative class, AdaC2 boosts more weights towards the positive class, and AdaC3 is a combinational result of AdaC1 and AdaC2. These analyses account for the observation that AdaC2 and AdaC3 achieve higher recall values than AdaC1. AdaCost [56] is a variation of AdaC1, in that it introduces a cost adjustment function instead of a cost item inside the exponential function. The cost adjustment function increases its weight "more" if misclassified and decreases the weight "less" otherwise. AdaCost therefore boosts more weights on the positive class than AdaC1. As a result, recall values obtained by AdaCost are usually higher than those of AdaC1. However, these plots indicate that AdaCost is not sensitive to the changes of cost setups with relatively flat lines. CSB2 increases weights more on False Negatives than False Positives, but decreases weights on true predictions equally. After normalization, it is not always guaranteed that the overall boosted weights on the positive class are more than those on the negative class, as samples of the positive class are few.

Table 4 shows the best $F$-measure values achieved by each boosting algorithm and the cost settings with which these values are achieved. To indicate at what recall and precision values these $F$-measure values are achieved, we also list the corresponding recall and precision values of the positive class. In these tables, "$F$" denotes $F$-measure, "$R$" recall and "$P$" precision of the positive class. Comparing with the $F$-measure (on the positive class) values obtained by the base classifications, those significantly better $F$-measure values through $t$-test with 95% confidence interval are presented in italics and the best results with respect to each base classifier applied to a data set are denoted in bold.

Comparing with the base classifications on the Cancer data, most cost-sensitive boosting algorithms obtained significantly better $F$-measure values except AdaC1 when applied to C4.5. On the Hepatitis data, when applied to C4.5, all cost-sensitive boosting algorithms achieve significantly better $F$-measure values; when applied to HPWR, AdaC1, AdaC2 and AdaC3 obtain significantly better $F$-measure values. On the Pima data, AdaC1 and AdaC3 when applied to HPWR get significantly better results. On the Sick data, except CSB2, the other boosting algorithms including AdaBoost achieve significantly better values when applied on HPWR. Taking one base classifier associated with one data set as one entity, among these 8 entities (2 base classifier crossing with 4 data sets), AdaBoost achieved significantly better results on 1 entity, AdaC1 on 4 entities, AdaC2 on 5 entities, AdaC3 on 6 entities, AdaCost on 4 entities and CSB2 on 3 entities. For the best performance out of the 8 entities, AdaC1 win 2 times, and AdaC2 and AdaC3 both win 3 times.

## 7. Conclusions and future research

In this paper, we investigate cost-sensitive boosting algorithms for advancing the classification of imbalanced data. Three new cost-sensitive boosting algorithms are studied along with several existing related algorithms. Several good features of AdaC2 indicate that AdaC2 is superior to its rivals:

1. Weight updating strategy of AdaC2 weighs each sample by its associated cost item directly. This enables a standard classification learning algorithm that optimizes error rate to optimize cost error rate according to the translation theorem.
2. Weight updating strategy of AdaC2 increases more weights on misclassified samples form the small class and less on those from the prevalent class, decreases less weights on correctly classified samples form the small class and more on those from the prevalent class. This ensures that more weights are always accumulated on the small class to bias the learning.
3. AdaC2 tallies with the stagewise additive modelling, where a steepest descent search is carried on to minimize the overall cost loss under the exponential function.
4. Experimental results indicate AdaC2 is sensitive to the cost setups. This observation indicates the effectiveness of the cost setup employed by AdaC2 in adjusting the learning focus and influencing the identification performance on the positive class.
5. AdaC2 has furnished better results in most experiments.

Some research issues are open for future investigation:

1. Cost-sensitive learning and/or measures assume that a cost matrix is known for different types of errors or samples. Given a data set, however, the cost matrix is often unavailable. In this paper, we test a range of cost factors manually. In practice, such a method may be inadequate as it increases learning cost. A notable task for future research is to fix cost factors using some more efficient methods.
2. In our experimental study on classifying imbalanced data, most of the data sets are medical diagnosis data taken from the UCI machine learning repository. The proposed cost-sensitive boosting algorithms can also be applied to other application domains, such as fraud detection and network intrusion, to explore their effectiveness in these specific domains.
3. In this research, cost-sensitive boosting algorithms are developed by adapting the discrete AdaBoost algorithm, which assumes the outputs of a base classifier are hard class labels. As a variation of AdaBoost, RealBoost was proposed to boost base classifiers with real-valued outputs of class probability estimates instead of class labels. The proposed cost-sensitive boosting methods are applicable to the RealBoost algorithm: integrating cost values into the framework

of RealBoost and developing cost-sensitive boosting algorithms using the same inference methods as introduced in this paper. Affected by the cost setup, weights boosted towards samples will be biased and class probability estimation will be varied. These factors on the small class will be strengthened when a higher cost value is associated. This analysis indicates that the cost-sensitive boosting approach should be as effective for the real-valued classifiers in tackling the imbalanced data as for the discrete classifiers. Further investigation with real applications is required.

There are of course many other worthwhile research possibilities that are not included in the list. We believe that because of the challenging topics and tremendous potential applications, the classification of imbalanced data will continue to receive more and more attention in both the scientific and the industrial worlds.

## References


[1] N.V. Chawla, N. Japkowicz, A. Kolcz, Editorial: special issue on learning from imbalanced data sets, SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets 6 (1) (2004) 1–6.

[2] T.E. Fawcett, F. Provost, Adaptive fraud detection, Data Mining Knowl. Discovery 1 (3) (1997) 291–316.

[3] M. Kubat, R. Holte, S. Matwin, Machine learning for the detection of oil spills in satellite radar images, Mach. Learn. 30 (1998) 195–215.

[4] P. Riddle, R. Segal, O. Etzioni, Representation design and brute-force induction in a boeing manufacturing domain, Appl. Artif. Intell. 8 (1991) 125–147.

[5] P.M. Murph, D.W. Aha, UCI repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, 1991.

[6] K. Ezawa, M. Singh, S.W. Norton, Learning goal oriented Bayesian networks for telecommunications risk management, in: Proceedings of the Thirteenth International Conference on Machine Learning, Bari, Italy, 1996, pp. 139–147.

[7] C. Cardie, N. Howe, Improving minority class predication using case-specific feature weights, in: Proceedings of the fourteenth International Conference on Machine Learning, Nashville, TN, July 1997, pp. 57–65.

[8] G.E.A.P.A. Batista, R.C. Prati, M.C. Monard, A study of the behavior of several methods for balancing machine learning training data, SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets, vol. 6(1), 2004, pp. 20–29.

[9] N. Japkowicz, S. Stephen, The class imbalance problem: a systematic study, Intell. Data Anal. J. 6 (5) (2002) 429–450.

[10] G. Weiss, Mining with rarity: a unifying framework, SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets, vol. 6(1), 2004, pp. 7–19.

[11] R. Akbani, S. Kwek, N. Jakowicz, Applying support vector machines to imbalanced datasets, in: Proceedings of European Conference on Machine Learning, Pisa, Italy, September 2004, pp. 39–50.

[12] B. Raskutti, A. Kowalczyk, Extreme rebalancing for SVMs: a case study, in: Proceedings of European Conference on Machine Learning, Pisa, Italy, September 2004, pp. 60–69.

[13] G. Wu, E.Y. Chang, Class-boundary alignment for imbalanced dataset learning, in: Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets, Washington, DC, August 2003.

[14] J. Zhang, I. Mani, KNN approach to unbalanced data distributions: a case study involving information extraction, in: Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data Sets, Washington, DC, August 2003.

[15] B. Liu, W. Hsu, Y. Ma, Mining association rules with multiple minimum supports, in: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, August 1999, pp. 337–341.

[16] A.K.C. Wong, Y. Wang, High order pattern discovery from discrete-valued data, IEEE Trans. Knowl. Data Eng. 9 (6) (1997) 877–893.

[17] N. Japkowicz, in: Proceedings of the AAAI'2000 Workshop on Learning from Imbalanced Data Sets, AAAI Tech Report WS-00-05, AAAI, 2000.

[18] N.V. Chawla, N. Japkowicz, A. Kotcz, in: Proceedings of the ICML'2003 Workshop on Learning from Imbalanced Data Sets, ICML, 2003.

[19] N.V. Chawla, N. Japkowicz, A. Kotcz, SIGKDD Explorations, Special Issue on Class Imbalances, vol. 6(1), ACM, New York, June 2004.

[20] N. Chawla, K. Bowyer, L. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[21] A. Estabrooks, A combination scheme for inductive learning from imbalanced data sets, Master's Thesis, Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada, 2000.

[22] M. Kubat, S. Matwin, Addressing the curse of imbalanced training sets: one-sided selection, in: Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, Los Altos, CA, 1997, pp. 179–186.

[23] M. Pazzani, C. Merz, P. Murphy, K. Ali, T. Hume, C. Brunk, Reducing misclassification costs, in: Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, NJ, July 1994, pp. 217–225.

[24] N. Japkowicz, Supervised versus unsupervised binary-learning by feedforward neural networks, Mach. Learn. 41 (1) (2001).

[25] Y. Freund, R.E. Schapire, Experiments with a new boosting algorithm, in: Proceedings of the Thirteenth International Conference on Machine Learning, 1996, The Mit Press, Cambridge, MA, Morgan Kaufmann, Los Altos, CA, pp. 148–156.

[26] Y. Freund, R.E. Schapire, A decision-theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.

[27] R.E. Schapire, Y. Singer, Improved boosting algorithms using confidence-rated predictions, Mach. Learn. 37 (3) (1999) 297–336.

[28] R.E. Schapire, Y. Singer, Boosting the margin: a new explanation for the effectiveness of voting methods, Mach. Learn. 37 (3) (1999) 297–336.

[29] J. Friedman, T. Hastie, R. Tibshirani, Additive logistic regression: a statistical view of boosting, Ann. Statist. 28 (2) (2000) 337–374.

[30] G. Ridgeway, The state of boosting, Comput. Sci. Statist. 31 (1999) 172–181.

[31] N. Japkowicz, Concept-learning in the presence of between-class and within-class imbalances, in: Proceedings of the Fourteenth Conference of the Canadian Society for Computational Studies of Intelligence, Ottawa, Canada, June 2001, pp. 67–77.

[32] M.V. Joshi, Learning classifier models for predicting rare phenomena, Ph.D. Thesis, University of Minnesota, Twin Cites, MN, USA, 2002.

[33] G. Weiss, F. Provost, Learning when training data are costly: the effect of class distribution on tree induction, J. Artif. Intell. Res. 19 (2003) 315–354.

[34] R.C. Prati, G.E.A.P.A. Batista, Class imbalances versus class overlapping: an analysis of a learning system behavior, in: Proceedings of the Mexican International Conference on Artificial Intelligence (MICAI), Mexico City, Mexico, April 2004, pp. 312–321.

[35] Z.H. Zhou, X.Y. Liu, Training cost-sensitive neural networks with methods addressing the class imbalance problem, IEEE Trans. Knowl. Data Eng. 18 (1) (2006) 63–77.

[36] J.R. Quinlan, Improved estimates for the accuracy of small disjuncts, Mach. Learn. 6 (1991) 93–98.

[37] B. Zadrozny, C. Elkan, Learning and making decisions when costs and probabilities are both unknown, in: Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, August 2001, pp. 204–213.

[38] Y. Lin, Y. Lee, G. Wahba, Support vector machines for classification in nonstandard situations, Mach. Learn. 46 (2002) 191–202.

[39] B. Liu, Y. Ma, C.K. Wong, Improving an association rule based classifier, in: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery, Lyon, France, September 2000, pp. 504–509.

[40] L.M. Manevitz, M. Yousef, One-class svms for document classification, J. Mach. Learn. Res. 2 (2001) 139–154.
[41] B. Zadrozny, J. Langford, N. Abe, Cost-sensitive learning by cost-proportionate example weighting, in: Proceedings of the Third IEEE International Conference on Data Mining, Melbourne, FL, November 2003, pp. 435–442.
[42] C.X. Ling, C. Li, Decision trees with minimal costs, in: Proceedings of the 21st International Conference on Machine Learning, Banff, Canada, July 2004.
[43] J. Bradford, C. Kunz, R. Kohavi, C. Brunk, C.E. Brodley, Pruning decision trees with misclassification costs, in: Proceedings of the Tenth European Conference on Machine Learning (ECML-98), Chemnitz, Germany, April 1998, pp. 131–136.
[44] P. Domingos, P. Metacost, Metacost: a general method for making classifiers cost sensitive, in: Advances in Neural Networks, International Journal of Pattern Recognition and Artificial Intelligence, San Diego, CA, 1999, pp. 155–164.
[45] N. Abe, B. Zadrozny, J. Langford, An iterative method for multi-class cost-sensitive learning, in: Proceedings of the tenth ACN SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, August 2004, pp. 3–11.
[46] M.V. Joshi, V. Kumar, R.C. Agarwal, Evaluating boosting algorithms to classify rare classes: Comparison and improvements, in: Proceedings of the First IEEE International Conference on Data Mining (ICDM'01), 2001.
[47] D. Lewis, W. Gale, Training text classifiers by uncertainty sampling, in: Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information, New York, NY, August 1998, pp. 73–79.
[48] P. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, Addison-Wesley, Reading, MA, 2006.
[49] F. Provost, T. Fawcett, Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions, in: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newportbeach, CA, August 1997, p. 43–48.
[50] J.A. Hanley, B.J. McNeil, The meaning and use of the area under a receiver operating characteristic (ROC) curve, Intell. Data Anal. J. 143 (1982) 29–36.
[51] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[52] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32.
[53] R.E. Schapire, The boosting approach to machine learning—an overview, in: MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA, March 2002, pp. 149–172.
[54] J. Kittler, M. Katef, R. Duin, J. Matas, On combining classifiers, IEEE Trans. Pattern Anal. Mach. Intell. 20 (3) (1998).
[55] M.S. Kamel, N. Wanas, Data dependence in combining classifiers, in: Proceedings of the Fourth International Workshop on Multiple Classifiers Systems, Surrey, UK, June 2003.
[56] W. Fan, S.J. Stolfo, J. Zhang, P.K. Chan, Adacost: misclassification cost-sensitive boosting, in: Proceedings of Sixth International Conference on Machine Learning (ICML-99), Bled, Slovenia, 1999, pp. 97–105.
[57] K.M. Ting, A comparative study of cost-sensitive boosting algorithms, in: Proceedings of the 17th International Conference on Machine Learning, Stanford University, CA, 2000, pp. 983–990.
[58] C. Chen, A. Liaw, L. Breiman, Using random forests to learn unbalanced data, ⟨http://stat-www.berkeley.edu/users/chenchao/666.pdf⟩, 2004, Technical Report 666, Statistics Department, University of California at Berkeley.
[59] P. Turney, Types of cost in inductive concept learning, in: Proceedings of Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning, Stanford University, CA, 2000, pp. 15–21.
[60] P. Viola, M. Jones, Fast and robust classification using asymmetric adaboost and a detector cascade, in: Proceedings of the Neural Information Processing Systems Conference, Vancouver, BC, Canada, December 2001, pp. 1311–1318.
[61] C. Elkan, The foundations of cost-sensitive learning, in: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, 2001, pp. 973–978.
[62] N.V. Chawla, A. Lazarevic, L.O. Hall, K.W. Bowyer, SMOTEBoost: improving prediction of the minority class in boosting, in: Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases, Dubrovnik, Croatia, 2003, pp. 107–119.
[63] H. Guo, H.L. Viktor, Learning from imbalanced data sets with boosting and data generation: the databoost-IM approach, SIGKDD Explorations Special Issue on Learning from Imbalanced Datasets 6 (1) (2004) 30–39.
[64] Y. Sun, A.K.C. Wong, Y. Wang, An overview of associative classifiers, in: The 2006 International Conference on Data Mining (DMIN'06), Las Vegas, Nevada, June 2006, pp. 138–143.
[65] Y. Wang, A.K.C. Wong, From association to classification: inference using weight of evidence, IEEE Trans. Knowl. Data Eng. 15 (3) (2003) 764–767.
[66] Y. Sun, Cost-sensitive boosting for classification of imbalanced data, Ph.D. Thesis, University of Waterloo, Waterloo, Ont., Canada, 2007.

**About the Author**—YANMIN SUN received the BS and MS degrees in Electrical and Computer Engineering Department from Wuhan Transportation University, China, in 1988 and 1999, respectively. She received his Ph.D. degree in the Electrical and Computer Engineering Department at the University of Waterloo, Canada, in April 2007. Her research interests include machine learning algorithms, data mining, and classification with imbalanced data.

**About the Author**—MOHAMED S. KAMEL received the B.Sc. (Hons) degree in Electrical Engineering from the University of Alexandria, Egypt, his M.Sc. degree in Computation from McMaster University, Hamilton, Canada, and his Ph.D. degree in Computer Science from the University of Toronto, Canada.

He joined the University of Waterloo, Canada in 1985 where he is at present Professor and Director of the Pattern Analysis and Machine Intelligence Laboratory at the Department of Electrical and Computer Engineering. Professor Kamel holds Canada Research Chair in Cooperative Intelligent Systems.

Dr. Kamel's research interests are in Computational Intelligence, Pattern Recognition, Machine Learning, and Cooperative Intelligent Systems. He has authored and co-authored over 350 papers in journals and conference proceedings, 7 edited volumes, 2 patents, and numerous technical and industrial project reports. Under his supervision, 67 Ph.D and M.A.SC. students have completed their degrees.

He is the Editor-in-Chief of the International Journal of Robotics and Automation, Associate Editor of the IEEE SMC, Part A, Pattern Recognition Letters, Cognitive Neurodynamics journal. He is also a member of the editorial advisory board of the International Journal of Image and Graphics and the Intelligent Automation and Soft Computing journal. He also served as Associate Editor of Simulation, the Journal of The Society for Computer Simulation.

Based on his work at the NCR, he received the NCR Inventor Award. He is also a recipient of the Systems Research Foundation Award for outstanding presentation in 1985 and the ISRAM best paper award in 1992. In 1994, he has been awarded the IEEE Computer Society Press outstanding referee award. He was also a co-author of the best paper in the 2000 IEEE Canadian Conference on electrical and Computer Engineering. Dr. Kamel is recipient of the University of Waterloo outstanding performance award, the faculty of engineering distinguished performance award.

Dr. Kamel is a member of ACM, PEO, Fellow of IEEE, and Fellow of EIC. He served as consultant for General Motors, NCR, IBM, Northern Telecom and Spar Aerospace. He is also a member of the board of directors and co-founder of Virtek Vision Inc. of Waterloo (www.virtek.ca).

**About the Author**—ANDREW K. C. WONG holds a Ph.D. from Carnegie Mellon University; B.Sc. (Hons) and M.Sc. from the Hong Kong University. He is an IEEE Fellow (for his contribution in machine intelligence, computer vision, and intelligent robotics).

Currently, Dr. Wong is a Distinguished Professor Emeritus (Systems Design Engineering) where he is also Adjunct to the School of Computer Sciences and the Electrical and Computer Engineering at the University of Waterloo (UW). He was the Founding Director of Pattern Analysis and Machine Intelligence Laboratory (PAMI Lab) at UW and a Distinguished Chair Professor at the Hong Kong Polytechnic University (00–03). His research areas cover machine intelligence, computer vision, intelligence robotics, pattern recognition, pattern mining, and bioinformatics. Dr. Wong has published close to 300 journal and conference papers and book chapters; and is holding five US Patents. He has served as consultant to many high-tech companies in USA and Canada.

Based on the core technologies he and his team have developed, three high tech companies were founded. Dr. Wong is a founder, and retired director (93–03) of Virtek Vision International Corporation, a publicly traded company and a leader in laser and vision technology. He was its president (86–93) and Chairman (93–97). In 1997, he co-founded Pattern Discovery Software Systems Ltd. and has served as Chairman ever since. In 2004, he helped founding Envisage Healthcare Solutions, a software company dedicated to research, development, and deployment of health information and bioinformatics for decision support in patients' diagnosis, therapeutic management, and intervention.

**About the Author**—YANG WANG received his B. Eng. degree in electronic engineering and his M. Eng. degree in systems engineering from Tsinghua University, Beijing, China, in 1989 and 1991, respectively. He received his Ph.D. degree in systems design engineering from the University of Waterloo, Waterloo, Ont., Canada, in 1997.

Dr. Wang is a co-founder of Pattern Discovery Software Systems Ltd., a software firm specialized in data mining solutions, and has been the CTO of the company since 1997. His current research interests include exploratory data mining, knowledge discovery, intelligent data analysis, and their applications. He is an adjunct professor of Systems Design Engineering at the University of Waterloo, where he supervises graduate students and conducts academic research.

Dr. Wang is a member of the IEEE Computer Society and ACM.