Cost-Sensitive Boosting Algorithms for Imbalanced Multi-instance Datasets

Xiaoguang Wang[1], Stan Matwin[2], Nathalie Japkowicz[1,3], and Xuan Liu[1]

[1] School of Electrical Engineering and Computer Science, University of Ottawa, Canada
Bwang009@eecs.uottawa.ca, Xliu107@uottawa.ca
[2] Faculty of Computer Science, Dalhousie University, Canada
Stan@cs.dal.ca
[3] Department of Computer Science, Northern Illinois University, USA
Nat@eecs.uottawa.ca

Abstract. Multi-instance learning is different than standard propositional classification, because it uses a set of bags containing many instances as input. The instances in each bag are not labeled, but the bags themselves are labeled positive or negative. Our research shows that classification of multi-instance data with imbalanced class distributions significantly decreases the performance normally achievable by most multi-instance algorithms, which is the same as the performance of most standard, single-instance classifier learning algorithms. In this paper, we present and analyze this multi-instance class imbalance problem, and propose a novel solution framework. We focus on how to utilize the extended AdaBoost techniques applicable to most multi-instance classifier learning algorithms. Cost-sensitive boosting algorithms are developed by introducing cost items into the learning framework of AdaBoost, to enable classification of imbalanced multi-instance datasets.

Keywords: multi-instance classification, class imbalance problem, AdaBoost, cost-sensitive learning.

1 Introduction

Multi-instance learning (MIL) differs from traditional supervised learning algorithms, in that a multi-instance dataset consists of bags of individual instances with unknown classifications, and only the bags are labeled. Each bag can contain several instances, but the number of instances in each is different, and the same instance can belong to several bags.

While MIL has been used in many applications, including drug activity recognition [3], text-categorization [15] and computer vision recognition [14], [20], there is a vast amount of research about, and many different approaches to, solving the MIL problem. For example, Diverse Density (DD) [4] and the Expectation-Maximization version [10] were proposed as general frameworks for solving multi-instance learning problems. The k Nearest Neighbour approach, known as Citation kNN, was adapted for MIL problems in [8]. Andrews et al. [15] proposed two approaches to modify Support Vector Machines: mi-SVM for instance-level classification, and MI-SVM for bag-level classification. For tree methods, Blockeel et al. [17] proposed a multi-instance tree method (MITI), and Bjerring et al. [21] extended this in their work by adopting MITI to learn rules (MIRI).

A dataset is imbalanced if the classes are not represented approximately equally. In a two-class imbalanced dataset, there are often far more negative examples than positive examples, and in this situation a default classifier will always predict 'negative'. In practice, one would want to penalize errors on positive examples more strongly than errors on negative examples. There have been attempts to deal with imbalanced datasets in real life domains, such as text classification [16], image classification [7], disease detection [18] and others [9], [13]. However, the data imbalance problem still exists for multi-instance classification without specialized solution provided. Although there are many published works about multi-instance classification, there is very little related discussion about imbalanced multi-instance classification problems. The multi-instance data imbalanced problem is presented in this paper, and cost-sensitive boosting algorithms are developed by introducing cost items into the learning framework of AdaBoost, for classification of imbalanced multi-instance datasets.

The paper examines two class classification problems, and the algorithms discussed can be extended to multi-class classification. The rest of the paper is organized as follows: Section 2 presents the class imbalance problem for multi-instance datasets and related concepts. In Section 3, the AdaBoost algorithm and its cost-sensitive adaptations for the single-instance class imbalance problem are discussed. Cost-sensitive boosting algorithms for the multi-instance class imbalance problem are presented in Section 4. Section 5 illustrates the efficiency of our algorithm as determined by experimentation, and offers some final remarks. Finally, Section 6 presents the conclusion and future work.

2 The Class Imbalance Problem of Multi-instance Datasets

The multi-instance learning problem can be defined as:
Given:

- A set of bags $\chi_i, i = 1, \ldots, N$, where each bag can consist of an arbitrary number of instances and a given label: $\chi_i = \{x_i^1, x_i^2, \ldots, x_i^{n_i}; y_i\}, i = 1, \ldots, N, y_i \in \{-1, +1\}$, where each instance $x_i^{n_i}$ is an M-tuple of attribute values belonging to a certain domain or instance space $\mathbb{R}$.
- The existence of an unknown function f that classifies individual instances as $+1$ or -1, and for which it holds that $c(\chi_i) = +1$ if and only if $\exists x_i^{n_i} \in \chi_i : f(x_i^{n_i}) = +1$. (multi-instance constraint, MIC)

From this definition we derive the standard assumption of MI learning, which is that a bag is negative if and only if all instances in the bag are negative; if the bag contains one or more positive instances, the bag is positive.

Although specific discussions about the multi-instance class imbalance problem are rarely found in previous work, the issue occurs frequently in real life application areas such as computer vision recognition and text mining. In our related research [23], multi-instance classification algorithms are used in underwater mine like object sonar image processing. Each target to be classified has many images from different angles and distances and these images build a multi-instance dataset. In real world environment the number of non-mine like objects is much greater than the number of mine like objects, so the class imbalance problem is an important factor affecting the performance of the classifiers. Our research shows that for most multi-instance dataset with imbalanced class distributions, classification of these datasets significantly decrease the performance normally achievable by most multi-instance algorithms but we can hardly find solutions to deal with this problem from these algorithms. This motivated us to investigate the problem of multi-instance class imbalance more thoroughly.

As we already know, a single-instance dataset is defined as imbalanced if at least one class is under-represented relative to others. For multi-instance datasets, the problem is similar but the circumstances are more complex. The class imbalance situation occurs not only at the instance-level, but also at the bag-level. Figure 1 shows the imbalanced multi-instance classification problem with the separating plane and the margin. Since the final margin of multi-instance classification is at bag-level, the default classifier would tend to penalize errors on positive bags more strongly than errors on negative bags. In Figure 1, there are far more majority bags than minority bags, and the margin learned by the default classifier is 'pushed' closer to the minority bags from the ideal margin.

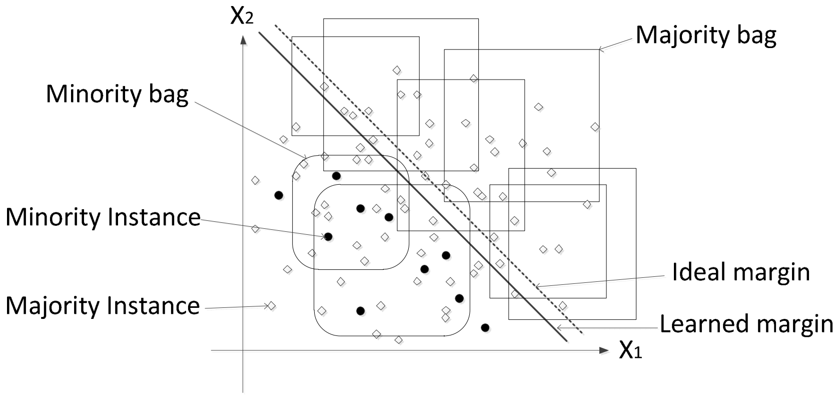

Fig. 1. The imbalanced multi-instance classification problem with the separating plane and the margin. Black dots denote instances of minority class while diamond dots denote instances of majority class. Rectangle frames with round corner denote minority bags and rectangle frames denote majority bags. The solid line denotes the learned margin by classifier and the dotted line denotes the ideal margin of two classes.

For the single-instance data imbalance problem, the machine learning community has addressed the issue of class imbalances in two different ways to solve the skewed vector space problem. The first method, which is classifier-independent, is to balance the distributions by considering the representative proportions of class examples in the distribution of the original data. The simplest way to balance a dataset is to under-sample or over-sample (randomly or selectively) the majority class, while maintaining the original minority class population [16]. One of the most common pre-processing methods to balance a dataset, Synthetic Minority Over-sampling Technique (SMOTE) [13], over-samples the minority class by taking each minority class sample and introducing synthetic examples along the line segments joining any or all of the k minority class nearest neighbors. Evidence shows that synthetic sampling methods are effective when dealing with learning from imbalanced data [9], [13], [16].

Working with classifiers to adapt datasets is another way to deal with the single-instance imbalanced data problem. The theoretical foundations and algorithms of cost-sensitive methods naturally apply to imbalanced learning problems [7], [11]. Thus, for imbalanced learning domains, cost-sensitive techniques provide a viable alternative to sampling methods. Recent research ([9], [11], [16]) suggests that assigning distinct costs to the training examples is a fundamental approach of this type, and various experimental studies of this ([5], [7], [18]) have been performed using different kinds of classifiers.

3 AdaBoost and Cost-Sensitive Adaptations

Boosting has been proven to be an effective method of combining multiple models in order to enhance the predictive accuracy of a single model [1], [6]. AdaBoost is a version of boosting that uses the confidence-rated predictions described in [1], [6]. It applies a base learner to induce multiple individual classifiers in sequential trials, and a weight is assigned to each example. After each trial, the vector of weights is adjusted to reflect the importance of each training example in the next induction trial. This adjustment effectively increases the weights of misclassified examples, and decreases the weights of correctly classified examples. Finally, the individual classifiers are combined to form a composite classifier.

Take as input the training set $(x_1, y_1), \dots, (x_m, y_m)$; $x_i \in \chi, y_i \in \{-1, +1\}$, 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. The key process of the AdaBoost.M1 method [1] is to iteratively update the distribution function over the training data. This means that for every iteration $t = 1, \dots, T$, where T is a given number of the total number of iterations, the distribution function D_t is updated sequentially, and used to train a new hypothesis:

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_t y_i h_t(x_i)\right)}{Z_t} \tag{1}$$

where $\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$ is the weight updating parameter, $h_t(x_i)$ is the prediction output of hypothesis h_t on the instance x_i, ε_t is the error of hypothesis h_t over the training data, and Z_t is a normalization factor.

Schapire and Singer [6] used a generalized version of Adaboost. As shown in [6], the training error of the final classifier is bounded as:

$$\frac{1}{m}|\{i: H(x_i) \neq y_i\}| \leq \prod_t Z_t \tag{2}$$

where

$$Z_t = \sum_i D_t(i) \exp(-\alpha_t y_i h_t(x_i)) \leq \sum_i D_t(i) \left(\frac{1 + y_i h_t(x_i)}{2} e^{-\alpha} + \frac{1 + y_i h_t(x_i)}{2} e^{\alpha} \right) \tag{3}$$

Minimizing Z_t on each round, α_t is induced as:

$$\alpha_t = \frac{1}{2} ln \left(\frac{\sum_{i, y_i = h_t(x_i)} D_t(i)}{\sum_{i, y_i \neq h_t(x_i)} D_t(i)} \right) \tag{4}$$

The weighting strategy of AdaBoost identifies samples on their classification outputs as correctly classified or misclassified. However, it treats samples of different classes equally. The weights of misclassified samples from different classes are increased by an identical ratio, and the weights of correctly classified samples from different classes are decreased by an identical ratio.

Since boosting is suitable for cost-sensitive adaption, motivated by [6]'s analysis and methods for choosing α_t, several cost-sensitive boosting methods for imbalanced learning have been proposed in recent years. Three cost-sensitive boosting methods, AdaC1, AdaC2 and AdaC3, were proposed in [18], which introduced cost items into the weight updating strategy of AdaBoost. AdaCost [5] is another cost-sensitive boosting algorithm that follows a similar methodology.

4 Proposed Methods

Similar to the methods for managing the single-instance class imbalance problem in Refs. [5], [7], [11], [18], the learning objective in dealing with the multi-instance class imbalance problem is to improve the identification performance on the minority class. In our research, the first strategy is to target the multi-instance imbalanced learning problem by using different cost matrices that describe the costs for misclassifying any particular data example. When used for single-instance learning, this method reportedly improved classification performance on class imbalanced datasets significantly [11], [12].

For multi-instance class imbalance datasets, the optimal prediction for a bag χ is the class i that minimizes

$$L(\chi, i) = \sum_j P(j|\chi) C(i, j) \tag{5}$$

where C denotes the cost matrix [11], and (i, j) is the cost of predicting class i when the true class is j. $P(j|\chi)$ denotes the probability of each class j being the true class of bag χ.

Our second strategy is to apply cost-minimizing techniques to the combination schemes of ensemble methods. This learning objective expects that the weighting strategy of a boosting algorithm will preserve a considerable weighted sample size of the minority class. A preferred boosting strategy is one that can distinguish different types of samples, and boost more weights on those samples associated with higher identification importance.

To denote the different identification importance among bags, each bag is associated with a cost item; the higher the value, the higher the importance of correctly identifying the sample. For an imbalanced multi-instance dataset, there are many more bags with class label $\mathrm{y} = -1$ than bags with class label $\mathrm{y} = +1$. Using the same learning framework as AdaBoost, the cost items can be fed into the weight update formula of AdaBoost (Eq. (1)) to bias the weighting strategy. The proposed methods are similar to those proposed in Ref. [18]. Figure 2 shows the proposed algorithms.

Given: A multi-instance training dataset with a set of bags $\chi_i, i = 1, \ldots, N$, where each bag can consist of an arbitrary number of instances and a given label: $\chi_i = \{x_i^1, x_i^2, \ldots, x_i^{n_i}; y_i\}, i = 1, \ldots, N, y_i \in \{-1, +1\}$, and each instance $x_i^{n_i}$ is an M-tuple of attribute values belonging to a certain domain or instance space $\mathbb{R}$.

Initialize $D_1(i) = 1/m$.

For $t = 1, \ldots, T$ && the constraint condition η is satisfied:

- Train a weak learner using distribution D_t.
- Get a weak hypothesis $h_t: \chi \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

$$D_{t+1}(i) = \frac{D_t(i) K_t(\chi_i, y_i)}{Z_t} \tag{6}$$

where $\mathrm{Z_t}$ is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(\chi) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\chi)\right) \tag{7}$$

Fig. 2. Cost-sensitive Adaboost for Multi-Instance Learning Algorithm

For the original adaboost, $K_t(\chi_i, y_i) = \exp(-\alpha_t y_i h_t(\chi_i))$, our proposed algorithms introduced four cost items into the weight update formula of AdaBoost: inside the exponent, outside the exponent, or in two ways both inside and outside the exponent. Each modification can be a new boosting algorithm, denoted as Ab1, Ab2, Ab3 and Ab4 respectively. Ab1, Ab2 and Ab3 are similar to AdaC1, AdaC2 and AdaC3 respectively for single-instance learning in Ref. [18]. The difference is, in our algorithms the training samples are bags of instances, not instances.

The modifications of $K_t(\chi_i, y_i)$ are then given by:

- Ab1:

$$K_t(\chi_i, y_i) = \exp(-C_i \alpha_t y_i h_t(\chi_i)) \tag{8}$$

- Ab2:

$$K_t(\chi_i, y_i) = C_i exp(-\alpha_t y_i h_t(\chi_i)) \tag{9}$$

- Ab3:

$$K_t(\chi_i, y_i) = C_i exp\,(-C_i\alpha_t y_i h_t(\chi_i)) \tag{10}$$

- Ab4:

$$K_t(\chi_i, y_i) = C_i^2 exp\,(-C_i^2\alpha_t y_i h_t(\chi_i)) \tag{11}$$

Now we induce the weight update parameter α_t and constraint condition η in Figure 2 for Ab4. From Eq. (11) we get:

$$D_{t+1}(i) = \frac{C_i^2 D_t(i)\exp\big(-C_i^2\alpha_t y_i h_t(\chi_i)\big)}{Z_t} = \frac{C_i^{2t}\exp\,(-C_i^2 y_i f(\chi_i))}{m\prod_t Z_t} \tag{12}$$

where

$$f(\chi) = \sum_t \alpha_t h_t(\chi) \tag{13}$$

and

$$Z_t = \sum_i C_i^2 D_t(i)\exp\big(-C_i^2\alpha_t y_i h_t(\chi_i)\big) \tag{14}$$

The overall training error is bounded as:

$$\frac{1}{m}|\{i: H(\chi_i) \neq y_i\}| \leq \frac{1}{m}\sum_i C_i^2\exp(-C_i^2 y_i f(\chi_i)) = \prod_t Z_t \sum_i \frac{C_i^2}{C_i^{2t}} D_{t+1}(i) \tag{15}$$

According to Ref. [6], for weak hypotheses $C_i^2\alpha_t y_i h_t(\chi_i) \in [-1, +1]$ with range $[-1, +1]$, α can be obtained by approximating Z as follows:

$$Z_t = \sum_i C_i^2 D_t(i)\exp\left(-C_i^2\alpha_t y_i h_t(\chi_i)\right) \leq \sum_i C_i^2 D_t(i)\left(\frac{1 + C_i^2 y_i h_t(\chi_i)}{2}e^{-\alpha_t} + \frac{1 - C_i^2 y_i h_t(\chi_i)}{2}e^{\alpha_t}\right) \tag{16}$$

Let

$$G(\alpha_t) = \sum_i C_i^2 D_t(i)\left(\frac{1 + C_i^2 y_i h_t(\chi_i)}{2}e^{-\alpha_t} + \frac{1 - C_i^2 y_i h_t(\chi_i)}{2}e^{\alpha_t}\right) \tag{17}$$

Our purpose is for α_t to minimize $G(\alpha_t)$, so we can obtain:

$$G'(\alpha_t) = \frac{dG}{d\alpha_t} = 0 \tag{18}$$

Next, we can analytically obtain α_t from Eq. (18), giving:

$$\alpha_t = \frac{1}{2}ln\left(\frac{\sum_i C_i^2 D_t(i) + \sum_{i,y_i=h_t(\chi_i)} C_i^4 D_t(i) - \sum_{i,y_i\neq h_t(\chi_i)} C_i^4 D_t(i)}{\sum_i C_i^2 D_t(i) - \sum_{i,y_i=h_t(\chi_i)} C_i^4 D_t(i) + \sum_{i,y_i\neq h_t(\chi_i)} C_i^4 D_t(i)}\right) \tag{19}$$

The sample weight updating goal of AdaBoost is to decrease the weight of the training samples that are correctly classified, and increase the weights of the opposite samples [1], [6]. Therefore, α_t should be a positive value, and the training error should be less than random guessing, based on the current data distribution.

To ensure that α_t is positive, we get

$$\sum_{i,y_i=h_t(\chi_i)} C_i^4 D_t(i) > \sum_{i,y_i\neq h_t(\chi_i)} C_i^4 D_t(i) \tag{20}$$

This is the constraint condition η in Figure 2.

Similarly, we can analytically choose α_t and constraint condition η for the other three modifications of Eq. (1) ([6], [18]). Table 1 lists all α_t and η of Ab1 to Ab4.

Table 1. Parameter α_t and η chosen for Figure 2

	α_t	η
Ab1	$\frac{1}{2} ln \left(\frac{1+\sum_{i,y_i=h_t(\chi_i)} C_i D_t(i) - \sum_{i,y_i\neq h_t(\chi_i)} C_i D_t(i)}{1-\sum_{i,y_i=h_t(\chi_i)} C_i D_t(i) + \sum_{i,y_i\neq h_t(\chi_i)} C_i D_t(i)}\right)$	$\sum_{i,y_i=h_t(\chi_i)} C_i D_t(i) > \sum_{i,y_i\neq h_t(\chi_i)} C_i D_t(i)$
Ab2	$\frac{1}{2} ln \left(\frac{\sum_{i,y_i=h_t(\chi_i)} C_i D_t(i)}{\sum_{i,y_i\neq h_t(\chi_i)} C_i D_t(i)}\right)$	$\sum_{i,y_i=h_t(\chi_i)} C_i D_t(i) > \sum_{i,y_i\neq h_t(\chi_i)} C_i D_t(i)$
Ab3	$\frac{1}{2} ln \left(\frac{\sum_i C_i D_t(i) + \sum_{i,y_i=h_t(\chi_i)} C_i^2 D_t(i) - \sum_{i,y_i\neq h_t(\chi_i)} C_i^2 D_t(i}{\sum_i C_i D_t(i) - \sum_{i,y_i=h_t(\chi_i)} C_i^2 D_t(i) + \sum_{i,y_i\neq h_t(\chi_i)} C_i^2 D_t(i}\right.$	$\sum_{i,y_i=h_t(\chi_i)} C_i^2 D_t(i) > \sum_{i,y_i\neq h_t(\chi_i)} C_i^2 D_t(i)$
Ab4	$\frac{1}{2} ln \left(\frac{\sum_i C_i^2 D_t(i) + \sum_{i,y_i=h_t(\chi_i)} C_i^4 D_t(i) - \sum_{i,y_i\neq h_t(\chi_i)} C_i^4 D_t(}{\sum_i C_i^2 D_t(i) - \sum_{i,y_i=h_t(\chi_i)} C_i^4 D_t(i) + \sum_{i,y_i\neq h_t(\chi_i)} C_i^4 D_t(}\right.$	$\sum_{i,y_i=h_t(\chi_i)} C_i^4 D_t(i) > \sum_{i,y_i\neq h_t(\chi_i)} C_i^4 D_t(i)$

We also used AdaCost [5] as a cost-sensitive boosting algorithm to deal with the multi-instance class imbalance problem. In Figure 2, AdaCost is developed when dealing with multi-instance classification by introducing:

$$K_t(\chi_i, y_i) = \exp\left(-\alpha_t y_i h_t(\chi_i) \beta(i)\right) \tag{21}$$

where

$$\beta(i) = \beta(\text{sign}(y_i h_t(\chi_i)), C_i) \tag{22}$$

Like Ab1, AdaCost introduces cost sensitivity inside the exponent of the weight updating formula of Adaboost. However, instead of applying the cost items directly, AdaCost employs a cost-adjustment function that aggressively increases the weights of costly misclassifications, while conservatively decreasing the weights of high-cost examples that are correctly classified.

5 Experiments

In this section, we explain our experiments to investigate and compare the following boosting and non-boosting algorithms: Base learner, Cost Sensitive, Adaboost, AdaCost, Ab1, Ab2, Ab3 and Ad4 with two weak learners using tree methods (MITI [17] and MIRI [21] respectively). Tree methods were chosen as the weak learners because they are 1) stable in multi-instance learning [17], [21], and 2) suitable to be weak learners in many related works [12], [18].

For these experiments parameter T, which governs the number of classifiers generated, was set to ten in each boosting algorithm. For the cost sensitive methods, the original costs were chosen according to the bag number of each class. For the cost sensitive boosting methods, the iteration rounds of boosting could be terminated through one of two conditions: a) the prefixed number T, or b) the constraint condition η in Figure 2. The Ten-fold Cross-Validation method was used in all experiments.

5.1 Details of Datasets

The first seven datasets used in our experiments are those employed in [19] and [21]. The original datasets are not imbalanced, so to make them all imbalanced we chose only a portion of the bags in one class.

Table 2 shows the details of the datasets used in our experiment. These datasets can be retrieved from http://www.eecs.uottawa.ca/~bwang009/.

Table 2. Details of Datasets ('#' denotes 'number of' and '%' denotes of 'percentage of').

Dataset	Size	#attribute	#minority bags	%minority bags	#minority instances	%minority instances
Elephant	125	230	25	20	150	19.69
Fox	121	230	21	17.36	134	20.71
Tiger	126	230	26	20.63	164	30.15
Mutagenesis_atom	167	10	42	25.15	365	34.02
Mutagenesis_bond	160	16	35	21.88	603	20.41
Mutagenesis_chain	152	24	27	17.76	514	12.49
Process	142	200	29	20.42	281	9.78

5.2 Experimental Results

When learning extremely imbalanced data, a trivial classifier that predicts every case as the majority class can still achieve very high accuracy. Thus, the overall classification accuracy is often not an effective measure of performance. We chose Gmean [2] as the measure for our algorithms and experiments. The definition of Gmean is found in Eq. (23) and the confusion matrix is defined in Table 3.

Table 3. Confusion Matrix

	Predicted Positive Class	Predicted Negative Class
Actual Positive class	TP (True Positive)	FN (False Negative)
Actual Negative class	FP (False Positive)	TN (True Negative)

$$\text{Gmean} = \left(\frac{TN}{TN+FP} \times \frac{TP}{TP+FN}\right)^{1/2} \quad (23)$$

Table 4. Experiment results on MITI (Gmean)

Dataset	Base	CS	AdaCost	Adaboost	Ab1	Ab2	Ab3	Ab4
Elephant	0.4313	0.6	0.5485	0.5238	0.5426	0.5514	0.5514	0.6099
Fox	0.2149	0.4337	0.4445	0	0.296	0.4655	0.5555	0.4928
Tiger	0.5463	0.7317	0.7038	0.6928	0.7805	0.7114	0.7671	0.6923
Mutagenesis_atom	0.5831	0.6309	0.6973	0.6234	0.5995	0.7059	0.657	0.6928
Mutagenesis_bond	0.5493	0.7538	0.7928	0.7085	0.6943	0.7335	0.7899	0.7783
Mutagenesis_chain	0.4251	0.6742	0.7272	0.503	0.7542	0.7242	0.7933	0.8327
Process	0.7096	0.7913	0.8156	0.8194	0.8593	0.8194	0.8043	0.842

Table 5. Experiment results on MIRI (Gmean)

Dataset	Base	CS	AdaCost	Adaboost	Ab1	Ab2	Ab3	Ab4
Elephant	0.5158	0.6536	0.5786	0.5571	0.5514	0.56	0.5817	0.6033
Fox	0.2116	0.4337	0.5014	0.4276	0.4756	0.3546	0.4499	0.4024
Tiger	0.5004	0.6844	0.6785	0.6725	0.7114	0.7981	0.7894	0.7442
Mutagenesis_atom	0.7005	0.7709	0.7358	0.7579	0.7554	0.7926	0.7687	0.7916
Mutagenesis_bond	0.7219	0.7746	0.7517	0.767	0.7783	0.7838	0.7697	0.7453
Mutagenesis_chain	0.631	0.7055	0.7694	0.6713	0.811	0.7976	0.8348	0.7797
Process	0.8358	0.8396	0.8993	0.8119	0.8554	0.8706	0.8502	0.8316

Tables 4 and 5 present the experimental results of the base learners and all the presented algorithms using the base learners. MITI [17] is chosen as the base learner in Table 4, and MIRI [21] is the base learner in Table 5. In the Tables, 'Base' denotes the base learner, and 'CS' indicates the cost sensitive method.

As Friedman's test [23] is a non-parametric statistical test for multiple classifiers and multiple domains, we performed it on the results in Tables 4 and 5. The null hypothesis for this test is that all the classifiers perform equally, and rejection of the null hypothesis means that there is at least one pair of classifiers with significantly different performance.

For Table 4 the test results are Friedman chi-squared = 25.0017, $df = 7$, and p-value = 0.0007583, and for Table 5 the results are Friedman chi-squared = 22.2381, $df = 7$, and p-value = 0.002311. The critical value for the chi-square distribution is 14.07 for a 0.05 level of significance for a single-tailed test. As 25.0017 and 22.2381 are both larger than 14.07, we can reject the hypothesis for both Tables 4 and 5.

We applied Nemenyi's post-hoc test [23] to determine which classifier has the best performance. First we ranked the Gmean values for each dataset with different classifiers. The sum of the ranks for all datasets is represented as $R_{.i}$, where i represents a classifier. Then we used the following formula to calculate the q value between different classifiers:

$$q_{ij} = (\bar{R}_i - \bar{R}_j) \Big/ \sqrt{\frac{k(k+1)}{6n}} \tag{24}$$

where k is the number of classifiers and n is the number of datasets. We then determined if one algorithm is better than another by comparing their q values with the critical value $q_\alpha = 3.19$ (gotten from [22]), as shown in Table 6. The result of 6-1-0 for Ab3 with MITI as the base learner means that the algorithm wins six times, is equal once, and loses zero times. If we set the scores as win=1, equal=0 and lose=-1, the total score of each algorithm in Tables 4 and 5 can be calculated. The result is shown in Table 6.

Table 6. Experiment result using the statistical test method (sorted by score from high to low)

	Ab3	Ab2	Ab4	AdaCost	Ab1	CS	Adaboost	Base
MITI	6-1-0	4-1-2	6-1-0	4-1-2	2-1-4	2-1-4	1-0-6	0-0-7
MIRI	6-1-0	6-1-0	2-3-2	2-3-2	2-3-2	2-3-2	1-0-6	0-0-7
Score	12	8	6	2	-2	-2	-10	-14

The experimental results show that all the proposed algorithms can improve the performance of the base learner. CS did not overcome the cost sensitive boosting methods, since it does not adopt a weight updating strategy. AdaCost did not outperform Ab3, Ab2 and Ab4, since in Ref. [5] AdaCost requires that β for the minority class be non-increasing with respect to C_n, which means the reward for correct classification is low when the cost is high [7]. The performance of Ab3 is the best in all experiments, and Ab2 and Ab4 are also competitive. The weighted updating strategy of cost-sensitive boosting algorithms increases the weights on the misclassified bags from the minority class more than it does on those from the majority class. Similarly, it decreases the weight on correctly classified bags from the minority class less than on those from the majority class.

6 Conclusions and Future Research

We have presented and analyzed the multi-instance class imbalance problem, and provided a novel framework for the design of cost-sensitive boosting algorithms for this problem. We compared the cost sensitive method, the Adacost algorithm, and the four proposed weight updating cost-sensitive boosting algorithms, along with two multi-instance tree methods.

Experimental evidence derived from standard datasets was presented to support the cost-sensitive optimality of the proposed algorithms. We found that cost-sensitive boosting consistently outperformed all other methods tested. In the future, we plan to investigate whether the proposed methods are sensitive to the cost setup. On the given datasets in our experiments, the updating strategies of Ab3 and Ab2 are more suitable than that of Ab4. Since the imbalance ratio is not very large for the chosen datasets in these experiments, in future work it would be worthwhile to investigate whether Ab4 can provide better result than Ab3 and Ab2 on highly class imbalanced multi-instance datasets. We also intend to research the application of the cost-sensitive boosting algorithms for multi-instance classification, and investigate other related algorithms.

Acknowledgement. The authors acknowledge the support of the Defence R&D Canada Centre for Operational Research and Analysis for this work. This work was funded by Defence R&D Canada Centre for Operational Research and Analysis under PWGSC Contract #W7714-115078/001/SV.

References

1. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference, pp. 148–156 (1996)
2. Kubat, M., Matwin, S.: Addressing the curse of imbalanced training sets: One-sided selection. In: Proceddings of the Fourteenth International Conference on Machine Learning, pp. 179–186 (1997)
3. Dietterich, T., Lathrop, R., Lozano-P´erez, T.: Solving the multiple instance problem with the axis-parallel rectangles. Artificial Intelligence 89(1-2), 31–71 (1997)
4. Maron, O., Lozano-Pérez, T.: A framework for multiple instance learning. In: Proc. of the 1997 Conf. on Advances in Neural Information Processing Systems 10, pp. 570–576 (1998)
5. Fan, W., Stolfo, S.J., Zhang, J., Chan, P.K.: AdaCost: Misclassification Cost-Sensitive Boosting. In: Proc. Int'l Conf. Machine Learning, pp. 97–105 (1999)
6. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Machine Learning 37(3), 297–336 (1999)
7. Ting, K.M.: A Comparative Study of Cost-Sensitive Boosting Algorithms. In: Proc. Int'l Conf. Machine Learning, pp. 983–990 (2000)
8. Wang, J., Zucker, J.D.: Solving the multiple-instance problem: A lazy learning approach. In: ICML (2000)
9. Japkowicz, N.: Learning from Imbalanced Data Sets: A Comparison of Various Strategies. In: Proc. Am. Assoc. for Artificial Intelligence (AAAI) Workshop Learning from Imbalanced Data Sets, pp. 10-15 (Technical Report WS-00-05) (2000)
10. Zhang, Q., Goldman, S.A.: EM-DD: An improved multiple instance learning technique. In: Neural Information Processing Systems 14 (2001)
11. Elkan, C.: The Foundations of Cost-Sensitive Learning. In: Proc. Int'l Joint Conf. Artificial Intelligence, pp. 973–978 (2001)
12. Ting, K.M.: An Instance-Weighting Method to Induce Cost-Sensitive Trees. IEEE Trans. Knowledge and Data Eng. 14(3), 659–665 (2002)
13. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research 16, 321–357 (2002)
14. Zhang, M.L., Goldman, S.: Em-dd: An improved multi-instance learning technique. In: NIPS (2002)
15. Andrews, S., Tsochandaridis, I., Hofman, T.: Support vector machines for multiple instance learning. Adv. Neural. Inf. Process. Syst. 15, 561–568 (2003)
16. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. ACM SIGKDD Explorations Newsletter 6(1), 20–29 (2004)
17. Blockeel, H., Page, D., Srinivasan, A.: Multi-instance tree learning. In: ICML (2005)
18. Sun, Y., Kamel, M.S., Wong, A.K.C., Wang, Y.: Cost-Sensitive Boosting for Classification of Imbalanced Data. Pattern Recognition 40(12), 3358–3378 (2007)

19. Foulds, J., Frank, E.: Revisiting multiple-instance learning via embedded instance selection. In: Wobcke, W., Zhang, M. (eds.) 21st Australasian Joint Conference on Artificial Intelligence Auckland, New Zealand, pp. 300–310 (2008)
20. Leistner, C., Saffari, A., Bischof, H.: MIForests: Multiple-instance learning with randomized trees. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010, Part VI. LNCS, vol. 6316, pp. 29–42. Springer, Heidelberg (2010)
21. Bjerring, L., Frank, E.: Beyond trees: Adopting MITI to learn rules and ensemble classifiers for multi-instance data. In: Wang, D., Reynolds, M. (eds.) AI 2011. LNCS (LNAI), vol. 7106, pp. 41–50. Springer, Heidelberg (2011)
22. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press (2011)
23. Wang, X., Shao, H., Japkowicz, N., Matwin, S., Liu, X., Bourque, A., Nguyen, B.: Using SVM with Adaptively Asymmetric Misclassification Costs for Mine-Like Objects Detection. In: ICMLA (2012)