Parameter Inference of Cost-Sensitive Boosting Algorithms

Yanmin Sun[1] A.K.C. Wong[1] Yang Wang[2]

[1] Pattern Analysis and Machine Intelligence Lab, University of Waterloo
sunym,akcwong@pami.uwaterloo.ca
[2] Pattern Discovery Software Ltd.
yang.wang@patterndiscovery.com

Abstract. Several cost-sensitive boosting algorithms have been reported as effective methods in dealing with class imbalance problem. Misclassification costs, which reflect the different level of class identification importance, are integrated into the weight update formula of AdaBoost algorithm. Yet, it has been shown that the weight update parameter of AdaBoost is induced so as the training error can be reduced most rapidly. This is the most crucial step of AdaBoost in converting a weak learning algorithm into a strong one. However, most reported cost-sensitive boosting algorithms ignore such a property. In this paper, we come up with three versions of cost-sensitive AdaBoost algorithms where the parameters for sample weight updating are induced. Then, their identification abilities on the small classes are tested on four "real world" medical data sets taken from UCI Machine Learning Database based on F-measure. Our experimental results show that one of our proposed cost-sensitive AdaBoost algorithms is superior in achieving the best identification ability on the small class among all reported cost-sensitive boosting algorithms.

1 Introduction

Reports from both academy and industry indicate that the class imbalance problem has posed a serious drawback of classification performance attainable by most standard learning methods which assume a relatively balanced distribution and equal error cost of the classes [6, 10]. *Class imbalance problem* can be interpreted in two aspects: the *imbalanced class distribution* and the *non-uniform misclassification costs*. Hence, the crucial learning issue is that the class distribution is skewed and the recognition importance on rare events is much higher than that on normal cases. Assuming the balanced class distribution and even recognition importance, traditional learning algorithms do not always produce classifiers which are capable of achieving satisfactory identification performances on rare classes.

AdaBoost (Adaptive Boosting) algorithm, introduced by Freund and Schapire [7, 12], is reported as an effective boosting algorithm to improve classification accuracy. In view that prevalent classes usually contribute more to the overall

classification accuracy, the weighting strategy of AdaBoost may bias towards the prevalent classes. Hence the desired identification ability on small classes is not guaranteed. Cost-sensitive boosting algorithms are therefore developed such that the boosting process may cater to the costly class [5, 13]. However, most reported cost-sensitive boosting algorithm neglect the effects of cost items when choosing the weight update parameter, which is crucial in converting a "weaker" learning algorithm into a strong one.

In this paper, we come up with three versions of cost-sensitive AdaBoost algorithms by inducing the misclassification costs into the weight update formula of AdaBoost in three different ways. For each version, weight update parameter is recalculated taking misclassification costs into consideration. These adaptations retain the good feature of AdaBoost while becoming sensitive to different level of learning importance of different classes. To evaluate their recognition abilities on small classes, four "real world" medical data sets are tested. These data are collections of typical disease diagnostics. Thus, the class imbalance problem prevails in these data sets. *F-measure* evaluation is adopted for performance comparisons.

This paper is organized as follows. Following the introduction in Section 1, section 2 describes the AdaBoost algorithm and addresses the problems of cost-sensitive learning. Section 3 details the methods of integrating misclassification cost into AdaBoost algorithm. Section 4 describes the experimental data, base learner and evaluation measurements. Section 5 compares the recognition abilities of different cost-sensitive boosting algorithms. Section 6 provides the conclusions.

2 AdaBoost and Cost-Sensitive Boosting

2.1 AdaBoost Algorithm

AdaBoost algorithm reported in [7, 12] takes as input a training set $\{(x_1, y_1), \cdots, (x_m, y_m)\}$ 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 as below:

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 using distribution D_t
2.Choose weight updating parameter: α_t
3.Update and normalize sample weights:

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

Where, Z_t is a normalization factor.
Output the final classifier: $H(x) = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$

It has been shown in [12] that the training error of the final classifier is bounded as below:

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leq \frac{1}{m}\sum_i exp(-y_i f(x_i)) = \prod_t Z_t \tag{1}$$

where,

$$Z_t = \sum_i D^{(t)}(i) exp(-\alpha_t h_t(x_i) y_i) \tag{2}$$

Minimize Z_t on each round, α_t is induced as

$$\alpha_t = \frac{1}{2} log \frac{\sum_{i,y_i=h_t(x_i)} D(i)^{(t)}}{\sum_{i,y_i \neq h_t(x_i)} D(i)^{(t)}} \tag{3}$$

To ensure that the selected value of α_t is positive, the following condition should hold

$$\sum_{i,y_i=h_t(x_i)} D(i)^{(t)} > \sum_{i,y_i \neq h_t(x_i)} D(i)^{(t)} \tag{4}$$

2.2 Cost-Sensitive Boosting

Cost-sensitive classification considers varying costs of different misclassification types. Thus the cost-sensitive learning process seeks to minimize the number of high cost errors and the total misclassification cost. Reported works on research in cost-sensitive learning can be categorized into three main groups related to the learning phases of a classifier: 1) Data preprocessing: modifying the distribution of the training set with regards to misclassification costs so that the modified distribution bias towards the costly classes [1, 3]; 2) Classifier Learning: making a specific classifier learning algorithm cost-sensitive [2, 8]; and 3) Classification: using Bayes risk theory to assign each sample to its lowest risk class [4].

Cost-sensitive learning methods in the first group, known as *cost-sensitive learning by example weighting* in [1], is very general since it applies to arbitrary classifier learners and does not change the underlying learning algorithms. In this method, an example-dependent cost is first converted into example weight. Then, a learning algorithm is applied to training examples drawn from this weighted distribution. Several variants of AdaBoost algorithm reported in [5, 13] are with this approach, known as cost-sensitive boosting algorithms.

These cost-sensitive boosting algorithms inherit the learning framework of AdaBoost algorithm. They feed the misclassification costs into the weight update formula of AdaBoost, so that the updated data distribution on the successive boosting round can bias towards the small classes. Except using cost items to update sample weights, CSB1 [13] does not use any α_t factor (or $\alpha_t = 1$), CSB2

[13] uses the same α_t as computed by AdaBoost, and AdaCost [5] introduces a cost adjustment function into weight update rule of AdaBoost. The requirement for this function is: for an instance with a higher cost factor, the function increases its weights "more" if the instance is misclassified, but decreases its weight "less" if otherwise.

The crucial step in AdaBoost algorithm is the selection of the weight update parameter which enables the training error be reduced rapidly. This process is an efficient boosting scheme to convert a weak learning algorithm into a strong one. When introducing the misclassification costs into the weight updating formula, it is necessary to integrate the cost items into the parameter calculation in order to maintain the boosting efficiency. Out of all reported cost-sensitive boosting algorithms, only in AdaCost misclassification costs are taken into consideration when calculating the weight update parameter α. However, the problems with this adaptation are: 1) the selection of the adjustment function is ad hoc; 2) when the cost items (C_P and C_N) are set to 1, AdaCost will not become the original AdaBoost algorithm, thus the steepest descent search of AdaBoost is varied by the cost adjustment function.

3 Cost-Sensitive AdaBoost Algorithms

In order to adapt the weight update strategy of AdaBoost algorithm for cost-sensitive learning, we propose three versions of cost-sensitive AdaBoost algorithms according to the ways we feed the the cost factor into the weight update formula of AdaBoost: inside the exponent, outside the exponent, and both inside and outside the exponent. Let $\{(x_1, y_1, c_1), \cdots, (x_m, y_m, c_m)\}$ be a sequence of training samples, where, as denoted previously, each x_i is an n-tuple of attribute values; y_i is a class label in Y where $Y = \{-1, +1\}$, and c_i is the cost factor belonging to the none-negative real domain R^+. Three modifications of the weight update formula of AdaBoost then become:

$$D^{(t+1)}(i) = \frac{D^{(t)}(i)exp(-\alpha_t c_i h_t(x_i)y_i)}{Z_t} \tag{5}$$

$$D^{(t+1)}(i) = \frac{c_i D^{(t)}(i)exp(-\alpha_t h_t(x_i)y_i)}{Z_t} \tag{6}$$

$$D^{(t+1)}(i) = \frac{c_i D^{(t)}(i)exp(-\alpha_t c_i h_t(x_i)y_i)}{Z_t} \tag{7}$$

Thus, respecting to each modification of weight update formula, a new α value should be calculated to minimize the weighted training error. Taking each modification as a new learning objective, three cost-sensitive AdaBoost algorithms can be developed. We denote them as AdaC1, AdaC2 and AdaC3 respectively. Adopting the inference method used in [12], the calculation of weight updating factor α for each algorithm can be presented in the following subsections.

3.1 AdaC1

Unraveling the weight update rule of Equation 5, we obtain

$$D^{(t+1)}(i) = \frac{exp(-\sum_t \alpha_t c_i y_i h_t(x_i))}{m \prod_t Z_t} = \frac{exp(-c_i y_i f(x_i))}{m \prod_t Z_t} \tag{8}$$

where,

$$Z_t = \sum_i D^{(t)}(i) exp(-\alpha_t c_i y_i h_t(x_i)) \tag{9}$$

Here, the training error bound as stated in Equation 1 still holds. Thus, the learning objective on each round is to find α_t and h_t so as to minimize Z_t (Equation 9). h_t can be trained while minimizing the weighted training error based on current data distribution. Then α_t is selected to minimize Equation 9. According to [12], once $c_i y_i h_t(x_i) \in [-1, +1]$, the following inequality holds

$$\sum_i D(i)^{(t)} exp(-\alpha c_i y_i h(x_i)) \leq \sum_i D(i)^{(t)} (\frac{1 + c_i y_i h_t(x_i)}{2} e^{-\alpha} + \frac{1 - c_i y_i h_t(x_i)}{2} e^{\alpha}) \tag{10}$$

By zeroing the first derivative of the right hand side of the inequality (10), α_t can be determined as:

$$\alpha_t = \frac{1}{2} log \frac{1 + \sum\limits_{i, y_i = h_t(x_i)} c_i D(i)^{(t)} - \sum\limits_{i, y_i \neq h_t(x_i)} c_i D(i)^{(t)}}{1 - \sum\limits_{i, y_i = h_t(x_i)} c_i D(i)^{(t)} + \sum\limits_{i, y_i \neq h_t(x_i)} c_i D(i)^{(t)}} \tag{11}$$

To ensure that the selected value of α_t is positive, the following condition should hold

$$\sum_{i, y_i = h_t(x_i)} c_i D(i)^{(t)} > \sum_{i, y_i \neq h_t(x_i)} c_i D(i)^{(t)} \tag{12}$$

3.2 AdaC2

Unraveling the weight update rule of Equation 6, we obtain

$$D^{(t+1)}(i) = \frac{c_i^t exp(-\sum_t \alpha_t y_i h_t(x_i))}{m \prod_t Z_t} = \frac{c_i^t exp(-y_i f(x_i))}{m \prod_t Z_t} \tag{13}$$

where,

$$Z_t = \sum_i c_i D^{(t)}(i) exp(-\alpha_t y_i h_t(x_i)) \tag{14}$$

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

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

There exists a constant γ such that $\forall i, \gamma < c_i^{t+1}$. Then,

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leq \prod_t Z_t \sum_i \frac{c_i D^t(i)}{c_i^{t+1}} \leq \frac{1}{\gamma}\prod_t Z_t \tag{16}$$

Since γ is a constant, the learning objective on each round is to find α_t and h_t so as to minimize Z_t (Equation 14). h_t can be trained while minimizing the weighted training error based on current data distribution. Then α_t is selected to minimize Equation 14 as:

$$\alpha_t = \frac{1}{2} log \frac{\sum\limits_{i, y_i = h_t(x_i)} c_i D(i)^{(t)}}{\sum\limits_{i, y_i \neq h_t(x_i)} c_i D(i)^{(t)}} \tag{17}$$

To ensure that the selected value of α_t is positive, the following condition should hold

$$\sum_{i, y_i = h_t(x_i)} c_i D(i)^{(t)} > \sum_{i, y_i \neq h_t(x_i)} c_i D(i)^{(t)} \tag{18}$$

3.3 AdaC3

The weight update formula (Equation 7) of AdaC3 is a combination of that of AdaC1 and AdaC2 (with the cost items both inside and outside the exponential function). Then the training error bound of AdaC3 could be expressed as

$$\frac{1}{m}|\{i : H(x_i) \neq y_i\}| \leq \frac{1}{\gamma}\prod_t Z_t \tag{19}$$

where, γ is a constant and $\forall i, \gamma < c_i^{t+1}$, and

$$Z_t = \sum_i c_i D^{(t)}(i) exp(-\alpha_t c_i y_i h_t(x_i)) \tag{20}$$

Since γ is a constant, the learning objective on each round is to find α_t and h_t so as to minimize Z_t (Equation 20). h_t can be trained while minimizing the weighted training error based on current data distribution. Then α_t is selected to minimize Equation 20.

According to [12], once $c_i y_i h_t(x_i) \in [-1, +1]$, the following inequality holds

$$\sum_i c_i D(i)^{(t)} exp(-\alpha c_i y_i h(x_i)) \leq \sum_i c_i D(i)^{(t)} (\frac{1 + c_i y_i h_t(x_i)}{2} e^{-\alpha} + \frac{1 - c_i y_i h_t(x_i)}{2} e^{\alpha}) \tag{21}$$

By zeroing the first derivative of the right hand side of inequality (21), α_t can be determined as:

$$\alpha_t = \frac{1}{2} log \frac{\sum_i c_i D(i)^{(t)} + \sum_{i, y_i = h_t(x_i)} c_i^2 D(i)^{(t)} - \sum_{i, y_i \neq h_t(x_i)} c_i^2 D(i)^{(t)}}{\sum_i c_i D(i)^{(t)} - \sum_{i, y_i = h_t(x_i)} c_i^2 D(i)^{(t)} + \sum_{i, y_i \neq h_t(x_i)} c_i^2 D(i)^{(t)}} \tag{22}$$

To ensure that the selected value of α_t is positive, the following condition should hold

$$\sum_{i, y_i = h_t(x_i)} c_i^2 D(i)^{(t)} > \sum_{i, y_i \neq h_t(x_i)} c_i^2 D(i)^{(t)} \tag{23}$$

4 Experiment Settings

4.1 Base Learner

To test these cost-sensitive AdaBoost algorithms, we select an associative classification learning system, namely High-Order Pattern and Weigh-of-Evidence Rule Based Classifier(HPWR) as the base learner. The selected base learner HPWR is a complete and independent system. Employing residual analysis and mutual information for decision support, it generates classification patterns and rules in two stages: 1) discovering high-order significant event associations using residual analysis in statistics to test the significance of the occurrence of a pattern candidate against its default expectation[15]; and 2) generating classification rules with weight of evidence attached to each of them to quantify the evidence of significant event associations in support of, or against a certain class membership[14] for a given sample. Hence, HPWR is a mathematically well-developed system with a more comprehensive and rigorous theoretical basis.

4.2 Data sets

We use four medical diagnosis data sets "Cancer", "Pima", "Hypothyroid" and "Sick-euthyroid" taken from UCI Machine Learning Database [11] to test the performances of these three cost-sensitive AdaBoost algorithms. These data sets all have two output labels: one denoting the disease category is treated as the positive class and another representing the normal category is treated as negative class. The percentages of the positive classes are 29.72%, 34.90%, 4.77% and 9.26% respectively.

In these experiments, continuous data in each data set is pre-discretized using the the commonly used discretization utility of MLC++ [9] on the default setting and missing values are treated as having the value "?". Five-fold cross-validation is used on all of the data sets. For consistency, exactly the same data are used to train and test all of these cost-sensitive boosting algorithms.

4.3 Cost Factor

The misclassification costs for samples in the same category are set the same value: C_P denoting the misclassification cost of the positive class and C_N representing that of the negative class. Conceptually, C_P should be greater than C_N. As constrained by the inferences of Inequality 10 and 21, their values should be no greater than 1. Thus, this condition should hold $1 \geq C_P \geq C_N > 0$. In these experiments, relative misclassification costs are set as $[1.0 : 1.0,\ 1.0 : 0.9,\ 1.0 : 0.8,\ 1.0 : 0.7,\ 1.0 : 0.6,\ 1.0 : 0.5,\ 1.0 : 0.4,\ 1.0 : 0.3,\ 1.0 : 0.2,\ 1.0 : 0.1]$ on two classes for all the data sets. Then, with each pair of cost settings, the performance of each learning algorithm is evaluated on 5-fold cross-validation.

4.4 F-measure for Performance Evaluation

In information retrieval, with respect to a given class, *Recall* is defined as the percentage of retrieved objects that are relevant; and *Precision* is defined as the percentage of relevant objects that are identified for retrieval. Clearly neither of these two measures are adequate by themselves to evaluate the recognition performance on a given class. Thus, the F-measure (F), a measure often-used by the Information Retrieval community for evaluating the performance of the right objects, is devised as a combination of Recall and Precision:

$$F = \frac{2RP}{R + P} \tag{24}$$

It follows that if the F-measure is high when both the recall and precision should be high. This implies that the F-measure is able to measure the "goodness" of a learning algorithm on the current class of interest.

5 Performance Comparisons

Table (1) shows the best F-measure value, as well as the misclassification cost setting of each algorithm on each data set. In general, over the 4 datasets, AdaC3 wins twice, AdaC2 and AdaCost each wins 1 time respectively and AdaC3 also achieves the highest average F-measure value over the four data sets. The performances of CSB1 and CSB2 are obviously not satisfactory.

The basic idea of cost-sensitive boosting algorithm in dealing with the class imbalance problem is to maintain a considerable weighted sample size of the positive class at each iteration of boosting. Then the recognition recall measurement is increased on the positive class. This is the critical step in dealing with

Table 1. F-measure Comparisons of Cost-Sensitive Boosting Algorithms

		HPWR	AdaBoost	AdaC1	AdaC2	AdaC3	AdaCost	CSB1	CSB2
Cancer	Cost			1:0.6	1:0.7	1:0.7	1:0.2	1:0.6	1:0.4
	$F_+(\%)$	40.21	47.10	50.64	53.98	**54.97**	50.75	47.88	50.73
Hypo	Cost			1:0.9	1:0.9	1:0.9	1:0.9	1:0.9	1:0.8
	$F_+(\%)$	55.84	81.99	84.20	**84.56**	83.02	82.15	82.72	69.42
Pima	Cost			1:0.5	1:0.6	1:0.7	1:0.3	1:0.6	1:0.9
	$F_+(\%)$	67.98	67.66	72.58	71.03	**73.61**	69.03	67.30	65.58
Sick	Cost			1:0.8	1:0.8	1:0.9	1:0.9	1:0.8	1:0.8
	$F_+(\%)$	69.22	79.77	82.51	78.05	81.04	**82.67**	77.77	62.89
Average	$F_+(\%)$	58.31	69.13	72.48	71.90	**73.16**	71.24	68.92	57.93

the class imbalance problem. However, there is a tradeoff between recall and precision: precision declines as recall increases. When the positive class is over resampled, recall of the positive class is greatly improved. Yet, more samples from the negative class are categorized to the positive class. As a consequence, the recognition precision on the positive class gets worse, and the F-measure cannot be satisfactory under this situation. Hence, to balance the tradeoff between recall and precision and get the best F-measure value, the boosted weights on the positive class should be in a proper degree which is adequate to obtain a satisfactory recall yet not too much to reduce the precision. Misclassification cost setting is one aspect that influences this issue. Table 1 shows the best ratio setting at which the best F-measure are obtained for each algorithm. Another important affect is related to the weighting scheme of each boosting algorithm. Experimental results reported in Table 1 show that AdaC3 achieves the best F-measure values on two data sets and also the highest average F-measure value over the four data sets.

6 Conclusion

In this paper, we have proposed three new cost-sensitive AdaBoost algorithms to tackle the class imbalance problem in the context of bi-class applications. Based on how cost items are used in the equation, three versions of cost-sensitive boosting algorithms, known as AdaC1, AdaC2 and AdaC3, are developed. To ensure boosting efficiency, α is recalculated taking misclassification costs into consideration for each version. We find that these adaptations retain the good feature of AdaBoost yet adequately sensitive to adjust to cope with different level of learning importance corresponding to different classes. To evaluate their recognition abilities on small classes, four "real world" medical data sets are tested. *F-measure* evaluation is adopted for performance comparisons. In our classification implementation and comparison, we select HPWR as the base learner. When comparing the recognition ability on the small class of each cost-sensitive AdaBoost algorithm, our experimental results show that AdaC3 is superior in

achieving the best performance among all reported cost-sensitive boosting algorithms. Further study on weight updating effect of each proposed cost-sensitive boosting algorithm is recommended for theoretically reasoning this observation.

References

1. N. Abe, B. Zadrozny, and 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*, pages 3–11, Seattle, WA, August 2004.
2. J. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C.E. Brodley. Pruning decision trees with misclassification costs. In *Proceedings of Tenth European Conference on Machine Learning(ECML-98)*, pages 131–136, Chemnitz, Germany, April 1998.
3. P. Chan and S. Stolfo. Toward scalable learning with non-uniform class and cost distributions. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data MIning*, pages 164–168, New York, NY, August 1998.
4. C. Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, Seattle, Washington, August 2001.
5. W. Fan, S.J. Stolfo, J. Zhang, and P.K. Chan. Adacost:misclasification cost-sensitive boosting. In *Proc. of Sixth International Conference on Machine Learning(ICML-99)*, pages 97–105, Bled, Slovenia, 1999.
6. T.E. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.
7. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an aplication to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, August 1997.
8. P. Geibel and F. Wysotzki. Perceptron based learning with example dependent and noisy costs. In T. Fawcett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 218–226. AAAI Press / Mit Press, 2003.
9. R. Kohavi, D. Sommerfield, and J. Dougherty. *Data Mining Using MLC++: A machine learning library in C++. Tools with Artificial Intelligence*. IEEE CS Press, 1996.
10. R. Kubat M., Holte and S. Matwin. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30:195–215, 1998.
11. P. M. Murph and D. W. Aha. *UCI Repository Of Machine Learning Databases*. Dept. Of Information and Computer Science, Univ. Of California: Irvine, 1991.
12. R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
13. K.M. Ting. A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the 17th International Conference on Machine Learning*, pages 983–990, Stanford University, CA, 2000.
14. Y. Wang and A. K. C. Wong. From association to classification: Inference using weight of evidence. *IEEE Trans. On Knowledge and Data Engineering*, 15(3):764–767, 2003.
15. A.K.C. Wong and Y. Wang. High order pattern discovery from discrete-valued data. *IEEE Trans. On Knowledge and Data Engineering*, 9(6):877–893, 1997.