Boosting Trees for Cost-Sensitive Classifications

Kai Ming Ting[1] and Zijian Zheng[2]

[1] Department of Computer Science, University of Waikato, Hamilton, New Zealand.
[2] School of Computing and Mathematics, Deakin University, Vic 3217, Australia.

Abstract. This paper explores two boosting techniques for cost-sensitive tree classifications in the situation where misclassification costs change very often. Ideally, one would like to have only one induction, and use the induced model for different misclassification costs. Thus, it demands robustness of the induced model against cost changes. Combining multiple trees gives robust predictions against this change. We demonstrate that the two boosting techniques are a good solution in different aspects under this situation.

1 Introduction

Most research on classifier learning has focused on minimum error classification. It aims to minimize the number of incorrect predictions or classifications made by classifiers. This kind of learning method ignores the differences between different types of incorrect prediction. It is very common in real world applications that different types of incorrect prediction cost differently. The cost of incorrect predictions is more important than the number of incorrect predictions in many real world domains such as in medical and financial areas. For example, in medical diagnosis, diagnosing someone as healthy when one has a life-threatening disease is usually considered to be more serious (thus higher cost) than another type of error—of diagnosing someone as ill when one is in fact healthy. Nevertheless, very little attention has been paid to cost-sensitive classification where the objective is to minimize the total cost of incorrect predictions or the number of high cost errors.

Moreover, in some cost-sensitive classification situations, misclassification costs may change very often. For example, in bank loan decision making, managers in different branches may assign different costs to the same type of incorrect decision. In addition, the costs may change from time to time even within the same branch. To the best knowledge of the authors, this situation has not been investigated. In this paper, we explore cost-sensitive classification techniques to handle this type of situation, and focus on decision tree learning in this study.

The most straightforward and simple approach to this problem is to alter the prediction selection process during classification, without modifying the classifier learning process. This can be done for a decision tree learning algorithm, such as C4.5 (Quinlan, 1993), in the following fashion. During the classification stage, an example to be classified is assigned the class with the minimum expected misclassification cost (Michie, Spiegelhalter, & Taylor, 1994) at the leaf to which the example is traced down, rather than the class with the maximum weight.

Because no modification to the tree induction process is required, the same tree can be re-used when the misclassification costs change. C4.5c is the variant of C4.5 modified in this manner and it is used as the base line of this research.

Intuitively, combining multiple models shall give more robust predictions than a single model under the situation where misclassification costs change very often. Boosting has been shown to be an effective method of combining multiple models in order to enhance the predictive accuracy of a single model (Quinlan, 1996; Schapire, Freund, Bartlett, & Lee, 1997). Thus, it is natural to think that boosting might also reduce the misclassification costs of C4.5c. In this paper, we explore two techniques of boosting C4.5c. The first technique is ordinary boosting combined with the minimum expected cost criterion. The second technique is a variant of ordinary boosting which utilizes the misclassification cost information during the induction of decision trees. We call the first method *Boosting*, and the second *Cost-Boosting*. We conduct empirical evaluation to assess the performance of Boosting and Cost-Boosting under the situation.

The next section describes the procedures used in Boosting and Cost-Boosting. Section 3 reports experiments with C4.5c, Boosting, and Cost-Boosting. We summarize our findings in the final section.

2 Boosting and Cost-Boosting

Here, Boosting is implemented by maintaining a weight for each training example (Quinlan, 1996) rather than drawing a succession of independent bootstrap samples from the original examples. Boosting induces multiple individual classifiers in sequential trials. At the end of each trial, the vector of weights is adjusted to reflect its importance for the next induction trial. This adjustment effectively increases the weights of misclassified examples. These weights cause the learner to concentrate on different instances in each trial and so lead to different classifiers. Finally, the individual trees are combined through voting to form a composite classifier. The Boosting procedure is shown as follows. Note that the weight adjustment formula in step (iii) below are from a new version of boosting (Schapire, Freund, Bartlett, & Lee, 1997).

Boosting procedure: Given a training set $\mathcal{T}$ containing N examples, $w_k(n)$ denotes the weight of the nth example at the kth trial, where $w_1(n) = 1/N$ for every n. In each trial $k = 1, \ldots, K$, the following steps are carried out.

(i) A decision tree T_k is constructed by using C4.5 from the training set under the weight distribution w_k.

(ii) $\mathcal{T}$ is classified using the decision tree T_k. Let $d(n) = 1$ if the nth example in $\mathcal{T}$ is classified incorrectly; $d(n) = 0$ otherwise. The error rate of this tree, ϵ_k, is defined as

$$\epsilon_k = \sum_n w_k(n)d(n). \tag{1}$$

If $\epsilon_k \geq 0.5$ or $\epsilon_k = 0$, then all $w_k(n)$ is set equal and perturbed with uniform noise and re-normalized, and carry on the process from step (i).

(iii) The weight vector $w_{(k+1)}$ for the next trial is created from w_k as follows:

$$w_{(k+1)}(n) = w_k(n)\frac{exp(-\alpha_k(-1)^{d(n)})}{z_k}, \tag{2}$$

where the normalizing term z_k and α_k are defined as

$$z_k = 2\sqrt{(1-\epsilon_k)\epsilon_k}, \qquad \alpha_k = \tfrac{1}{2}ln((1-\epsilon_k)/\epsilon_k). \tag{3}$$

After K trials, the decision trees $T_1, \ldots, T_K$ are combined to form a single composite classifier. Given an example, the final classification of the composite classifier relies on the votes of all the individual trees. The vote of the tree T_k is worth α_k units. Since we use the expected misclassification cost to select the predicted class, the voting is not simply summing up the vote of every individual tree. Instead, the following computation is performed.

Let $t_k(x)$ be the leaf of the tree T_k where the example x falls into, and $W_i(t_k(x))$ be the total weight of class i examples in $t_k(x)$. The expected misclassification cost for class j with respect to the example x and the composite classifier consisting of trees $T_1, \ldots, T_K$ is given by:

$$EC_j(x) \propto \sum_i^I \sum_k^K \alpha_k W_i(t_k(x)) cost(i,j), \tag{4}$$

where $cost(i,j)$ is the misclassification cost of classifying a class i example as class j; and I is the total number of classes.

To classify a new example x, $EC_j(x)$ is computed for every class. The example x is assigned to class j with the smallest value for $EC_j(x)$. That is, $EC_j(x) < EC_{j'}(x)$ for all $j' \neq j$.

From the description above, it can be seen that Boosting only utilizes the misclassification cost information during classification through Equation (4). Its classifier induction process does not employ the cost information. This allows a single Boosting induction to be used for different misclassification costs.

One can modify the Boosting procedure so that the weights of misclassified examples are updated according to the costs associated with these misclassifications. Thus, each subsequent tree is cost-sensitive. Based on this idea, Boosting is modified to create a variant: **Cost-Boosting**. Cost-Boosting uses the same procedure as Boosting except the weight adjustment process in step (iii). We assume a unity condition $cost(i,j) \geq 1, \forall i \neq j$ (see details in the next section); and the weight adjustment is re-defined as follows.

$$w_{(k+1)}(n) = \frac{w'_{(k+1)}(n)}{\sum_n w'_{(k+1)}(n)}, \tag{5}$$

$$w'_{(k+1)}(n) = \begin{cases} cost(actual(n), predicted(n)), & \text{if } actual(n) \neq predicted(n); \\ Nw_k(n), & \text{otherwise.} \end{cases} \tag{6}$$

Because all trees (except the first one) are cost-sensitive, Cost-Boosting needs to perform induction every time the misclassification costs change.

During the classification stage, Cost-Boosting also uses Equation (4) for selecting the class with the minimum expected cost except that each individual tree in Cost-Boosting is worth 1 unit for voting, that is, $\alpha_k = 1$.

As Boosting and Cost-Boosting, the base line algorithm C4.5c also employs the same formulae for selecting the class, in which $K = 1$ and $\alpha_k = 1$.

Note that the first tree in both Boosting and Cost-Boosting is exactly the same as that produced by C4.5c.

3 Experiments

In this section, we empirically evaluate Boosting and Cost-Boosting by comparing with C4.5c. Twenty natural domains from the UCI machine learning repository (Merz & Murphy, 1997) are used in the experiments. This test suite covers a wide variety of different domains with respect to dataset size, the number of classes, the number of attributes, and types of attributes.

The misclassification cost information is provided in the form of a cost matrix of size $I \times I$, where I is the number of classes in a domain. The off-diagonal entries contain the costs of misclassifications, while the entries on the diagonal contain the cost of correct classifications which are equal to zero.

Since no datasets from real-world domains where misclassification costs often change are available to us, we simulate this type of situation by artificially generating cost matrices. A cost matrix for each domain is randomly generated for each experimental run. In each matrix, the costs in the off-diagonal entries are any randomly generated integer between 1 and 10. In two-class domains, one of the two off-diagonal entries must be 1 and the other more than 1. In multi-class domains, at least one of the entries is 1. The only reason using this *unity condition* is to allow us to measure the number of high cost errors.

One 10-fold cross-validation is carried out in each domain, except in the Waveform domain where 10 pairs of training set of size 300 and test set of size 5000 are randomly generated. In each fold, we conduct 10 runs on the same training and test sets using 10 randomly generated cost matrices to simulate the cost changing situation. In each run, the same cost matrix is employed in training and testing. All reported results are averaged over 100 runs.

We use two measures to evaluate the performance of the algorithms employed for cost-sensitive classification. The first measure is the *total cost of misclassifications* made by a classifier on a test set (i.e., $\sum_m cost(actual(m), predicted(m))$). The second measure is the *number of high cost errors*. It is the number of misclassifications associated with costs higher than 1 made by a classifier on a test set. Note that the lowest misclassification cost is 1 in a normalized cost matrix. A good cost-sensitive classifier should have low total misclassification cost, or small number of high cost errors, or both.

The parameter K controlling the number of classifiers generated in both Boosting and Cost-Boosting is set at 10 for the experiments. It is interesting to see the performance improvement that can be gained by a single order of magnitude increase in computation. All C4.5c parameters have their default values, and only pruned trees are used.

Table 1. Comparison of C4.5c, Boosting and Cost-Boosting

Datasets	C4.5c		Boosting vs C4.5c		Cost-Boosting vs C4.5c		Cost-Boosting vs Boosting	
	cost	#hce	cost ratio	#hce ratio	cost ratio	#hce ratio	cost ratio	#hce ratio
Echocardiogram	7.9	0.82	.82	.16	.81	.32	.99	2.00
Hepatitis	7.5	0.93	.71	.13	.68	.42	.96	3.25
Heart	19.6	2.67	.64	.13	.62	.31	.97	2.40
Horse	17.0	1.62	1.08	.04	.93	.74	.86	20.00
Credit	24.5	2.88	1.15	.26	.81	.59	.70	2.25
Breast-W	12.8	1.85	.66	.21	.51	.36	.78	1.76
Diabetes	39.0	3.75	.92	.22	.84	.43	.92	1.98
Hypothyroid	8.4	1.20	1.42	.47	.90	.78	.63	1.65
Euthyroid	21.2	3.00	1.26	.42	.89	.73	.70	1.74
Coding	1277.9	139.68	.70	.07	.67	.13	.95	1.91
Lymphography	14.9	2.52	.95	1.17	.89	1.10	.94	.94
Glass	38.1	6.41	.67	.82	.69	.84	1.03	1.03
Waveform	7330.9	1189.75	.61	.72	.60	.70	.99	.98
Soybean	29.6	5.21	.90	.99	.72	.89	.80	.90
Annealing	30.0	5.52	.86	.97	.82	.98	.95	1.01
Vowel	103.8	17.27	.56	.73	.60	.81	1.07	1.10
Splice	96.7	15.38	1.05	1.19	.83	.94	.79	.79
Abalone	691.8	118.70	.81	.92	.77	.87	.95	.94
Nettalk(s)	475.7	82.42	.79	.93	.75	.87	.95	.95
Satellite	466.9	77.84	.67	.85	.64	.79	.96	.93
Mean			.86	.57	.75	.68	.89	2.43

Table 1 shows the misclassification costs and the number of high cost errors of C4.5c, and the ratios for the pair-wise comparison among C4.5c, Boosting, and Cost-Boosting are presented in the last three columns. A ratio of less than 1 for Boosting vs C4.5c, for example, represents an improvement due to Boosting. The mean ratios over 20 domains are shown in the last row.

Boosting reduces the misclassification costs of C4.5c in 15 out of the 20 domains, and increases the misclassification costs of C4.5c in the other 5 domains. On average, Boosting achieves 14% reduction over C4.5c in terms of the misclassification costs. Cost-Boosting further reduces the misclassification costs of C4.5c. Cost-Boosting obtains lower costs than C4.5c in all 20 domains. The average reduction is 25%. Compared with Boosting, Cost-Boosting has lower costs in 18 domains, and higher costs in only 2 domains. On average, Cost-Boosting achieves 11% lower misclassification costs than Boosting. These results clearly show the advantage of Boosting over C4.5c, and the advantage of Cost-Boosting over both C4.5c and Boosting in terms of misclassification costs.

In terms of the number of high cost errors, Boosting improves C4.5c dramatically. It achieves 43% reduction over C4.5c on average. In comparison to C4.5c, Cost-Boosting achieves the average reduction of 32%. Comparing Cost-Boosting directly to Boosting, the former has the number of high cost errors 2.43 times larger than the latter. Note that one single domain, Horse, makes a significant contribution to this increase. In this domain, Boosting has 0.06 high cost errors, while Cost-Boosting has 1.20 high cost errors. This gives a ratio of 20.00 for Cost-Boosting vs Boosting.

Due to lack of space, results of investigations on some related issues, such as the effect of K, are not reported here. They can be found in the full report (Ting & Zheng (1998) at [http://www.cs.waikato.ac.nz/cs/Pub/Staff/kaiming.html]).

4 Summary

This paper has explored two techniques for dealing with cost-sensitive decision tree classification in the situation where misclassification costs change very often. One is Boosting—the ordinary boosting with the minimum expected cost criterion. It makes use of the cost information during classification stage by using the minimum expected cost criterion to select the predicted class.

Another technique is Cost-Boosting, a variant of Boosting, designed specifically for cost-sensitive classification in this paper. This technique takes the advantage of the available misclassification cost information during training, which makes the boosting procedure more sensitive to the cost of misclassification. However, this advantage comes at a price of extra computation—Cost-Boosting needs to create new classifiers every time misclassification costs change.

Experimental results show that both Boosting and Cost-Boosting can significantly reduce the misclassification cost and the number of high cost errors of a single decision tree under the frequent cost change situation—combining multiple trees in Boosting and Cost-Boosting gives more robust predictions against cost changes. In terms of misclassification cost, Cost-Boosting is a better choice than Boosting. When the aim is to minimize the number of high cost errors, we strongly recommend to use Boosting.

References

Merz, C.J. & P.M. Murphy (1997), *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Michie, D., D.J. Spiegelhalter, & C.C. Taylor (1994), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Limited.

Pazzani, M., C. Merz, P. Murphy, K. Ali, T. Hume, & C. Brunk (1994), Reducing misclassification costs, in *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 217-225, Morgan Kaufmann.

Quinlan, J.R. (1993), *C4.5: Program for machine learning*, Morgan Kaufmann.

Quinlan, J.R. (1996), Bagging, boosting, and C4.5, in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 725-730, AAAI Press.

Schapire, R.E., Y. Freund, P. Bartlett, & W.S. Lee (1997), Boosting the margin: A new explanation for the effectiveness of voting methods, in *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 322-330.

Ting, K.M. & Z. Zheng (1998), Boosting Trees for Cost-Sensitive Classifications, *Working Paper 1/98*, Dept. of Computer Science, University of Waikato.

This article was processed using the LaTeX macro package with LLNCS style