Working Paper Series
ISSN 1170-487X

Boosting Trees for Cost-Sensitive Classifications

by Kai Ming Ting and Zijian Zheng

Working Paper 1/98
January 1998

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

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 classification 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 ordinary boosting combined with the minimum expected cost criterion to select the prediction class is a good solution under this situation. We also introduce a variant of the ordinary boosting procedure which utilizes the cost information during training. We show that the proposed technique performs better than the ordinary boosting in terms of misclassification cost. However, this technique requires to induce a set of new trees every time the cost changes. Our empirical investigation also reveals some interesting behavior of boosting decision trees for cost-sensitive classification.

Keywords: inductive learning, boosting, cost-sensitive classification, decision tree learning, machine learning.

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, the cost committed in diagnosing someone as healthy when one has a life-threatening disease is usually considered to be much higher than another type of error—of diagnosing someone as ill when one is in fact healthy. The former type of error is more serious than the latter type of error. 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 at the leaf.[3] The expected misclassification cost of a class is the expected cost when predicting this class. Its value at a leaf is estimated by computing the cost of classifying all training examples at that leaf to the class. Because no modification to the tree induction process is required, the same tree can be re-used when the misclassification costs change. A variant of C4.5 modified in this manner is used as the base line of this research. It is called C4.5c.

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; Freund & Schapire, 1996). 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

[3] The class with the maximum weight is the same as the majority class if all examples have the same weight.

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. Section 4 discusses some related issues. Section 5 describes related work, and 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 (Freund & Schapire, 1996). 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:[4] 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 three 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 continue the boosting process from step (i).

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

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

[4] Here, we use the decision tree learning algorithm, C4.5, for our classifier learning, but this boosting procedure can be used for any type of classifier learning.

where the normalizing term z_k and α_k are defined as

$$z_k = 2\sqrt{(1-\epsilon_k)\epsilon_k}, \tag{3}$$

$$\alpha_k = \frac{ln((1-\epsilon_k)/\epsilon_k)}{2}. \tag{4}$$

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:

$$EC_j(x) \propto \sum_{i}^{I} \sum_{k}^{K} \alpha_k W_i(t_k(x)) cost(i,j), \tag{5}$$

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 the computation of the expected misclassification cost. 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 the detail in the next section); and the weight adjustment is re-defined as follows.

$$w_{(k+1)}(n) = \frac{w'_{(k+1)}(n)}{\sum_n w'_{(k+1)}(n)}, \tag{6}$$

$$w'_{(k+1)}(n) = \begin{cases} cost(actual(n), predicted(n)), & \text{if } actual(n) \neq predicted(n); \\ Nw_k(n), & \text{otherwise.} \end{cases} \tag{7}$$

Equation (7) replaces an instance's weight with the misclassification cost if it is misclassified; otherwise the current weight (with proportion of the total weight)

is retained. Equation (6) performs normalization to ensure $\sum_n w_{k+1}(n)N = N$. The normalization is important because there is no reason to alter the size of training set, which is equivalent to the sum of all training instance weights, while the individual instance weights are adjusted to reflect the relative importance of instances for making future prediction with respect to cost-sensitive classification.

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 (5) 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. Table 1 gives a description of these domains, including the number of examples, number of classes, number of binary, nominal, and continuous attributes, and default accuracy. 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 row of the matrix indicates the actual class, and the column indicates the predicted class. The off-diagonal entries contain the costs of misclassifications, while the entries on the diagonal contain the cost of correct classifications which are usually equal to zero. For example, the entry on the position (i, j) of the matrix is the cost of classifying a class i example as class j. It is denoted as $cost(i, j)$.

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 for this *unity condition* is to allow us to measure the number of high cost errors, which is one of two important measures in cost-sensitive classifications to be defined later.

One 10-fold cross-validation (Breiman, Friedman, Olshen, and Stone, 1984) 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

Table 1. Description of the domains used in the experiments.

Datasets	# Examples	# Classes	# Attr	Default Accuracy (%)
Echocardiogram	131	2	1B 6C	67.2
Hepatitis	155	2	13B 6C	79.4
Heart	303	2	13C	54.1
Horse	368	2	3B 12N 7C	63.0
Credit	690	2	4B 5N 6C	55.5
Breast-W	699	2	9C	65.5
Diabetes	768	2	8C	65.1
Euthyroid	3163	2	18B 7C	90.7
Hypothyroid	3163	2	18B 7C	95.2
Coding	20000	2	15N	50.0
Lymphography	148	4	9B 9N	54.7
Glass	214	6	9C	35.7
Waveform	300–5000	3	40C	33.3
Soybean	683	19	16B 19N	23.8
Annealing	898	6	19B 13N 6C	76.2
Vowel	990	11	10C	9.5
Splice	3177	3	60N	51.9
Abalone	4177	3	1N 7C	52.7
Nettalk(s)	5438	5	7N	40.1
Satellite	6435	6	36C	23.8

N: nominal; B: binary; C: Continuous

generated cost matrices to simulate the cost changing situation. In each run, the same cost matrix is employed in training, if it is utilized, and testing. All reported results are averaged over 100 runs, except stated otherwise.

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., $\sum_m cost(actual(m), predicted(m))$). A good cost-sensitive classifier should have low total misclassification cost. This is a more meaningful performance measure than the number of misclassification errors for cost-sensitive classification. 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 also have small number of high cost errors.

The following subsection compares C4.5c, Boosting, and Cost-Boosting. Subsection 3.2 studies the effect of K on Boosting and Cost-Boosting for cost-sensitive classification.

3.1 Comparison of C4.5c, Boosting, and Cost-Boosting

The parameter K controlling the number of classifiers generated in both Boosting and Cost-Boosting is set at 10 for the experiments in this subsection. 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.

Tables 2(a) and 2(b) show, respectively, the misclassification costs and the number of high cost errors of C4.5c, Boosting, and Cost-Boosting. The ratios for the pair-wise comparison among the three algorithms are also presented. For example, a ratio of less than 1 for Boosting vs C4.5c 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. Only in 2 domains, does Boosting obtain larger number of high cost errors than C4.5c. In comparison to C4.5c, Cost-Boosting achieves the average reduction of 32%. Only in 1 out of the 20 domains, does Cost-Boosting has larger number of high cost errors than C4.5c. 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.

It is interesting to note that in all two-class domains (the first 10 domains in the table), Boosting has fewer high cost errors than Cost-Boosting. However, in 7 out of the 10 multi-class domains, Boosting has more high cost errors than Cost-Boosting. Only in 3 multi-class domains, does Boosting have slightly fewer high cost errors than Cost-Boosting.

3.2 The Effect of K on Boosting and Cost-Boosting

It has been shown that boosting almost always reduces errors of its base classifier when the number of trials, K, increases (Schapire *et al.*, 1997). In this subsection, we study its effect on Boosting and Cost-Boosting for cost-sensitive classification.

Table 3 presents the results of $K = 10$ and $K = 50$ with Boosting and Cost-Boosting. The results show that, by changing the value of K from 10 to 50, the misclassification cost of Boosting *increases* by 3% on average. This increase

Table 2. Comparison of C4.5c, Boosting and Cost-Boosting

(a) Misclassification costs

Datasets	C4.5c	Boosting vs C4.5c		Cost-Boosting vs C4.5c		Cost-Boosting vs Boosting
	cost	cost	ratio	cost	ratio	ratio
Echocardiogram	7.9	6.5	.82	6.4	.81	.99
Hepatitis	7.5	5.3	.71	5.1	.68	.96
Heart	19.6	12.6	.64	12.2	.62	.97
Horse	17.0	18.4	1.08	15.8	.93	.86
Credit	24.5	28.3	1.15	19.8	.81	.70
Breast-W	12.8	8.4	.66	6.6	.51	.78
Diabetes	39.0	35.7	.92	32.7	.84	.92
Hypothyroid	8.4	11.8	1.42	7.5	.90	.63
Euthyroid	21.2	26.7	1.26	18.8	.89	.70
Coding	1277.9	896.8	.70	853.2	.67	.95
Lymphography	14.9	14.1	.95	13.3	.89	.94
Glass	38.1	25.5	.67	26.3	.69	1.03
Waveform	7330.9	4467.2	.61	4414.9	.60	.99
Soybean	29.6	26.8	.90	21.3	.72	.80
Annealing	30.0	25.8	.86	24.5	.82	.95
Vowel	103.8	58.0	.56	62.3	.60	1.07
Splice	96.7	101.6	1.05	80.8	.83	.79
Abalone	691.8	562.7	.81	533.0	.77	.95
Nettalk(s)	475.7	376.6	.79	356.9	.75	.95
Satellite	466.9	311.2	.67	297.3	.64	.96
Mean			.86		.75	.89

(b) Number of high cost errors.

Datasets	C4.5c	Boosting vs C4.5c		Cost-Boosting vs C4.5c		Cost-Boosting vs Boosting
	#hce	#hce	ratio	#hce	ratio	ratio
Echocardiogram	0.82	0.13	.16	0.26	.32	2.00
Hepatitis	0.93	0.12	.13	0.39	.42	3.25
Heart	2.67	0.35	.13	0.84	.31	2.40
Horse	1.62	0.06	.04	1.20	.74	20.00
Credit	2.88	0.76	.26	1.71	.59	2.25
Breast-W	1.85	0.38	.21	0.67	.36	1.76
Diabetes	3.75	0.82	.22	1.62	.43	1.98
Hypothyroid	1.20	0.57	.47	0.94	.78	1.65
Euthyroid	3.00	1.26	.42	2.19	.73	1.74
Coding	139.68	9.59	.07	18.28	.13	1.91
Lymphography	2.52	2.94	1.17	2.77	1.10	.94
Glass	6.41	5.23	.82	5.39	.84	1.03
Waveform	1189.75	856.64	.72	837.41	.70	.98
Soybean	5.21	5.15	.99	4.66	.89	.90
Annealing	5.52	5.33	.97	5.40	.98	1.01
Vowel	17.27	12.69	.73	13.94	.81	1.10
Splice	15.38	18.28	1.19	14.43	.94	.79
Abalone	118.70	109.78	.92	102.79	.87	.94
Nettalk(s)	82.42	76.24	.93	72.07	.87	.95
Satellite	77.84	66.21	.85	61.69	.79	.93
Mean			.57		.68	2.43

Table 3. Comparison of $K = 10$ and $K = 50$ with Boosting and Cost-Boosting

(a) Misclassification cost.

Datasets	Boosting			Cost-Boosting		
	$K = 10$	$K = 50$		$K = 10$	$K = 50$	
	cost	cost	ratio	cost	cost	ratio
Echocardiogram	6.5	6.5	1.01	6.4	6.4	1.00
Hepatitis	5.3	6.2	1.16	5.1	5.0	.99
Heart	12.6	13.3	1.06	12.2	11.5	.94
Horse	18.4	19.0	1.03	15.8	15.7	.99
Credit	28.3	30.1	1.07	19.8	19.9	1.00
Breast-W	8.4	8.2	.98	6.6	6.2	.94
Diabetes	35.7	37.3	1.05	32.7	32.0	.98
Hypothyroid	11.8	13.9	1.17	7.5	7.4	.98
Euthyroid	26.7	36.1	1.35	18.8	19.3	1.03
Coding	896.8	927.6	1.03	853.2	865.0	1.01
Lymphography	14.1	13.1	.93	13.3	12.3	.93
Glass	25.5	23.5	.92	26.3	25.1	.96
Waveform	4467.2	4225.9	.95	4414.9	4223.7	.96
Soybean	26.8	29.6	1.11	21.3	20.5	.96
Annealing	25.8	26.5	1.03	24.5	24.2	.99
Vowel	58.0	52.0	.90	62.3	60.2	.97
Splice	101.6	105.2	1.04	80.8	75.7	.94
Abalone	562.7	555.4	.99	533.0	519.1	.97
Nettalk(s)	376.6	381.1	1.01	356.9	339.1	.95
Satellite	311.2	282.2	.91	297.3	288.2	.97
Mean			1.03			.97

(a) Number of high cost errors.

Datasets	Boosting			Cost-Boosting		
	$K = 10$	$K = 50$		$K = 10$	$K = 50$	
	#hce	#hce	ratio	#hce	#hce	ratio
Echocardiogram	0.13	0.09	.69	0.26	0.22	.85
Hepatitis	0.12	0.08	.67	0.39	0.33	.85
Heart	0.35	0.14	.40	0.84	0.59	.70
Horse	0.06	0.00	.00	1.20	1.05	.88
Credit	0.76	0.37	.49	1.71	1.65	.96
Breast-W	0.38	0.25	.66	0.67	0.52	.78
Diabetes	0.82	0.40	.49	1.62	1.38	.85
Hypothyroid	0.57	0.54	.95	0.94	0.87	.93
Euthyroid	1.26	0.80	.63	2.19	2.08	.95
Coding	9.59	2.20	.23	18.28	10.86	.59
Lymphography	2.94	2.88	.98	2.77	2.55	.92
Glass	5.23	5.14	.98	5.39	5.31	.99
Waveform	856.64	826.74	.97	837.41	818.02	.98
Soybean	5.15	5.48	1.06	4.66	4.61	.99
Annealing	5.33	5.60	1.05	5.40	5.44	1.01
Vowel	12.69	10.89	.86	13.94	13.53	.97
Splice	18.28	18.95	1.04	14.43	13.59	.94
Abalone	109.78	111.01	1.01	102.79	101.19	.98
Nettalk(s)	76.24	77.85	1.02	72.07	68.85	.96
Satellite	66.21	61.91	.94	61.69	60.60	.98
Mean			.76			.90

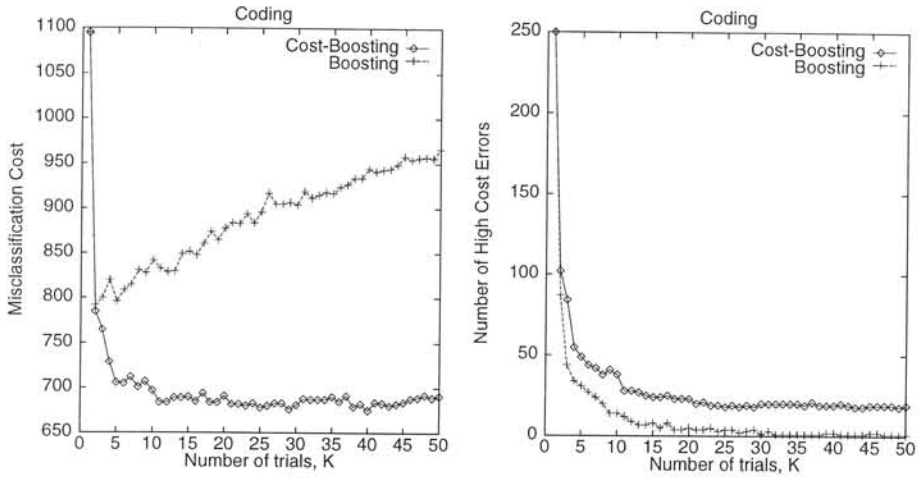

Fig. 1. The effect of K on Boosting and Cost-Boosting in the Coding domain

occurs in 13 out of the 20 domains, and more often in two-class domains (9 out of 10) than the multi-class domains (4 out of 10). In contrast, the misclassification cost of Cost-Boosting decreases by 3% on average. This decrease occurs in 16 out of the 20 domains. In only 2 domains, does the cost increase up to 3%. This suggests that increasing the value of K can improve the performance of Cost-Boosting in terms of the misclassification cost, but it does not help Boosting to reduce misclassification cost.

It is interesting to note that Boosting reduces the number of high cost errors despite the fact that it increases the misclassification costs when the value of K is increased from 10 to 50. Boosting with $K = 50$ has fewer high cost errors in 15 out of 20 domains, with an average of 24% reduction. Note that this amount of reduction is mainly due to the large reductions in 9 out of the 10 two-class domains. In contrast, Cost-Boosting with $K = 50$ only has a modest average reduction of 10% (but it has fewer high cost errors in 19 out of 20 domains).

Comparing Boosting to Cost-Boosting, the former has larger reductions in high cost errors in all two-class domains except the Hypothyroid domain; and has absolute fewer high cost errors in all two-class domains. But, the difference in multi-class domains is not so clear-cut.

3.3 Explanations for performance differences

In terms of misclassification cost, the performance difference in the two-class domains is highlighted in Figure 1, which compares Boosting and Cost-Boosting in the Coding domain, as a function of the number of trials. This experiment is carried out in a single run using $cost(2,1) = 3$ and $cost(1,2) = 1$. In general, Cost-Boosting continues to reduce its misclassification cost as K increases; though the reduction is marginal after K exceeds 10. In sharp contrast, Boosting

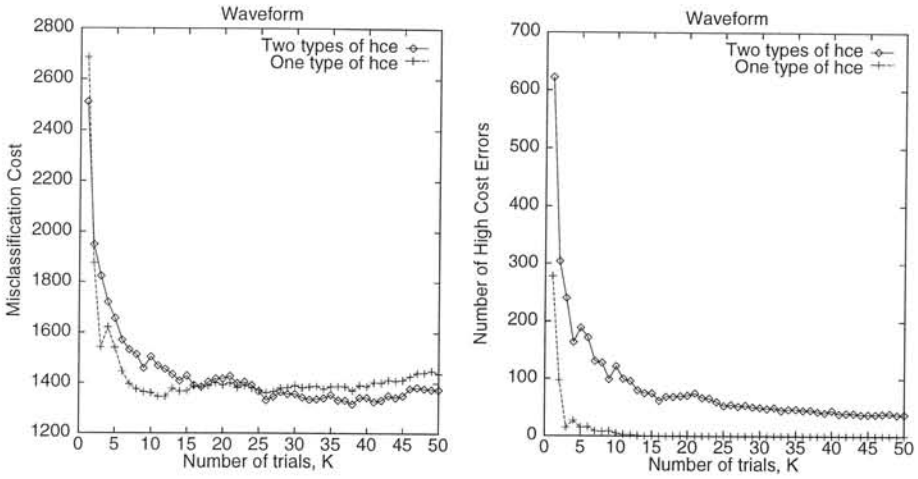

Fig. 2. Boosting with one or two types of high cost error in the Waveform domain

decreases the misclassification cost at the beginning, but clearly increases the cost after K exceeds 2.

Boosting's sharp increase in misclassification cost in some domains, when K increases, is a uncharacteristic behavior in the general framework of boosting (Schapire *et al.*, 1997). This can be explained as follows. While Boosting concentrates to reduce the number of high cost errors, it might inadvertently increase the number of low cost errors. In some cases, the reduction might not off-set the increase. This results an increase in total misclassification cost. This phenomenon is also observed while inducing a single cost-sensitive tree (Ting, 1997) in datasets with highly skewed class distribution.

In most cases, Boosting reduces the number of high cost errors faster than Cost-Boosting as K increases. This suggests that Cost-Boosting's weight adjustment procedure is weaker than that of Boosting in reducing high cost errors.

In terms of the number of high cost errors, the performance difference between two-class domains and multi-class domains can be explained as follows. The cost matrix in two-class domains is simple, i.e., it has only one type of high cost error. Thus, both boosting procedures can concentrate on reducing this type of error. In multi-class domains, there are more than one type of high cost error. The boosting procedures may waver among several types of high cost error, apart from the fact that there are more high cost errors. This explains why the degree of reduction in the number of high cost errors in multi-class domains is much less than that in two-class domains. Figure 2 demonstrates this effect on Boosting in the Waveform domain, a three-class domain. This experiment is carried out in a single run using $cost(2,1) = 5$ for 'one type of hce', and $cost(2,1) = 2$ & $cost(1,3) = 3$ for 'two types of hce', and the rest of the off-diagonal entries in the cost matrix has unit cost. Although both have about the same total misclassification cost, Boosting with 'one type of hce' clearly reduces the number of high cost errors with a faster rate than Boosting with 'two types

of hce'. For example, the former reduces from 278 at $K = 1$ to 5 at $K = 10$, a ratio of 0.018; the later reduces from 622 at $K = 1$ to 122 at $K = 10$, a ratio of 0.196.

4 Discussion

Following Breiman's (1996) recommendation, our experiments in Section 3 allow both Boosting and Cost-Boosting to continue generating trees until the set limit K, even when either condition $\epsilon_k \geq 0.5$ or $\epsilon_k = 0$ is met. Boosting meets the condition $\epsilon_k \geq 0.5$ only before the set limit at $K = 50$ in seven domains (i.e., Echocardiogram, Horse, Diabetes, Soybean, Annealing, Splice and Nettalk(s)). To check whether the recommendation is suitable for cost-sensitive classification, we further conduct an experiment in these domains using a variant of Boosting which stops generating trees when the condition $\epsilon_k \geq 0.5$ is met. In agreement with Breiman's (1996) results on minimum error classification, Boosting performs consistently better, though marginally, than the variant for cost-sensitive classification in all datasets. For example, in the Nettalk(s) domain, continue to generate trees until the set limit at $K = 50$ reduces the misclassification cost from 389.9 to 381.1.

One weakness of Cost-Boosting's weight adjustment procedure is that if the first tree misclassifies with unit cost only, then all the subsequent trees would be exactly the same as the first one since the weights stay the same with equal values (i.e., $cost(actual(n), predicted(n)) = Nw_k(n) = 1$). One simple remedy is to employ Boosting's weight adjustment procedure in the first trial, and switch back to employ Cost-Boosting's own procedure from the second trial onward.

Instead of equal initial weights, one can make use of the misclassification cost information to set the initial weights (Ting, 1997) of Cost-Boosting. Thus, all trees would be cost-sensitive, including the first one. However, this makes little difference in performance, both in terms of the misclassification costs and the number of high cost errors, especially when using high value of K.

While the minimum expected cost criterion can be employed in minimum error learners (e.g., C4.5 and ordinary boosting) to handle the situation where misclassification costs change very often, we show that it is not the best method to minimize misclassification costs. Ting (1997) reports the similar finding with the induction of a single tree for cost-sensitive classification.

As any other methods which employ multiple models, one loses comprehensibility of the induced trees in Boosting and Cost-Boosting.

5 Related Work

Cost-Boosting is inspired by the instance weight adjustment method in decision tree learning for boosting (Quinlan, 1996) and for effective cost-sensitive tree induction (Ting, 1997). The principle of using the minimum expected cost criterion for cost-sensitive classification is described by Michie *et al.* (1994).

There are some research on the induction of a single cost-sensitive tree. Breiman *et al.* (1984), Knoll, Nakhaeizadeh, & Tausend (1994), Pazzani, Merz,

Murphy, Ali, Hume, & Brunk (1994), Webb (1996), and Ting (1997) investigate methods of incorporating variable misclassification costs into the processes of tree generation, tree pruning and tree specialization for cost-sensitive classification. Furthermore, there are a few decision tree learning algorithms that consider the costs of tests, such as EG2 (Núñez, 1991), CS-ID3 (Tan, 1993), and IDX (Norton, 1989). Turney (1995) studies both the misclassification cost and the test cost using a genetic algorithmic search in decision tree induction. All the above systems induce a single cost-sensitive tree, given a cost matrix. None of them has explored methods to handle the situation where misclassification costs change very often.

Recent research in boosting has provided promising results from both theoretical and empirical aspects (Freund & Schapire, 1996; Quinlan, 1996; Schapire *et al.*, 1997). Nevertheless, boosting has not been applied to cost-sensitive classification yet.

Our proposal of using Boosting in situation where costs change very often is reminiscent of the two-tier representation for learning (Michalski, 1987; Kubat, 1996) which suggests explicit separation of induction and classification strategies.

6 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 does not consider misclassification cost during induction of classifiers, and only 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 the ordinary boosting approach, 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 in two-class domains.

Our investigation on the behavior of Boosting reveals that increasing the number of models too much can cause the misclassification cost to rise. Though this is a uncharacteristic behavior in the general framework of boosting, we have provide an explanation to this effect for cost-sensitive classification.

References

Breiman, L. (1996), Bias, Variance, and Arcing Classifiers, *Technical Report 460*, Department of Statistics, University of California, Berkeley, CA.

Breiman, L., J.H. Friedman, R.A. Olshen, & C.J. Stone (1984), *Classification And Regression Trees*, Belmont, CA: Wadsworth.

Freund, Y. & R.E. Schapire (1996), Experiments with a new boosting algorithm, in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148-156, Morgan Kaufmann.

Knoll, U., G. Nakhaeizadeh & B. Tausend (1994), Cost-sensitive pruning of decision trees, in *Proceedings of the Eighth European Conference on Machine Learning*, pp. 383-386. Berlin, Germany: Springer-Verlag.

Kubat, M. (1996), Second Tier for Decision Trees, in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 293-301.

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.

Michalski, R.S. (1987), How to learn imprecise concepts: a method employing a two-tiered Knowledge Representation for learning, in *Proceedings of the Fourth International Workshop on Machine Learning*, pp. 50-58.

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

Norton, S.W. (1989), Generating better decision trees, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 800-805, Morgan Kaufmann.

Núñez, M. (1991), The use of background knowledge in decision tree induction, *Machine Learning, 6*, pp. 231-250.

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. Morgan Kaufmann.

Tan, M. (1993), Cost-sensitive learning of classification knowledge and its applications in robotics, *Machine Learning, 13*, pp. 7-33.

Ting, K.M. (1997), Inducing Cost-Sensitive Trees via Instance-Weighting, *Working Paper 97/22*, Department of Computer Science, University of Waikato.

Turney, P.D. (1995), Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm, *Journal of Artificial Intelligence Research, 2*, pp. 369-409.

Webb, G.I. (1996) Cost-sensitive specialization, in *Proceedings of the 1996 Pacific Rim International Conference on Artificial Intelligence*, pp. 23-34, Springer-Verlag.

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