Working Paper Series
ISSN 1170-487X

Inducing Cost-Sensitive Trees via Instance-Weighting

by Kai Ming Ting

Working Paper 97/22
September 1997

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

Inducing Cost-Sensitive Trees via Instance-Weighting

Kai Ming Ting KAIMING@CS.WAIKATO.AC.NZ
Department of Computer Science, University of Waikato, Hamilton, New Zealand.

Abstract

We introduce an instance-weighting method to induce cost-sensitive trees in this paper. It is a generalization of the standard tree induction process where only the initial instance weights determine the type of tree (i.e., minimum error trees or minimum cost trees) to be induced. We demonstrate that it can be easily adopted to an existing tree learning algorithm.

Previous research gave insufficient evidence to support the fact that the greedy divide-and-conquer algorithm can effectively induce a truly cost-sensitive tree directly from the training data. We provide this empirical evidence in this paper. The algorithm employing the instance-weighting method is found to be comparable to or better than both C4.5 and C5 in terms of total misclassification costs, tree size and the number of high cost errors. The instance-weighting method is also simpler and more effective in implementation than a method based on altered priors.

Keywords: Cost-sensitive trees, instance weights.

1. Introduction

Cost-sensitive classifications have received much less attention than minimum error classifications in empirical learning research. Classifiers that minimize the number of misclassification errors are inadequate in problems with variable misclassification costs. Many practical classification problems have different costs associated with different types of error. For example, in medical diagnosis, the errors committed in diagnosing someone as healthy when one has a life-threatening disease is usually considered to be far more serious (thus higher costs) than the opposite type of error—of diagnosing someone as ill when one is in fact healthy.

A line of research in cost-sensitive tree induction employing the greedy divide-and-conquer algorithm demands further investigation. Breiman *et al.* (1984) describe two different methods of incorporating variable misclassification costs into the process of tree induction. These methods adapt the test selection criterion in the tree growing process. Pazzani *et al.* (1994) reported negative empirical results when using one of the Breiman *et al.*'s formulation to induce cost-sensitive trees. They found that the cost-sensitive trees do not always have lower misclassification costs, when presented with unseen test data, than those trees induced without cost consideration. Knoll, Nakhaeizadeh & Tausend (1994) studied several cost-sensitive pruning methods, but the result is inconclusive. Using a post-processing approach, Webb (1996) shows that applying a cost-sensitive specialization technique to a minimum error tree can reduce its misclassification costs by about 3% on average. Employing the greedy divide-and-conquer algorithm, the research so far does not show convincingly that a truly cost-sensitive tree can/cannot be effectively learned, directly from the training data. We investigate this issue specifically in this paper.

This paper presents the instance-weighting method to induce cost-sensitive trees that minimize misclassification costs. This method is inspired by instance weight modification in boosting decision trees by Quinlan (1996). Boosting generates multiple classifiers in sequential steps. At the end of each step, the weight of each instance in the training set is adjusted to reflect its importance for the next induction step. These weights cause the learner to concentrate on different instances in each step and so lead to different classifiers. These classifiers are then combined by voting to form a composite classifier. Boosting begins with *equal* initial weights in the first step. The intuition for the cost-sensitive induction in this paper is to have *different* initial weights which reflect the (given) costs of misclassification. This effectively influences the learner to focus on instances which have high misclassification costs. We demonstrate that this is a viable method and can be easily adopted to an existing learning algorithm. We show convincingly that a truly cost-sensitive tree can be effectively learned using this method—an algorithm incorporating the instance-weighting method achieves a 29% reduction in misclassification costs over the same algorithm without it. It is also found to be competitive with a recent program C5 (Quinlan, 1997) and better in some aspects.

Section 2 presents the proposed instance-weighting method. In Section 3, we define three different types of cost matrix and conduct a systematic investigation using each one of them. We find that the type of cost matrix affects the behavior of the instance-weighting cost-sensitive tree induction algorithm. Most previous research only employs one of the three types of cost matrix defined in this paper. Section 4 reports our empirical investigation.

Section 5 relates the instance-weighting method to a previous method using altered priors. It is then followed by discussion and conclusions.

2. Cost-Sensitive Tree Induction via Instance-Weighting

Let N be the total number of instances from the given training set, and N_j be the number of class j instances. Similarly, let $N(t)$ and $N_j(t)$ be the number of instances and class j instances in node t of a decision tree. Let $C(j)$ be the cost of misclassifying a class j instance. $C(j)$ is computed according to the type of cost matrix which will be described in the next section.

In the case of unit costs, assume the weight of each training instance is unity. The intuition is to modify the weight of an instance proportional to the cost of misclassifying the class the instance belonged, and the sum of all training instance weights is still equal to N. The last condition is important because there is no reason to alter the size of the 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. Thus, the weight of a class j instance can be computed as

$$w(j) = C(j)\frac{N}{\sum_i C(i)N_i}, \tag{1}$$

such that the sum of all instance weights is $\sum_j w(j)N_j = N$.

The probability that an instance is in class j given that it falls into node t is given by

$$p(j|t) = \frac{w(j)N_j(t)}{\sum_i w(i)N_i(t)}. \tag{2}$$

The greedy divide-and-conquer procedure for inducing minimum error trees (e.g., Breiman *et al.* (1984) and Quinlan (1993)) can then be used without modification, except that Equation (2) is used instead of $N_j(t)/\sum_i N_i(t)$ in the computation of the test selection criterion. This can be done by modifying an existing algorithm, e.g., C4.5 (Quinlan, 1993) accordingly; where the test selection criterion is to choose the test which has the maximum gain in entropy (i.e., $-\sum_j p(j|t)log[p(j|t)]$) in node t.

An alternative and coherent way is to initialize the weight of each instance according to Equation (1) and perform a *summation* of class j instance weights in node t, i.e., $W_j(t)$, (instead of *counting* the number of instances, $N_j(t)$) to compute $p(j|t)$. This is given by

$$p(j|t) = \frac{W_j(t)}{\sum_i W_i(t)}. \tag{3}$$

We modified C4.5 to create C4.5CS using Equations (1) and (3). Since Equation (3) is already in place in C4.5,[1] we only need to initialize the training instance weights to $w(j)$.

1. The fractional weights are used in C4.5 for the treatment of missing values. See Quinlan (1993) for details.

The advantage of the second approach is that the whole process of tree growing and tree pruning is the same as that used to induce minimum error trees. This can be viewed as a generalization of the standard tree induction process where *only the initial instance weights determine the type of tree (i.e., minimum error trees or minimum cost trees) to be induced.*

Figure 1 shows an example of a split on the same attribute test using unit instance weights (in the left figure) and different instance weights (in the right figure). The sum of the instance weights for each class are shown in each node. With unit weights, each sum is equivalent to the number of instances for each class, $N_j(t)$. This example has two equiprobable classes, where $N_1 = N_2 = 50$ at the root of the tree. The right figure shows the result of the same split when $C(1) = 3$ and $C(2) = 1$. Employing Equation (1), the weights of all instances are modified to $w(1) = 1.5$ and $w(2) = 0.5$. As a result, the sums of the class j instance weights at the root are $W_1 = 75$ and $W_2 = 25$. This example shows that initialize the instance weights to $w(j)$ amounts to changing the class distribution of the training data.

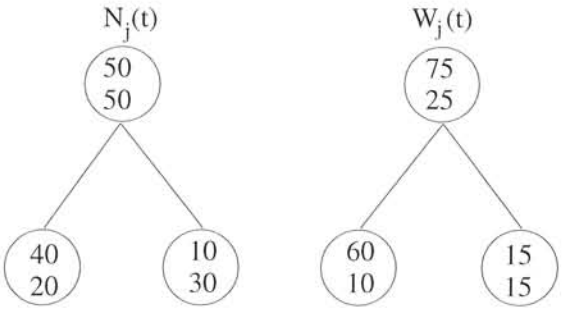

Figure 1: Splitting on the same test—using unit instance weights (left) and different instance weights (right).

To classify a new instance, C4.5CS predicts the class which has the maximum weights at a leaf, as in C4.5.

3. Types of Cost Matrix

In a classification task of K classes, the misclassification costs can be specified in a cost matrix of size $K \times K$. The row of the matrix indicates the predicted class, and the column indicates the actual class. The off-diagonal entries contain the costs of misclassifications; and on the diagonal lie the costs for correct classifications which are zero in this case, since our main concern here is total misclassification costs of an induced tree.

A cost matrix must be converted to a cost vector $C(j)$ in order to use Equation (1) for instance-weighting. Here we define three types of cost matrix, which give rise to different definitions of $C(j)$. Let $cost(i, j)$ be the cost of misclassifying an instance belonging to class j as belonging to class i. In all cases, $cost(i, j) = 0.0$, for $i = j$. The three types of cost matrix are defined as

Table 1: Examples of three types of cost matrix, $cost(i, j)$.

Type (a)

		j = 1	2	3
i	1	0.0	3.0	6.0
	2	1.0	0.0	1.0
	3	1.0	1.0	0.0

Type (b)

		j = 1	2	3
	1	0.0	3.0	7.0
	2	1.0	0.0	7.0
	3	1.0	3.0	0.0

Type (c)

		j = 1	2	3
	1	0.0	3.0	2.0
	2	1.0	0.0	7.0
	3	10.0	5.0	0.0

(a) $cost(i, j) > H$ only for a single value of $i = I$; and $cost(i \neq I, j) = H$ for all $i \neq j$. We define

$$C(j) = cost(I, j) \text{ for } j \neq I, \text{ and } C(I) = H.$$

(b) This type of cost matrix has a constant misclassification cost for each class, i.e., $cost(i, j) = H_j \geq 1.0$ for each $j \neq i$. Thus, it can be expressed as a cost vector

$$C(j) = H_j.$$

(c) In this more general case, $cost(i, j) \geq 1.0$ for all $i \neq j$. A possible form of $C(j)$ (Breiman *et al.*, 1984) is

$$C(j) = \sum_i cost(i, j).$$

Examples of these matrices are shown in Table 1.

In our experiments, without loss of generality, we impose the following unity condition. For type (a) cost matrix, $H = 1.0$; for type (b) cost matrix, at least one $H_j = 1.0$; and for type (c) cost matrix, at least one $cost(i, j) = 1.0$. The only reason to have this unity condition or normalization[2] is to allow us to measure the number of *high cost errors*, which is defined as the number of misclassification errors that have costs more than 1.0. This is another important measure for cost-sensitive classification; where the aim of cost-sensitive classification is to minimize the number of high cost errors or misclassification costs, or both. Note that in two-class datasets under this condition, all three types will have similar cost matrices and $C(j)$, i.e., one of the two nonzero entries in the cost matrix or one of the two $C(j)$'s is a unit cost.

In the following experiments, a cost matrix is randomly generated at the beginning of each trial. Each entry more than unity in the cost matrix is assigned an integer randomly generated between 2 to 10. Class I in type (a) cost matrix is also randomly selected.

4. Experiments

Four measures are used to evaluate the performance of the cost-sensitive tree induction algorithms. They are total misclassification costs (i.e., $\sum_l^{N'} cost(\text{predicted-class}(l), \text{actual-class}(l))$,

2. Note that an arbitrary cost matrix can be normalized to become one of the three types of cost matrix under this unity condition.

Table 2: Details of the datasets used in the experiment.

Datasets	# Instances	# Classes	# Attr			Default Accuracy %
Echocardiogram	131	2	1B		6C	67.2
Hepatitis	155	2	13B		6C	79.4
Heart(Statlog)	270	2			13C	55.6
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
GermanCredit	1000	2			24C	70.0
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.

where N' is the number of instances in the unseen test set), tree size (i.e., total number of internal nodes and leaves), the number of high cost errors, and the total number of misclassification errors on unseen data. The first and the third are the most important measures in cost-sensitive classifications. A good cost-sensitive classifier will have as low as possible in these two measures. Every thing being equal, a tree induction algorithm is better than the other if it induces smaller trees. An average of multiple experimental trials for each measure is reported.

We conduct experiments using twenty datasets obtained from the UCI repository of machine learning databases (Merz & Murphy, 1996) and two datasets with specified cost matrices (i.e., Heart(Statlog) and GermanCredit) used in the Statlog project (Michie, Spiegelhalter & Taylor, 1994). They consist of twelve two-class datasets and ten multi-class datasets. The details of these datasets are given in Table 2.

Random cost assignments, as described in Section 3, are used in all datasets except the Heart(Statlog) and GermanCredit datasets. In the later cases, the costs (i.e., $cost(1,2) = 1.0$ and $cost(2,1) = 5.0$) specified in Michie, Spiegelhalter & Taylor (1994) are used.

Ten 10-fold cross-validations are carried out in each dataset, except in the Waveform dataset where randomly generated training data size of 300 and test data size of 5000 are used in the 100 trials.

In Section 4.1, we compare C4.5CS with C4.5 to evaluate whether trees induced by C4.5CS are more cost sensitive than those produced by C4.5. Note that *the only difference between C4.5CS and C4.5 is the initial weight setting.* Any performance differences are due to this initial weight setting. In section 4.2, we compare C4.5CS with a recent improved version of C4.5, i.e., C5.

4.1 Can C4.5CS effectively induce cost-sensitive trees?

We begin our investigation first using type (a) cost matrix, then using the other two types in the multi-class datasets in the following two subsections. In Section 4.1.3, we evaluate C4.5CS with and without the minimum expected cost criterion.

4.1.1 Using type (a) cost matrix

Given a training set and a cost matrix, C4.5CS induces a cost-sensitive tree which minimizes the total misclassification costs. C4.5 produces a tree which minimizes the total misclassification errors. Both trees are then tested using a separate test set, and the total misclassification costs are measured according to the given cost matrix.

Tables 3 and 4 present averages, over 100 trials, for the misclassification costs, the tree size, the number of high cost errors and the total errors for both C4.5CS and C4.5 in each dataset. The ratio (C4.5CS/C4.5) for each of these measures is also presented—a value less than 1 represents an improvement due to C4.5CS. The bottom rows in both tables present the means of these ratios for C4.5CS against C4.5 over all datasets.

In terms of misclassification costs, C4.5CS achieves a 29% reduction as compared to C4.5 on average. Note that for half of all datasets, C4.5CS reduces *ge* 29% in costs. On a treatment by treatment basis, the effect of cost reduction varies from 1% for the Vowel dataset to 58% for the GermanCredit dataset. This constitutes all datasets except two. The Soybean and Hypothyroid datasets are the only exceptions in which C4.5CS has 16% and 3% increase in costs.

In terms of tree size, C4.5CS produces trees 31% smaller than those produced by C4.5 on average. In thirteen out of the twenty-two datasets, C4.5CS produces smaller trees than this average. The effect of smaller trees varied from 5% for the Annealing dataset to 99% for the GermanCredit dataset. This constitutes all but two datasets. C4.5CS produces trees which are 101% and 68% larger than those produced by C4.5 in the Hypothyroid and Euthyroid datasets respectively. This is because the two datasets have very skew class distribution (i.e., 90.7% and 95.2% of the total instances belong to one of the two classes in these two datasets). A high cost $C(j)$ assigned to the class which has small number of instances effectively reduces the class distribution skewness. This leads to larger trees as a result.

C4.5CS has 33% more errors than C4.5 on average, but makes 64% fewer high cost errors than C4.5. The reduction in the number of high cost errors varies from 14% for the Hypothyroid dataset to 99% for the GermanCredit dataset. Note that C4.5CS is making fewer high cost errors in all datasets. On the other hand, C4.5CS makes more errors than C4.5 in all datasets.

Table 3: C4.5CS versus C4.5 (Misclassification Cost and Tree Size).

Datasets	Misclassification Cost			Tree Size		
	C4.5CS	C4.5	ratio	C4.5CS	C4.5	ratio
Echocardiogram	7.5	14.1	.53	6.4	10.8	.59
Hepatitis	5.4	12.0	.45	11.0	17.0	.65
Heart(Statlog)	10.9	17.1	.64	16.7	35.6	.47
Heart	13.3	23.2	.57	19.5	39.5	.49
Horse	16.2	20.2	.80	8.1	11.6	.70
Credit	21.1	37.3	.57	9.6	33.2	.29
Breast-W	8.6	10.9	.79	15.5	23.8	.65
Diabetes	34.4	69.8	.49	18.6	41.9	.44
GermanCredit	30.3	72.8	.42	2.2	149.3	.01
Hypothyroid	8.5	8.2	1.03	24.5	12.2	2.01
Euthyroid	21.0	23.4	.90	42.5	25.3	1.68
Coding	930.4	2062.4	.45	412.8	2805.6	.15
Lymphography	5.5	7.7	.71	15.0	27.4	.55
Glass	9.1	12.7	.72	33.5	45.5	.74
Waveform	2646.2	4108.8	.64	30.4	51.0	.60
Soybean	8.0	6.9	1.16	84.9	96.4	.88
Annealing	9.8	11.3	.87	72.7	76.6	.95
Vowel	28.7	28.9	.99	151.7	187.0	.81
Splice	46.1	47.7	.97	123.4	171.6	.72
Abalone	200.7	430.9	.47	292.3	579.2	.50
Nettalk(s)	148.1	205.9	.72	1353.7	2061.6	.66
Satellite	115.5	152.4	.76	390.1	561.2	.70
Mean			.71			.69

Table 4: C4.5CS versus C4.5 (No. High Cost Errors and No. Errors).

Datasets	No. High Cost Errors			No. Errors		
	C4.5CS	C4.5	ratio	C4.5CS	C4.5	ratio
Echocardiogram	0.59	2.06	.29	5.2	4.7	1.11
Hepatitis	0.54	1.78	.30	3.6	3.4	1.06
Heart(Statlog)	1.02	2.79	.37	6.8	5.9	1.15
Heart	1.21	3.35	.36	8.6	6.6	1.30
Horse	1.32	2.90	.46	11.8	5.8	2.03
Credit	1.63	5.33	.31	13.2	10.2	1.29
Breast-W	0.90	1.58	.57	4.7	3.6	1.31
Diabetes	2.19	10.20	.21	26.8	19.8	1.35
GermanCredit	0.10	11.43	.01	29.9	27.1	1.10
Hypothyroid	1.03	1.20	.86	3.6	2.3	1.57
Euthyroid	2.52	3.39	.74	9.2	6.3	1.46
Coding	31.16	284.53	.11	871.5	554.8	1.57
Lymphography	0.28	0.91	.31	4.3	3.2	1.34
Glass	0.31	1.09	.28	7.7	7.1	1.08
Waveform	227.48	514.56	.44	1655.0	1528.1	1.08
Soybean	0.14	0.23	.61	7.4	5.6	1.32
Annealing	0.17	0.99	.17	9.4	6.7	1.40
Vowel	0.60	1.76	.34	26.4	20.3	1.30
Splice	2.86	6.16	.46	35.0	18.6	1.88
Abalone	6.01	54.21	.11	180.7	161.9	1.12
Nettalk(s)	4.70	21.37	.22	129.2	94.4	1.37
Satellite	4.81	13.43	.36	96.3	87.7	1.10
Mean			.36			1.33

Table 5: Mean ratios for C4.5CS against C4.5 over ten multi-class datasets.

	Multi-class			Two-class*
Cost matrix type	(a)	(b)	(c)	
Misclassification Cost ratio	.80	.87	.98	.64
Tree Size ratio	.71	.70	.85	.66
No. High Cost Errors ratio	.33	.90	1.02	.38
No. Errors ratio	1.30	1.30	1.06	1.36

*: average over twelve two-class datasets.

Table 6: Mean ratios for C4.5CS against C4.5CS_mc over ten multi-class datasets.

	Multi-class			Two-class*
Cost matrix type	(a)	(b)	(c)	
Misclassification Cost ratio	.99	.99	1.06	.95
Tree Size ratio	1.00	1.00	1.00	1.00
No. High Cost Errors ratio	1.73	.99	1.02	1.60
No. Errors ratio	.93	.93	.88	.82

4.1.2 Multi-Class Problems

In this section, we study the behavior of C4.5CS using all three types of cost matrix in the ten multi-class datasets.

Tables 5 presents the mean ratios for C4.5CS against C4.5 using the three types of cost matrix defined in Section 3. Results for the two-class datasets from Section 4.1.1 are also summarized in the last column of this table for ease of reference.

In comparison to C4.5, C4.5CS performs better in terms of misclassification costs, tree size and the number of high cost errors for all three types of cost matrix, except that it is making 2% more high cost errors when using type (c) cost matrix. Among these cost matrices, C4.5 and its cost-sensitive counterpart seem to be performing most similarly using type (c) cost matrix, and least similarly using type (a) cost matrix. This is evident since the mean ratios for misclassification costs and the number of high cost errors increase from type (a) to type (b) and to type (c) cost matrices. This suggests that the conversion from $cost(i, j)$ to $C(j)$ is most effective for type (a) cost matrix but least effective for type (c) cost matrix.

4.1.3 The minimum expected cost criterion

In this section, we investigate a variant, C4.5CS_mc, which makes use of the minimum expected cost criterion rather than the maximum weight criterion in selecting a prediction class during classification. The expected misclassification cost for class j is given by

$$\begin{aligned} EC_j(x) &= \sum_i p(i|t(x))cost(i,j) \\ &\propto \sum_i W_i(t(x))cost(i,j), \end{aligned} \tag{4}$$

where $t(x)$ is the leaf of the tree where instance x falls into; $p(i|t(x))$ and $W_i(t(x))$ are the class i probability and instance weight sum as defined in Equation (3).

Table 6 presents the mean ratios for C4.5CS against C4.5CS_mc. The effect of using the minimum expected cost criterion seems to be different for two-class and multi-class datasets. This criterion generally performs better for multi-class datasets (it decreases misclassification costs by 6% for type (c) cost matrix and maintains about the same for the other two types of cost matrix; and it decreases the number of high cost errors by 73% for type (a) cost matrix and maintains about the same for type (b) and type (c) cost matrices). But, for two-class datasets, this criterion increases misclassification costs by 5% though it decreases the number of high cost errors by 60%.[3]

Thus, it is advisable to employ C4.5CS_mc for multi-class datasets, and also for two-class datasets if the aim is to minimize the number of high cost errors.

It is interesting to note that the minimum expected cost criterion has a minor impact on misclassification costs (vary from –6% to +5%) in comparison to the impact on the number of high cost errors (vary from –73% to +1%).

SUMMARY

We summarize the findings as follows.

- C4.5CS performs comparably to or better than C4.5 in terms of misclassification costs, tree size and the number of high cost errors.
- The minimum expected cost criterion minimizes the number of high cost errors but not necessary the misclassification costs.

4.2 How does C4.5CS compare to C5?

In this experiment, we compare C4.5CS to the improved version of C4.5, i.e., C5, which can produce a cost-sensitive tree, given a cost matrix. We first present the detail results using type (a) cost matrix, then the summary results using all three types of cost matrix

Tables 7 and 8 present the results in the same format as in Tables 3 and 4. The results show that C4.5CS performs comparably to C5 in terms of misclassification costs. The mean ratio is .99 over twenty-two datasets. In only seven datasets are the ratios for misclassification costs outside the interval 1.00±.10.

In terms of tree size, C4.5CS produces smaller trees than those produced by C5 in twenty out of the twenty-two datasets. On average, C4.5CS induces trees which are 14% smaller. It is interesting to note that the two exceptions, the Annealing and Nettalk(s) datasets, are due to a simpler C5 representation for nominal attribute tests next to the leaves of the tree. In C4.5, when a decision node next to the leaves is a test using a nominal attribute, the number of leaves is the same as the number of possible values for this attribute. This usually results in many leaves with no covered instance; especially when the nominal attribute has many values. C5 simplifies the representation by collecting all zero-instance leaves into a

3. There is another way of applying the minimum expected cost criterion. In this case, one uses $EC_j(x) \propto \sum_i N_i(t(x))cost(i,j)$ instead of Equation (4). This implementation gives a modest reduction of 5% in high cost errors in comparison to C4.5CS; and the misclassification costs stay the same.

Table 7: C4.5CS versus C5 (Misclassification Cost and Tree Size).

Datasets	Misclassification Cost			Tree Size		
	C4.5CS	C5	ratio	C4.5CS	C5	ratio
Echocardiogram	7.5	7.3	1.04	6.4	6.8	.94
Hepatitis	5.4	5.5	.98	11.0	12.7	.87
Heart(Statlog)	10.9	11.6	.94	16.7	20.9	.80
Heart	13.3	14.7	.90	19.5	24.5	.80
Horse	16.2	16.1	1.00	8.1	9.7	.84
Credit	21.1	20.9	1.01	9.6	11.6	.83
Breast-W	8.6	9.4	.91	15.5	18.2	.85
Diabetes	34.4	34.7	.99	18.6	22.5	.83
GermanCredit	30.3	30.4	1.00	2.2	2.5	.88
Hypothyroid	8.5	8.7	.98	24.5	27.0	.91
Euthyroid	21.0	20.3	1.04	42.5	47.5	.89
Coding	930.4	927.0	1.00	412.8	607.5	.68
Lymphography	5.5	5.6	.97	15.0	20.8	.72
Glass	9.1	9.7	.94	33.5	46.0	.73
Waveform	2646.2	3234.6	.82	30.4	44.9	.68
Soybean	8.0	6.7	1.20	84.9	93.0	.91
Annealing	9.8	9.4	1.05	72.7	67.3	1.08
Vowel	28.7	27.9	1.03	151.7	185.4	.82
Splice	46.1	41.3	1.12	123.4	187.4	.66
Abalone	200.7	229.5	.87	292.3	470.7	.62
Nettalk(s)	148.1	129.5	1.14	1353.7	747.8	1.81
Satellite	115.5	133.0	.87	390.1	554.0	.70
Mean			.99			.86

Table 8: C4.5CS versus C5 (No. High Cost Errors and No. Errors).

Datasets	No. High Cost Errors			No. Errors		
	C4.5CS	C5	ratio	C4.5CS	C5	ratio
Echocardiogram	0.59	0.54	1.09	5.2	5.2	1.00
Hepatitis	0.54	0.54	1.00	3.6	3.7	.97
Heart(Statlog)	1.02	1.23	.83	6.8	6.6	1.03
Heart	1.21	1.48	.82	8.6	8.6	1.00
Horse	1.32	1.16	1.14	11.8	12.5	.94
Credit	1.63	1.66	.98	13.2	13.0	1.02
Breast-W	0.90	1.04	.87	4.7	4.8	.98
Diabetes	2.19	2.34	.94	26.8	26.4	1.02
GermanCredit	0.10	0.14	.71	29.9	29.9	1.00
Hypothyroid	1.03	1.01	1.02	3.6	3.7	.97
Euthyroid	2.52	2.51	1.00	9.2	8.8	1.05
Coding	31.16	34.30	.91	871.5	857.0	1.02
Lymphography	0.28	0.40	.70	4.3	4.1	1.05
Glass	0.31	0.52	.60	7.7	7.0	1.10
Waveform	227.48	361.34	.63	1655.0	1534.3	1.08
Soybean	0.14	0.08	1.75	7.4	6.3	1.17
Annealing	0.17	0.19	.89	9.4	8.5	1.11
Vowel	0.60	1.33	.45	26.4	21.5	1.23
Splice	2.86	4.10	.70	35.0	23.3	1.50
Abalone	6.01	13.57	.44	180.7	175.3	1.03
Nettalk(s)	4.70	5.30	.89	129.2	106.6	1.21
Satellite	4.81	9.74	.49	96.3	88.7	1.09
Mean			.86			1.07

Table 9: Mean ratios for C4.5CS_mc against C5.

	Multi-class			Two-class	
Cost matrix type	(a)	(b)	(c)		$\star$
Misclassification Cost ratio	1.00	1.05	1.04	1.05	.98
Tree Size ratio	.74	.74	.86	.85	.85
No. High Cost Errors ratio	.35	1.06	1.03	.67	.94
No. Errors ratio	1.28	1.16	1.05	1.29	1.00

$\star$: Mean ratios for C4.5CS against C5.

single leaf with a subset branch. When we simplify the C4.5CS representation as in C5, the tree sizes reduce to 55.5 for the Annealing dataset and 566.2 for the Nettalk(s) dataset, which are both smaller than the trees produced by C5. Taking these new figures into account, the mean ratio for tree size reduces to .80 from .86.

Although trees produced by C4.5CS have more misclassification errors (i.e., 7% on average) than those by C5, C4.5CS is less likely to make high cost errors. This feature is particularly important for most cost-sensitive classifications, which aim to minimize the number of high cost errors and are more relaxed on low cost errors. C4.5CS makes 14% fewer high cost errors than C5 on average. This effect is apparent in sixteen out of the twenty-two datasets.

We employ C4.5CS_mc in the following experiment using all three types of cost matrix. Tables 9 presents mean ratios for C4.5CS_mc against C5 using the three types of cost matrix in the multi-class datasets and in the two-class datasets. Results for the two-class datasets from Tables 7 and 8 are also summarized in the last column of this table for ease of reference.

In multi-class datasets, the results show that C4.5CS_mc is comparable to or marginally worse than C5 in terms of misclassification costs, but induces significantly smaller trees when using all three types of cost matrix. C4.5CS_mc has significantly fewer high cost errors when using type (a) cost matrix, but marginally more high cost errors when using the other two types of cost matrix.

In two-class datasets, employing C4.5CS_mc instead of C4.5CS reduces the mean ratio for the number of high cost errors from .94 to .67 but increases the mean ratio for misclassification costs from .98 to 1.05. Tree size maintains at the same ratio.

Summary

We summarize the findings of comparison with C5 as follows.

- In the two-class datasets or using type (a) cost matrix in the multi-class datasets, C4.5CS_mc is clearly a better choice than C5 in terms of the number of high cost errors and tree size.
- Using type (b) and type (c) cost matrices in the multi-class datasets, C4.5CS_mc is marginally worse than C5 in terms of misclassification costs and the number of high cost errors, but is significantly better in terms of tree size.

Table 10: Average misclassification costs for five classifiers

Dataset	C4.5CS	C5	CART	Discrim	NaiveBayes
Heart(Statlog)	0.404	0.430	†0.452 (8)	†0.393 (2)	†0.374 (1)
GermanCredit	0.303	0.304	0.613 (8)	0.535 (1)	0.703 (12)

†: the 9-fold cross-validation is used, and others use 10-fold.
(n): ranking in the Statlog project; n=1 is the best result.

4.3 Comparison with the results from the Statlog Project

The Statlog project (Michie, Spiegelhalter & Taylor, 1994) studied a number of classifiers for cost-sensitive classification in the Heart and GermanCredit datasets. We re-state their results of three classifiers for comparison. The classifiers are CART (Breiman *et al.*, 1984), linear discriminant (Discrim) and NaiveBayes. The Statlog project uses average misclassification costs (i.e., the ratio of the total misclassification costs and the total number of test instances) for evaluation. We convert our results for C4.5CS and C5 to this measure, and the results for the five classifiers are tabulated in Table 10.

The results show that C4.5CS performs better than CART in both datasets. It performs marginally worse than Discrim and NaiveBayes in the Heart dataset, but significantly better than both classifiers in the GermanCredit dataset.

5. Relation to Altered Priors

Breiman *et al.* (1984) discuss a method of incorporating variable misclassification costs via altered priors for cost-sensitive tree induction. Let priors, $\pi(j) = N_j/N$, and $C(j)$ as defined in Section 3, then the altered priors are given by

$$\pi'(j) = \frac{C(j)\pi(j)}{\sum_i C(i)\pi(i)} = \frac{C(j)N_j}{\sum_i C(i)N_i}.$$

In the instance-weighting method, every instance is weighted proportional to $C(j)$. The weight of a class j instance is computed as

$$w(j) = \frac{C(j)N}{\sum_i C(i)N_i} = \pi'(j)N/N_j = \frac{\pi'(j)}{\pi(j)}.$$

Thus, the instance weight is a ratio of the altered prior and the original prior. Both methods share the same idea of changing the class distribution according to the given misclassification costs, but one implementation is simpler and more effective than the other. In fact, our formulae for $p(j|t)$ in Equation (2) is equivalent to that used in CART (Breiman *et al.*, 1984) which employs altered priors. Implementation using Equation (2) or by merely modifying instance weights will produce the same tree at the end of the tree growing process. But, the former would require an equivalent modification in the pruning process; otherwise, it will perform poorly. This is demonstrated by modifying C4.5 according to Equation (2), to yield C4.5(π'). Because instance weights are not altered, the tree induced by C4.5(π') will be pruned according to unit instance weights.

Table 11: Mean ratios for C4.5CS against C4.5(π') over twenty-two datasets.

	Type (a) cost matrix
Misclassification Cost ratio	.72
Tree Size ratio	1.10
No. High Cost Errors ratio	.38
No. Errors ratio	1.26

Table 11 reports the mean ratios (C4.5CS/C4.5(π')) for the four measures in an experiment using type (a) cost matrix. These figures are averaged over twenty-two datasets. C4.5(π') is significantly worse than C4.5CS for the two important measures in cost-sensitive classifications (i.e., misclassification costs and the number of high cost errors). The poor result of C4.5(π') is due to the inconsistent use of instance weights from the tree growing process to the tree pruning process.

6. Discussion

The results presented in Tables 3, 4 and 5 show clearly that the greedy divide-and-conquer algorithm can effectively induce a cost-sensitive tree directly from the training set, given a cost matrix. A previous reported method—cost-sensitive specialization (Webb, 1996), which also applied to C4.5, only achieves a cost reduction of 3% on average over twenty-two datasets. This method maintains the same size of the tree produced by C4.5. In comparison, the instance-weighting method achieves a 29% reduction in costs, and produces trees which are 31% smaller.

The idea of using fractional instance weights in decision tree induction has been applied in C4.5 (Quinlan, 1993). In the case of missing value, C4.5 partitions the training set using fractional weights in evaluating a test during tree induction and classification. Quinlan (1996) further develops the idea to boosting decision tree algorithms. In both cases, *equal* initial weights are used. We extend this idea to *different* initial weights, which we demonstrate to be a viable and simple approach to extend the tree induction algorithm to include cost-sensitive classifications.

Pazzani *et al.* (1994) show that trees induced using the Gini criterion (i.e., the test selection criterion as used in CART) with altered priors perform worse than those induced using only the Gini criterion in terms of misclassification costs in both two-class and multi-class datasets. Pazzani *et al.* (1994) attribute the negative result to the conversion of cost matrix $cost(i, j)$ to cost vector $C(j)$. The results presented in Table 5 show that this is only partly true for type (c) cost matrix in the multi-class datasets. Using type (a) and type (b) cost matrices or two-class datasets, our results indicate that a tree induced with altered priors, together with the corresponding modification in the pruning method, should perform better than those without. Also, we think the difference is not due to the test selection criterion employed (C4.5 uses the entropy criterion rather than the Gini criterion).

However, the conversion does result a weakness in the instance-weighting method. This method will induce the same tree for different cost matrices whenever these matrices convert to the same cost vector $C(j)$. This is undesirable, especially in multi-class

problems using type (c) cost matrix. While other forms of conversion are possible, e.g., $C(j) = max_i\ cost(i, j)$, they too suffer from the same problem. One possible remedy is to conduct tree pruning which uses $cost(i, j)$ directly. However, we have not found an effective method for this kind of pruning so far.

A simple method to use a minimum error tree for cost-sensitive classifications is to employ the minimum expected cost criterion in selecting a predicted class during classification (Michie, Spiegelhalter & Taylor, 1994). We have conducted a comparison between C4.5CS_mc and C4.5_mc, both using the minimum expected cost criterion, and obtained similar results to those in Table 5; despite the fact that the mean ratios for misclassification costs are higher in this case. For example, in two-class datasets, the mean ratio for misclassification costs for C4.5CS_mc against C4.5_mc is .86 as compared to .64 (for C4.5CS/C4.5). Both results show that it is better to induce a cost-sensitive tree than a minimum error tree for cost-sensitive classifications.

Unfortunately, no description is given regarding the cost-sensitive induction method employed in C5 (Quinlan, 1997), and we have no access to source code, except those used for classifying from an induced tree which are held in public domain. The later reveals that C5 predicts a class which has the minimum expected costs; whereas C4.5CS, similar to C4.5, predicts a class which has the maximum weights. Our experiment in Section 4.1.3 indicates that the minimum expected cost criterion used in C5 has indeed an advantage over C4.5CS in terms of making fewer high cost errors; yet, C5 still makes more high cost errors than C4.5CS when using type (a) cost matrix. The significantly smaller trees, produced by C4.5CS using all three types of cost matrix, are also another indication that the cost-sensitive induction method used in C5 is different from that in C4.5CS.

There are several tree induction algorithms that consider the costs of tests, such as EG2 (Núñez, 1991), CS-ID3 (Tan, 1993), IDX (Norton, 1989). Turney (1995) investigates both the costs of tests and misclassifications using a genetic algorithmic search in tree induction. We do not consider costs of tests in this paper to avoid complication of the issue under investigation.

7. Conclusions

We introduce an instance-weighting method to induce cost-sensitive trees, and demonstrate that it is a viable approach and simple to implement or adopt to an existing learning algorithm. It is a generalization of the standard tree induction process to include both minimum error trees and minimum cost trees. This work refutes an earlier negative result with regard to cost-sensitive tree induction employing the greedy divide-and-conquer algorithm. Our empirical results show convincingly that the greedy divide-and-conquer algorithm can effectively induce a truly cost-sensitive tree directly from the training data.

The algorithm employing the instance-weighting method is found to be comparable to or better than both C4.5 and C5 in terms of total misclassification costs, tree size and the number of high cost errors. This is despite the fact that the current instance-weighting method has a weakness that requires the conversion of cost matrix to cost vector. The instance-weighting method is also simpler and more effective in implementation than the method based on altered priors.

Acknowledgment

The author is grateful to the New Zealand Marsden Fund for financial support for this research.

References

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., Nakhaeizadeh, G., & Tausend, B. (1994), Cost-Sensitive Pruning of Decision Trees, in *Proceedings of the Eighth European Conference on Machine Learning*, pp. 383-386. Berlin, Germany: Springer-Verlag.

Merz, C.J. & Murphy, P.M. (1996), *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.

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), Boosting, Bagging, and C4.5, in *Proceedings of the 13th National Conference on Artificial Intelligence*, pp. 725-730, AAAI Press.

Quinlan, J.R. (1997), *C5*, [http://rulequest.com].

Tan, M. (1993), Cost-Sensitive Learning of Classification Knowledge and its applications in robotics, *Machine Learning, 13*, pp. 7-33.

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.