Inducing Cost-Sensitive Trees via Instance Weighting

Kai Ming Ting

School of Computing and Mathematics, Deakin University, Vic 3168, Australia.

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 to be induced—*minimum error* trees or *minimum high cost error* trees. We demonstrate that it can be easily adapted 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 incorporating the instance-weighting method is found to be better than the original algorithm in terms of total misclassification costs, the number of high cost errors and tree size in two-class datasets. The instance-weighting method is simpler and more effective in implementation than a previous method based on altered priors.

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. 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 convinc-

ingly 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 seeks to minimize the number of high cost errors and total 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 adapted 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 substantial reduction in misclassification costs, the number of high cost errors and tree size over the same algorithm without it in two-class domains.

The proposed instance-weighting method changes the class distribution such that the tree so induced is in favor of the class with high weight/cost and is less likely to commit errors with high cost. This usually reduces the total misclassification costs as a consequence. Smaller trees are a natural product of the tree induction procedure when presented with training dataset of skewed class distribution, which is a result of weighting instances in dataset with relatively balanced class distribution. We present the proposed instance-weighting method in the next section.

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. The probability that an instance is in class j given that it falls into node t is given by the ratio of the total number of class j instances to the total number of instances in this node.

$$p(j|t) = \frac{N_j(t)}{\sum_i N_i(t)}. \tag{1}$$

When node t contains instances that belong to a mixture of classes, the standard greedy divide-and-conquer procedure for inducing trees (e.g., Breiman *et al.* (1984) and Quinlan (1993)) uses a test selection criterion to choose a test at this node such that the training instances which fall into each branch, as a result of the split, become more homogeneous. One of the commonly used criterion is entropy i.e., $-\sum_j p(j|t)log[p(j|t)]$. At each node, the tree growing process selects

a test which has the maximum gain in entropy until the node contains only a single-class collection of instances.

To avoid overfitting, a pruning process is employed to reduce the size of the tree such that the estimated error is minimum. In short, the standard tree induction procedure seeks to produce a *minimum error* tree.

Our intuition for cost-sensitive tree induction is to modify the weight of an instance proportional to the cost of misclassifying the class to which the instance belonged, leaving the sum of all training instance weights 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.

Let $C(j)$ be the cost of misclassifying a class j instance, then the weight of a class j instance can be computed as

$$w(j) = C(j)\frac{N}{\sum_i C(i)N_i}, \tag{2}$$

such that the sum of all instance weights is $\sum_j w(j)N_j = N$. For $C(j) \geq 1$, $w(j)$ has the smallest value, $0 < \frac{N}{\sum_i C(i)N_i} < 1$, when $C(j) = 1$; and the largest value, $w(j) = \frac{C(j)\sum_i N_i}{\sum_i C(i)N_i} > 1$, when $C(j) = max_i C(i)$.

Similar to $p(j|t)$, $p_w(j|t)$ is defined as the ratio of the total weight of class j instances to the total weight in node t.

$$p_w(j|t) = \frac{W_j(t)}{\sum_i W_i(t)} = \frac{w(j)N_j(t)}{\sum_i w(i)N_i(t)}. \tag{3}$$

The standard greedy divide-and-conquer procedure for inducing minimum error trees can then be used without modification, except that $W_j(t)$ is used instead of $N_j(t)$ in the computation of the test selection criterion in the tree growing process and the error estimation in the pruning process. Thus, both processes are affected due to this change.

We modified C4.5 (Quinlan, 1993) to create C4.5CS. We only need to initialize the training instance weights to $w(j)$ since C4.5 has already employed $W_j(t)$ for the computation discussed above.[1]

This modification effectively converts the standard tree induction procedure that seeks to minimize *the number of errors*, regardless of cost, to a procedure that seeks to minimize *the number of errors with high weight/cost.* Note that minimizing the later does not guarantee that the total misclassification cost is minimized. This is because the number of low cost errors is usually increased as a result.

The advantage of this 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

[1] C4.5 uses fractional weights for the treatment of missing values. See Quinlan (1993) for details.

be viewed as a generalization of the standard tree induction process where *only the initial instance weights determine the type of tree to be induced—minimum error trees or minimum high cost error trees.*

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

Cost Matrix and C(j). In a classification task of I classes, the misclassification costs can be specified in a cost matrix of size $I \times I$. 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.[2]

Let $cost(i, j)$ be the cost of misclassifying a class j instance as belonging to class i. In all cases, $cost(i, j) = 0.0$, for $i = j$. A cost matrix must be converted to a cost vector $C(j)$ in order to use Equation (2) for instance-weighting. In this paper, we employ the form of conversion suggested by Breiman *et al.* (1984):

$$C(j) = \sum_{i}^{I} cost(i, j). \tag{4}$$

In our experiments, without loss of generality, we impose a unity condition—at least one $cost(i, j) = 1.0$. The only reason to have this unity condition or normalization[3] 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.

Note that the cost matrix to cost vector conversion is expected to work well with the cost-sensitive tree induction, as described in this section, when there are only two classes. But it might be inappropriate when there are more than two classes because it collapses $I \times I$ numbers to I. In order to investigate the potential problem due to this conversion, we explicitly divide the experimental datasets into two groups: two-class and multi-class. Any performance discrepancy between these two groups is due to this conversion.

3 Experiments

Four measures are used to evaluate the performance of the cost-sensitive tree induction algorithm in this paper. They are total misclassification costs (i.e., $\sum_{l}^{N'} cost(\text{predicted-class}(l), \text{actual-class}(l))$, where N' is the number of instances in the unseen test set), pruned 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. The aim of cost-sensitive classification is to minimize the number of high cost errors or misclassification costs, or both. Every thing being equal, a tree induction algorithm is better than the other if it induces smaller trees.

[2] In general, the costs of correct classifications can be non-zero. Minimizing the costs of correct classifications is a different issue outside the scope of this paper.

[3] Note that an arbitrary cost matrix can be normalized to become a cost matrix satisfying this unity condition.

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_S and German) used in the Statlog project (Michie, Spiegelhalter & Taylor, 1994). The datasets are selected to cover a wide variety of different domains with respect to dataset size, the number of classes, the number of attributes, and types of attributes. They consist of twelve two-class datasets and ten multi-class datasets.

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.

Random cost assignments with the unity condition are used in all datasets except the Heart_S and German 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. In the former cases, a cost matrix is randomly generated at the beginning of each trial. Each non-diagonal entry in the cost matrix is assigned an integer randomly generated between 1 to 10.

We first 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.

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

Given a training set and a cost matrix, C4.5CS induces a cost-sensitive tree which seeks to minimize the number of high cost errors and total misclassification costs. C4.5 produces a tree which seeks to minimize 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.

Table 1 presents 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 means of these ratios are given for the twelve two-class datasets as well as the ten multi-class datasets.

In terms of misclassification costs, C4.5CS achieves a mean reduction of 38% as compared to C4.5 in two-class datasets; but a mean reduction of only 2% in multi-class datasets.

In terms of tree size, C4.5CS produces trees 34% smaller than those produced by C4.5 in two-class datasets; and only 15% smaller in multi-class datasets. In only two datasets (Hypothyroid and Euthyroid), C4.5CS produces trees which are larger than those produced by C4.5. This is because the two datasets have very skewed class distribution (i.e., 95.2% and 90.7% of the total instances belong to one of the two classes in these two datasets, respectively). 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. Although the costs are randomly assigned without reference to the original class distribution,

Table 1. C4.5CS versus C4.5 in terms of misclassification cost, tree size, number of high cost errors and total number of errors.

Datasets	Cost			Tree Size			No. HC Errors			No. Errors*	
	C4.5CS	C4.5	ratio	C4.5CS	C4.5	ratio	C4.5CS	C4.5	ratio	C4.5CS	ratio
Echo	7.9	18.2	.44	6.0	10.8	.56	0.5	2.5	.22	5.4	1.15
Hepa	6.0	12.3	.48	9.5	17.0	.56	0.4	1.6	.27	3.9	1.15
Heart_S	10.9	17.1	.64	16.7	35.6	.47	1.0	2.8	.37	6.8	1.15
Heart	14.2	25.2	.56	18.2	39.5	.46	1.1	3.3	.34	8.7	1.32
Horse	16.4	21.8	.75	8.6	11.6	.74	1.1	2.9	.39	11.6	2.00
Credit	21.3	36.5	.58	10.5	33.2	.32	1.5	4.9	.31	13.7	1.34
Breast	9.9	14.4	.69	14.8	23.8	.62	1.0	2.0	.51	4.9	1.36
Diab.	35.8	69.6	.51	18.4	41.9	.44	2.0	9.4	.21	27.5	1.39
German	30.3	72.8	.42	2.2	149.3	.01	0.1	11.4	.01	29.9	1.10
Hypo	9.0	8.7	1.03	24.6	12.2	2.02	0.9	1.1	.81	4.0	1.74
Euthyr	20.7	24.2	.85	39.8	25.3	1.57	2.1	3.3	.64	10.1	1.60
Coding	942.2	2058.6	.46	302.7	2805.6	.11	22.2	278.2	.08	880.8	1.59
Mean			.62			.66			.35		1.41
Lympho	17.4	18.2	.96	19.0	27.4	.69	2.9	2.9	1.00	3.3	1.03
Glass	35.8	38.4	.93	39.2	45.5	.86	6.0	6.3	.95	6.9	.97
Wave	7119.0	7709.3	.92	42.7	51.0	.84	1177.9	1232.4	.96	1522.1	1.00
Soybean	33.8	30.4	1.11	91.1	96.4	.95	5.7	5.2	1.11	6.2	1.11
Anneal	35.4	35.7	.99	76.7	76.6	1.00	6.4	6.0	1.06	7.3	1.09
Vowel	115.1	111.8	1.03	175.2	187.0	.94	19.1	18.2	1.05	21.1	1.04
Splice	95.7	95.6	1.00	157.1	171.6	.92	16.2	15.3	1.06	22.7	1.22
Abalone	708.7	799.2	.89	402.1	579.2	.69	124.6	129.2	.96	168.4	1.04
Net(s)	507.9	514.2	.99	1650.4	2061.6	.80	86.0	84.2	1.02	99.6	1.06
Sat.	475.3	478.5	.99	472.4	561.2	.84	78.4	78.5	1.00	88.1	1.00
Mean			.98			.85			1.02		1.06

* the column on C4.5 is omitted because of lack of space.

reduction in skewness seems to have a larger effect than increase in skewness in these two datasets.

C4.5CS makes 65% fewer high cost errors than C4.5 in two-class datasets; but 2% more high cost errors in multi-class datasets. On the other hand, C4.5CS has 41% more errors than C4.5 in two-class datasets, but only 6% more errors in multi-class datasets.

Hypothyroid is the only two-class dataset in which C4.5CS has higher misclassification costs (by 3%) than C4.5. While C4.5CS is able to reduce the number of high cost errors by 19% in this highly skewed class distribution dataset, but the 74% increase in total errors outweighs this reduction which results a net increase in total misclassification costs.

3.2 Minimum expected cost criterion

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). It is interesting to find out how the proposed method compared to this simple method.

The expected misclassification cost for predicting class i with respect to the example x is given by:

$$EC_i(x) \propto \sum_j W_j(t(x))cost(i,j), \tag{5}$$

where $t(x)$ is the leaf of the tree where instance x falls into, and $W_j(t)$ is the total weight of class j training instances in node t.

To classify a new example x using a minimum error tree and the minimum expected cost criterion, $EC_i(x)$ is computed for every class. The example x is assigned to class i with the smallest value for $EC_i(x)$. That is, $EC_i(x) < EC_{i'}(x)$ for all $i' \neq i$.

A comparison between C4.5CS_mc and C4.5_mc, both using the minimum expected cost criterion, is conducted here. The results in Table 2 show that it is still better to induce a cost-sensitive tree than a minimum error tree for cost-sensitive classifications in two-class datasets, even using the minimum expected cost criterion.

Table 2. Mean ratios for C4.5CS_mc against C4.5_mc.

	Two-class	Multi-class
Misclassification Cost ratio	.86	.99
Tree Size ratio	.66	.85
No. High Cost Errors ratio	.36	1.02
No. Errors ratio	1.57	1.07

3.3 Summary

We summarize the findings so far as follows.

- In terms of misclassification costs and the number of high cost errors, C4.5CS performs better than C4.5 in two-class datasets but only comparably in multi-class datasets.
- The relative poor performance of C4.5CS in multi-class datasets is due to the cost matrix to cost vector conversion.
- C4.5CS always makes fewer high cost errors than C4.5 in two-class datasets; but in datasets with highly skewed class distribution, C4.5CS might have higher total misclassification costs than C4.5.
- C4.5CS produces smaller trees than C4.5 because instance weighting effectively increases the skewness of the otherwise more balanced class distribution.
- Even using the minimum expected cost criterion, it is better to induce a cost-sensitive tree than to induce a minimum error tree for cost-sensitive classifications in two-class datasets.

4 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 Equation (4), then the altered priors are given by (Breiman *et al.*, 1984)

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

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. Implementation using altered priors 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 accordingly 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.

The mean ratios (C4.5CS/C4.5(π')) for misclassification cost and the number of high cost errors are .70 and .43 respectively. These figures are averaged over the twelve two-class datasets in which C4.5CS performs well. C4.5(π') is significantly worse than C4.5CS for the two important measures in cost-sensitive classifications. 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.

5 Conclusions

We have introduced an instance-weighting method to induce cost-sensitive trees, and demonstrated that it is a viable approach and simple to implement or adapt to an existing learning algorithm. It is a generalization of the standard tree induction process to include both minimum error trees and minimum high cost error trees. The instance-weighting method is simpler and more effective in implementation than a previous method based on altered priors.

Our empirical results show convincingly that the greedy divide-and-conquer procedure can effectively induce a truly cost-sensitive tree directly from the training data. This work refutes an earlier negative result (Pazzani *et al.*, 1994) with regard to cost-sensitive tree induction employing the greedy divide-and-conquer procedure in two-class datasets.

The algorithm incorporating the instance-weighting method is found to be better than the original algorithm in two-class datasets in terms of the number of

high cost errors, total misclassification costs and tree size. The instance weighting which changes the class distribution directly contributes to this improved performance.

The current instance-weighting method has two weaknesses: (1) it requires the conversion of cost matrix to cost vector which hampers its performance in multi-class datasets, and (2) it might not perform well in terms of total misclassification costs in datasets with highly skewed class distribution. We have suggested a cost-sensitive version of boosting (Schapire *et al.*, 1997) to address these weaknesses, reported in Ting & Zheng (1998).

6 References

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

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, Springer-Verlag.

Merz, C.J. & Murphy, P.M. (1996), *UCI Repository of machine learning databases* [http://www.ics.uci. edu/mlearn/MLRepository.html]. University of California, Dept. 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), Boosting, Bagging, 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.

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.

Ting, K.M. & Z. Zheng (1998), Boosting Trees for Cost-Sensitive Classifications, *Proceedings of the Tenth European Conference on Machine Learning*, Berlin: Springer-Verlag, pp. 190-195.