Cost-Sensitive Decision Trees with Multiple Cost Scales

Zhenxing Qin, Shichao Zhang, and Chengqi Zhang

Faculty of Information Technology, University of Technology, Sydney,
PO Box 123, Broadway, Sydney, NSW 2007, Australia
{zqin, zhangsc, chengqi}@it.uts.edu.au

Abstract. How to minimize misclassification errors has been the main focus of Inductive learning techniques, such as CART and C4.5. However, misclassification error is not the only error in classification problem. Recently, researchers have begun to consider both test and misclassification costs. Previous works assume the test cost and the misclassification cost must be defined on the same cost scale. However, sometimes we may meet difficulty to define the multiple costs on the same cost scale. In this paper, we address the problem by building a cost-sensitive decision tree by involving two kinds of cost scales, that minimizes the one kind of cost and control the other in a given specific budget. Our work will be useful for many diagnostic tasks involving target cost minimization and resource consumption for obtaining missing information.

1 Introduction

Inductive learning techniques have met great success in building models that assign testing cases to classes (Mitchell 1997, Quinlan 1993). How to minimize misclassification errors has been the main focus of Inductive learning techniques, such as CART (Breiman, Friedman, Olshen and Stone 1984) and C4.5 (Quinlan 1993). However, misclassification error is not the only error in classification problem. Numbers of different types of classification errors are listed in (Turney 2000), and the costs of different types of errors are often very different.

More recently, researchers have begun to consider both test and misclassification costs: (Turney 1995, Greiner, Grove and Roth 2002). The objective is to minimize the expected total cost of tests and misclassifications.

Ling, Yang, Wang and Zhang (2004) proposed a new method for building and testing decision trees that minimizes the sum of the misclassification cost and the test cost. It assumes a static cost structure where the cost is not a function of time or cases. It also assumes the test cost and the misclassification cost have been defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis.

But in practice application, Cost may be measured in very different units. Sometimes we may meet difficulty to define the multiple costs on the same cost scale. It is not only a technology issue, but also a social issue. In medical diagnosis, how much money you should assign for a misclassification cost? Sometimes, a misclassification may hurt a patient's life. And from a point of view from social issue, life is invaluable. So we need to involve both of the two cost scales.

G.I. Webb and Xinghuo Yu (Eds.): AI 2004, LNAI 3339, pp. 380–390, 2004.

On the other hand, a static cost structure may not enough to handle multiple cost scales. In real world, when involving at least two performance metrics, it is not realistic to expect to minimize both of them always. At that time, a trade-off is needed.

For example, a diagnosis cost may include two kinds of costs in monetary units (test fee - dollars) and temporal units (test time - seconds). For each individual user, it may pay more attention to a specific cost scale. A millionaire prefers a minimal diagnosis mistake (it means minimal misclassification cost), and he would like to pay much more money for more detail tests. But someone else can accept a tolerant misclassification cost by controlling the diagnosis fee in a specific budget (such as the insurance cover limit).

In this paper, we address the problems above by building a cost-sensitive decision tree by involving two kinds of cost scales, which minimizes the one kind of cost and control the other in a given specific budget.

The rest of the paper is organized as follows. In Section 2, we first review the related works. In section 3, we simply introduce the tree-building algorithm based on single cost scale, and discuss the new issues and properties as involving resource control on decision tree. After that, we consider several testing strategies and analyze their relative merits in section 4. Finally, we present our experimental results in section 5 and conclude the work with a discussion of future work in Section 6.

2 Previous Works

More recently, researchers have begun to consider both test and misclassification costs: (Turney 1995, Greiner, Grove and Roth 2002). The objective is to minimize the expected total cost of tests and misclassifications. In Turney's survey article (Turney 2000), a whole variety of costs in machine learning are analyzed, the first two types of costs are the misclassification costs that are the costs incurred by misclassification errors and test costs these are the costs incurred for obtaining attribute values.

In (Zubek, Dietterich, 2002), the cost-sensitive learning problem is cast as a Markov Decision Process (MDP), and an optimal solution is given as a search in a state space for optimal policies. For a given new case, depending on the values obtained so far, the optimal policy can suggest a best action to perform in order to both minimize the misclassification and the test costs.

Similar in the interest in constructing an optimal learner, Greiner, Grove and Roth (2002) studied the theoretical aspects of active learning with test costs using a PAC learning framework. Turney (1995) presented a system called ICET, which uses a genetic algorithm to build a decision tree to minimize the cost of tests and misclassification.

Ling, Yang, Wang and Zhang (2004) proposed a new method for building and testing decision trees that minimizes the sum of the misclassification cost and the test cost. We simply introduce It assumes a static cost structure where the cost is not a function of time or cases. It also assumes the test cost and the misclassification cost have been defined on the same cost scale, such as the dollar cost incurred in a medical

diagnosis. We will simply introduce the tree building based on single cost scale as following:

To minimize the total target cost, at each leaf, the algorithm labels the leaf as either positive or negative (in a binary decision case) by minimizing the target misclassification cost. Let us look at a concrete example in (Ling, Yang, Wang and Zhang 2004). Assume that during the tree building process, there is a set of P and N positive and negative examples respectively to be further classified by possibly building a sub-tree. If we assume that $P \times FN > N \times FP$, then if no sub-tree is built, the set would be labeled as positive, and thus, the total target misclassification cost is

$$T = N \times FP$$

Suppose that an attribute A with a test cost C1 is considered for a potential splitting attribute. Assume that A has two values, and there are $P1$ and $N1$ positive and negative examples with the first value, $P2$ and $N2$ positive and negative examples with the second value, and $P0$ and $N0$ positive and negative examples with A's value unknown. Then the total test cost would be

$$(P1+N1+P2+N2) \times C1$$

(i.e., cases with unknown attribute values do not incur test costs). Assume that the first branch will be labeled as positive (as $P1 \times FN1 > N1 \times FP1$), and the second branch will be labeled as negative, then the total misclassification cost of the two branches would be

$$N1 \times FP1 + P2 \times FN1$$

As we have discussed earlier in this section, examples with the unknown value of A stay with the attribute A, and we have assumed that the original set of examples is labeled as positive. Thus, the misclassification cost of the unknowns is $N0 \times FP$. The total cost of choosing A as a splitting attribute would be:

$$T_A = (P1+N1+P2+N2) \times C1 + N1 \times FP1 + P2 \times FN1 + N0 \times FP1$$

If $T_A < T$, *where* $T = N \times FP1$, then splitting on A would reduce the total cost of the original set, and we will choose such an attribute with the minimal total cost as a splitting attribute. We will then apply this process recursively on examples falling into branches of this attribute. If $T_A \geq T$ for all remaining attributes, then no further sub-tree will be built, and the set would become a leaf, with a positive label. Table 1 is a concrete example Ecoli dataset and figure 1 is the corresponding decision tree.

Table 1. Test and misclassification costs set for Ecoli dataset

A1	A2	A3	A4	A5	A6	FP/FN
50	50	50	50	50	20	800/800

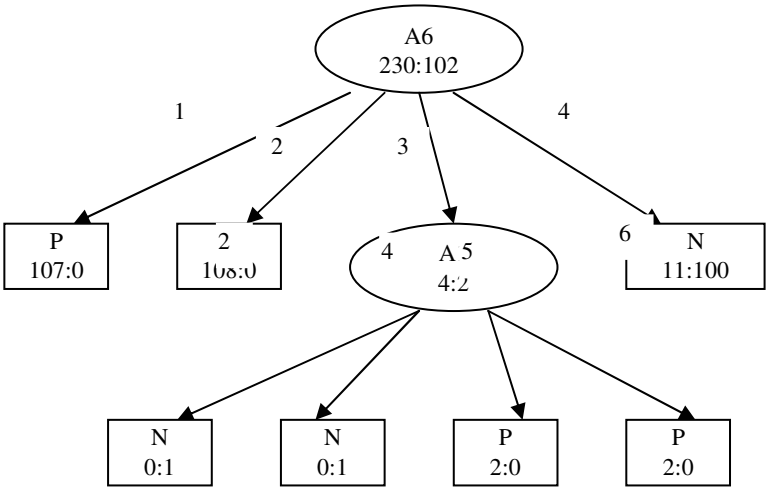

Fig. 1. A decision tree built from the Ecoli dataset (costs are set as in Table 2)

3 Building Decision Tree with Minimal Costs Under Resource Constrains

The goal of our decision-tree learning algorithm is to minimize the sum of target cost on misclassification and test, at the same time, resource cost must less than the resource budget.

We assume test and the misclassification cost contain two kinds of cost – target and resource. Both of the target and resource have been defined on two different cost scales relatively, such as dollar cost and time cost incurred in a medical diagnosis. We assume there is a maximum limit on resource, called resource budget.

Table 2 shows a sample of two cost scales on "Ecoli" dataset. From table 2, we can see that, there are two kinds of costs, cost1 is the target cost, and cost2 is the resource consumption. For example, FP1 = 800 is the target misclassification cost and FP2 = 150 is the resource misclassification cost of false positive.

Table 2. Test and misclassification costs set for "Ecoli" dataset

	FP	FN	A1	A2	A3	A4	A5	A6
Target	800	800	50	60	60	50	50	30
Resource	150	100	10	20	10	10	10	10

3.1 Tree Building Based on Target-First Strategy

There are at least two strategies can be used to involving resource cost in the cost-sensitive decision tree building. The first one is called target-first strategy. It comes from the point of view social issue: target cost is invaluable. It exactly ignore the resource issue and attempts to minimize the total target cost on misclassification and test cost, so we will follow the same building procedure like (Ling, Yang, Wang and Zhang 2004). Exactly the tree-building algorithm is a special case of our target-first strategy when we set all the resource consumption as zero.

As the tree totally ignore resource cost in tree building phase. It means we may pay 100 resource cost to decrease 110 target cost rather than paying 50 resource cost to decrease 100 target. It considers the resource at testing phase. Given a test example, we explore the tree and perform all need test. Once resource budget is exhausted, we stop performing any test and give a result.

3.2 Tree Building Based on Performance-First Strategy

This strategy is exactly the idea of trade-off between target and resource. It uses the target gain ratio to choose potential splitting attributes. Follow the example above, the total target cost of choosing A as a splitting attribute is:

$$T_A = (P1+N1+P2+N2)\times C1 + N1\times FP1 + P2\times FN1 + N0\times FP1$$

We also can calculate the resource consumption of A is

$$C_A = (P1+N1+P2+N2)\times C2 + N1\times FP2 + P2\times FN2 + N0\times FP2$$

If $T_A < T$, *where* $T = N\times FP1$, then the target cost gain is T- T_A, the gain ratio of choosing A as a splitting attribute is

$$R_A = (T- T_A)/C_A$$

Since performance-first strategy involves resource cost during decision tree building, so it is expected to explore deeper along the tree with limited resource. At the same time, we may not have enough resource to perform a test during exploring the tree in testing phase. It means we may stop at an internal node and give a result at once. So the potential result of this internal node should be reserved.

Definition 1: For a internal decision tree node, *potential label* is its class label if the node is labeled as a leaf, and relative target and resource misclassification cost is called *potential leaf cost.*

In testing phase, it uses the similar exploring method as in target-first strategy. Since it involves resource cost during tree building, so test with highest target/resource performance will be perform fist. It is to explore deeper along the tree with limited resource. An example of target-resource cost decision tree is shown as in Figure 2. It is extended from the single scale tree in figure 1.

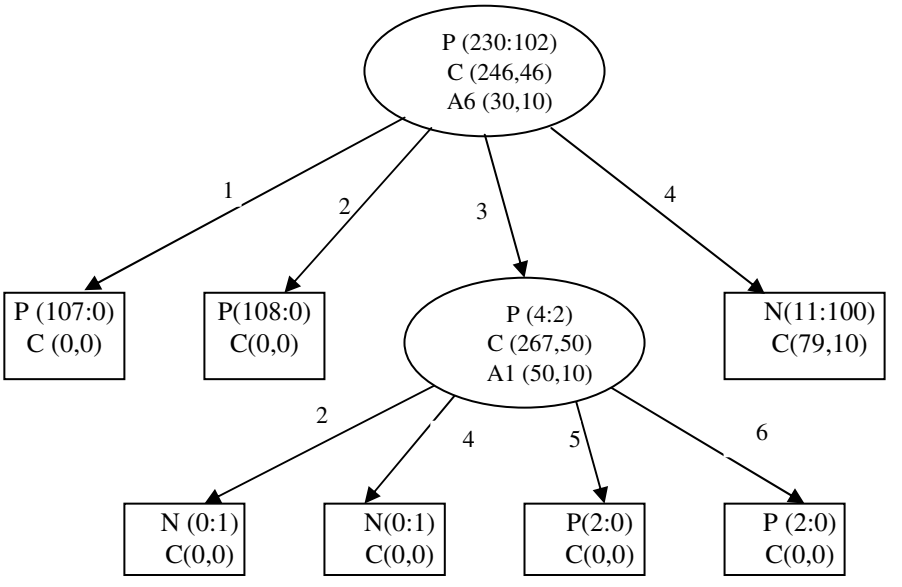

Fig. 2. A decision tree built from the Ecoli dataset (costs are set as in Table 2)

From the Figure 2, we can see that, in each leaf node, we record the class label P or N to represent positive and negative, and the training example data distribution, such as (107,0) at the left child of root node means there is 107 positive and 0 negative examples, finally C(0,0) means 0 target minimal cost and 0 resource cost. In each internal node, we also record the *potential label* with training example distribution and relative cost consumption, such as P (230:102) in root node, and the splitting attribute with test costs, such as A6 (30,10) in root node.

3.3 Resource Control Issues

At the same time of to minimizing the total target cost, we must control the resource consumption less then the specific budget, noted by B. Once resource is exhausted, we will stop exploring further sub-tree and output a leaf according to the target cost. The first issue is how to deal with the cases just going though the threshold B? Firstly, we introduce two concepts first: *confirmed node* and *proposed node.*

Definition 2: Given a test example S and a resource budget B, exploring the decision tree from root node, when we reach an internal node N with the total resource consumption $R(N) \leq B$, attribute value in node N is known but no more resource performing test for the value, then node N is called *proposed node,* and the parent of *N is called confirmed node.*

We stop exploring the decision tree once resource budget can not support further explore, and give users a result based on *confirmed node* and *proposed node.* The

former tells users current best decision with resource B, the later tells users the resource needed for further test.

The second issue, how we get minimal target cost with limited resource budget? It exactly comes from single scale tree. Originally, decision tree was built to minimize the misclassification cost based on the statistics information of splitting attributes. Exploring further branches means smaller subset and better class prediction. But further exploring means more tests, also mean more test cost, so minimizing the sum of target cost is also a trade-off problem. We expect our performance-first strategy can provide a best overall performance since the test with best performance was chosen in each branch of decision tree.

4 Performing Tests on Testing Examples with Resource Control

In this section, we discuss some new issues in testing strategies as involving the resource controlling on the cost-minimal decision tree. Our aim is to predict the class of the testing examples with many missing values with the minimal total target cost, and control resource cost in a specific budget. We also use the same test case in (Ling, Yang, Wang and Zhang 2004) to illustrate our test strategies.

Table 3. An example testing case with several unknown values. The true values are in parenthesis and can be obtained by performing the tests (with costs list in Table 2)

A1	A2	A3	A4	A5	A6	Class
? (6)	2	? (1)	2	2	? (3)	P

Cost-minimal decision tree in (Ling, Yang, Wang and Zhang 2004) shows an amazing performance in dealing with testing examples with many missing values. And in order to predict the class of the testing examples with the minimal total cost for this case, four testing strategies were studied. We will briefly introduce them as following, noted as M1 to M4. When meet an unknown value in test example:

The strategy M1, called Optimal Sequential Test (OST), performs extra tests on the unknown values. It uses the tree built with the minimal cost to decide what tests must be performed in sequence.

The strategy M2 stops right there, and uses the ratio of positive and negative examples in that (internal) node to predict the testing example (recall that these ratios are calculated based on training cases which also have unknown values at this node).

The third strategy M3 uses the C4.5's strategy in dealing with missing values by choosing a value according the probabilities of the attribute's all values. Instead of stopping at the node whose attributes value is unknown in the testing case, this strategy will "split" the testing case into fractions according to the training examples, and go down all branches simultaneously.

The fourth and final strategy M4 ignores the attributes with unknown values and uses rest attributes to build a new tree for the test sample.

We can see that M1 performs extra tests for unknown values. So M1 strategy is only for the case with enough resource. Once resource is exhausted, we can choose one of other three strategies to give a result. Strategies M2 & M3 avoid performing tests and predict the testing example with statistics information in nodes. M4 ignores the attributes with unknown values and building a new tree, but it still need to consider the resource consumption as in the original tree. All those three strategies have not any test costs but they may meet the problem of no enough resource as reaching leaf node.

For instance, we test the example of table 3 in decision tree in Figure 2. Assuming we got resource budget B= 10, so we can perform the test in root node, got A6 = 3. Then the example goes down to the 3rd branch of root node, additional resource cost 10 is needed but no enough resource to perform a test for the attribute A1. What should we do now? First, the node is marked as proposed node (proposed to be labeled as Negative with shortage of resource 10). Then we go back to its parent node (root node here) and output it as confirmed node with class label P. We will conduct experiments to compare the three tree building strategies in next section.

5 Experiments

We conducted experiments on five real-world datasets (Ling, Yang, Wang and Zhang 2004, Blake and Merz 1998) and compared the target-first and performance-first tree building strategies against C4.5. These datasets are chosen because they have at least some discrete attributes, binary class, and a good number of examples. The numerical attributes in datasets are discretized first using minimal entropy method (Fayyad and Irani 1993) as our algorithm can currently only deal with discrete attributes. The datasets are listed in Table 4.

Table 4. Datasets used in the experiments

	No. of attributes	No. of examples	Class distribution (P/N)
Ecoli	6	332	230/102
Breast	9	683	444/239
Heart	8	161	98/163
Thyroid	24	2000	1762/238
Austrilia	15	653	296/357

First, we compare the target cost and resource consumption of target-first and performance-first tree building strategies against C4.5 with OST (we assume our resource budget can only support 50 percent of all tests) on all five dataset. The re-

sults of target cost and resource consumption are shown as in figure 3 and 4 relatively.

From Figure 3, we can see that performance-first tree strategy outperform the other two in target cost. It means performance-first strategy got a better overall performance with limit resource budget. And in Figure 4, we can see that performance-first strategy also consumes less resource than other two strategies. It means performance-first strategy got a better overall performance, which can get a lower target cost with less resource consumption.

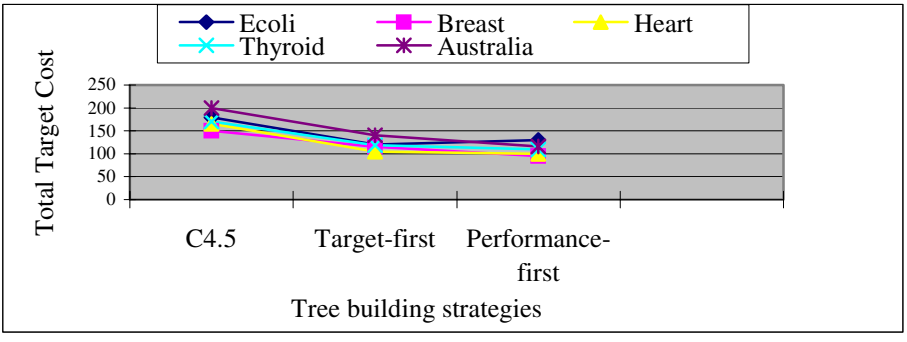

Fig. 3. Comparing of total target cost of three tree building strategies on different datasets

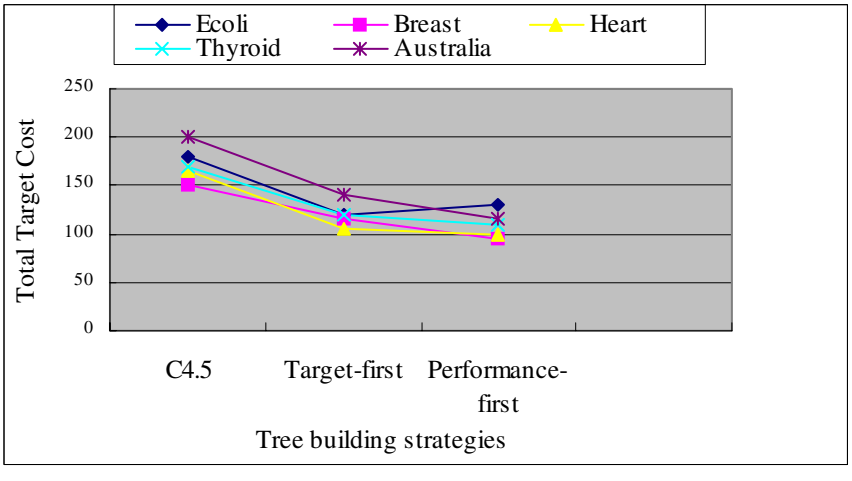

Fig. 4. Comparing of total resource of three tree building strategies on different datasets

To compare the influence of resource budget on three strategies, we conducted an experiment on all the datasets with varying budget B to support a part of all needed tests from 20 to 100 percent. For the more completely usage of resource, we use OST

testing strategy first, once the cost is exhaust we use M2 testing strategy to give a result. The result is shown in figure 5. From figure 5, we can see that all target cost will go down as the test examples can explore further branches, then lower total cost are obtained. The performance-first strategy also outperforms the other two in target cost with same resource consumption.

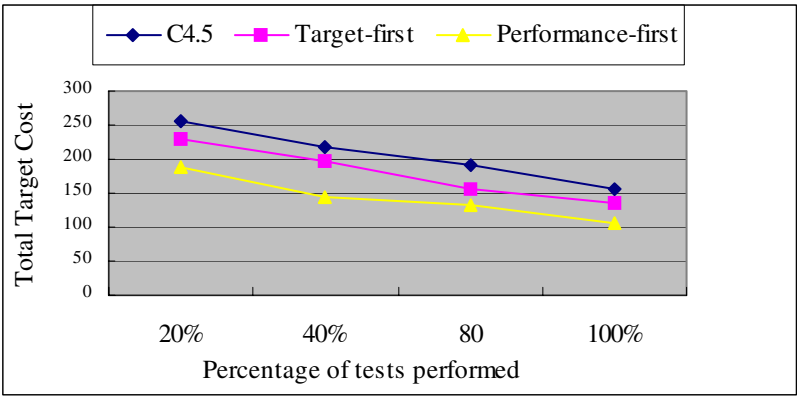

Fig. 5. Comparing of total target cost of three tree building strategies on percentage of tests performed under resource Budget

6 Conclusions and Future Work

In this paper, we presented a simple and novel method to overcome difficulty to define the multiple costs on the same cost scale in building decision trees that minimize the sum of the misclassification cost and the test cost. Our method involves two kinds of cost scales, and minimizes the one kind of cost as control the other one in a given budget. We proposed a new performance-based splitting criterion for attribute selection, and discussed several intelligent testing strategies in single cost scales as involving resource control. Our experiments show that our new decision-tree-building algorithm with performance-based splitting criterion dramatically outperforms the target-first tree building which simply add a resource control on single scales tree. In addition, compared to other related works, our algorithm has a lower cost consumption on most of testing strategies, and is thus more robust and practical.

In the future, we plan to consider how to minimize the total target cost with partial cost-resource exchanging. In some situations, such as medical diagnosis, this scenario is more practical since lot of hospitals provide VIP services. We also want to extend our Optimal Sequential Test to Optimal Batch Test, Also pruning can be introduced in our tree-building algorithm to avoid over-fitting of the data.

References

[1] Charles Ling, Qiang Yang, Jianning Wang and Shichao Zhang (2004), Decision Trees with Minimal Costs. In: *Proceedings of 21st International Conference on Machine Learning*, Banff, Alberta, Canada, July 4-8, 2004.

[2] Turney, P. D. (2000), Types of cost in inductive concept learning, *Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning,* Stanford University, California.

[3] Blake, C. L., and Merz, C. J. (1998), *UCI Repository of machine learning databases* (See [http://www.ics.uci.edu/~mlearn/MLRepository.html]). Irvine, CA: University of California, Department of Information and Computer Science.

[4] Turney, P. D. (1995), Cost-sensitive classication: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Articial Intelligence Research*, 2: 369-409, 1995.

[5] Mitchell, T.M. (1997), *Machine Learning*. *McGraw Hills*

[6] Zubek, V. B., Dietterich, T. G. (2002), Pruning Improves Heuristic Search for Cost-Sensitive Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*. pp. 27-34, Sydney, Australia.

[7] Greiner, R., Grove, A. J., and Roth D. (2002), Learning cost-sensitive active classiers. Articial Intelligence, 139(2): 137-174, 2002.

[8] Quinlan, J. R. (1993), *C*4.5*: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, California, 1993.

[9] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984), *Classification and Regression Trees*. Wadsworth, Monterey, California, 1984.