Decision Trees with Minimal Costs

Abstract

We propose a simple, novel and yet effective method for building and testing decision trees that minimizes the sum of the misclassification cost and the test cost. More specifically, we first propose a novel and simple splitting criterion for attribute selection in tree building. Our tree-building algorithm has many desirable properties for a cost-sensitive learning system that considers both types of costs. Then, assuming that the testing cases may have a large number of missing values for certain attributes, we design several intelligent testing strategies that can suggest how to obtain the missing values with new tests that come with a cost in order to minimize the total cost. We experimentally compare these strategies and C4.5, and demonstrate that our new algorithms significantly outperform C4.5 and its variations. In addition, the complexity of our algorithm is similar to C4.5, and is much lower than many previous works. Our work will be useful for many diagnostic tasks where one must consider both the misclassification cost and the test cost for obtaining missing information.

1. Introduction

Inductive learning techniques have met great success in building models that assign testing cases to classes (Quinlan 1993; Mitchell 1997). However, much previous inductive learning research has focused on how to minimize classification errors. The classification errors are useful in deciding whether a learned model tends to make correct decisions on assigning class labels for new cases, and as such they are an important factor to consider in practice. However, there are different types of classification errors, and the costs of different types of errors are often very different. For example, in a binary classification task in the medical domain, the cost of false positive (FP) and the cost of false negative (FN) are often very different. In addition, misclassification costs are not the only costs to consider when applying the model to new cases; we also consider the "test cost" that is as important as the misclassification cost when testing cases themselves do not provide all the values for their attributes that may be necessary for classification. That is, the testing strategies may suggest, with additional costs, obtaining information for the missing values on test cases.

Inductive learning methods that consider a variety of costs are often referred to as cost-sensitive learning (Turney 2002), and tasks that incur both misclassification and test costs abound in practice. As an example, consider again the task of a medical practice that examines incoming patients based on previous experiences. Suppose that these experiences have been compiled into a model such as a decision tree (Quinlan 1993). When dealing with a case for a new patient, it is often the case that certain information for this patent may yet be known; for example, the blood tests or the X-ray test may not have been done yet. One possible approach to solving this type of problems is to use the strategy in C4.5 (Quinlan 1993) in dealing with missing values. That is, when a testing case is classified by the decision tree, and is stopped at an attribute whose value is unknown, no test will be performed to obtain its value; instead, the testing case is distributed into branches of the attribute and the classification results are weighted on all of the branches. The problem with this approach is that it ignores the possibility of obtaining the missing value with a cost, and thus reducing the misclassification cost and the total cost. One of our testing methods (the third method discussed in Section 4) uses the C4.5's strategy, and is shown to be inferior to our new method proposed in this paper (see Sections 4 and 5). Another possible approach is to perform all tests for unknown values. This is clearly not optimal either as some of such tests can be very expensive. A third possible approach is to utilize the decision tree built by C4.5 to guide which tests should be performed. Again when a testing case is classified by the decision tree, and is stopped at an attribute whose value is unknown, the tree naturally suggests that this test should be done with a cost, and the testing case can follow the right branch, until it can be classified in a leaf. The problem with this approach is that when building the decision tree, the costs of obtaining these test results are completely ignored. As a consequence, tests that incur heavy costs may be placed on top of the tree, requiring all future patients to complete these tests. This may greatly increase the total test cost, and thus, the total cost. We compare this approach with our new methods proposed in this paper, and show that it is again inferior to the best method we propose (Sections 4 and 5).

In this paper, we study a tree-building strategy that minimizes the sum of the misclassification cost and the test cost, and a set of testing strategies that may suggest additional tests to be done with a cost to minimize the total cost on testing cases. Our tree-building algorithm has a number of very desirable properties for cost-sensitive learning systems, including some important properties as

pointed out by (Turney, 2000; Turney 1995). For example, if all test costs are larger than the misclassification cost, then no test should be performed and a one-node decision tree will be returned. As important as building a tree with the minimal total cost, we formulate several strategies to deal with the unknown values by taking into account both the misclassification and the test costs. These strategies are compared against each other and the best strategy is selected.

The rest of the paper is organized as follows. We first review the related work in Section 2. Then we present our new tree-building algorithm, and show it has many desirable properties (Section 3). After that, we consider several testing strategies and analyze their relative merits (Section 4). Finally, we present our experimental results (Section 5) and conclude the work with a discussion of future work (Section 6).

2. Review of Pervious Work

Much work has been done in machine learning on minimizing the classification errors. This is equivalent to assigning the same cost to each type of classification errors (for example, FP and FN), and then minimizing the total misclassification costs. In his survey article (Turney 2000), a whole variety of costs in machine learning is analyzed, and the test cost is singled out as one of the least considered areas in machine learning. In particular, (Turney 2000) considered the following types of costs in machine learning:

- Misclassification costs: these are the costs incurred by misclassification errors. Works such as (Domingos 1999, Elkan 2001) considered machine learning with non-uniform misclassification costs
- Test costs: these are the costs incurred for obtaining attribute values. Some previous work such as (Nunez, 1991; Tan 1993) considered the test cost alone without incorporating misclassification cost. As pointed out by (Turney 2000) it is obviously an oversight. As far as we know, the only work considering both misclassification and test costs includes (Turney 1995; Zubek and Dietterich 2002; Greiner et al. 2002). We discuss these works in detail below.

In (Zubek and Dieterrich 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. While related to our work, their research adopts an optimal strategy, which may take very high computational cost to conduct the search process. In contrast, we adopt the local search algorithm of (Quinlan 1993) using a polynomial time algorithm to build a model, which returns a new decision tree. Then when performing the testing, our testing strategy (see later) together with the decision tree will suggest whether to do a test or not. Thus, our algorithm follows the direction of approximation rather than optimal algorithms.

Similar in the interest in constructing an optimal learner, (Greiner et al. 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. Our work also considers the decision tree model, where we additionally consider both the minimization of misclassification cost on training data and the formulation of a testing strategy for minimizing the test costs on the testing data. As mentioned above, because our algorithm essentially adopts the same decision-tree building framework as in (Quinlan 1993), our algorithm is expected to be more efficient than Turney's genetic algorithm based approach.

3. Building Decision Tree With Minimal Costs

We assume that the training data may consist of some missing values (whose values cannot be obtained). We also assume a static cost structure where the cost is not a function of time or cases. Further, we assume that 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.

Our new decision-tree learning algorithm is quite simple. For simplicity, we consider discrete attribute and binary class labels; extensions to other cases can be made likewise. We assume that *FP* is the cost of one false positive example, and *FN* is the cost of one false negative example. Our algorithm uses a new splitting criterion of *minimal total cost* on training data, instead of minimal entropy, to build decision trees. This cost measure is equivalent to the expected total cost measure used in the works of (Turney 1995; Zubek and Dietterich 2002; Greiner et al. 2002). More specifically, at each step, rather than choosing an attribute that minimizes the entropy (as in *C4.5*), our algorithm chooses an attribute that reduces and minimizes the total cost, which is the sum of the test cost and the misclassification cost, for the split. Then, similar to *C4.5*, our algorithm chooses a locally optimal attribute without backtracking. Thus the resulting tree may not be globally optimal. However, the efficiency of the tree-building algorithm is generally high. A concrete example is given later in this section.

A fine point of our new algorithm is the way it deals with attributes with unknown values in the training set. In many variations of decision tree algorithms, the unknown value is treated as one of the ordinary values. However, in our work, the strategy is that all unknown values (we use "?" for the unknown value) are treated as a special

"value": no leaf or sub-tree will be built for examples with the "?" value. This is because it is unrealistic to assume the unknown values would be as useful for classification as the known values. In addition, when a testing example is stopped at an attribute whose value is unknown, if the attribute has a "?" branch, it is impossible to decide whether the test should be performed by the tree. Therefore, the examples with unknown attribute values will not be grouped together as a leaf, or to build a sub-tree; instead, they are "gathered" inside the node that represents that attribute. We then calculate the ratio of the positive and negative examples in that internal node. See the concrete example given later for more details. Our second testing algorithm (see Section 4) will use such ratios in making prediction.

Another important point is how the leaves are labeled. In traditional decision tree algorithms, the majority class is used to label the leaf node. In our case, as the decision tree is used to make predictions in order to minimize the total cost, the leaves are labeled also to minimize the total cost. That is, at each leaf, the algorithm labels the leaf as either positive or negative (in a binary decision case) by minimizing the misclassification cost. More specifically, suppose that the leaf has P positive examples, and N negative examples. If $P \times FN > N \times FP$ (i.e., the cost of predicting negative is greater than the cost of predicting positive), then the leaf is labeled as positive; otherwise it is labeled as negative. Therefore, the label of a leaf does not just depend on the majority class of the leaf, but also the cost of misclassification.

Let us look at a concrete example. 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 misclassification cost is $T = N \times FP$. Suppose that an attribute A with a test cost C 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 $(P1+N1+P2+N2) \times C$ (i.e., cases with unknown attribute values do not incur test costs). Assume that the first branch will be labeled as positive (as $P1 \times FN > N1 \times FP$), and the second branch will be labeled as negative, then the total misclassification cost of the two branches would be $N1 \times FP + P2 \times FN$. 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:

$$T_A = (P1+N1+P2+N2) \times C + N1 \times FP + P2 \times FN + N0 \times FP$$

If $T_A < T$, *where* $T = N \times FP$, 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.

Finally, as our tree attempts to minimize the total cost, it may also overfit the training dataset. Traditional decision tree algorithms such as C4.5 incorporate a post-tree pruning procedure to simplify the tree. In the current version of our algorithm, however, we do not yet perform tree pruning. As all of our tree building algorithms (see Section 3) build unpruned trees, our experiment comparisons (Section 5) are still fair and valid. It remains our future work to include pruning in our tree-building algorithm with the minimal total cost.

Aimed at minimizing the total cost of test and misclassification, our new decision-tree algorithm has several desirable features. We will discuss these features below, using the dataset "Ecoli" as an example (Blake & Merz 1998). This dataset has 332 labelled examples, which are described by 6 attributes. The numerical attributes are first discretized using the minimal entropy method (Fayyad & Irani 1993), as our tree building algorithm can currently only accept discrete attributes (but it is straightforward to extend our algorithm to accept continuous attributes as C4.5 does). The attribute values are renamed as 1, 2, 3, and so on. More details on this and other datasets used in experiments can be found in Section 5.

The first property, as discussed in the Introduction, is that the relative difference between misclassification and test costs can affect the tree dramatically. If the former is less than the latter, then no test should be performed, and the decision tree would be simply a one-node leaf. On the other hand, if the former is much larger than the latter, then all tests should be done, as long as they are relevant; i.e., they can improve the predictive accuracy. This can be seen clearly from the "Ecoli" dataset. Indeed, if the misclassification cost is set to 200 for both FP and FN, and all test cost is set to 300, then the algorithm returns a one-leaf node as shown in Figure 1 (a). On the other hand, when all test costs are set to 0, then the tree is the "largest"; in this case, the tree has 13 nodes in total, and can be seen in Figure 1 (c). As an "intermediate" case, if all test costs are set to 20, then the decision tree with the minimal cost has 6 nodes in total, and the tree can be seen in Figure 1 (b).

The second important and desirable property is that for attributes with different test costs, our new algorithm is likely to choose an attribute with zero or small cost at the top (or root) of the tree. This is because the attribute at the root will be tested by all examples, and thus the total attribute cost would be relatively high. Choosing an

attribute with zero or small cost helps reduce the total cost. Of course attribute selection also depends on the distributions of attribute values and class labels of the training examples.

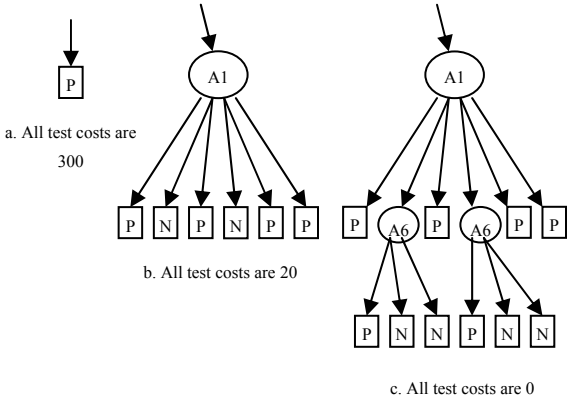

Figure 1. Three different decision trees built with different test costs.

Table 1 shows three cases in which attribute costs are different. In the first case (the baseline), all attribute costs are set to 20. In the second and third cases attribute costs are set differently. The misclassification cost is set at 800 for both FP and FN. As we can see, in the second case, the attribute A2 has the smallest test cost, and it is indeed chosen as the root of the tree as shown in Figure 2(b). In the third case, attribute A5 has the smallest test cost, and it is chosen as the root (Figure 2(c)).

Table 1. Three different sets of attribute costs.

COST	A1	A2	A3	A4	A5	A6
Tree # 1	20	20	20	20	20	20
Tree # 2	200	20	100	100	200	200
Tree # 3	200	100	100	100	20	200

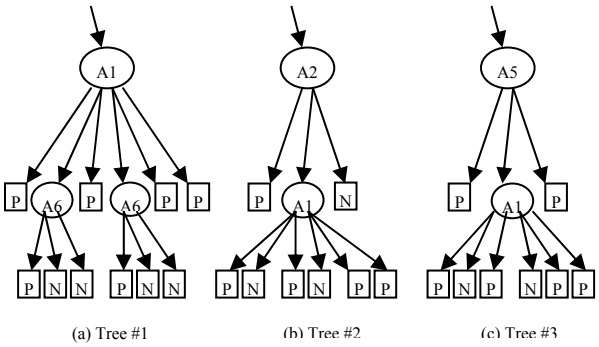

Figure 2. Three different decision trees built with three different test costs as in Table 1.

The third property, related to the second one, is that when the test cost of an attribute is increased, that test attribute will be "pushed" down in the tree, until it "falls out" of the tree (when the test cost becomes too large). Figures 3 (a) to (c) show the trees with the test cost of A1 set to 20, 50, and 80, respectively, while all other attribute costs are fixed, and the misclassification cost is 800. We can see clearly that with the increase of the test cost of A1, the attribute moves down the tree, until it falls out of the tree in the end.

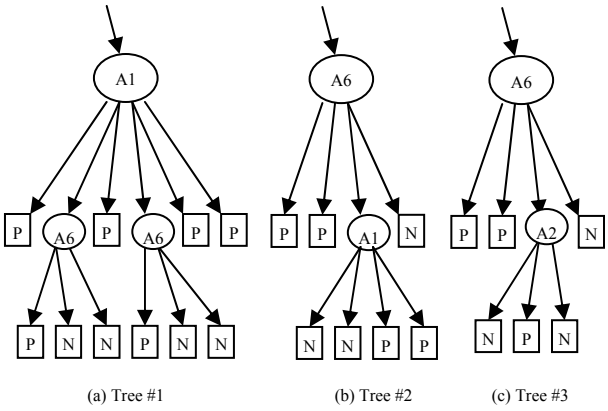

Figure 3. Three different decision trees built with A1 cost as 20, 50, and 80.

4. Performing Tests on Testing Examples

After the minimal-cost decision tree is built, the next interesting question is how this tree can be used to deal with testing examples with many missing values, in order to predict the class of the testing examples with the minimal total cost for this case. Deciding which tests should be performed is a part of the testing strategy.

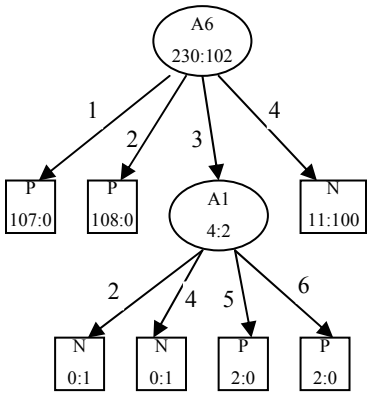

Figure 4. A decision tree built from the Ecoli dataset (costs are set as in Table 2).

We will study four testing strategies. We use the decision tree in Figure 4, built with the test cost in Table 2, and a testing example in Table 3 to illustrate the four strategies described below. Bear in mind that this is only for one

particular testing case. The overall performance of these strategies will be compared in the next section with a large number of testing examples.

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

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

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

The first strategy, called Optimal Sequential Test (OST), is very simple and intuitive. It uses the tree built with the minimal cost to decide what tests must be performed in sequence. More specifically, each test example goes down the tree until an attribute whose value is unknown is met in the testing example. As the tree was built to minimize the total cost, this tree would suggest that this test should be performed with the cost, and its value would decide which branch to go down. For example, when the testing example in Table 3 goes down the tree in Figure 4, it will stop at the node A6. Then the test is done with a cost 20, and it reveals the value 3. Then the example goes down to node A1, and a test on A1 is performed with a cost 50, with the value 6. Thus, it falls into the rightmost leaf under A1, which predicts the class P. The prediction is the same as the true class of the testing case, so there is no misclassification cost. Thus the total cost is 20 + 50 = 70.

This strategy, while optimal based on the minimal cost of the training set, is sequential. That is, one must wait for the result of the first test before a next test could be determined. In some medical diagnosis, doctors cannot afford to wait the result of the first test before other tests can be done; they normally order a set of tests to be done at once. This "Optimal Batch Test" can be modeled in our decision tree easily. The basic idea is that when a testing case is stopped at the first attribute whose value is unknown, all unknown values under that attribute must be obtained. Clearly this strategy will return the same prediction as OST (i.e., same misclassification cost), but it would incur a higher test cost.

The second strategy uses the same decision to make prediction, but it stipulates that no further tests should be done. More specifically, when a testing example is stopped at an attribute whose value is unknown, it 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). Using the same example, when the testing example stops at the node A6, it would predict that the testing example is of class for the node A6 (which is positive by $P \times FN > N \times FP$ i.e. $230 \times 800 > 102 \times 800$). As no test is done, there is no test cost. The total cost is thus 0 here.

The third strategy is a variation of the second strategy. 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 final class is a weighted sum of the class in each branch. Note that this is essentially C4.5's strategy in dealing with missing values. Notice that even though the testing case goes down from the attribute with a cost, there is no testing cost involved, as it merely "guesses" the values of the attributes. Using the same example to illustrate this strategy, the testing case will not stop at node A6 this time; instead, it will distribute into four branches with a ratio 107/108/6/111. The first two branches make a correct prediction with no misclassification cost. The last branch makes a wrong prediction, with a misclassification cost of 800. The third branch encounters another unknown value, so it is distributed further down in the tree, with a ratio of 1/1/2/2. The first two branches make a wrong prediction (costing 800), while the next two branches make a correct prediction. With the total number of 432 (230+102) training examples in the tree building, the weighted cost for this testing example is thus: $800 \times (1+1+111)/432 = 209.3$.

The fourth and final strategy also stipulates that no further tests should be done, but it utilizes the existing attribute values to the full extent. More specifically, for each testing example, a new (and different) decision tree is built dynamically from all of the training examples with only those attributes whose values are known in the testing example. In this way, the new decision tree only uses attributes with known values in the testing example, and thus, no new test would be needed during the testing. As an example, as A2, A4, and A5 are the only known attributes, a new decision tree using the training examples with A2, A4, and A5 as attributes will be built as in Figure 5. From this tree, the testing examples can easily be classified by the tree, as values of attributes used in the tree are all known in the testing example. In this case, as A5's value is 2, it goes down to the second branch, and predicts N. It thus incurs a misclassification cost of 800.

This final strategy in itself is interesting, and it is a kind of lazy learning algorithms where the learning model is built only during testing and can be affected by the testing examples (see, for example, LazyDT by (Friedman et al 1996)). Here, as testing examples may have a different set of known attributes, the trees from different testing examples can be different too.

We expect that our first testing strategy, the Optimal Sequential Test, would be best with overall lowest total cost, as it is based on minimizing the total cost in the

training set. The fourth method, building different trees for different testing cases, would be second, as it utilizes fully the training data, and like lazy learning, it explores the search space in the local region. The second and third methods would probably perform the worst. In the next section, we will perform extensive experiments to compare and evaluate these methods with real-world datasets.

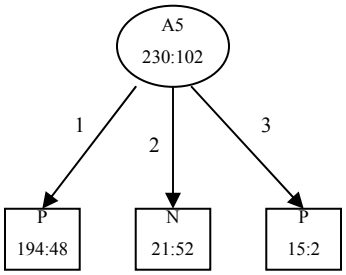

Figure 5. A decision tree built only from attributes A2, A4 and A5.

5. Experiments

We ran experiments on five real-world datasets and compared the four testing strategies against the baseline C4.5. In C4.5, we use the information gain to build a decision tree (without pruning). Missing values are ignored in training examples as done in C4.5. Then the tree is used to predict the testing examples. The testing process is similar to our first testing strategy (Optimal Sequential Test). That is, when the testing example is classified by the tree, and if an attribute value is unknown, a test is done with a cost. The testing example then goes down further according to the value obtained, until it reaches the leaf, where a prediction is made.

We use five datasets in our experiments. 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 & 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
Australia	15	653	296/357

For the experiments, each dataset is split into two parts: the training set (60%) and the testing set (40%). A decision tree is built from the training set using our new algorithm that minimizes the total cost (Section 3). For our fourth lazy-style testing method, a different tree is built for each testing case. The decision tree is then used to predict the testing examples, and to decide what tests, if any, should be performed to minimize the total cost.

As we discussed in the Introduction section, testing examples would often have more unknown values, as it is part of the testing process to decide what tests need to be performed. Therefore, a certain percentage of attributes are randomly selected and marked as unknown. If the testing algorithm decides to perform a test on an unknown attribute, then its real value is revealed and a cost is accumulated.

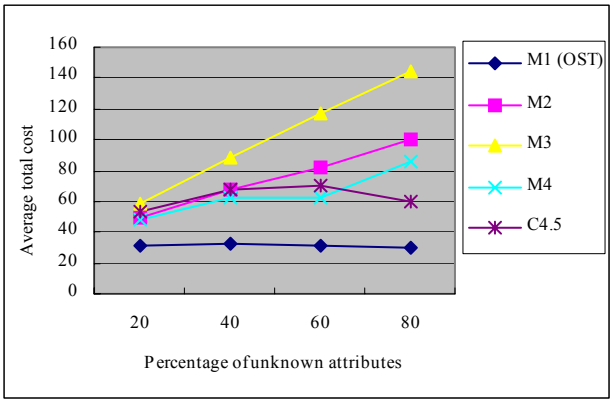

Figure 6. Total cost comparison under different unknowns.

Figure 6 shows different algorithms in terms of the different percentages of unknown attribute values in the testing examples. This figure shows the graph for the Ecoli dataset, whereas the other figures for other datasets are similar and thus are omitted. The scales on the x-axis (20%, 40%, and so on) represent the percentage of unknown attributes in the testing sets. The curve represents the average total cost of a testing case of 5 different testing strategies, averaged over 5 runs. In this set of experiments, the misclassification cost is set as 400/400 (400 for FN and 400 for FP), and the test costs are set randomly between 0 and 100.

From this experiment, we can draw several interesting conclusions. First, our first method (M1), Optimal Sequential Test (OST), is clearly the winner. The total cost is always the lowest, and it does not increase much when the percentage of unknown values increases. This is mainly because the test costs are relatively cheap, and with the suggestions of tests performed by OST, the final prediction is quite accurate (so small misclassification cost). Second, our fourth method (M4), a lazy-style decision tree algorithm, is the second best when the percentage of unknown attributes is less than 60%. It is because it utilizes fully the known attributes by building a

new decision tree for each testing example. However, when there are too many unknown attributes (such as 80%), the decision tree built from only 20% of the known attributes is obviously inaccurate, thus the misclassification cost increases dramatically, increasing the total cost as well. Third, C4.5 is the third best overall, and similar to OST, the total cost does not increase with more missing values in testing cases. This shows that doing tests (as in Optimal Sequential Test and C4.5) is better than not doing tests (as in Methods 2, 3 and 4) when the test cost is not too large. However, C4.5 is not as good as OST as test costs are not taken into consideration in the decision tree built. Fourth, the second and third methods (M2 and M3) are worse, because they use a single decision tree built from the training set, and it does not perform tests to improve predictive accuracy. Here we can see that their performance degrades as the unknown values increase. Fifth, it is clear from the graphs that the more unknown attribute values, the higher the total cost for testing strategies without doing the test, and the more adventurous our first method Optimal Sequential Test and C4.5 would be compared to other methods.

Recall that M3 is essentially C4.5's strategy in dealing with unknown values. It is surprising to see that in this and later experiments M3 (the C4.5 strategy) seems to be worse than the naïve strategy M2. We think that the main reason is that distribution into branches of the tree due to unknown values accumulates a large misclassification cost overall. It would be interesting to see if C4.5 would be better off if it uses the naïve strategy as in M2 in dealing with missing values.

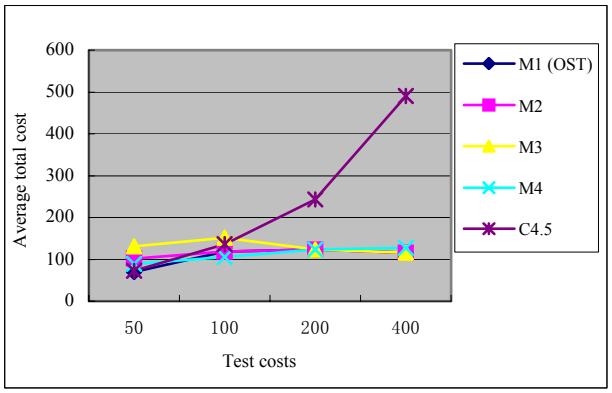

Figure 7. Comparison under different test costs.

The next set of experiments compares different algorithms in terms of magnitudes of the testing costs while the classification cost is fixed at 400/400. In Figure 7, which plots the result on the Ecoli dataset, the costs of the tests (attributes) range from 50 to 400. The percentage of unknown attribute values is set to be 60%. Again we can make many interesting conclusions, some of which are similar to what we have drawn before, but some are quite different. First, our first method (M1), Optimal Sequential Test (OST), is still clearly the winner. But other testing strategies we proposed (M2, M3, and M4) become very similar to OST when the test cost increases. This is expected as the test costs increase, our tree-building algorithm will prefer not to build a tree (or only to build a one-node tree) to save the total cost and this may end up with lower total costs. When this happens, the four testing strategies we proposed may become the same. Second, C4.5 performs much worse when the test cost increases. This is also expected as C4.5 builds the same decision tree independent of the test cost, and thus the total costs become much larger when the test costs increase. Third, when the test cost is still relatively small (from 50 to 100), our fourth method (M4), a lazy-style decision tree algorithm, is still next in ranking. Because again it utilizes fully the known attributes by building a new decision tree for each testing example. Fourth, it is clear from the graphs that the total test cost of all of our testing strategies does not increase much when the test cost increases. This is again because our tree-building algorithm and testing strategies aim at minimizing the total cost of misclassifications and tests.

Our last set of experiments is similar to the one above, but with a dramatically unbalanced misclassification costs. More specifically, the FP and FN costs are set to 200/1000. Test costs are set randomly, and the percentage of unknown attributes values is 60%. In this case, we confirmed our expectation that C4.5 would not perform well as it does not distinguish the two types of misclassification costs while our methods do. The result is presented in Figure 8.

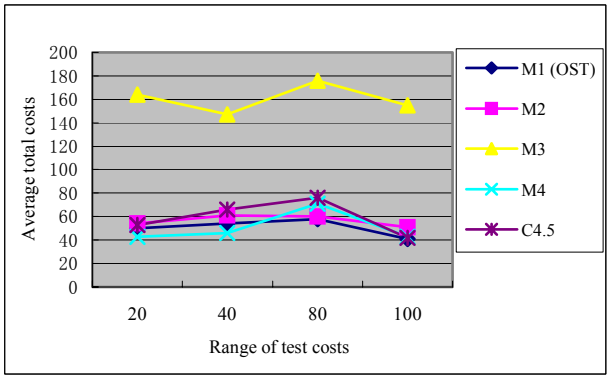

Figure 8. Comparing unbalanced misclassification costs.

One of our significant results is that the best performance as demonstrated by the method M1 on the Ecoli dataset is repeatable throughout all datasets that we consider (see Table 4). Figures 9 (a) and (b) show the performance of M1 for different datasets. As can be seen, as the percentage of unknowns and the test costs change, the performance of M1 is consistent across different datasets. We can conclude that the superiority of M1 over the other methods is in fact a general phenomenon.

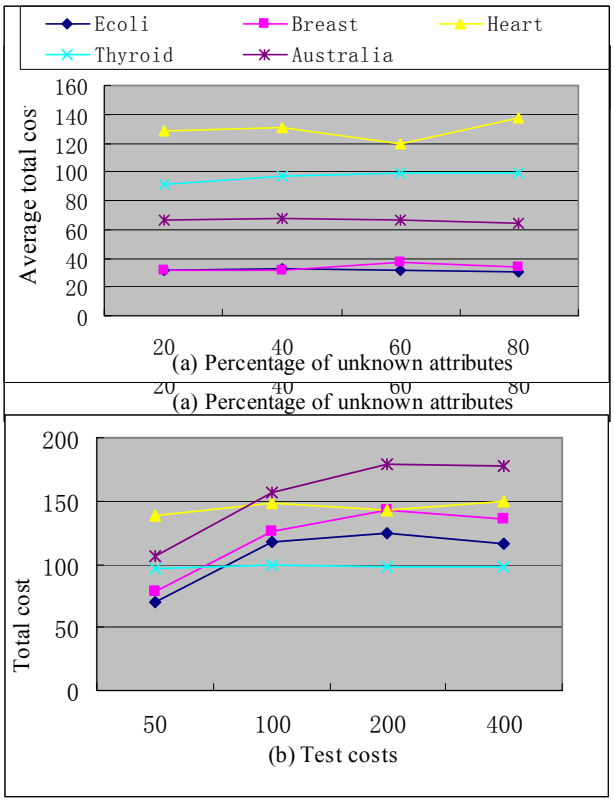

Figure 9 M1 for different datasets with varying unknowns (a) and test costs (b)

6. Conclusions and Future Work

In this paper, we presented a simple and novel method for effectively building decision trees that minimize the sum of the misclassification cost and the test cost. Our method utilizes a new cost-based splitting criterion for attribute selection, and incorporates several intelligent testing strategies that can suggest how to obtain missing values with new tests. Our experiments show that our new decision-tree-building algorithm, together with the best testing strategy, Optimal Sequential Test, can dramatically outperform a number of other competing algorithms, including C4.5. In addition, compared to other related works, our algorithm has a much lower computational complexity, and is thus more practical.

In the future, we plan to consider several extensions of this work. One important direction is to consider minimize the total cost when new tests are done in one shot, rather than in a sequential manner. In some situations, such as medical diagnosis for human and plants (Veeramachaneni and Avesani 2003), this scenario is more practical since doctors and scientists often suggest several medical tests to be done at once. We did extend our Optimal Sequential Test to Optimal Batch Test in Section 4, but it would be interesting to find more effective methods. Also pruning can be introduced in our tree-building algorithm to avoid overfitting of the data.

Blake, C.L., & Merz, C.J. (1998). *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.

Domingos, P. (1999) MetaCost: A General Method for Making Classifiers Cost-Sensitive. In *Knowledge Discovery and Data Mining*, Pages 155-164.

Elkan. C. (2001) *The Foundations of Cost-Sensitive Learning*. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (IJCAI'01), pp. 973-978.

Fayyad, U. M., & Irani, K. B. (1993). Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1022--1027. Morgan Kaufmann,

Friedman, J. Yun, Y. and Kohavi, R. Lazy decision trees, in *Proc. 13th Nat'l. Conf. Artificial Intelligence*, 1996.

Greiner, R, Grove A. and Roth D. (2002) Learning Cost-Sensitive Active Classifiers, *Artificial Intelligence Journal* 139:2, pp. 137-174.

Margineantu, D. (2001). Methods for cost-sensitive learning. *Dissertation*, Oregon State Univ.

Mitchell, T.M. (1997) *Machine Learning* McGraw Hill

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

Quinlan, J. R. (1993) C4.5: Programs for Machine Learning. *Morgan Kaufmann Publishers.*

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

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.

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, 1995.

Veeramachaneni S. and Avesani P. (2003) Active Sampling for Feature Selection. In Proceedings of the Third IEEE International Conference on Data Mining. Pp. 665-668. Florida, USA IEEE Computer Society.

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