Simple Test Strategies for Cost-Sensitive Decision Trees

Shengli Sheng[1], Charles X. Ling[1], Qiang Yang[2]

[1]Department of Computer Science, The University of Western Ontario
London, Ontario N6A 5B7, Canada
{cling, ssheng}@ csd.uwo.ca
[2]Department of Computer Science, Hong Kong UST, Hong Kong
qyang @ cs.ust.hk

Abstract. We study cost-sensitive learning of decision trees that incorporate both test costs and misclassification costs. In particular, we first propose a lazy decision tree learning that minimizes the total cost of tests and misclassifications. Then assuming test examples may contain unknown attributes whose values can be obtained at a cost (the test cost), we design several novel test strategies which attempt to minimize the total cost of tests and misclassifications for each test example. We empirically evaluate our tree-building and various test strategies, and show that they are very effective. Our results can be readily applied to real-world diagnosis tasks, such as medical diagnosis where doctors must try to determine what tests (e.g., blood tests) should be ordered for a patient to minimize the total cost of tests and misclassifications (misdiagnosis). A case study on heart disease is given throughout the paper.

1 Introduction

In many real-world machine learning applications, minimizing misclassification error is often not the ultimate goal, as "errors" can cost differently. This type of learning is called cost-sensitive learning. Turney [13] surveys a wide range of costs in cost-sensitive learning, among which two types of costs are singled out as most important: misclassification costs and test costs. For example, in a binary classification task, the costs of false positive (FP) and false negative (FN) are often very different. In addition, attributes may have costs (test costs) when acquiring values. The goal of learning is to minimize the total cost of misclassifications and tests.

Tasks involving both misclassification and test costs are abundant in real-world applications. For example, when building a model for medical diagnosis from the training data, we must consider the cost of tests (such as blood tests, X-ray, etc.) and the cost of misclassifications (errors in the diagnosis). Further, when a doctor sees a new patient (a test example), tests are normally ordered, at a cost to the patient or the insurance company, to better diagnose or predict the disease of the patient (i.e., reducing the misclassification cost). Doctors must balance the trade-off between potential

misclassification costs and test costs to determinate which tests should be ordered, and at what order, to reduce the expected total cost. A case study on heart disease is given in the paper.

In this paper, we propose a lazy-tree learning that improves on a previous decision tree algorithm that minimizes the total cost of misclassifications and tests. We then describe several novel "test strategies" to determine what tests should be performed, and at what order, for attributes with unknown values in test examples such that the total expected cost is minimum. Extensive experiments have been conducted to show the effectiveness of our tree building and test strategies compared to previous methods.

2 Review of Previous Work

Cost-sensitive learning has received an extensive attention in recent years. Much work has been done in considering non-uniform misclassification costs (alone), such as [4, 5, 7]. Those works can often used to solve the problem of learning with very imbalanced datasets [3]. Some previous work, such as [11], considers the test cost alone without incorporating misclassification cost. As pointed out by [13] it is obviously an oversight. A few previous works consider both misclassification and test costs, and they are reviewed below.

In [14], the cost-sensitive learning problem is cast as a Markov Decision Process (MDP). They adopt an optimal search strategy, which may incur a high computational cost. In contrast, we adopt the local search similar to C4.5 [10], which is very efficient. Lizotte et al. [9] study the theoretical aspects of active learning with test costs using naïve Bayes classifiers. Turney [12] presents a system called ICET, which uses a genetic algorithm to build a decision tree to minimize the cost of tests and misclassifications. Our algorithm again is expected to be more efficient than Turney's genetic algorithm.

Ling et al. [8] propose a new decision tree learning algorithm that uses minimum total cost of tests and misclassifications as the attribute split criterion. However, a single tree is built for all test examples. The information of some known attributes in a test example is ignored if they do not appear in the path through which the test example goes down the tree to a leaf, and their test strategies are very simple. In this paper, we propose a lazy-tree learning to minimize the total cost of misclassifications and tests. But it can make use of the known attributes in each test example to reduce the total cost. We also propose an improved attribute selection criterion to split the training data. In addition, we propose several novel and sophisticated test strategies for obtaining missing attribute values when classifying new test examples that, as far as we know, have not been published previously.

Chai et al. [1] propose a naïve Bayes based algorithm, called CSNB, which searches for minimal total cost of tests and misclassifications. Our test strategies utilize the tree structure while naïve Bayes does not. Experiments show that our tree-based test strategies outperform CSNB in most situations (see experimental comparisons later in the paper).

3 Lazy Decision Trees for Minimum Total Cost

We assume that we are given a set of training data (with possible missing attribute values), the misclassification costs (FP and FN), and test costs for each attribute. Instead of building a single decision tree for all test examples [8], we propose a lazy-tree approach to utilize as much information in the known attributes as possible. More specifically, given a test example with known and unknown attributes, we first reassign the test cost of the known attributes to be 0 while the cost of the unknown attributes remains unchanged. For example, suppose that there are 3 attributes and their costs are \$30, \$40, and \$60 respectively. If in a test example, the second attribute value is unknown (obtain by testing), then the new test costs would be reset to \$0, \$40, and \$0 respectively. Then a tree is built using the split criterion that minimizes the total cost of tests and misclassifications. Clearly our method builds different trees for test examples with different sets of unknown attributes. As our lazy-tree learning approach utilizes as much information in the known attributes as possible, we expect it will reduce the total cost in testing significantly. The rationale is that attributes with the zero test cost are more likely to be chosen early during the tree building process. When a test example is classified by this specific tree, it is less likely to be stopped by unknown attributes near the top of the decision tree. This tends to reduce the total test cost, and thus the total cost, as shown later in the experiments.

Another improvement we made over [8] is that we use the *expected* total misclassification cost when selecting an attribute for splitting. This gives a more accurate choice for attribute selection. That is, an attribute may be selected as a root node of a decision tree if the sum of the test cost and the expected misclassification costs of all branches is minimum among other attributes, and is less than that of the root. For a subset of examples with *tp* positive examples and *tn* negative examples, if $C_P = tp \times TP + tn \times FP$ is the total misclassification cost of being a positive leaf, and $C_N = tn \times TN + tp \times FN$ is the total misclassification cost of being a negative leaf, then the probability of being positive is estimated by the relative cost of C_P and C_N; the smaller the cost, the larger the probability (as minimum cost is sought). Thus, the probability of being positive is: $1 - \frac{C_P}{C_P + C_N} = \frac{C_N}{C_P + C_N}$. The expected misclassification cost of being positive is: $E_P = \frac{C_N}{C_P + C_N} \times C_P$. Similarly, the probability of being a negative leaf is $\frac{C_P}{C_P + C_N}$; and the expected misclassification cost of being negative is: $E_N = \frac{C_P}{C_P + C_N} \times C_N$. Therefore, without splitting, the expected total misclassification cost of a given set of examples is: $E = E_P + E_N = \frac{2 \times C_P \times C_N}{C_P + C_N}$. If an attribute A has l branches, then the

expected total misclassification cost after splitting on A is: $E_A = 2 \times \sum_{i=1}^{l} \frac{C_{P_i} \times C_{N_i}}{C_{P_i} + C_{N_i}}$.

Thus, $(E - E_A - T_C)$ is the expected cost reduction splitting on A, where T_C is the total test cost for all examples on A. it is easy to find out which attribute has the smallest expected total cost (the sum of the test cost and the expected misclassification cost), and if it is smaller than the one without split (if so, it is worth to split). With the expected total misclassification cost described above as the splitting criterion, the lazy-tree learning algorithm is shown as follows.

LazyTree(*Examples, Attributes, TestCosts, testExample*)

1. For each attribute
   a. If its value is known in *testExample*, its test cost is assigned as *0*
2. call *CSDT*(*Examples, Attributes, TestCostsUpdated*) to build a cost-sensitive decision tree

CSDT(*Examples, Attributes, TestCosts*)

1. Create a *root* node for the tree
2. If all examples are positive, return the single-node tree, with *label* = +
3. If all examples are negative, return the single-node tree, with *label* = -
4. If attributes is empty, return the single-node tree, with label assigned according to *min* (E_P, E_N)
5. Otherwise Begin
   a. If *maximum cost reduction* < *0* return the single-node tree, with label assigned according to *min* (E_P, E_N)
   b. *A* is an attribute which produces maximum cost reduction among all the remaining attributes
   c. Assign the attribute *A* as the tree *root*
   d. For each possible value vi of the attribute *A*
      i. Add a new branch below root, corresponding to the test $A=v_i$
      ii. Segment the training examples into each branch *Example*_v_i
      iii. If no examples in a branch, add a leaf node in this branch, with label assigned according to *min* (E_P, E_N)
      iv. Else add a subtree below this branch, CSDT(*examples*_v_i, *Attributes-A, TestCosts*)
6. End
7. Return *root*

One weakness of our method is higher computational cost associated with lazy learning. However, our tree-building process has the same time complexity as C4.5, so it is quite efficient. In addition, lazy trees for the same set of unknown attributes are the same. Trees frequently used can be stored in memory for the speed trade-off.

4 A Case Study on Heart Disease

We apply our lazy decision tree learning on a real dataset, the Heart Disease, with known test costs. The dataset was used in the cost-sensitive genetic algorithm by [12]. The learning problem is to predict the coronary artery disease from the 13 non-invasive tests on patients. The class label 0 or negative class indicates a less than 50% of artery narrowing, and 1 indicates more than 50%. The costs of the 13 non-invasive tests are in Canadian dollars, and were obtained from the Ontario Health Insurance Program's fee schedule [12]. These individual tests and their costs are: age ($1), sex ($1), cp (chest pain type, $1), trestbps (resting blood pressure, $1), chol (serum cholesterol in mg/dl, $7.27), fbs (fasting blood sugar, $5.20), restecg (resting electrocardiography results, $15.50), thalach (maximum heart rate achieved, $102.90), exang (exercise induced angina, $87.30), oldpeak (ST depression induced by exercise, $87.30), slope (slope of the peak exercise ST segment, $87.30), ca (number of major vessels colored by fluoroscopy, $100.90), and thal ($102.90). Tests such as thalach, exang, oldpeak, and slope are electrocardiography results when the patient runs on a treadmill, and are usually performed as a group. Tests done in a group may be discounted in costs, but this is not considered in this paper (see future work). However, no information about misclassification costs was given. After consulting a researcher in the Heart-Failure Research Group in the local medical school, a positive prediction normally entails a more expensive and invasive test, the angiographic test, to be performed, which accurately measures the percentage of artery narrowing. A negative prediction may prompt doctors to prescribe medicines, but the angiographic test may still be ordered if other diseases (such as diabetes) exist. An angiographic test costs about $600. Thus, it seems reasonable to assign false positive and false negative to be $600 and $1000 respectively.

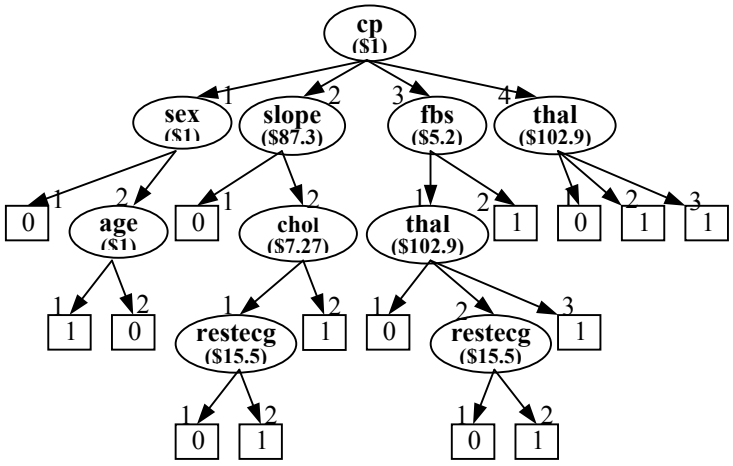

Fig. 1. Lazy tree for the test case with missing values for all attributes

Assuming in a new test example all attribute values are missing (as seeing a completely new patient), the original test costs given above are used directly for the tree building. The numerical attributes in datasets are discretized into integers (1, 2, …) using the minimal entropy method of [6]. We apply our lazy decision tree learning for this test case, and obtain a decision tree shown in Figure 1.

We can see that often less expensive tests are used in the top part of the tree. For example, cp is selected as the root of the tree, sex and fbs are in the second level of the tree. But slope and thal, expensive tests, are also selected in the second level, since they have higher merit to reduce the total cost. That is, the splitting criterion selects tests according to their relative merit of reducing the total cost. When this tree is presented to the Heart-Failure researcher, he thinks that the tree is reasonable in predicting artery narrowing. Note that it is not feasible for us to compare our results on this dataset with [12] and other previous work using the same dataset, as they have very different settings. Here we present this case to show intuitively how our lazy-tree building algorithm and test strategies (to be discussed next) work.

5 Two Categories of Test Strategies

We define two categories of test strategies: Sequential Test and Single Batch Test. For a given test example with unknown attributes, the Sequential Test can request only one test at a time, and wait for the test result to decide which attribute to be tested next, or if a final prediction is made. The Single Batch Test, on the other hand, can request one set (batch) of one or many tests to be done simultaneously before a final prediction is made.

The related test strategies have many corresponding applications in the real world. In medical diagnoses, for example, doctors normally order one set of tests (at a cost) to be done at once. This is the case of the Single Batch Test. If doctors only order one test at a time (this can happen if tests are very expensive and/or risky), this is the case of the Sequential Test. In the next two subsections the two types of test strategies will be discussed in great details.

5.1 Lazy-trees Optimal Sequential Test (LazyOST)

Recall that Sequential Test allows one test to be performed (at a cost) each time before the next test is determined, until a final prediction is made. Ling et al. [8] described a simple strategy called Optimal Sequential Test (or OST in short) that directly utilizes the decision tree built to guide the sequence of tests to be performed in the following way: when the test example is classified by the tree, and is stopped by an attribute whose value is unknown, a test of that attribute is made at a cost. This process continues until the test case reaches a leaf of the tree. According to the leaf reached, a prediction is made, which may incur a misclassification cost if the prediction is wrong. Clearly the time complexity of OST is only linear to the depth of the tree.

One weakness with this approach is that it uses the same tree for all testing examples. In this work, we have proposed a lazy decision-tree learning algorithm (Section 3) that builds a different tree for each test example. We apply the same test process above in the lazy tree, and call it Lazy-tree Optimal Sequential Test (LazyOST). Note that this approach is "optimal" by the nature of the decision tree built to minimize the total cost; that is, subtrees are built because there is a cost reduction in the training data. Therefore, the tree's suggestions for tests will also result in minimum total cost. (Note the terms such as "optimal" and "minimum" used in this paper do not mean in the absolute and global sense. As in C4.5, the tree building algorithm and test strategies use heuristics which are only locally optimal).

Note that it is not obvious that this lazy-tree Optimal Sequential (LazyOST) Test should always produce a small total cost compared to the single-tree OST. This is because in both approaches, the test costs of the known attributes do not count during the classifying of a test example. However, when we build decision tree specifically for a test example, the tree minimizes the total cost without counting the known attributes in the training data. This would produce a smaller total cost for that test example. In contrast, in the single tree approach, only one tree is built for all test examples, and specific information about known and unknown attributes in each test example is not utilized. In Section 4.1.2 we will compare LazyOST and OST on ten real-world datasets to see which one is better in terms of having a smaller total cost.

Case Study on Heart Disease Continued. Continuing on the heart-disease example, we next choose a test example with most attribute values known from the dataset, as the known values serve as the test results. The discretized attribute values for this test case are: age=1, sex=2, cp=3, trestbps=1, chol=1, fbs=1, restecg=1, thalach=1, exang=2, oldpeak=2, slope=1, ca=?, thal=2, and class=0 (a negative case). We apply LazyOST on the tree in Figure 1. Again assuming all values are unknown, LazyOST requests the sequence of tests as: cp (=3), fbs (=1), thal (=2), and restecg (=1), with a total test cost of $124.60. The prediction of the tree is 0 (correct), thus the misclassification cost is 0. Therefore, the total cost for this test case is $124.60.

Comparing Sequential Test Strategies. To compare various sequential test strategies, we choose 10 real-world datasets, listed in Table 1, from the UCI Machine Learning Repository [1]. These datasets are chosen because they are binary class, have at least some discrete attributes, and have a good number of examples. Each dataset is split into two parts: the training set (60%) and the test set (40%). Unlike the case study of heart disease, the detailed test costs of these datasets are unknown. To make the comparison possible, we simply choose randomly the test costs of all attributes to be some values between 0 and 100. This is reasonable because we compare the relative performance of all test strategies under the same chosen costs. The misclassification cost is set to 200/600 (200 for false positive and 600 for false negative). For test examples, a certain ratio of attributes (0.2, 0.4, 0.6, 0.8, and 1) are randomly selected and marked as unknown to simulate test cases with various degrees of missing values. Three Sequential Test strategies, OST [8], LazyOST (our work), and CSNB [2] are compared. We repeat this process 25 times, and the average total costs for the 10 datasets are plotted in Figure 2.

Table 1. Datasets used in the experiments

	No. of Attributes	No. of Examples	Class dist. (N/P)
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
Tic-tac-	9	958	332/626
Mushroom	21	8124	4208/3916
Kr-vs-kp	36	3196	1527/1669
Voting	16	232	108/124
Cars	6	446	328/118

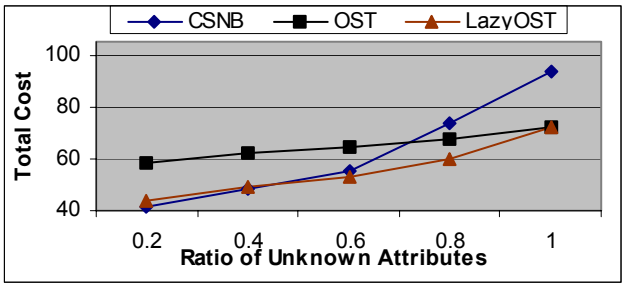

Fig. 2. Comparing our new Sequential Test strategy LazyOST with CSNB and OST

We can make several interesting conclusions. First, we can see clearly that LazyOST outperforms OST on all 10 datasets under every unknown attribute ratio, except 1. When all attributes are unknown, the eager and lazy tree learners produce the same tree. Second, the difference between OST and LazyOST is larger at a lower ratio of unknown attributes compared to a higher ratio. This is because when the ratio is low, most attributes are known, and LazyOST takes advantages of these known attributes for individual test examples while OST does not. This confirms our early expectation that our new lazy trees learning algorithm produces a tree with smaller total costs compared to the previous single tree approach. Last, we also see that the CSNB [2] performs better than OST when the ratio of unknown attributes is less than 0.7 (confirming results in [2]), since CSNB has a lower misclassification cost than OST with lower ratios of unknown attributes. However, LazyOST performs best among the three strategies when the ratio of unknown attributes is greater than 0.3.

5.2 Single Batch Tests

The Sequential Test Strategies discussed in the previous section have to wait for the result of each test to determine which test will be the next one. Waiting not only agonizes

the patient in medical diagnosis, it may also be life threatening if the disease is not diagnosed and treated promptly. Thus doctors normally order one set (batch) of tests to be done at once. This is the case of the Single Batch Test. Note that results of the tests in the batch can only be obtained simultaneously after the batch is determined.

In [8] a very simple heuristic is described. The basic idea is that when a test example is classified by a minimum-cost tree and is stopped by the first attribute whose value is unknown in the test case, all unknown attributes under and including this first attribute would be tested, as a single batch. Clearly, this strategy would have exactly the same misclassification cost as the Optimal Sequential Test, but the total test cost is higher as extra tests are performed. We call this strategy Naïve Single Batch (NSB).

We propose two new and more sophisticated Single Batch Test strategies, and discuss their strengths and weaknesses. We will show experimentally that they are better than the Naïve Single Batch and the single batch based on naïve Bayes [2].

Greedy Single Batch (GSB). The rationale behind GSB is to find the most likely leaf (the most typical case) that the test example may fall into, and collect the tests on the path to this leaf for the batch test (to "confirm" the case). More specifically, it first locates all "reachable" leaves under the first unknown attribute (let us call it u) when the test example is classified by the tree. Reachable leaves are the leaves that can be possibly reached from u given the values of known attributes and all possible values of the unknown attributes under u. Then a reachable leaf with the maximum number of training examples is located, and the unknown attributes on the path from u to this leaf are collected as the batch of tests to be performed.

Intuitively this strategy reduces the total test cost than the Naïve Single Batch as only a subset of the tests is performed. However, it may increase the misclassification costs compared to the Optimal Sequential Test, as the greedy "guesses" may not be correct, in which case the test example will not reach a leaf, and must be classified by an internal node in the decision tree, which is usually less accurate than a leaf node. This will incur a higher misclassification cost.

Optimal Single Batch (OSB). The Optimal Single Batch (OSB) seeks a set of tests to be performed such that the sum of the test costs and expected misclassification cost after those tests are done is optimal (minimal). Intuitively, it finds the expected cost reduction for each unknown attribute (test), and adds a test to the batch if the cost reduction is positive and maximum (among other tests). More specifically, when a test example is classified by the tree, and is stopped by the first unknown attribute u in the tree, the total expected cost *misc(u)* can be calculated. At this point, *misc(u)* is simply the expected misclassification cost of u, and there is no test cost. If u is tested at a cost C, then the test example is split according to the percentage of training examples that belong to different attribute values, and is duplicated and distributed into different branches of the tree (as we do not know *u's* value since this is a batch test), until it reaches some leaves, or is stopped by other unknown attributes. For each such reachable leaf or unknown attribute, the expected cost can be calculated again, and the weighted misclassification cost can be obtained (let us call it S). The sum of C and S is then the expected cost if u is tested, and the difference between *misc(u)* and $C+S$ is the cost reduction $E(u)$ if u is tested. If such a cost reduction is positive, then u is put into the batch of tests. Then from the current set of reachable unknown attributes, a node with the maximum positive cost reduction is

chosen, and it is added into the current batch of tests. This process is continued until the maximum cost reduction is no longer greater than 0, or there is no reachable unknown attributes (all unknown attributes under *u* are in the batch, reducing to Naïve Single Batch). The batch of tests is then discovered. The pseudo-code of OSB is shown here.

In the pseudo-code, *misc(.)* is the expected misclassification cost of a node, *c(.)* is the test cost of an attribute, *R(.)* is all reachable unknown nodes and leaves under a node, and *p(.)* is the probability (estimated by ratios in the training data) that a node is reached. Therefore, the formula *E(i)* in the pseudo-code calculates the cost difference between no test at *i* (so only misclassification cost at *i*) and after testing *i* (the test cost plus the weighted sum of misclassification costs of reachable nodes under i). That is, *E(i)* is the expected cost reduction if *i* is tested. Then the node t with the maximum cost reduction is found, and if such reduction is positive, *t* should be tested in the batch. Thus, *t* is removed from *L* and added into the batch list *B*, and all reachable unknown nodes or leaves of *t*, represented by the function *r(t)*, is added into *L* for further consideration. This process continues until there is no positive cost reduction or there is no unknown nodes to be considered (i.e., *L* is empty). The time complexity is linear to the size of the tree, as each node is considered only once.

L = empty /* list of reachable and unknown attributes */
B = empty /* the batch of tests */
u = the first unknown attribute when classifying a test case
Add *u* into *L*
Loop
  For each $i \in L$, calculate *E(i)*:
    $E(i) = misc(i) - [c(i) + \sum p(R(i)) \times misc(R(i))]$
  E(t) = max E(i) /* *t* has the maximum cost reduction */
  If *E(t)* > 0 then add *t* into *B*, delete *t* from *L*, add *r(t)* into *L*
    else exit Loop /* No positive cost reduction */
Until *L* is empty
Output *B* as the batch of tests

Comparing the two new single batch strategies, Greedy Single Batch (GSB) is simple and intuitive; it finds the most likely situation (leaf) and requests tests to "confirm" it. The time complexity is linear to the depth of the tree. It works well if there is a reachable leaf with a large number of training examples. The time complexity of the Optimal Single Batch (OSB) is linear to the size of the tree, but it is expected to have a smaller total cost than GSB. Both GSB and OSB may suggest tests that may be wasted, and test examples may not fall into a leaf.

Case Study on Heart Disease Continued. We apply GSB and OSB on the same test case as in Section 4.1.1 with the decision tree in Figure 1. The GSB suggests the (single) batch of (cp, and thal), while the OSB suggests the single batch of (cp, sex, slope, fbs, thal, age, chol, and restecg) to be tested. With both GSB and OSB, the test case does not go into a leaf, and some tests are wasted. The test cost is $103.9 for GSB and $221.17 for OSB, while the misclassification costs are 0 for both GSB and OSB. Thus, the total cost

for the test case is $103.9 and $221.17 for GSB and OSB respectively. Note that we cannot conclude here GSB is better than OSB as this is only for a test case.

Comparing Single Batch Test Strategies. We use the same experiment procedure on the same 10 datasets to compare various Single Batch Test strategies including CSNB-SB [2]. The misclassification costs are set to 2000/6000. These costs are set to larger values so the trees will be larger to show more clearly the effect of batch tests. The total costs for the 10 datasets are compared and the average total costs for the 10 datasets are plotted in Figure 3.

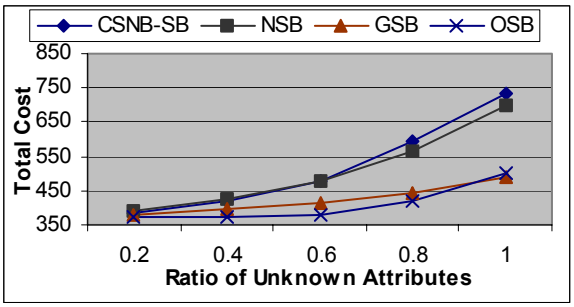

Fig. 3. Comparing our new Single Batch Test strategies GSB and OSB CSNB-SB and NSB

From Figure 3 we can clearly see that Optimal Single Batch (OSB) performs the best among other single batch test strategies. When the ratio of missing attributes is relatively small (0.2), the three tree-based single batch test strategies are similar, as very few attributes would need to be tested. When the ratio of missing attributes increases, the differences become more evident, especially between Naïve Single Batch and Greedy Single Batch. All the tree-based single batch strategies perform better than the single batch with naïve Bayes. The reason is that the structure of the decision tree is utilized when deciding the single batch, while naïve Bayes has no such structure to rely on.

6 Conclusions and Future Work

In this paper, we present a lazy decision tree learning algorithm to minimize the total cost of misclassifications and tests. We then design two categories of test strategies: Sequential Test and Single Batch Test, to determine which unknown attributes should be tested, and in what order, to minimize the total cost of tests and misclassifications. We evaluate the performance (in terms of the total cost) empirically, compared to previous methods using a single decision tree and naïve Bayes. The results show that the new test strategies, Lazy-tree Optimal Sequential Test, and Optimal Single Batch, work best in the corresponding categories. The time complexity of these new test strategies is linear to the tree depth or the tree size, making them efficient for testing a large number of test cases.

These strategies can be readily applied to large datasets in the real world. A detailed case study on heart disease is given in the paper.

In our future work we plan to continue to work with medical doctors to apply our algorithms to medical data with real costs. We also plan to consider discounts when groups of tests are ordered at the same time, and to incorporate other types of costs in our decision tree learning and test strategies.

References

1. Blake, C.L., and Merz, C.J. 1998. *UCI Repository of machine learning databases (website)*. Irvine, CA: University of California.
2. Chai, X., Deng, L., Yang, Q., and Ling,C.X.. 2004. Test-Cost Sensitive Naïve Bayesian Classification. *In Proceedings of the Fourth IEEE International Conference on Data Mining.* Brighton, UK : IEEE Computer Society Press.
3. Chawla,N.V., Japkowicz, N., and Kolcz, A. eds. 2004. *Special Issue on Learning from Imbalanced Datasets. SIGKDD*, 6(1): ACM Press.
4. Domingos, P. 1999. MetaCost: A General Method for Making Classifiers Cost-Sensitive. *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 155-164. San Diego, CA: ACM Press.
5. Elkan, C. 2001. The Foundations of Cost-Sensitive Learning. *In Proceedings of the Seventeenth International Joint Conference of Artificial Intelligence*, 973-978. Seattle, Washington: Morgan Kaufmann.
6. Fayyad, U.M., and 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*, 1022-1027. France: Morgan Kaufmann.
7. Ting, K.M. 1998. Inducing Cost-Sensitive Trees via Instance Weighting. *In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery,* 23-26. Springer-Verlag.
8. Ling, C.X., Yang, Q., Wang, J., and Zhang, S. 2004. Decision Trees with Minimal Costs. *In Proceedings of the Twenty-First International Conference on Machine Learning,* Banff, Alberta: Morgan Kaufmann.
9. Lizotte, D., Madani, O., and Greiner R. 2003. Budgeted Learning of Naïve-Bayes Classifiers. *In Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*. Acapulco, Mexico: Morgan Kaufmann.
10. Quinlan, J.R. eds. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.
11. Tan, M. 1993. Cost-sensitive learning of classification knowledge and its applications in robotics. *Machine Learning Journal,* 13:7-33.
12. Turney, P.D. 1995. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* 2:369-409.
13. Turney, P.D. 2000. Types of cost in inductive concept learning. *In Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, Stanford University, California.
14. Zubek, V.B., and Dietterich, T. 2002. Pruning improves heuristic search for cost-sensitive learning. *In Proceedings of the Nineteenth International Conference of Machine Learning,* 27-35, Sydney, Australia: Morgan Kaufmann.