Test Strategies for Cost-Sensitive Decision Trees

Charles X. Ling, Victor S. Sheng, and Qiang Yang, *Senior Member*, *IEEE*

Abstract—In medical diagnosis, doctors must often determine what medical tests (e.g., X-ray and blood tests) should be ordered for a patient to minimize the total cost of medical tests and misdiagnosis. In this paper, we design cost-sensitive machine learning algorithms to model this learning and diagnosis process. Medical tests are like attributes in machine learning whose values may be obtained at a cost (attribute cost), and misdiagnoses are like misclassifications which may also incur a cost (misclassification cost). We first propose a lazy decision tree learning algorithm that minimizes the sum of attribute costs and misclassification costs. Then, we design several novel "test strategies" that can request to obtain values of unknown attributes at a cost (similar to doctors' ordering of medical tests at a cost) in order to minimize the total cost for test examples (new patients). These test strategies correspond to different situations in real-world diagnoses. We empirically evaluate these test strategies, and show that they are effective and outperform previous methods. Our results can be readily applied to real-world diagnosis tasks. A case study on heart disease is given throughout the paper.

Index Terms—Induction, concept learning, mining methods and algorithms, classification.

1 Introduction

Inductive learning techniques have had great success in building classifiers and classifying test examples into classes with a high accuracy or low error rate. However, in many real-world applications, reducing misclassification errors is not the final objective since different errors can cost quite differently. This type of learning is called cost-sensitive learning. Turney [18] surveys a whole range of costs in cost-sensitive learning, among which two types of costs are most important: misclassification costs and attribute costs. (Turney used "test costs." To reduce overuse of the word "test," we use "attribute cost" throughout this paper.) For example, in a binary classification task, the cost of false positive (FP) and the cost of false negative (FN) are often very different. In addition, attributes (similar to medical tests) may have different costs, and acquiring values of attributes may also incur costs. The goal of learning in this paper is to minimize the sum of the misclassification costs and the attribute costs.

Tasks involving both misclassification and attribute costs are abundant in real-world applications. In medical diagnosis, medical tests are like attributes in machine learning whose values may be obtained at a cost (attribute cost), and misdiagnoses are like misclassifications which may also bear a cost (misclassification cost). When building a classification model for medical diagnosis from the training data, we must consider both the attribute costs (medical tests such as blood tests) and misclassification costs (errors in the diagnosis). Further, when a doctor sees a new patient (a test example), additional medical tests may be 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). We use the term "test strategy" to describe a process that allows the learning algorithm to obtain attribute values at a cost when classifying test examples. The goal of the test strategies in this paper is to minimize the sum of attribute costs and misclassification costs, similar to doctors' goal to minimize the total cost to the patients (or the whole medical system). A case study on heart disease is given in the paper.

In this paper, we first propose a lazy-tree learning that improves on a previous decision tree algorithm [8] that minimizes the sum of misclassification costs and attribute costs. We then describe several novel test strategies to determine what values of unknown attributes should be obtained, and at what order, such that the total expected cost is minimum. Extensive experiments have been conducted to show the effectiveness of our tree building algorithms and the new test strategies compared to previous methods.

The rest of paper is organized as follows: We first review the related work in Section 2. Then, we describe a tree-building algorithm that is improved from a previous work in Section 3. In Section 4, we propose several novel test strategies when the tree is used to predict new test cases with unknown attributes, and compare these strategies on 13 data sets. Throughout these sections, we provide a case study on a real-world data set of the heart disease. Finally, we conclude the work in Section 5.

2 Review of Previous Work

Gorry and Barnett [7] suggested a "myopic approach" to request more information during decision making, but the

- *C.X. Ling and V.S. Sheng are with the Department of Computer Science, The University of Western Ontario, London, Ontario N6A 5B7, Canada. E-mail: {cling, ssheng}@csd.uwo.ca.*
- *Q. Yang is with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong. E-mail: qyang@cs.ust.hk.*

Manuscript received 18 May 2005; revised 27 Oct. 2005; accepted 23 Jan. 2006; published online 19 June 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0196-0505.

method focuses only on prediction accuracy; there is no cost involved in getting the information. Cost-sensitive learning has received extensive attentions in recent years. Turney [18] analyzes a variety of costs in machine learning, such as misclassification costs, attribute costs, active learning costs, computation cost, human-computer interaction cost, etc. Two types of costs are singled out as the most important in machine learning: misclassification costs and attribute costs. Much work has been done in considering nonuniform misclassification costs (alone), such as [3], [4], [19], and [16]. Those works can often be used to solve the problem of learning with imbalanced data sets [1]. Some previous work, such as [12] and [15], considers the attribute costs alone without incorporating misclassification costs. As pointed out by [18], it is obviously an oversight. Melville et al. [10], [11] studied how to achieve a desired model accuracy by acquiring missing values in identified training examples with minimum cost. However, they do not minimize the total cost (misclassification and attribute costs) of test examples with missing values. As far as we know, the only work considering both misclassification and attribute costs includes [17], [20], [9], [8], [14], and [2]. We discuss these works in detail below.

In [20], 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. While related to our work, their research adopts an optimal search strategy, which may incur very high computational cost to conduct the search. In contrast, we adopt heuristics and local search similar to [13] using a polynomial time algorithm to build a new decision tree, and our test strategies are also polynomial to the tree size. Our results may be only locally optimal. Lizotte et al. [9] studied the theoretical aspects of active learning with attribute costs using a PAC learning framework, which models how to use a budget to collect the relevant information for the real-world applications with no actual data at beginning. Our algorithm builds a model from historical data to minimize the sum of misclassification costs and attribute costs for a new case with missing attribute values. Turney [17] presented a system called ICET, which uses a genetic algorithm to build a decision tree to minimize the sum of attribute costs and misclassification costs. Our algorithm adopts the same decision-tree building framework as in [13], and it is expected to be more efficient than Turney's genetic algorithm-based approach. More specifically, ICET [17] is a two-tiered search strategy. On the lower tier, EG2 is applied to build the individual decision tree. On the upper tier, ICET uses a genetic algorithm (GENESIS) to evolve a population of 50 decision trees which have different biases. In total, there are 1,000 trials in ICET. We only need to build one decision tree for test examples with the same subset of missing values (see Section 3.1.1). It is obvious that our algorithm is much faster than ICET. In addition, the individual decision tree is ICET is based on EG2, which does not integrate misclassification costs and attribute costs directly. Our decision tree integrates the two costs together naturally. ICET considers two kinds of tests: "immediate" tests and "delayed" tests. Our algorithm introduces the "waiting cost" (see Section 4.3), which quantifies the delay. Finally, the test strategy of ICET is similar to our sequential test strategy, while we have also designed single batch test strategies (Section 4.2) and the multiple batch test strategies (Section 4.3).

Ling et al. [8] proposed a new decision tree learning program that uses minimum total cost of attributes and misclassifications as the attribute split criterion. They also propose several simple test strategies to handle the missing attributes in the test data, and compare their results to C4.5. However, in their method, only a single tree is built for all test examples. Our work is an extension of [8] and [14] as we propose a new lazy decision tree algorithm that builds different decision trees for different test examples to utilize as much information in the known attributes as possible (Section 3.1.1). We also incorporate discounts in attribute costs when tests are performed together (Section 3.1.3). We show that the same sequential test strategy produces smaller total cost compared to [8] (see Section 4.1.2). In addition, we propose several novel single batch strategies (Section 4.2) and multiple batch strategies (Section 4.3) that, as far as we know, have not been published previously.

Chai et al. proposed a naïve Bayesian-based cost-sensitive learning algorithm, called CSNB [2], which reduces the total cost of attributes and misclassifications. They also propose a single batch test strategy that may obtain several attribute values simultaneously. Our single batch strategies utilize the tree structure when deciding the batch of attribute values to be obtained at the same time. Experiments show that our tree-based test strategies outperform CSNB in terms of the total cost of misclassifications and attributes in many cases (Section 4.1.2).

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, and attribute costs. Ling et al. [8] describes a C4.5-like decision tree learning algorithm, which uses the minimal total cost (or maximum cost reduction) as a tree-split criterion, similar to maximum information gain ratio. More specifically, given a set of training examples, a cost metric of misclassifications, and attribute costs, the algorithm calculates the misclassification cost without splits. Then, for each attribute, the sum of the attribute cost and misclassification costs after the split is calculated. That the previous one subtracts the later one is the cost reduction. The algorithm chooses an attribute with the maximum cost reduction (equivalent to minimal total cost after split) to split the training examples. This process is applied recursively to build subtrees. With this method, a single decision tree is built based on the training examples, and the tree is used for predicting all test examples.

3.1 An Improved Algorithm for Constructing Cost-Sensitive Decision Trees

We have designed and implemented three improvements over [8] in constructing cost-sensitive decision trees.

3.1.1 Improvement 1: Lazy Tree

Instead of building a single decision tree for all test examples as in [8], we propose a lazy-tree approach [6] 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 cost of the known attributes to be \$0 while the cost of the unknown attributes remains unchanged (see line $1(i)$ of *LazyTree* in the pseudocode). For example, suppose that there are three attributes and their costs are \$30, \$40, and \$60, respectively. If in a test example, the second attribute value is unknown, then the new attribute costs would be reset to \$0, \$40, and \$0, respectively. Then, a cost-sensitive tree is built using [8]. As the attribute costs of the known attributes are reset as \$0, the first and third attributes most possibly appear at the top of the tree. Although this is a small change, we expect that it will reduce the total cost in testing significantly. The rationale is that attributes with the zero attribute cost are more likely to be chosen early during the tree building process. When a test example is classified by this decision tree, it is less likely to encounter unknown attributes (the second attribute in above example) near the top of the tree. This tends to reduce the total attribute cost, and thus the total cost, as to be shown in the experiments in Section 4.1.2.

Clearly, our lazy-tree approach builds different trees for test examples with different sets of unknown attributes, as attribute costs are set differently. One weakness of our method is higher computational cost associated with lazy learning because a tree is specifically built for each test example. However, our tree-building process has the same time complexity as C4.5, so it is quite efficient (see Section 4.1.3 for comparing running time of various algorithms). In addition, lazy trees for the same set of known attributes are the same, and they can be saved for predicting future examples with the same set of known attributes. That is, trees frequently used can be stored in memory for speedup.

3.1.2 Improvement 2: Expected Cost Calculation

Another improvement we made over [8] is that we use the *expected* total misclassification cost when selecting attributes for splitting to produce trees with a smaller total cost (see lines 3 and 4 of CSDT in the pseudocode). This improved split criterion tends to produce a more accurate split and build a tree with small total cost (see Section 4.1.2). For a subset of examples, if $C_P = tp \times TP + fp \times FP$ is the total misclassification cost of being a positive leaf, and $C_N = tn \times TN + fn \times FN$ 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 a positive leaf is: $1 - \frac{C_P}{C_P+C_N} = \frac{C_N}{C_P+C_N}$, and 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: $E_N = \frac{C_P}{C_P+C_N} \times C_N$.[1] 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 cost of testing examples on attribute A. After calculating the expected cost reduction for all attributes, it is easy to find out which one is the largest. If the largest expected cost reduction is greater than 0, then the attribute is chosen as the split (otherwise, it is not worth to build the tree further, and a leaf is returned). Section 4.1.2 compares the improved tree algorithm with the original one [8].

1. The expected misclassification cost E_P of being a positive leaf for a given set of examples is indeed the same as the expected misclassification cost E_N of being a negative leaf for the set of examples. This is possible as both are expected costs.

3.1.3 Improvement 3: Considering Group Discount

A third improvement is that we have incorporated possible *discounts* in obtaining values of a group of attributes with missing values in the tree building process. This is a special case of conditional attribute costs [17], which allow attribute costs to vary with the choice of prior tests. Often, medical tests are not performed independently, and when certain medical tests are performed together, it is cheaper to do these tests in a group than individually. For example, the cholesterol test and fasting blood sugar attribute cost \$7.27 and \$5.20 individually. As both are blood tests, if both are performed together, it would be cheaper to do the second test. In general, attributes can be partitioned into groups, and each group has a particular discount amount. When the first attribute in a group is called for, the attribute cost is the original cost (the full cost). However, if any additional attributes in the same group are requested for their values, their costs would be the original (full) costs minus the discounted cost. In implementing the cost-sensitive decision-tree building process, if an attribute in a group is selected as a split attribute, the costs of other attributes in the group are simultaneously reduced by the discount amount for the future tree-building process (see line 5 of *CSDT* in the pseudocode). As the attribute costs are discounted, the tests in the same group would more likely be picked as the next node in the future tree building—a property that is desirable for the cost-sensitive trees which prefer attributes with lower costs [8]. An example of group discounts will be given in the next section.

The pseudocode for the improved cost-sensitive learning algorithm is given below:

LazyTree(Examples, Attributes, TestCosts, testExample)

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

CSDT(Examples, Attributes, TestCosts)

1. If all examples are positive/negative, return *root*
2. If maximum expected cost reduction <0, return root
3. Let A be an attribute with maximum expected cost reduction
4. *root* $\leftarrow A$
5. Update *TestCosts* if discount applies
6. For each possible value vi of the attribute A
   i) Add a new branch $A = vi$ below *root*
   ii) Segment the training examples *Example_vi* into the new branch

TABLE 1
Attribute Costs (in $) and Group Discounts for Heart Diseases

Tests (attributes)	Description	Individual Costs	Group A discount	Group B discount	Group C discount
age	age of the patient	$1.00			
sex	sex	$1.00			
cp	chest pain type	$1.00			
trestbps	resting blood pressure	$1.00			
chol	serum cholesterol in mg/dl	$7.27	$2.10		
fbs	fasting blood sugar	$5.20	$2.10		
restecg	resting electrocardiography results	$15.50			
thalach	maximum heart rate achieved	$102.90		$101.90	
exang	exercise induced angina	$87.30			$86.30
oldpeak	ST depression induced by exercise	$87.30			$86.30
slope	slope of the peak exercise ST segment	$87.30			$86.30
ca	number of major vessels colored by fluoroscopy	$100.90			
thal	finishing heart rate	$102.90		$101.90	

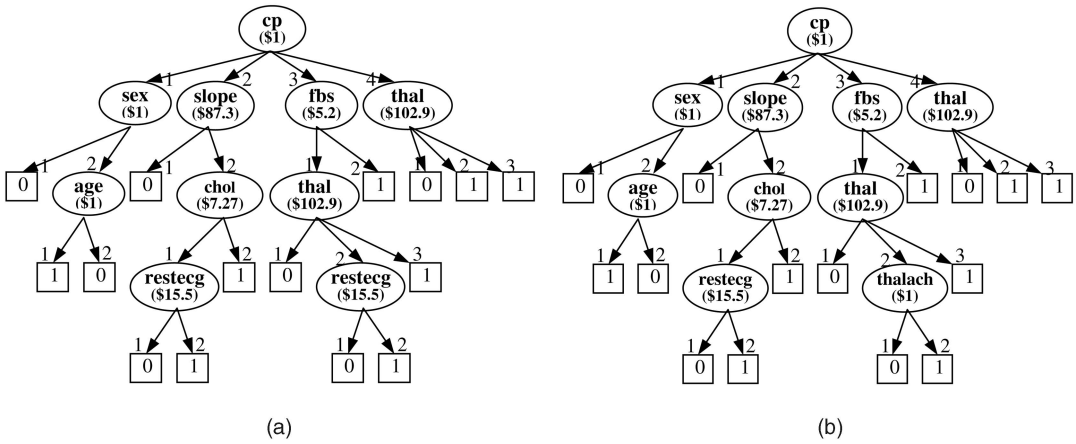

Fig. 1. Lazy trees for the test case with missing values for all attributes.

iii) Call *CSDT(examples_vi, Attributes-A, TestCosts)* to build subtree

3.2 A Case Study on Heart Disease

We apply our lazy decision-tree learning on a real application example that involves the diagnosis of the Heart Disease, where the attribute costs are obtained from medical experts and insurance programs. The data set was used in the cost-sensitive genetic algorithm by [17]. The learning problem is to predict the coronary artery disease from the 13 noninvasive tests on patients, as listed in Table 1. The attributes on patients profile, such age, sex, etc., are also regarded as "tests" with a very low cost (such as $1) to obtain their values. The costs of the 13 noninvasive tests are in Canadian dollars ($), and were obtained from the Ontario Health Insurance Program's fee schedule [17]. These individual tests and their costs are also listed in Table 1. Tests such as 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 are discounted in costs, and Table 1 also lists these groups and the discount amount of each group. Each patient is classified into two classes: the class label 0 or negative class indicates a less than 50 percent of artery narrowing, and 1 indicates more than 50 percent. There are a total of 294 cases in the data set, with 36.1 percent positive cases (106 positive cases).

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 more 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 attribute costs about $600. Thus, it seems reasonable to assign false positive and false negative to be $600 and $1,000, respectively.

Assuming that in a new test example all attribute values are missing (as seeing a completely new patient), the original attribute costs given in Table 1 are used directly for decision-tree building. The numerical attributes in data sets are discretized into integers (1, 2, ...) using the minimal entropy method of Fayyad and Irani in [5]. We apply our lazy decision tree learning for this test case, and obtain two decision trees shown in Fig. 1. Fig. 1a is the tree without considering group discounts, while Fig. 1b is the tree considering group discounts (as in Table 1).

We can easily see that the two trees are very similar, except for one test, thalach, which replaces the test restecg in Fig. 1a. Since tests thal and thalach belong to the same group, after thal is tested at the full cost, thalach costs only $1 and is selected after thal when group discounts are considered. This would clearly reduce the total cost when the tree in Fig. 1b classifies test examples. Also, we can see that in general less expensive tests (attributes) are used in the top part of the trees. The splitting criterion selects attributes according to their relative merit of reducing the total cost. When these trees are presented to the Heart-Failure researcher, he thinks that they are quite reasonable in predicting artery narrowing.

TABLE 2
The Features of 13 Data Sets 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-toe	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
Thyroid_i	24	1939	1762/167
Kr-vs-kp_i	36	1661	1527/134
Heart-D	13	294	188/106

4 Three Categories of Test Strategies

In this paper, a "test strategy" is a method to classify a test example from a cost-sensitive decision tree during which it is possible to "perform tests," that is, to obtain values of unknown attributes, at a cost (attribute costs). It mimics the diagnosis process of doctors in which additional medical tests may be requested at a cost to help their diagnoses (predicting the disease of the patients).

We define and study three categories of test strategies: Sequential Test (Section 4.1), Single Batch Test (Section 4.2), and Multiple Batch Test (Section 4.3). For a given test example with unknown attributes, the Sequential Test can request only one test (attribute value) 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 Multiple Batch Test can request a series of batches of tests, each after the results of the previous batch are known, before making a final prediction. Clearly, the Multiple Batch Test is most general, as the other two are special cases of it—Sequential Test is when each batch only contains one test, and Single Batch is when the number of batches can only be one.

The three categories of test strategies correspond well to different situations in real-world medical diagnosis. For example, when seeing a new patient, doctors often order one set of tests (at a cost) to be done together. This is the case of the Single Batch Test. Sometimes doctors may order a second (and a third) set of tests to be done, based on the results of the previous set of tests. This is the case of the Multiple 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. The goal of these test strategies again is to minimize the total cost of tests (attributes) requested and the misclassification costs associated with the final prediction. In the next three sections, the three types of test strategies will be discussed in great details.

4.1 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 requested and 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 paper, 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 attribute 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 11 real-world data sets and two generated data sets (Table 2) to see which one is better in terms of having a smaller total cost.

4.1.1 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 data set, as the known values serve as the test results. The discretized attribute values for this test case are: age = 1,

TABLE 3
The Total Cost (in $) for Our LazyOST Compared to Previous Strategies (CSNB and OST)

Dataset Name	Strategy	Unknown Attribute Ratio				
		0.2	0.4	0.6	0.8	1.0
Ecoli	CSNB	28.0	31.3	33.6	63.6	84.8
	OST	45.7	45.8	45.8	54.8	54.9
	LazyOST	11.9	11.9	11.9	10.7	27.2
Breast	CSNB	8.7	10.1	13.7	28.6	56.3
	OST	35.0	38.1	44.3	45.1	51.0
	LazyOST	22.9	22.5	27.4	27.7	40.4
Heart	CSNB	61.7	66.0	76.5	83.3	92.4
	OST	80.0	80.0	80.7	81.4	81.4
	LazyOST	69.7	72.1	66.4	61.1	95.5
Thyroid	CSNB	18.2	30.2	35.1	49.2	80.0
	OST	15.5	25.2	25.8	30.7	40.3
	LazyOST	12.6	12.5	12.6	14.4	60.8
Australia	CSNB	49.2	58.9	83.7	104.8	123.8
	OST	85.2	86.4	88.5	89.8	90.3
	LazyOST	46.2	46.4	49.2	76.5	82.8
Tic-tac-toe	CSNB	69.1	72.1	76.5	78.9	83.7
	OST	70.8	70.8	70.8	70.8	70.8
	LazyOST	52.4	52.4	64.4	69.4	71.5
Mushroom	CSNB	16.9	18.0	19.7	22.7	49.5
	OST	15.8	23.6	23.6	23.7	47.0
	LazyOST	0.1	0.1	0.2	0.5	58.9
Kr-vs-kp	CSNB	47.4	73.6	82.4	145.8	163.8
	OST	95.9	95.9	95.9	95.9	95.9
	LazyOST	8.6	9.8	11.3	17.6	97.4
Voting	CSNB	42.3	51.1	52.9	60.2	73.7
	OST	32.0	45.0	51.9	53.1	53.5
	LazyOST	21.9	22.2	26.8	26.8	28.8
Cars	CSNB	73.8	75.7	75.8	100.4	125.7
	OST	105.3	107.4	108.0	116.9	121.9
	LazyOST	46.5	46.5	45.3	54.1	91.1
Thyroid_i	CSNB	39.4	93.9	112.3	161.2	267.6
	OST	34.1	60.4	61.2	74.4	100.5
	LazyOST	36.2	35.9	36.3	40.2	108.9
Kr-vs-kp_i	CSNB	11.5	13.9	20.4	100.2	217.4
	OST	12.9	12.9	12.9	38.8	64.7
	LazyOST	0	0	0	2.0	50.0
Heart-D	CSNB	139.2	143.8	141.8	142.0	236.9
	OST	171.7	172.0	172.0	172.4	167.2
	LazyOST	151.6	154.1	154.9	153.3	202.1

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 Fig. 1b which considers the group discount to the test case. Again assuming all values are unknown, LazyOST requests the sequence of tests as: cp (= 3), fbs (= 1), thal (= 2), and thalach (= 1), with a total attribute cost of $110.10. The prediction of the tree is 0 (it is correct), thus the misclassification cost is $0. Therefore, the total cost for this test case is $110.10. However, if the test example is classified by the tree in Fig. 1a (group discount is not considered), the test restecg will be requested instead of the much cheaper thalach. In this case, the total cost would be $124.60, higher than the tree that considers the group discount.

4.1.2 Comparing Total Cost for Sequential Test Strategies

To compare the overall performance of various sequential test strategies, we choose 10 real-world data sets, listed in Table 2, from the UCI Machine Learning Repository [1]. These data sets are chosen because they are binary class, have at least some discrete attributes, and have a good number of examples. To create data sets with more imbalanced class distribution, two data sets (thyroid and kr-vs-kp) are resampled to create a small percentage of positive examples. They are called thyroid_i and kr-vs-kp_ik, respectively. Each data set is split into two parts: the training set (60 percent) and the test set (40 percent). Unlike the case study of heart disease, the attribute costs and misclassification costs of these data sets are unknown. To make the comparison possible, we simply assign certain values as Canadian dollars (or any other unit) for these costs. We randomly choose the attribute 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 FP/FN is set to $200/$600 for the more balanced data sets (the minority class is greater than 10 percent) and $200/$3000 for the imbalanced data sets (the minority class is less than 10 percent as in thyroid_i and kr-vs-kp_i). As the group discount of attributes is unknown, all attributes are assumed to be independent in their attribute costs. To make the comparison complete, the heart disease data set used in the case study (called Heart-D at the bottom of Table 2) is also added in the comparison (with its own attribute costs). 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 (this work), and CSNB [2] are compared in Table 3. The average total costs (in $) for the 13 data sets are plotted in Fig. 2.

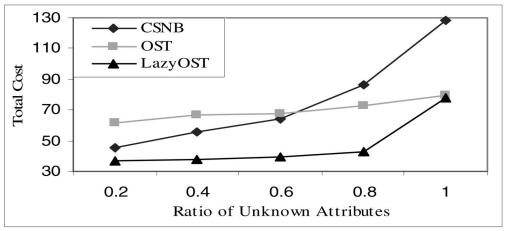

Fig. 2. Comparing the average total cost (in $) of our new Sequential Test strategy LazyOST to previous methods CSNB and OST. The smaller the total cost, the better.

We can make several interesting conclusions. First, we can see clearly that LazyOST outperforms OST on all the 13 data sets (balanced or imbalanced) under almost all unknown attribute ratios. Note that OST, LazyOST, and CSNB are all heuristic algorithms and, therefore, occasionally, LazyOST may not be better than OST or CSNB. When all attributes are unknown, the eager and lazy tree learners produce the same tree, as expected. 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 advantage 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 slightly better than OST when the ratio of unknown attributes is less than about 0.6 (confirming results in [2]) since CSNB has a lower misclassification

TABLE 4
The Total Cost (in $) for LazyOST, OST, and OOST on the Data Set Kr-vs-kp

Strategy	Unknown Attribute Ratio				
	0.2	0.4	0.6	0.8	1
OST	95.9	95.9	95.9	95.9	95.9
OST + expected	26.7	26.7	26.7	30.3	97.4
LazyOST	8.6	9.8	11.3	17.6	97.4

TABLE 5
The Average Running Time (in Milliseconds) of a Test Example in Ecoli

	Ratios of Unknown Attributes				
	0.2	0.4	0.6	0.8	1.0
CSNB	1.29	1.93	2.74	5.02	6.51
OST	0.02	0.02	0.02	0.02	0.02
LazyOST	0.04	0.05	0.05	0.05	0.05

cost than OST with lower ratios of unknown attributes. However, LazyOST performs best among the three strategies on average, as witnessed by Fig. 2.

We can obtain from Table 3 that LazyOST outperforms OST by 32.8 percent on average over all data sets used in the experiments. However, two improvements in the cost-sensitive decision tree (lazy learning and expected cost calculation; see Sections 3.3.1 and 3.3.2) may have been responsible. To see how much of an effect each of the two improvements in LazyOST has over OST, we choose one data set, Kr-vs-kp, and run OST, OST plus expected cost calculation, and OST plus expected cost calculation and plus lazy tree (that is, LazyOST). The results, shown in Table 4, clearly indicate that most cost reduction of LazyOST over OST (81.2 percent on average) comes from the new lazy learning improvement, while the expected cost calculation also has an effect (18.8 percent on average).

4.1.3 Comparing Speed for Sequential Test Strategies

In our experiments, we find that the running speed of the three sequential test strategies is quite different. Table 5 lists the running time for each test example on the data set Ecoli with 40 percent as test examples. Note that all the strategies studied and compared in this paper are test strategies (on test examples). The running time on each test example after the decision tree is built is linear to the depth or size of the tree, and it has little to do with the size of training or test sets. (The LazyOST does rebuild the decision tree for each test example, but similar to C4.5, the training process is quite efficient). The All algorithms are implemented in Java running on Windows XP on a Pentium IV (2.4 GHz) with 512GB memory. We can see clearly that CSNB is much slower than OST and LazyOST. This is because the cost-sensitive naïve Bayes searches for attributes to test while the cost-sensitive decision tree directly uses the tree structure to determine attributes to be tested for each test example. Although LazyOST is several times slower than OST, but both are very fast. Also, we can observe that the time of CSNB becomes longer when the ratio of unknown attributes increases as there are more attributes that should be searched and considered. However, both OST and LazyOST use about the same time since the tree sizes of both lazy and eager decision trees remain roughly the same when the ratio of unknown attributes increases.

4.2 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 patients in medical diagnosis, it may also be life threatening if the disease is not diagnosed and treated promptly. Thus, doctors normally order one set 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 attribute 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].

4.2.1 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 as in the single batch of tests (to "confirm" the case, similar to the diagnosis process of many doctors). 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, which is used to estimate the most likely leaf for the test example, 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 would produce a smaller total attribute 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.

4.2.2 Optimal Single Batch (OSB)

The Optimal Single Batch (OSB) seeks a set of tests to be performed such that the sum of the attribute 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 expected 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 attribute 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 *expected 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 pseudocode of OSB is shown below:

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 $E(i)$:
    $E(i) = misc(i) - [c(i) + \sum p(R(i)) \times misc(R(i))]$
  $E(t) = maxE(i)$ /* 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

In the pseudocode, $misc(.)$ is the expected misclassification cost of a node, $c(.)$ is the attribute cost, $R(.)$ 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 pseudocode calculates the cost difference between no test at i (so only misclassification cost at i) and after testing i (the attribute 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 a 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.

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.

4.2.3 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 Fig. 1b. 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 attribute 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 whether GSB is better than OSB as this is only for one test case.

4.2.4 Comparing Total Cost for Single Batch Strategies

We use the same experiment procedure on the same 13 data sets used in Section 4.1.2 (see Table 2) to compare the overall performance of various Single Batch Test strategies including CSNB-SB [2]. The only change is that we set the misclassification costs to be much higher: \$2,000/\$6,000 (\$2,000 for false positive and \$6,000 for false negative) for the more balanced data sets (the minority class is greater than 10 percent) and \$2,000/\$30,000 for the imbalanced data sets (the minority class is less than 10 percent). This will result in larger trees, and thus, the batch effect is more evident. The total costs for the 13 data sets are listed in Table 6 and the average total costs are plotted in Fig. 3.

From Fig. 3, we can clearly see that overall our new single batch strategies OSB and GSB outperform previous methods CSNB-SB and NSB. 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 our newly proposed strategies (GSB and OSB) and previous methods (NSB and CSNB-SB). In general, the tree-based single batch strategies perform better than the naïve Bayes-based one. The reason is again that the structure of the minimum-cost decision tree is utilized when deciding the single batch, while naïve Bayes has no such structure to rely on.

4.2.5 Comparing Speed for Single Batch Strategies

We also find the running speed of CSNB-SB and other tree-based single batch strategies (NSB, GSB, and OSB) are about the same. Thus, we only compare the speed of NSB with that of CSNB-SB. The average running time for a test example on the data set Ecoli is showed in Table 7.

From Table 7, we can see NSB is much faster than CSNB-SB. It is because the tree-based test strategies can utilize the

TABLE 6
The Total Cost (in $) on 13 Data Sets for Our New Single Batch Test Strategies GSB and OSB Compared to Previous Methods CSNB-SB and NSB

Dataset Name	Strategy	Unknown Attribute Ratio				
		0.2	0.4	0.6	0.8	1.0
Ecoli	CSNB-SB	174.6	174.7	190.0	256.9	288.7
	NSB	131.9	133.3	139.5	157.7	187.0
	GSB	121.7	123.2	121.7	138.7	156.2
	OSB	120.2	121.7	120.7	138.3	156.2
Breast	CSNB-SB	82.1	104.3	159.2	337.3	472.2
	NSB	247.4	267.8	284.3	332.6	393.4
	GSB	247.4	262.7	273.5	259.7	259.7
	OSB	247.4	261.6	280.5	272.1	261.2
Heart	CSNB-SB	615.4	648.6	720.4	768.1	927.3
	NSB	854.0	886.0	953.5	1019.3	1117.0
	GSB	824.7	807.3	774.3	699.7	713.5
	OSB	803.0	757.9	691.1	677.0	683.9
Thyroid	CSNB-SB	179.1	242.4	270.3	347.4	596.7
	NSB	192.5	260.4	306.0	406.9	638.5
	GSB	176.1	180.9	176.7	176.4	220.4
	OSB	167.1	170.4	164.3	169.7	220.4
Australia	CSNB-SB	477.7	534.6	692.5	824.1	1014.8
	NSB	700.4	734.0	811.9	944.4	1091.1
	GSB	662.4	651.9	674.9	675.7	739.9
	OSB	654.2	641.4	656.8	692.8	776.4
Tic-tac-toe	CSNB-SB	725.9	770.8	817.1	858.5	907.2
	NSB	692.0	729.7	854.2	979.6	1013.4
	GSB	658.0	630.9	697.8	731.8	707.8
	OSB	656.2	605.1	665.7	708.3	707.7
Mushroom	CSNB-SB	74.3	81.3	94.7	258.4	474.6
	NSB	17.4	30.9	46.2	67.4	180.0
	GSB	23.3	44.0	51.1	63.2	129.4
	OSB	23.3	44.0	51.1	63.2	129.4
Kr-vs-kp	CSNB-SB	524.2	573.5	674.7	880.4	1280.7
	NSB	72.6	161.3	290.4	497.5	973.3
	GSB	127.2	222.1	290.0	395.2	564.6
	OSB	172.2	256.8	325.7	515.8	959.3
Voting	CSNB-SB	428.1	490.8	606.4	706.7	750.3
	NSB	257.7	303.4	341.6	374.9	434.4
	GSB	265.7	291.4	307.3	293.8	228.4
	OSB	259.7	252.2	243.0	240.4	228.4
Cars	CSNB-SB	562.3	565.2	543.0	677.7	646.4
	NSB	723.1	758.1	757.8	846.2	948.4
	GSB	636.3	596.3	556.0	694.9	865.3
	OSB	636.3	596.3	556.0	694.9	865.3
Thyroid_i	CSNB-SB	387.8	596.5	582.1	712.8	608.4
	NSB	658.8	761.5	820.8	967.5	1057.3
	GSB	585.1	570.2	542.4	493.2	488.1
	OSB	553.7	509.9	447.9	409.4	442.8
Kr-vs-kp_i	CSNB-SB	82.4	84.8	113.6	346.1	700.3
	NSB	12.9	12.9	12.9	38.8	64.7
	GSB	12.9	12.9	12.9	38.8	64.7
	OSB	12.9	12.9	12.9	38.8	64.7
Heart-D	CSNB-SB	1 3 6 . 6	145.4	159.4	208.7	640.3
	NSB	158.9	165.0	171.5	196.0	548.0
	GSB	157.6	163.8	162.9	187.5	223.1
	OSB	157.7	163.7	161.9	179.3	157.5

tree structure in deciding which tests to perform, so the time complexity of the testing strategies is only liner to the tree depth or tree size, while CSNB-SB is based on Naïve Bayes, and a search is performed to find the tests with small costs. Thus, the average test time is much longer, and it increases with the unknown attribute ratio. The tree-based test strategies, however, have similar test time for different unknown attribute ratios.

4.3 Multiple Batch Tests

The Multiple Batch Test naturally combines the Sequential Test and the Single Batch Test, in that batches of tests can be requested in sequence. To make the Multiple Batch Test meaningful, one must assume and provide a "batch cost," the extra cost of each additional batch test (there is no batch cost for the first batch). This cost reflects the cost of "waiting" for the test results of previous batches. Ideally, if the batch cost is

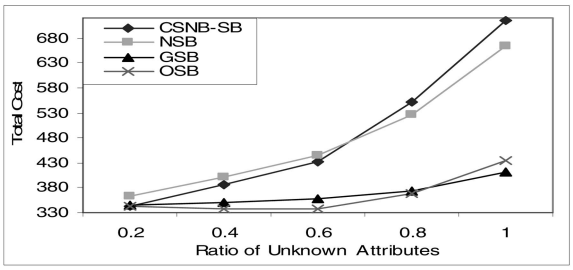

Fig. 3. Comparing the total cost (in $) of our new single batch test strategies GSB and OSB with previous methods CSNB-SB and NSB.

$0, then there is no need to do more than one test at a time, and the Multiple Batch Test becomes simply the Sequential Test. If the batch cost is infinitely large, then one cannot afford to do more than one batch of tests, and the Multiple Batch Test becomes the Single Batch Test.

The following two Multiple Batch Test strategies are an extension (in the same way) of the previous Single Batch Test strategies: Greedy Single Batch and Optimal Single Batch. Thus, we simply change "single" to "multiple" in naming the two new strategies.

4.3.1 Greedy Multiple Batch Test (GMB)

We first describe the Greedy Multiple Batch Test based on the Greedy Single Batch Test. Recall that Greedy Single Batch Test guesses the most probability path for a test example. All the tests on this path are collected. In Greedy Multiple Batch Test, we keep on adding tests on the paths into the current batch until the cumulative ROI (return of investment) does not increase. The rationale behind this (heuristic) strategy is that attributes that bring a larger ROI should be worth including in the same batch test. The ROI is equal to the return (the total cost reduction) divided by the investment (sum of the total attribute cost and the batch cost). That is,

$$ROI = \frac{\sum Costreduction}{BatchCost + \sum AttributeCost}.$$

After the first batch of tests is determined and tested with values revealed, the test example can be classified further down in the tree according to the test results until it is stopped by another unknown attribute. The same process applies, until no more batches of tests are required. The time complexity of this strategy is linear to the size of the tree, as each node in the tree would be considered at most once.

The algorithm described above is highly heuristic as it often adds more than one test into the current batch at a time. Thus, even if the batch cost is $0, the strategy is not equivalent to Sequential Test as in the ideal situation, but to Greedy Single Batch. Also, if the batch cost is very large, the current batch will grow until all the remaining unknown attributes are added into the current batch, and the strategy is similar to the Naïve Single Batch.

4.3.2 Optimal Multiple Batch Test (OMB)

Here, we extend Optimal Single Batch to Optimal Multiple Batch in the very same way as we extend Greedy Single Batch to Greedy Multiple Batch. Recall that in the Optimal Single Batch Test, an unknown attribute is added into the batch if the successive cost reduction of testing it is positive and maximum among the current reachable unknown attributes. In the Optimal Multiple Batch Test, we keep on adding more unknown attributes into the current batch until the accumulative ROI (defined and used in Greedy Multiple Batch) does not increase.

The algorithm described above is heuristic but it is close to the ideal one: It guarantees that if the batch cost is $0, then no extra tests will be added into the current batch, and the strategy is equivalent to the Sequential Test. On the other hand, if the batch cost is very large, the current batch will grow until the cost reduction of the remaining unknown attributes is no longer greater than 0, and the strategy is equivalent to the Optimal Single Batch.

TABLE 7
The Average Running Time (in Milliseconds) of a Test Example of Ecoli

	0.2	0.4	0.6	0.8	1.0
CSNB-SB	0.96	1.14	1.58	2.32	2.57
NSB	0.04	0.05	0.05	0.05	0.05

4.3.3 Case Study on Heart Disease Continued

We apply the Greedy Multiple Batch on the same test example with the tree in Fig. 1b. Assuming the batch cost is $50.00, the strategy decides that two batches of tests are needed for the test case. The first batch of tests contains cp and thal. As the value of cp is 3, not 4 (expected, as the test example can go down to the thal node if cp = 4), the second batch is needed (containing cp, fbs, thal, and thalach). As cp and thal are already tested, only fbs and thalach have attribute costs. Thus, the total attribute costs for the test case is $161.1 (including the batch cost $50.00), while the misclassification cost is 0. Thus, the total cost is $161.1. The Optimal Multiple Batch Test suggests two batches of tests. The first batch has just two tests, cp and fbs. After the values of cp and fbs are obtained (as cp = 3 and fbs = 1), the second batch also contains two tests, thal and thalach. The misclassification cost is 0, while the total attributes costs for the test case is also $161.1 (including the batch cost of $50.00). Thus, the total cost for the test case is also $161.1.

4.3.4 Comparing Total Cost for Multiple Batch Strategies

We conduct experiments with the two Multiple Batch Test Strategies on the same 13 data sets using the same procedure as in earlier sections. We vary the batch cost to be $0, $100, and $200. Table 8 lists the number of batches under different batch costs for the 13 data sets, and Fig. 4 plots the average results of the 13 data sets. Note that the number of batches is often less than 1 especially when the ratios of missing attributes are small (such as 0.2). This is because often test examples do not encounter unknown attributes when they are classified by the decision tree and, thus, no tests are needed. As the tree would likely use known attributes near the top of the tree, a feature of the lazy tree discussed earlier, often test examples can reach a leaf (classification result) without encountering any unknown attribute. (We have also observed this in the

TABLE 8
Number of Batches for the Two Multiple Batch Test Strategies

Dataset Name	Strategies	Batch Cost (in $)	Unknown Attribute Ratio				
			0.2	0.4	0.6	0.8	1.0
Ecoli	OMB	0	0.71	0.71	0.70	0.97	1.06
		100	0.69	0.68	0.67	0.93	0.99
		200	0.67	0.67	0.67	0.92	0.99
	GMB	all	0.66	0.65	0.66	0.92	1.00
Breast		0	0.28	0.64	0.74	1.03	1.27
		100	0.28	0.64	0.74	0.93	1.05
		200	0.28	0.64	0.74	0.92	0.99
	GMB	all	0.28	0.64	0.74	0.90	1.00
Heart	OMB	0	0.72	1.02	1.43	1.91	2.42
		100	0.67	0.95	1.21	1.49	1.78
		200	0.66	0.92	1.15	1.43	1.70
	GMB	all	0.64	0.84	0.96	0.99	1.00
Thyroid	OMB	0	0.30	0.80	0.81	0.95	1.17
		100	0.30	0.79	0.79	0.92	1.08
		200	0.29	0.79	0.78	0.91	1.08
	GMB	all	0.26	0.60	0.64	0.72	1.02
Australia	OMB	0	0.46	0.83	1.41	1.98	2.41
		100	0.46	0.81	1.32	1.81	2.33
		200	0.46	0.80	1.29	1.74	2.08
	GMB	all	0.44	0.76	0.96	1.01	1.01
Tic-tac-toe	OMB	0	0.63	1.03	1.39	1.54	1.40
		100	0.58	0.92	1.20	1.29	1.11
		200	0.57	0.89	1.14	1.21	1.01
	GMB	all	0.53	0.84	0.95	0.92	0.80
Mushroom	OMB	0	0.36	0.56	0.67	0.76	1.48
		100	0.36	0.56	0.67	0.76	1.47
		200	0.36	0.56	0.67	0.76	1.47
	GMB	all	0.36	0.49	0.57	0.66	1.00
Kr-vs-kp	OMB	0	0.60	0.94	1.36	1.96	2.37
		100	0.53	0.84	1.21	1.67	2.01
		200	0.52	0.82	1.17	1.63	1.93
	GMB	all	0.49	0.67	0.76	0.90	1.00
Voting	OMB	0	0.55	1.11	1.37	1.40	1.59
		100	0.54	1.10	1.37	1.40	1.59
		200	0.54	1.10	0.36	1.40	1.59
	GMB	all	0.47	0.86	0.99	1.00	1.00
Cars	OMB	0	0.73	0.89	0.90	1.04	1.18
		100	0.71	0.83	0.85	0.97	1.07
		200	0.71	0.82	0.84	0.95	1.07
	GMB	all	0.72	0.84	0.84	0.94	1.00
Thyroid_i	OMB	0	0.73	1.52	1.87	2.29	2.77
		100	0.73	1.50	1.86	2.28	2.75
		200	0.73	1.50	1.85	2.27	2.74
	GMB	all	0.54	0.92	1.01	1.05	1.02
Kr-vs-kp_i	OMB	0	0.20	0.20	0.20	0.60	1.00
		100	0.20	0.20	0.20	0.60	1.00
		200	0.20	0.20	0.20	0.60	1.00
	GMB	all	0.20	0.20	0.20	0.60	1.00
Heart-D	OMB	0	0.03	0.03	0.11	0.53	2.37
		100	0.03	0.03	0.10	0.48	1.90
		200	0.03	0.03	0.10	0.48	1.83
	GMB	all	0.03	0.03	0.08	0.43	1.03

Optimal Sequential Test and Single Batch Test discussed earlier.) We also note that the number of batches is almost the same for the Greedy Multiple Batch Test under different batch costs, so we list only one value (the average) in Table 8 and plot only one curve in Fig. 4.

From Fig. 4, we can draw several interesting conclusions for the two Multiple Batch Test Strategies. For Optimal Multiple Batch, first of all, the smaller the batch cost, the more the number of batches (for the same ratio of unknown attributes). This meets our expectation as when the batch cost is small, it is cheaper to wait and, thus, more desirable to actualize the tests and decide the subsequent tests. The Optimal Multiple Batch Test becomes the Optimal Sequential Test when the batch cost is zero. Second, when the ratios of missing values in the test examples increase, the average number of batches also increases. This is also expected, as

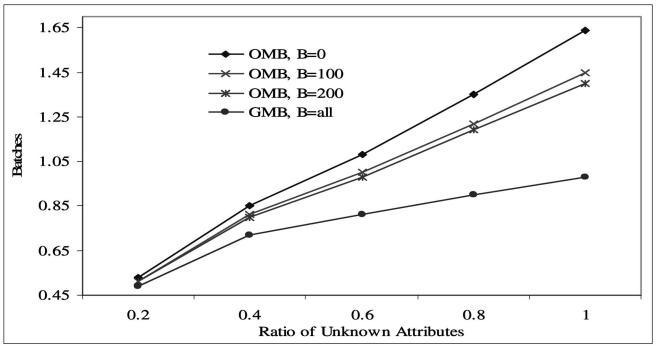

Fig. 4. The average number of batches for the two Multiple Batch Test Strategies.

with more unknown attributes, there is more opportunity to do more batches of tests on the unknowns. On the other hand, the number of batches of the Greedy Multiple Batch Test does not vary with the batch cost. In most situations, only one batch is sufficient, if tests are required. This is due to the heuristic nature of the Greedy Multiple Batch, as it adds multiple tests into the batch in each step (while the Optimal Multiple Batch adds only one test at a time). We can expect that the Optimal Multiple Batch would outperform (in terms of the total cost) the Greedy Multiple Batch, as shown below.

Table 9 reports the total cost under different batch costs and ratios of missing values in test examples for the 13 data sets, and Fig. 5 plots the average of the 13 data sets. Similarly, the total cost of the Greedy Multiple Batch Test is very similar (less than 1 percent difference) with various batch costs, so we list its results in one row in Table 9 and plot in one curve in Fig. 5. For Optimal Multiple Batch, we can see clearly that when the ratio of missing values increases, the total cost also increases, as expected. What is more interesting is that when the batch cost increases, the total cost also increases. This is because Multiple Batch Test is a trade-off between the best situation of Sequential Test (B = \$0) and the worst situation of Naïve Single Batch Test ($B = \infty$). Comparing Greedy Multiple Batch and Optimal Multiple Batch, we can see that at a low ratio of missing values (such as 0.2), the former is similar to the latter with a small batch cost (B = \$0). This is because when the batch cost is small, the Optimal Multiple Batch Test becomes Optimal Sequential Test, which is better but similar to Greedy Single Batch; both perform tests from the first unknown attribute to a leaf. At a high ratio of missing values (such as 1), Greedy Multiple Batch has a similar total cost as Optimal Multiple Batch when the batch cost is high (such as \$200). This is because at this stage, the Optimal Multiple Batch becomes Optimal Single Batch, and as Fig. 3 shows, Optimal Single Batch has a similar total cost with Greedy Single Batch. The results suggest to us that under low ratios of missing values, the simpler and faster Greedy Multiple Batch algorithm is preferred, while under high ratios of unknown values, the Optimal Multiple Batch should be applied.

TABLE 9
The Average Total Cost (in $) for our Multiple Batch Test Strategies

Dataset Name	Strategies	Batch Cost (in $)	Unknown Attribute Ratio				
			0.2	0.4	0.6	0.8	1.0
Ecoli	OMB	0	120.1	122.3	120.5	138.2	167.0
		100	153.4	154.0	150.4	190.7	245.7
		200	161.8	165.6	154.4	192.5	245.9
	GMB	all	128.9	129.8	127.7	137.6	152.1
Breast	OMB	0	247.4	265.1	284.2	312.3	328.4
		100	251.8	279.5	307.4	367.9	457.7
		200	251.8	280.6	311.2	393.2	498.1
	GMB	all	247.4	266.5	282.8	319.2	365.2
Heart	OMB	0	845.9	854.4	891.2	903.4	930.5
		100	888.0	921.2	1019.6	1083.0	1111.4
		200	899.2	938.4	1059.1	1161.4	1200.9
	GMB	all	852.7	876.2	936.5	1009.7	1117.0
Thyroid	OMB	0	182.8	186.2	192.2	203.9	234.9
		100	192.6	215.4	220.4	249.3	384.1
		200	197.9	237.6	238.5	277.7	436.1
	GMB	all	190.2	251.8	290.6	361.5	508.5
Australia	OMB	0	686.4	701.2	714.1	736.7	715.9
		100	705.8	734.8	803.3	904.3	940.4
		200	709.0	746.3	852.1	1010.4	1108.4
	GMB	all	698.1	725.8	797.7	923.4	1056.8
Tic-tac-toe	OMB	0	674.6	691.0	758.2	820.8	795.1
		100	714.6	761.1	855.8	940.2	891.1
		200	726.7	788.8	904.9	1001.6	956.5
	GMB	all	689.0	721.1	843.4	966.8	1013.4
Mushroom	OMB	0	17.4	25.2	29.5	37.7	70.0
		100	30.1	52.5	64.0	72.7	163.0
		200	30.1	59.6	73.5	83.3	210.5
	GMB	all	17.4	30.9	46.2	59.8	164.5
Kr-vs-kp	OMB	0	54.5	95.4	138.3	174.0	341.5
		100	89.3	156.3	248.3	321.6	495.6
		200	95.0	187.7	308.3	411.1	614.7
	GMB	all	66.2	152.9	276.4	468.3	942.0
Voting	OMB	0	253.1	269.3	282.8	279.4	275.2
		100	280.7	334.1	367.1	362.8	374.9
		200	288.0	358.5	406.0	405.8	435.9
	GMB	all	257.0	299.9	337.2	363.4	410.8
Cars	OMB	0	723.0	753.1	742.6	825.6	919.5
		100	759.0	793.3	786.2	885.7	1014.7
		200	773.5	807.8	809.3	903.2	1021.8
	GMB	all	716.9	748.2	744.2	833.7	948.4
Thyroid_i	OMB	0	604.3	597.1	586.9	549.2	500.7
		100	634.8	672.7	692.2	689.7	685.4
		200	660.4	740.5	790.6	824.1	860.8
	GMB	all	650.0	726.2	772.7	824.3	968.9
Kr-vs-kp_i	OMB	0	12.9	12.9	12.9	38.8	64.7
		100	25.9	25.9	25.9	77.6	129.3
		200	25.9	25.9	25.9	77.6	129.3
	GMB	all	12.9	12.9	12.9	38.8	64.7
Heart-D	OMB	0	157.6	162.8	164.5	183.8	224.1
		100	158.7	163.9	168.4	195.3	366.9
		200	159.4	164.5	171.8	201.0	443.5
	GMB	all	159.1	165.0	168.8	194.9	405.7

5 Conclusions and Future Work

In this paper, we present a lazy decision tree learning algorithm to minimize the sum of misclassification costs and attribute costs. We then design three categories of test strategies: Sequential Test, Single Batch Test, and Multiple 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, Optimal Single Batch, and Optimal Multiple 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 data sets in the real world. A detailed case study on heart disease is given in the paper.

In our future work, we will allow variable wait costs for different tests; this can probably unify Single Batch and Sequential Test Strategies discussed in this paper. We also plan to continue to work with medical doctors to apply our algorithms to medical data with real costs. Last, we plan to study the effect of the pruning in cost-sensitive decision trees, and incorporate other types of costs in our decision tree learning and test strategies.

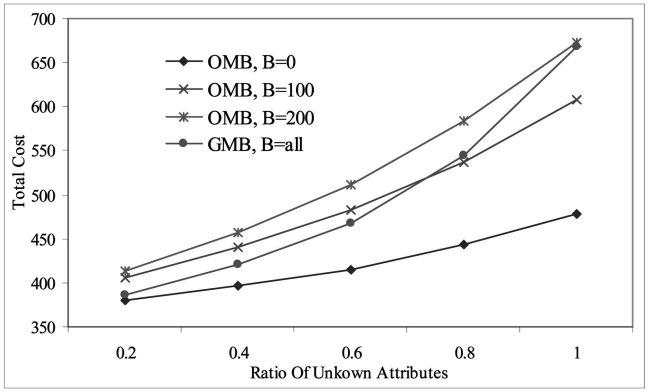

Fig. 5. The average total costs (in $) under different batch costs for our Multiple Batch Test Strategies.

ACKNOWLEDGMENTS

The authors would like to thank Peter Turney for his helpful discussions and suggestions during this research. They also thank reviewers for insightful comments. Charles X. Ling and Victor S. Sheng thank NSERC, and Qiang Yang thanks Hong Kong RGC (RGC Project HKUST 6187/04E), for support of their research. This paper is an extended version of our AAAI 2006 paper [14].

REFERENCES

[1] C.L. Blake and C.J. Merz UCI Repository of Machine Learning Databases (Web Site), Dept. of Information and Computer Science, Univ. of California, Irvine, 1998.
[2] X. Chai, L. Deng, Q. Yang, and C.X. Ling, "Test-Cost Sensitive Naïve Bayesian Classification," *Proc. Fourth IEEE Int'l Conf. Data Mining,* 2004.
[3] P. Domingos, "MetaCost: A General Method for Making Classifiers Cost-Sensitive," *Proc. Fifth Int'l Conf. Knowledge Discovery and Data Mining,* pp. 155-164, 1999.
[4] C. Elkan, "The Foundations of Cost-Sensitive Learning," *Proc. 17th Int'l Joint Conf. Artificial Intelligence,* pp. 973-978, 2001.
[5] U.M. Fayyad and K.B. Irani, "Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning," *Proc. 13th Int'l Joint Conf. Artificial Intelligence,* pp. 1022-1027, 1993.
[6] J. Friedman, Y. Yun, and R. Kohavi, "Lazy Decision Trees," *Proc. 13th Nat'l Conf. Artificial Intelligence,* 1996.
[7] G. Gorry and G. Barnett, "Experience with a Model of Sequential Diagnosis," *Computers and Biomedical Research,* 1968.
[8] C.X. Ling, Q. Yang, J. Wang, and S. Zhang, "Decision Trees with Minimal Costs," *Proc. 21st Int'l Conf. Machine Learning,* 2004.
[9] D. Lizotte, O. Madani, and R. Greiner, "Budgeted Learning of Naïve-Bayes Classifiers," *Proc. 19th Conf. Uncertainty in Artificial Intelligence,* 2003.
[10] P. Melville, M. Saar-Tsechansky, F. Provost, and R.J. Mooney, "Active Feature Acquisition for Classifier Induction," *Proc. Fourth Int'l Conf. Data Mining,* 2004.
[11] P. Melville, M. Saar-Tsechansky, F. Provost, and R.J. Mooney, "Economical Active Feature-Value Acquisition through Expected Utility Estimation," *Proc. Workshop Utility-Based Data Mining,* 2004.
[12] M. Nunez, "The Use of Background Knowledge in Decision Tree Induction," *Machine Learning,* vol. 6, pp. 231-250, 1991.
[13] *C4.5: Programs for Machine Learning,* J.R. Quinlan, ed. Morgan Kaufmann, 1993.
[14] V.S. Sheng, C.X. Ling, A. Ni, and S. Zhang, "Cost-Sensitive Test Strategies," *Proc. 21st Nat'l Conf. Artificial Intelligence,* 2006.
[15] M. Tan, "Cost-Sensitive Learning of Classification Knowledge and Its Applications in Robotics," *Machine Learning J.,* vol. 13, pp. 7-33, 1993.
[16] K.M. Ting, "Inducing Cost-Sensitive Trees via Instance Weighting," *Proc. Second European Symp. Principles of Data Mining and Knowledge Discovery,* pp. 23-26, 1998.
[17] P.D. Turney, "Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm," *J. Artificial Intelligence Research,* vol. 2, pp. 369-409, 1995.
[18] P.D. Turney, "Types of Cost in Inductive Concept Learning," *Proc. Workshop Cost-Sensitive Learning, 17th Int'l Conf. Machine Learning,* 2000.
[19] B. Zadrozny and C. Elkan, "Learning and Making Decisions When Costs and Probabilities are Both Unknown," *Proc. Seventh Int'l Conf. Knowledge Discovery and Data Mining,* pp. 204-213, 2001.
[20] V.B. Zubek and T. Dietterich, "Pruning Improves Heuristic Search for Cost-Sensitive Learning," *Proc. 19th Int'l Conf. Machine Learning,* pp. 27-35, 2002.

Charles X. Ling received the MSc and PhD degrees from the Department of Computer Science at the University of Pennsylvania in 1987 and 1989, respectively. Since then, he has been a faculty member in the Department of Computer Science at the University of Western Ontario, Canada. His main research areas include machine learning (theory, algorithms, and applications), cognitive modeling, and AI in general. He has published more than 100 research papers in journals (such as, *Machine Learning*, *JMLR*, *JAIR*, *Bioinformatics*, the *IEEE Transactions on Knowledge and Data Engineering*, and *Cognition*) and international conferences (such as IJCAI, ICML, and ICDM). He has been an associate editor for the *IEEE Transactions on Knowledge and Data Engineering*, and guest editor for several journals. He is also the director of the Data Mining Lab, leading data mining development in CRM, Bioinformatics, and the Internet. He has managed several data mining projects for major banks and insurance companies in Canada.

Victor S. Sheng received the BE degree from Chongqing University, China, in 1993, the ME degree in computer engineering from Suzhou University, China, in 1999, and the MSc degree in computer science from the University of New Brunswick in 2004. He is currently a PhD candidate in computer science at University of Western Ontario, Canada. He is interested in data mining research and applications.

Qiang Yang received the PhD degree from the University of Maryland, College Park. He is a faculty member at the Hong Kong University of Science and Technology in the Department of Computer Science. His research interests are AI planning, machine learning, case-based reasoning, and data mining. He is a senior member of the IEEE and an associate editor for the *IEEE Transactions on Knowledge and Data Engineering* and *IEEE Intelligent Systems*.