Test-Cost Sensitive Classification on Data with Missing Values

Qiang Yang, *Senior Member*, *IEEE*, Charles Ling, Xiaoyong Chai, and Rong Pan

Abstract—In the area of cost-sensitive learning, inductive learning algorithms have been extended to handle different types of costs to better represent misclassification errors. Most of the previous works have only focused on how to deal with misclassification costs. In this paper, we address the equally important issue of how to handle the test costs associated with querying the missing values in a test case. When an attribute contains a missing value in a test case, it may or may not be worthwhile to take the extra effort in order to obtain a value for that attribute, or attributes, depending on how much benefit the new value will bring about in increasing the accuracy. In this paper, we consider how to integrate test-cost-sensitive learning with the handling of missing values in a unified framework that includes model building and a testing strategy. The testing strategies determine which attributes to perform the test on in order to minimize the sum of the classification costs and test costs. We show how to instantiate this framework in two popular machine learning algorithms: decision trees and naive Bayesian method. We empirically evaluate the test-cost-sensitive methods for handling missing values on several data sets.

Index Terms—Cost-sensitive learning, decision trees, naive Bayes.

1 Introduction

Inductive learning techniques, such as the naive Bayesian and decision tree algorithms, have met great success in building classification models with an aim to minimize the classification errors [1], [2]. However, much previous inductive learning research has only focused on how to minimize classification costs such as the cost of false positive (FP) and the cost of false negative (FN). The classification errors are useful in deciding whether a learned model tends to make correct decisions on assigning class labels for new cases and is useful for dealing with data with unbalanced classes. However, misclassification costs are not the only costs to consider in practice. When performing classification on a new case, values for some attributes may be missing. In such a case, we may have the option of performing additional tests in order to obtain a value for these attributes. However, performing these additional tests may incur more costs, where some costs are in the form of lengthy waiting time and others include monetary payment. Still, some tests are worthwhile to perform because having the additional values might greatly increase the classification accuracy. Thus, we often must consider the "test cost" when missing values must be obtained through physical "tests" which incur costs themselves. These costs are often as important as the misclassification costs.

As an example, consider the task of a medical practice that examines incoming patients for a certain illness (see Table 1). Suppose that the doctors' previous experience has been compiled into a classification model such as a naive Bayesian classifier. When dealing with an incoming patient, it is often the case that certain information for this patent may not yet be known; for example, the blood tests or the X-ray test may not have been done yet. At this point, the doctor (that is, the classification model) must exercise its judgement appropriately: Performing these tests will incur certain extra costs, but some tests may provide useful informational benefits toward reducing the classification costs. In the end, it is the balancing act of the two types of costs—namely, the classification costs and the test costs—that will determine which tests will be done.

Tasks that incur both misclassification and test costs associated with missing values abound in industrial practice ranging from medical diagnosis to scientific research and to drug design. In the past, inductive learning methods that consider a variety of costs are often referred to as cost-sensitive learning [3], [4]. In this paper, we present a unified framework for how to integrate both cost-sensitive learning and the treatment of missing values in test data. To distinguish from methods that only consider misclassification costs, we call this method *test-cost-sensitive* learning methods.

Our test-cost-sensitive learning framework, which is called the *TCSL* framework, consists of two parts. First, in part one, when training a model, we consider both the misclassification costs and the potential test costs associated with the attributes in order to minimize the potential future total costs. Then, in part two, we design test strategies that are tailored for each individual test example in order to

- *Q. Yang and R. Pan are with the Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. E-mail: {qyang, panrong}@cs.ust.hk.*
- *C. Ling is with the Department of Computer Science, The University of Western Ontario, London, Ontario N6A 5B7, Canada. E-mail: cling@csd.uwo.ca.*
- *X. Chai is with the Department of Computer Science, Purdue University, 250 N. University Street, West Lafayette, IN 47907. E-mail: chai@cs.purdue.edu.*

Manuscript received 17 May 2005; revised 9 Oct. 2005; accepted 27 Oct. 2005; published online 17 Mar. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0192-0505.

TABLE 1
An Example of a New Case Containing Missing Values and Their Associated Costs for Getting a Value

Tests:	**Blood Test**	**X-ray Test**	**CAT Scan**	**Ultrasound**	**MRI Scan**	**Diagnosis**
	?	Normal	?	?	Abnormal	to be decided
Costs:	\$500		\$200	\$300		$MC + TC$

MC is the misclassification cost, and TC is the test cost.

exploit the known information and propose a plan to acquire the unknown.

For training a model, we apply our TCSL framework to two machine learning algorithms, decision trees and naive Bayesian, and evaluate their relative merits. For testing, when a new case contains missing values for its attributes, decisions must be made whether to obtain values of these attributes through tests. In this paper, we consider two test strategies: a sequential test strategy and a batch test strategy. The former takes tests for missing values sequentially. Decisions on whether an additional test is needed or which attribute with missing value should be tested next are made based on the outcome of the previous test results. The batch test strategy requires several new tests to be done all together rather than in a sequential manner.

The novelty of our work can be seen from several angles, as follows:

1. Most previous work on cost-sensitive learning has mostly considered how to increase the classification accuracy by considering the false positive and negative costs. In our TCSL framework, we additionally consider the test cost, by minimizing the sum of the classification and test costs together.
2. We not only consider how to build a TCSL model from the training data, but we also consider how to handle the test cases by considering the sequential and batch test strategies for obtaining the missing values.

2 Related Work

Much work has been done in machine learning on minimizing the classification errors. This is equivalent to assigning the same cost to each type of classification errors (for example, FP and FN), and then minimizing the total misclassification costs. In Turney's survey papere [3], a whole variety of costs in machine learning are analyzed, and the test cost is singled out as one of the least considered areas in machine learning. In particular, [3] considered the following types of costs in machine learning:

- Misclassification costs: These are the costs incurred by misclassification errors. Works such as [5], [4], [6] considered machine learning with nonuniform misclassification costs.
- Test costs: These are the costs incurred for obtaining attribute values. Some previous work such as [7], [8] considered the test cost alone without incorporating misclassification cost. As pointed out in [3], it is obviously an oversight.

As far as we know, the only works that considered both misclassification and test costs include [9], [10], [11], [12]. Of these works, [9] explicitly considered how to directly incorporate both types of costs in decision tree building processes and in determining the next attribute to test, should the attribute contain a missing value. An advantage of the minimal-test-cost decision tree method is that it naturally extends the decision tree construction algorithm by considering both the misclassification costs and the test cost in a local search framework. It additionally considers whether a test should be conducted and how to select the next attribute to test. Through experimentation with this method, however, we have also found some shortcomings. Because decision trees are aimed at serializing attribute tests along the path of a tree, it is not well-suited for performing batch tests that involve a number of tests to be done together. Furthermore, because it places different levels of importance on the attributes by the natural organization of the tree, it cannot be easily fitted to make decisions on testing strategies that select attributes for the tests. In contrast, the naive Bayesian-based algorithms overcome these difficulties more naturally. As we will see, the performance offered by the test-cost sensitive naive Bayesian is significant over its decision-tree counterpart.

In [11], the cost-sensitive learning problem is cast as a Markov Decision Process (MDP), and an optimal solution is given as a search in a state space for optimal policies. For a given new case, depending on the values obtained so far, the optimal policy can suggest a best action to perform in order to both minimize the misclassification and the test costs. While related to our work, their research adopts an optimal strategy, which may take very high computational cost to conduct the search process. In contrast, we adopt the local search algorithm the concept of a utility gain, which is more efficient to compute and, as we will show, attains a high level of classification accuracy. Thus, our algorithm follows the direction of approximation rather than optimal algorithms.

Similarly, Greiner et al. in the interest in constructing an optimal learner, [12] studied the theoretical aspects of active learning with test costs using a PAC learning framework. Turney [10] presented a system called ICET, which uses a genetic algorithm to build a decision tree to minimize the cost of tests and misclassification. As mentioned above, because our algorithm essentially adopts the conditional probability-based framework, which requires only a linear scan through the data set, our algorithm is expected to be more efficient than Turney's genetic algorithm-based approach.

In the past, test costs have also been separately considered in [9], where decision-tree-based testing strategies were explored along with several testing strategies, and in [13], where a Naive Bayesian-based method is proposed with sequential strategies. However, these methods have never been placed under a unified test-cost-sensitive learning framework, nor have they been compared thoroughly together. With the unified framework, it is now possible to compare not only sequential test strategies, but also batch test strategies where several tests are selected to be done together.

3 Test-Cost Sensitive Learning

3.1 Problem Formulation

The Test-Cost-Sensitive classification learning problem can be formulated as follows:

Description of **Algorithm** TCSL

Given: (D, R, T), where

- D is a data set consisting of N samples $(x_1, x_2, \cdots, x_N)$ from P classes $(c_1, c_2, \cdots, c_P)$. Each sample x_i is described by M attributes $(a_1, a_2, \cdots, a_M)$ among whom there can be missing values.
- R is a misclassification cost matrix. Each entry $r_{ij} \triangleq R(i,j)$ specifies the cost of classifying one sample from class c_i as class c_j $(1 \leq i, j \leq P)$. Usually, $R_{ii} = 0$ $(1 \leq i \leq P)$.
- T is a test cost vector. Each entry $T_k \triangleq T(k)$ specifies the cost of taking a test on attribute a_k.

Build: a classifier for minimizing the total cost of classification that includes both the test cost and the misclassification cost for the training examples, and a test strategy for deciding, given a test case, an order in which to obtain the missing values.

Our aim is to minimize the sum of classification costs C_{mc} and test costs for every test case, C_{test}.

Subject to: constraints such as the total cost is within a user specified upper bound.

The above formulation provides a more general framework than the traditional cost-sensitive learning formulations. Actually, the latter is just a special case of the TCSL framework where the test cost is sufficiently large so that no test will be performed. Also, the conditional risk [14] can be equivalently implemented by setting the misclassification cost matrix properly.

TCSL provides a more general framework than the traditional supervised learning frameworks. Standard supervised learning is just a special case of the TCSL because no missing values are acquired in the traditional methods. That is, if we set the test costs T to positive infinite so that no test will be performed, then the TCSL retrogresses to a standard supervised learning algorithm which makes prediction based only on the known attributes. For example, the conditional risk [14] can be equivalently implemented by setting the misclassification cost matrix R.

By imposing different constraints on how to obtain the attributes, we can realize different test strategies. An important constraint is the available resource constraint used to limit the total number of test costs. As an example, consider the task of a medical practice that patients have a limited amount of money for doing medical tests, each test having a different test cost. Therefore, the total test fees cannot exceed the money limitation. When the money limitation is reached, the patients cannot afford any more tests, even if there are some tests that can reduce the risk of false diagnosis (misclassification). Then, the problem of the doctors is to design an optimal sequential test strategy within the test cost constraint. The strategy is sequential since the decision on the next test to perform can be dependent on the outcome of the previous one.

In addition to doing the test in sequence, in practice, there is also a great need for batch tests. In medical diagnosis, doctors cannot afford to wait for the result of the first test before other tests can be done. They normally order a set of tests to be done at one shot. This kind of constraint can be viewed as limiting the times of doing tests in addition to the resources available. It is more practical and decisions must be made altogether before any test result comes out. In this situation, the problem is to design an optimal batch test strategy instead of doing tests one by one in a sequential manner. It is interesting how to design such an "Optimal Batch Test" for the TCSL. We will discuss the batch test strategy in detail in Section 5.4.

Note that a main contribution of our work is the unification of attribute test and misclassification costs. However, in many real-world domains, these two costs carry different meanings and, therefore, how to combine them is a domain-dependent issue. For simplicity, however, in this work, we simply assume that they are comparable and can be added together to get the total cost.

4 Test-Cost-Sensitive Decision Trees

4.1 Model Construction

The test-cost-sensitive learning framework can include different classification algorithms. In this section, we review the test-cost-sensitive decision tree algorithm [9] and model it under the general test-cost-sensitive learning framework.

We assume that the training data may consist of some missing values (whose values cannot be obtained). We also assume the test costs are associated with attributes (that is, to perform a CAT Scan in a medical case, the cost to the patient is the same regardless of the outcome). Furthermore, we assume that the test cost and the misclassification cost have been defined on the same cost scale, such as the dollar cost incurred in a medical diagnosis. For simplicity, we consider discrete attributes and binary class labels; extensions to other situations, such as numerical classes, can be made similarly. We assume that FP is the cost of one false positive example and FN is the cost of one false negative example.

Our decision-tree learning algorithm uses a new splitting criterion of minimal total cost on training data, instead of minimal entropy, to build decision trees. This cost measure is equivalent to the expected total cost measure used in the works of [3], [11], [12]. More specifically, at each step, rather than choosing an attribute that minimizes the entropy (as in C4.5), our algorithm chooses an attribute that reduces and minimizes the total cost, which is the *sum of the test cost and the misclassification cost*, for the split. With the total cost formula, similar to C4.5, our algorithm chooses a locally optimal attribute without backtracking. Another similarity is the way it treats missing values in the *training data set*. Thus, even though the resulting tree may not be globally optimal, the efficiency of the tree-building algorithm is generally high. A concrete example is given later in this section.

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

Algorithm csDT-learn() (see Algorithm 1) lists the general input and output required to learn a model in test-cost-sensitive decision-tree learning. Once a model is built on the training data, the model can then be applied to a test case x, as shown in Algorithm 2 in the next section.

```
**Algorithm 1** csDT-learn$(D, A, CL, R, T, TCF)$
    **Input**:
    $D$—a data set of samples $\{x_1, x_2, \dots, x_N\}$,
    $A$—a set of attributes $\{A_1, A_2, \dots, A_M\}$, where
        $A_m \in \{v_{m,1}, v_{m,2}, \dots, v_{m,|A_m|}\}$,
    $CL$—predefined classes $\{c_1, c_2, \dots, c_P\}$,
    $R$—misclassification cost matrix (for binary class
        problems, this defines the $FP$ and $FN$ costs),
    $T$—a test cost vector, which is an array listing the
        cost value for each attribute,
    $TCF$—a total cost formula for each outcome.
    **Output**:
    $Model(c)$—the learned model that predicts the class
    value of a new case $c$ with a probability measure.

Step 1: Let $T$ be the minimum misclassification cost of
        stopping at $D$ with a class label $c_D$, among all
        possible $c_D \in CL$;
Step 2: For each attribute $A_i, i = 1, 2, \dots, m$ in $D$, calculate
        the total cost $T_i$ of testing and splitting $D$ on $A_i$'s
        values;
Step 3: If $T \le T_i$ for all $i$, then stop at $D$ with a class label $c_D$
        from Step 2;
Step 4: Select the $A_i$ with the minimum total cost $T_i$ among
        all $A_i$ in $D$, and split the data set $D$ according the
        values of $A_i$, producing data sets $D_{ij}, j = 1, 2, \dots k$;
Step 5: Apply TCSL-learn algorithm to each $D_{ij}$
        recursively.
```

Consider a concrete example to illustrate the calculation of the total cost. Assume that, during the tree building process, there is a set of P and N positive and negative examples, respectively, to be further classified by possibly building a subtree. If $P * FN > N * FP$, then, if no subtree is built, the set would be labeled as positive (+). Thus, the total misclassification cost is $T = N * FP$. Suppose that an attribute A with a test cost C is considered for a potential splitting attribute. Assume that an attribute A has two values, and there are $P1$ and $N1$ positive and negative examples with the first value, respectively, and $P2$ and $N2$ positive and negative examples with the second value, respectively. Then, the total *test* cost would be $(P1 + N1 + P2 + N2) * C$.

Assume that the first branch will be labeled as positive (as $P1 * FN > N1 * FP$), and the second branch will be labeled as negative, then the total misclassification cost of the two branches would be $N1 * FP + P2 * FN$. Based on these considerations, the total cost of choosing A as a splitting attribute, when processing the training examples, would be:

$$T_A = (P1 + N1 + P2 + N2) * C + N1 * FP + P2 * FN.$$

If $T_A < T$, where $T = N * FP$, then splitting on A would reduce the total cost of the original set, and we will choose such an attribute with the minimal total cost as a splitting attribute. We will then apply this process recursively on examples falling into branches of this attribute. If $T_A \ge T$ for all remaining attributes, then no further subtree will be built, and the set would become a leaf with a positive label.

4.2 Test Strategy on New Examples

After the minimal-cost decision tree is built, the next interesting question is how this tree can be used to deal with testing examples with many missing values in order to predict the class of the testing examples with the minimal total cost for this case. A test strategy determines an order to obtain missing values for each particular testing test case. Different test strategies can result in different total costs. In this paper, we consider a simple sequential strategy: to simply follow the tree built in the previous section.

The testing strategy we consider in this paper is to simply follow the tree (see Algorithm 2) because the decision trees have already specified an order in which to perform the tests. During the process of classifying a single example, should there be any missing values for an attribute, we always do a test on that attribute in order to obtain a value. This process continues until a leaf node is reached.

```
**Algorithm 2** csDT-test$(Model, C, T, T_{\text{total}}, x)$
    **Input**:
    $Model()$—a test-cost-sensitive classification model,
    $R$—a misclassification cost matrix,
    $T$—a test cost vector,
    $T_{total}$—the total resources available,
    $x$—a testing example.
    **Output**:
    Label—the predicted class,
    $T_{test}$—the test cost for the example.
Step 1: If a batch-test strategy is followed, then obtain all
        missing values of $x$ with tests, and then apply the
        decision tree classification to $x$; output the class
        label and total test cost $T_{test}(x)$, and stop.
```

Step 2: If a sequential-test strategy is followed, then apply the tree to x starting from the root node;

Step 3: Repeat: at any node N, if the corresponding value of x is missing, then perform a test on this attribute if the total test cost so far is less than T_{total}; if exceeding T_{total} already, then stop and report the minimal cost class label at node N;

Step 4: If a leaf node is met, then report the class label C with the total test cost so far $T_{test}(x)$ and stop.

The decision-tree-based sequential test strategy, while optimal based on the minimal cost of the training set, is *sequential* in nature. A *batch test strategy* can be modeled in our decision tree as well, although, as we will see in the next section, not as natural as in a Naive Bayesian algorithm. The basic idea is that, when a testing case is stopped at the first attribute whose value is unknown, all unknown values under that attribute must be obtained. Clearly, this batch strategy will return the same prediction as the decision-tree-based test strategy (i.e., same misclassification cost), but it would incur a higher test cost in general because some test results are not used in classification.

5 Test-Cost-Sensitive Naive Bayes

5.1 Costs in Naive Bayesian Classification

Decision trees determine a natural sequence of tests to be done, but may be too strict in some cases. To allow a more flexible testing strategy, we now turn to Naive Bayesian classifiers, which are shown to perform very well in practice [15]. Below, we consider the cost-sensitive Naive Bayesian Learning [13] under our test-cost sensitive learning framework. For classification, the standard naive Bayes algorithm computes the posterior probability $P(c_j|x)$ of sample x belonging to class c_j according to Bayes' rule:

$$P(c_j|x) = \frac{P(x|c_j)P(c_j)}{P(x)},$$

where

$$P(x|c_j) = \prod_{m=1}^{|A|} P(A_m = v_{m,k}|c_j)$$

is the product of $|A|$ likelihoods. $|A|$ is the number of attributes describing x and $v_{m,k}$ is one possible value of attribute A_m. Then, sample x is predicted to belong to class j^* which has the highest value $P(c_{j^*}|x)$. When there exist missing values in sample x, the corresponding attributes are simply left out in likelihood computation and the posterior probability is computed only based on the known attributes $\widetilde{A} \subseteq A$. Therefore, $P(x|c_j) = \prod_{A_m \in \widetilde{A}} P(A_m = v_{m,k}|c_j)$.

The standard naive Bayesian algorithm can be extended to take into account the misclassification cost. Suppose that C_{ij} is the cost of predicting an example of class i as belonging to class j. In this situation, the *expected total cost* (that is, the T_{total} in Algorithm 1) of predicting a single sample x as a class j is known as the *conditional risk* [14] and is defined as: $R(j|x) = \sum_i C_{ij} \times P(i|x)$, where $P(i|x)$ is the posterior probability. Then, sample x is predicted to belong to the class j^* which has the minimal conditional risk $R(j|x)$.

TABLE 2
Likelihoods of Attributes

Attri-butes	Test Cost	Positive (c_1)		Negative (c_2)	
		Yes	No	Yes	No
liver firm	45	51.9%	48.1%	59.8%	40.2%
spleen	42	61.3%	38.7%	84.9%	15.1%
spiders	40	29.0%	71.0%	75.6%	24.4%
ascites	45	54.8%	45.2%	95.0%	5.0%

Before considering the test costs, let us consider an example. Suppose that, in the medical diagnosis of a hepatitis case, 21 percent of patients are positive (have hepatitis, c_1) and 79 percent of patients are negative (healthy, c_2). Therefore, the priors are $P(c_1) = 21$ percent and $P(c_2) = 79$ percent, respectively. Assume the misclassification costs (conditional risk) are $C_{12} = 400$, $C_{21} = 100$, and $C_{11} = C_{12} = 0$.

There are four attributes to characterize a patient, and the test cost of each attribute and the likelihoods are listed in Table 2.

When diagnosing a new patient, the doctor faces the decision of whether a test should be performed and, if so, which one. Since each test on an unknown attribute has its own discriminating power on disease and also brings a certain amount of cost, decisions must be made by considering both factors. Later, we will see that the attribute "ascites" is the best first choice since it trades off the misclassification cost and test costs. However, if its test cost is raised from 45 to 47, the attribute "liver firm" will be more preferable for testing first. This example shows that the testing strategy is related to the test costs.

In practice, the problem is even more complicated when more attributes are involved and some tests are with delayed results. For example, the blood tests are usually shipped to a laboratory and the results are sent back to doctors the next day. In this case, for the sake of patients, doctors often ask for a batch of tests simultaneously. Therefore, the test strategy must consider more than one test to be done, under different constraints.

5.2 Model Construction in csNB

The procedure of learning a csNB classifier is basically an estimation of the distribution parameters as in standard naive Bayes. Algorithm 3 outlines the learning procedure.

```
Algorithm 3 csNB-learn(D, A, CL)
    Input:
    $D$—a data set of samples $\{x_1, x_2, \ldots, x_N\}$,
    $A$—a set of attributes $\{A_1, A_2, \ldots, A_M\}$, where
        $A_m \in \{v_{m,1}, v_{m,2}, \ldots, v_{m,|A_m|}\}$,
    $CL$—predefined classes $\{c_1, c_2, \ldots, c_P\}$.
    Output:
    $\hat{P}(c_j)$—the estimated prior probabilities,
    $\hat{P}(A_m = v_{m,k}|c_j)$—the estimated likelihoods, where
                    $0 < k <= |A_m|$.
```

Steps:
```
1: for each class $c_j$ do
2: $\hat{P}(c_j) = \frac{\sum_{i=1}^{N} P(c_j|x_i)}{N}$
3: end for
4: for each class $c_j$ do
5:    for each attribute $A_m$ do
6:       for each value $v_{m,k}$ of attribute $A_m$ in class $c_j$ do
7:          $\hat{P}(A_m = v_{m,k}|c_j)$
            $= \frac{\lambda + \Sigma_{i=1}^{N} Num(A_m = v_{m,k}, x_i) P(c_j|x_i)}{\lambda|A_m| + \Sigma_{l=1}^{|A_m|} \Sigma_{i=1}^{N} Num(A_m = v_{m,l}, x_i) P(c_j|x_i)}$
8:       end for
9:    end for
10: end for
```

In the above equations, λ is the smoothing factor. $\lambda = 1$ is known as the Laplacian smoothing which we use in our experiments. The $Num(A_m = v_{m,k}, x_i)$ is a function counting the number of times that attribute A_m has the value $v_{m,k}$. Moreover, $P(c_j|x_i) \in \{0, 1\}$, $j \in \{1, \cdots, P\}$; if x_i belongs to the class j, the value is 1; otherwise, the value is 0.

5.3 Sequential Test Strategies

After a naive Bayes classifier is built, an interesting question now is how the classifier performs tests when the testing examples have missing values. Instead of treating test costs C_{test} and misclassification costs C_{mc} separately, we offer testing strategies for minimizing the sum of C_{test} and C_{mc} for different situations. In this section, we consider the sequential test strategy and leave the batch test strategy to Section 5.4.

The sequential test strategy is as follows: During the process of classification, decisions are made sequentially on whether a test on an unknown attribute (an attribute with missing value) should be performed based on the results of previous tests, and, if so, which attribute is selected.

Formally, let $S = < D_1, D_2, \cdots >$ denote the strategy of a sequence of decisions where $D_i = A_j$ represents the ith decision of selecting an unknown attribute A_j for testing. Sequential test means that decision D_{i+1} is made dependent on the result of decision D_i, more specifically, the outcome of the test.

Suppose that $x = (a_1, a_2, \cdots, a_M)$ is a testing example. Each attribute a_i can be either known or unknown. Let $\widetilde{A}$ denote the set of all known attributes among attributes A and $\overline{A}$ the unknown attributes. The misclassification cost of classifying x as class c_j is:

$$R(c_j|x) = \sum_{i=1}^{P} C_{ij} \times P(c_j|x), \quad 1 \leq j \leq P, \tag{1}$$

where $P(c_j|x) = \frac{P(x|c_j)P(c_j)}{P(x)}$ is the posterior probability obtained using Bayes' rule. The class c_j^* with the minimum cost is then viewed as the predicted label and the value $R(c_j^*|x)$ is the misclassification cost C_{mc} with the current values of attributes in $\widetilde{A}$.

Prediction can be made based on $\widetilde{A}$. However, a test or a sequence of tests on some unknown attributes may be more preferable to reduce the misclassification cost while minimizing the total cost. To decide whether a further test is needed and, if so, which attribute $\overline{A}_i \in \overline{A}$ to select, we introduce the *utility* of testing an unknown attribute $\overline{A}_i$ as follows:

$$Util(\overline{A}_i) = Gain(A, \overline{A}_i) - C_{test}(\overline{A}_i). \tag{2}$$

$C_{test}(\overline{A}_i)$ is the test cost of $\overline{A}_i$ given by T_i. $Gain(A, \overline{A}_i)$ is the reduction in misclassification cost obtained from knowing $\overline{A}_i$'s true value, which is given by:

$$Gain(\widetilde{A}, \overline{A}_i) = C_{mc}(\widetilde{A}) - C_{mc}(\widetilde{A} \cup \overline{A}_i). \tag{3}$$

$C_{mc}(\widetilde{A}) = \min_j R(c_j|\widetilde{A})$ is easily obtained using (1). However, it is not trivial as the calculation of $C_{mc}(\widetilde{A} \cup \overline{A}_i)$ since the value of unknown $\overline{A}_i$ is not revealed until the test is performed. We calculate it by taking expectation over all possible values of $\overline{A}_i$ as follows:

$$\begin{aligned} C_{mc}(\widetilde{A} \cup \overline{A}_i) &= E\big[\min_j \big(R(c_j|\widetilde{A} \cup \overline{A}_i)|\widetilde{A}\big)\big] \\ &= \sum_{k=1}^{||\overline{A}_i||} \Big(P(\overline{A}_i = v_{i,k}|\widetilde{A}) \quad (4) \\ &\quad \times \min_j R(c_j|\widetilde{A}, \overline{A}_i = v_{i,k})\Big). \quad (5) \end{aligned}$$

In (4), the expected minimum value of misclassification cost is dependent on the values of attributes $\widetilde{A}$ known so far. In the expended form (5), the minimum misclassification given $\overline{A}_i = v_{i,k}$ is weighted by the conditional probability $P(\overline{A}_i = v_{i,k}|\widetilde{A})$ which can be obtained using Bayes' rule.

Overall, by using (2) to calculate all the utilities of testing unknown attributes in $\overline{A}$, we can decide whether a test is needed ($\exists_i Util(\overline{A}_i) > 0$) and which attribute $\overline{A}_i^*$ should be tested ($\arg\max_i Util(\overline{A}_i)$).

After the attribute $\overline{A}_i^*$ is selected and tested, the set of known attributes $\widetilde{A}$ is extended to $\widetilde{A} \cup \{\overline{A}_i^*\}$ and the corresponding $\overline{A}$ is reduced to $\overline{A}/\{\overline{A}_i^*\}$. Such a selection process is repeated until all the utilities of testing unknown attributes left is negative (unworthy) or there is no unknown attribute left.

Finally, a class label is predicted based on the extended unknown attribute set $\widetilde{A}$. The misclassification cost C_{mc} is then C_{ij} if the example of true class c_i is predicted as c_j. All the costs brought by the attribute tests are comprised of the test costs C_{test}. Consequently, the total cost $C_{total} = C_{mc} + C_{test}$ can be obtained. The details of the csNB-sequential prediction are given in Algorithm 4. As the output, the algorithm gives the prediction of a testing example x as well as the test costs C_{test} involved for some unknown attributes.

Algorithm 4 csNB-sequential-predict(NBC, R, T, x)

Input:
NBC—a cost-sensitive naive Bayes classifier,
R—a misclassification cost matrix,
T—a test cost vector,
x—a testing example.
Output:
Label—the predicted class,
C_{test}—the test costs.

```
Steps:
 1: Let $\widetilde{A}$ and $\overline{A}$ denote the set of known attributes and the
    set of unknown attributes of $x$.
 2: Set $C_{test} = 0$.
 3: while $\overline{A}$ is not empty do
 4:     for all $\overline{A}_i \in \overline{A}$ do
 5:       Calculate $Util(\overline{A}_i)$ by using (2);
 6:     end for
 7:     if not $\exists_i Util(\overline{A}_i) > 0$ then
 8:       break;
 9:     end if
10:   $\overline{A}_i^* = \max_i Util(\overline{A}_i)$
11:   Reveal $\overline{A}_i$'s value $v$.
12:   $C_{test} = C_{test} + T_{\overline{A}_i^*}$
13:   $\widetilde{A} \leftarrow \widetilde{A} \cup \{\overline{A}_i = v\}$
14:   $\overline{A} \leftarrow \overline{A}/\{\overline{A}_i^*\}$
15: end while
16: Calculate the misclassification costs $R(c_j|\widetilde{A})$ by (1).
17: Label = $\arg\min_j R(c_j|\widetilde{A})$.
```

Back to the example in Section 5.1, the utilities of the four attributes are 9.6, 8.6, 9.0, and 11.1, respectively. Therefore, the unknown attribute "ascites" will be selected in the first place. Through further calculation, a test sequence can be obtained.

A desirable property is that even though all the test costs are zero, the csNB may not do tests for all missing attributes. One reason is that the gain from knowing a new attribute value $\overline{A_i}$ is not always positive. According to (4), if the misclassification cost $C_{mc}(\widetilde{A} \cup \overline{A}_i)$ is equal to or even larger than the original one $C_{mc}(\widetilde{A})$, the gain is nonpositive. This creates a paradox: Adding new features (especially unrelated features) to a naive Bayes classifier may actually lead to a worse prediction accuracy. The basic source can be traced back to the wrong independent assumption of naive Bayes [14]. For the same reason, adding these features to the csNB can increase the misclassification costs and is therefore not preferred. Also, another reason is that the characteristics of some misclassification cost matrix can affect the testing process dramatically. As an example, suppose the entries C_{ij_0} in the j_0th column of misclassification matrix C is much smaller than other entries in C, so that the minimizing function $\min_j R(c_j|\widetilde{A})$ and (5) always have j_0 returned. In this case, the gain from any unknown attribute $\overline{A_i}$ is always zero and csNB will not do any tests even if their costs are zero.

5.4 Batch Test Strategies

The sequential test strategy is optimal in the sense that 1) it takes expectation on all possible outcomes of attribute tests and 2) the decisions are made in a sequential manner such that the next test is dependent on the previous one. However, in many situations, tests are required to be done at once due to some practical constraints, such as time. In these situations, unknown attributes are predetermined for testing in a batch manner.

Specifically, the batch test strategy is different from the sequential test strategy mainly in two facets. First, tests on unknown attributes must be determined in advance before any one of them is carried out (time constraints). Second, because of the limitation on available resources, it is impossible to test all the unknown attributes and, consequently, selection must be made among those attributes (resource constraints). Therefore, under the paradigm of batch test, a strategy $S = \{D_1, D_2, \cdots, D_t\}$ is no longer a sequence of, but a set of decisions. Still, $D_i = A_j$ represents the selection of unknown attribute A_j for testing. While two decisions are correlated due to the constraints imposed, one decision is no longer dependent on the outcome of the other as in the sequential test strategy.

While finding the optimal batch test strategy S by examining all possible subset of unknown attributes $\overline{A}$ is computationally difficult, we assume the conditional independence assumption of attributes and map the problem of finding strategy S as a knapsack problem [18]. Given a knapsack and a set of items, each with a volume and a value, the problem is to determine the number of each item to include in the knapsack so that the total volume is less than the volume of the knapsack and the total value is as large as possible.

The equivalent knapsack problem is defined as follows: Each unknown attribute $\overline{A}_i \in \overline{A}$ is viewed as an item in the currently missing item set $\overline{A}$. The volumes of each item $\overline{A}_i$ is the test cost $T_{\overline{A}_i}$ of it given by the test cost vector T. The gain of $\overline{A}_i$ given by (3) is cast as its value. Furthermore, the resource constraint T_{total} is viewed as the whole volume of the knapsack. Then, the problem is to find the most valuable set of items (a subset of $\overline{A}$) that fit in the knapsack of fixed volume T_{total} which can be solved using dynamic programming in $\Theta\{|\overline{A}|T_{total}\}$.The resulting algorithm is given in Algorithm 5.

```
Algorithm 5 csNB-batch-predict(NBC, R, T, T_{total}, $x$)
    Input: NBC—a cost-sensitive naive Bayes classifier,
    $R$—a misclassification cost matrix,
    $T$—a test cost vector,
    $T_{total}$—the total resources available,
    $x$—a testing example.
    Output:
    Label—the predicted class,
    $C_{test}$—the test costs.
Steps:
 1: Let $\widetilde{A}$ and $\overline{A}$ denote the set of known attributes and the
    set of unknown attribute of $x$.
 2: Set $C_{test} = 0$.
 3: Set $Values = \{\}$ and $Volumes = \{\}$.
 4: for all $\overline{A}_i \in \overline{A}$ do
 5:     Calculate $Gain(\widetilde{A} \cup \overline{A}_i)$ by (3)
 6:     if $Gain(\overline{A}_i) > C_{test}(\overline{A}_i) = T_{\overline{A}_i}$ then
 7:         $Values \leftarrow Values \cup \{Gain(\overline{A}_i)\}$
 8:         $Volumes \leftarrow Volumes \cup \{T_{\overline{A}_i}\}$
 9:     end if
10:   end for
11: $(\{\overline{A}_i^*\}, C_{test}) = \mathrm{KNAPSACK}(Values, Volumes)$
12: Reveal the values of attributes in $\{\overline{A}_i^*\}$.
13: $\widetilde{A} \leftarrow \widetilde{A} \cup \{\overline{A}_i^* = v_{i,k}^*\}$
14: Calculate the misclassification costs $R(c_j|\widetilde{A})$ by (1).
15: Label = $\arg\min_j R(c_j|\widetilde{A})$.
```

TABLE 3
Data Sets Used in the Experiments

Name of datasets	No. of attributes	Name of datasets	No. of attributes
Ecoli	6	Breast	9
Heart	8	Thyroid	24
Australia	15	Cars	6
Voting	16	Mushroom	22

In Step 11 of the csNB-batch algorithm, the outputs of the function $\text{KNAPSACK}(\cdot)$ are the selected items $\overline{A}_i^*$ (unknown attributes) and the total test costs $C_{test} \leq T_{total}$. Similarly, the misclassification cost C_{mc} can be calculated in the same way as in csNB-sequential and, subsequently, C_{total} can be obtained.

6 Experiments

In order to evaluate the performance of the TCSL framework for both decision-trees and naive Bayesian methods, in both sequential and batch test manners, several experiments were carried out on data sets from the UCI ML repository [15] as well as a real-world data set from an insurance company. The eight data sets used are listed in Table 3. These data sets were chosen because they have discrete attributes, binary class, and a sufficient number of examples. We only consider binary class problems (positive and negative) in the following experiments, although our TCSL algorithms can be used in multiple class problems naturally. The numerical attributes in data sets were discretized using minimal entropy method [18].

We ran a three-fold cross validation on these data sets. For the testing examples, a certain percentage (missing rate) of attributes are randomly selected and marked as unknown. If during testing, an algorithm decides to perform a test on an unknown attribute, the real value of the attribute is revealed and the test cost C_{test} is accumulated. Finally, the misclassification cost C_{mc} can be obtained by comparing the predicted label with the true class label. The performance of the algorithms is therefore measured in terms of the total cost C_{total}, the sum of C_{test} and C_{mc}. To the two-class problems, let c_1 be the positive class and c_2 the negative class. The misclassification matrix was set as $R_{12} = R_{21} = 400$ and $R_{11} = R_{22} = 0$, where R_{12} can be interpreted as false negative and R_{21} false positive. Test costs of each attributes in T are set randomly between 0 and 100.

For algorithm comparison, two variations of traditional naive Bayes classifiers were used as the baselines. The first one is the naive Bayes classifier augmented to minimize the misclassification cost (conditional risk), as given in [14]. This classifier is termed Lazy Naive Bayes (LNB) since it simply predicts class labels based on the known attributes and requires no further tests to be done on unknown ones. The second variation is the naive Bayes classifier extended further from LNB. It requires all the missing values to be made up before prediction. Since this classifier allows no missing values, it is therefore termed Exacting Naive Bayes (ENB). Comparisons were also made between csNB and the Test-Cost-Sensitive Decision Trees (csDT) proposed in Section 4.

In summary, four methods were examined:

- LNB—Lazy naive Bayes,
- ENB—Exacting naive Bayes,
- csNB—Cost-sensitive naive Bayes, and
- csDT—test-cost-sensitive decision trees with the sequential test strategy (follow the tree).

We note that the decision tree algorithm we use here is a rather simple minded one: It simply follows the tree sequentially to know which test to perform in order to obtain a next missing value. Other more sophisticated methods can be further developed, which is our ongoing work.

6.1 Comparing Decision Tree and Naive Bayesian

We first compared the cost-sensitive decision-tree algorithm with sequential test strategy (follow the tree) with the

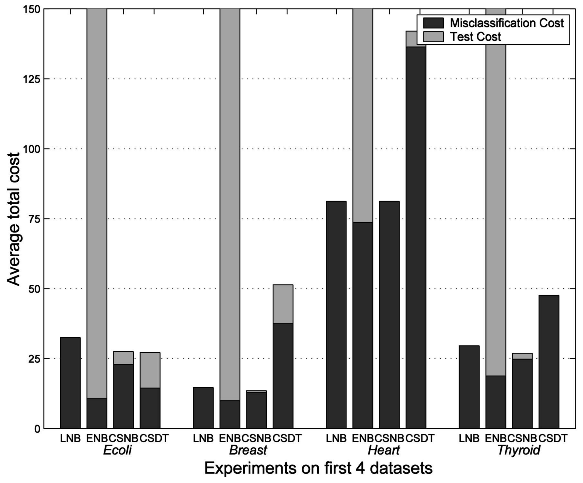

Fig. 1. Average total cost comparisons of four methods on data sets: Ecoli, Breast, Heart, and Thyroid.

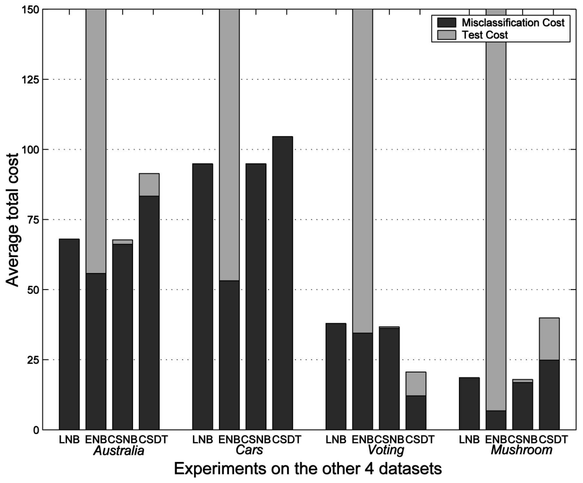

Fig. 2. Average total cost comparisons of four methods on data sets: Australia, Cars, Voting, and Mushroom.

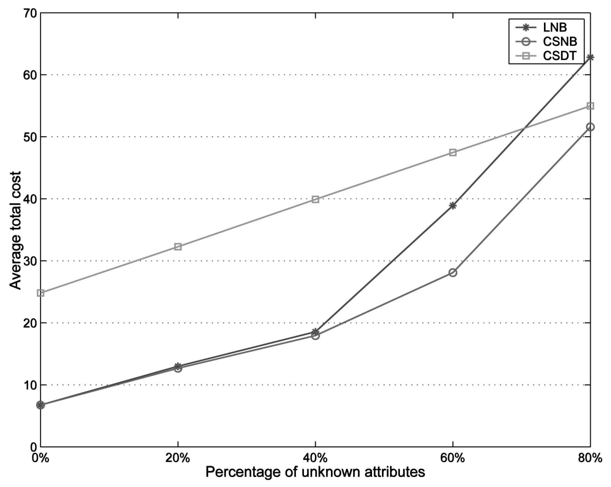

Fig. 3. Comparisons with varying missing rates.

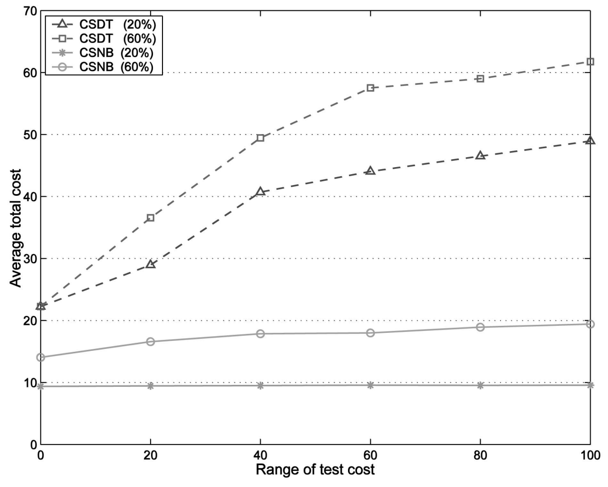

Fig. 4. Comparisons on the Breast data set with varying test costs.

different cost-sensitive naive Bayesian algorithms. Fig. 1 and Fig. 2 show the results of different algorithms in sequential test strategy on all the eight data sets. In these experiments, the percentage of unknown attributes (missing rate) was 40 percent. Each group of four bars represents the runs of four algorithms on one particular data set. The height of a bar represents the average total cost and, therefore, the lower the better. Each bar consists of two parts: The lower dark portion stands for the misclassification cost while the upper light portion stands for the test costs.

There are several interesting observations from these experiments. First, although the misclassification costs of the ENB method are almost always the lowest among the four methods, the average total costs are the highest. This is because the low misclassification costs are achieved at the cost of testing all unknown attributes, which is costly when the missing rate is high.

Second, despite its lazy nature, the LNB method performs surprisingly well, even better than the csDT method. This can be explained by the fact that, while csDT uses the splitting criterion of minimal total costs for attribute selection in tree building, whenever trees are built, all tests are fixed. Only the values of those attributes associated with tree nodes along the paths are examined and the others are omitted. However, the other attributes that are on other branches of a tree can still be informative for classification. LNB, on the other hand, is capable of making use of these attributes. For the example in Section 5.1, suppose that the attributes "ascites" and "Spiders" are known. csDT may choose "ascites" as a splitting node and leave "Spider" out, while csNB still utilizes both attributes.

To investigate the impact of the percentage of unknown attributes on the total costs, experiments were carried out on average total cost with the increasing percentage of unknown attributes. Fig. 3 shows the results on the Mushroom data set (other figures are spared for space). As we can see, when the percentage increases (> 40 percent), the average total cost of LNB increases significantly and surpasses that of csDT. Again, csNB is better than the other two over the whole range.

Another set of experiments was conducted to compare two cost-sensitive algorithms csDT and csNB in test costs. The misclassification costs are still fixed at 400 (FP) and 400 (FN). Fig. 4 and Fig. 5 are the results on the Breast and Mushroom data sets with both the missing rate 20 percent and 60 percent. One can see that, when the test costs are small (below 20), csDT wins overall. However, as the test costs increase, csNB outperforms csDT. One thing to note is that csNB is less sensitive to the test costs than csDT. This reveals that the csNB algorithm is better at balancing the misclassification and test costs.

6.2 Comparison to Other Cost-Sensitive Learning Systems

In addition, we consider comparing with three main previous works in cost sensitive learning. Incorporating attribute costs into decision trees is an established research direction in coping with cost-sensitive learning. As we reviewed in Section 2, several approaches have been proposed that have mainly focused on fixing existing decision tree-based algorithms. They include ICET [19],

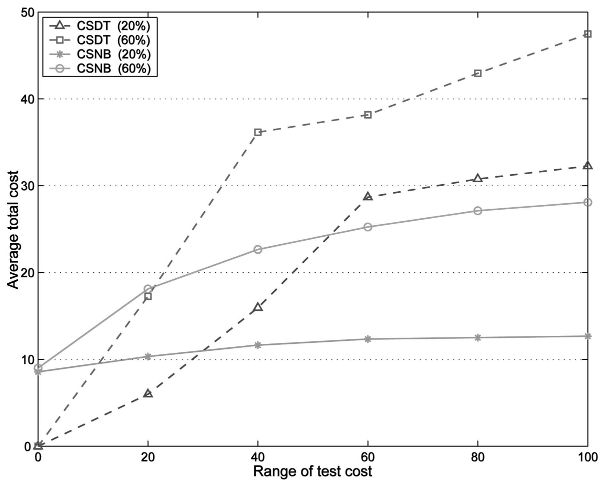

Fig. 5. Comparisons on the Mushroom data set with varying test costs.

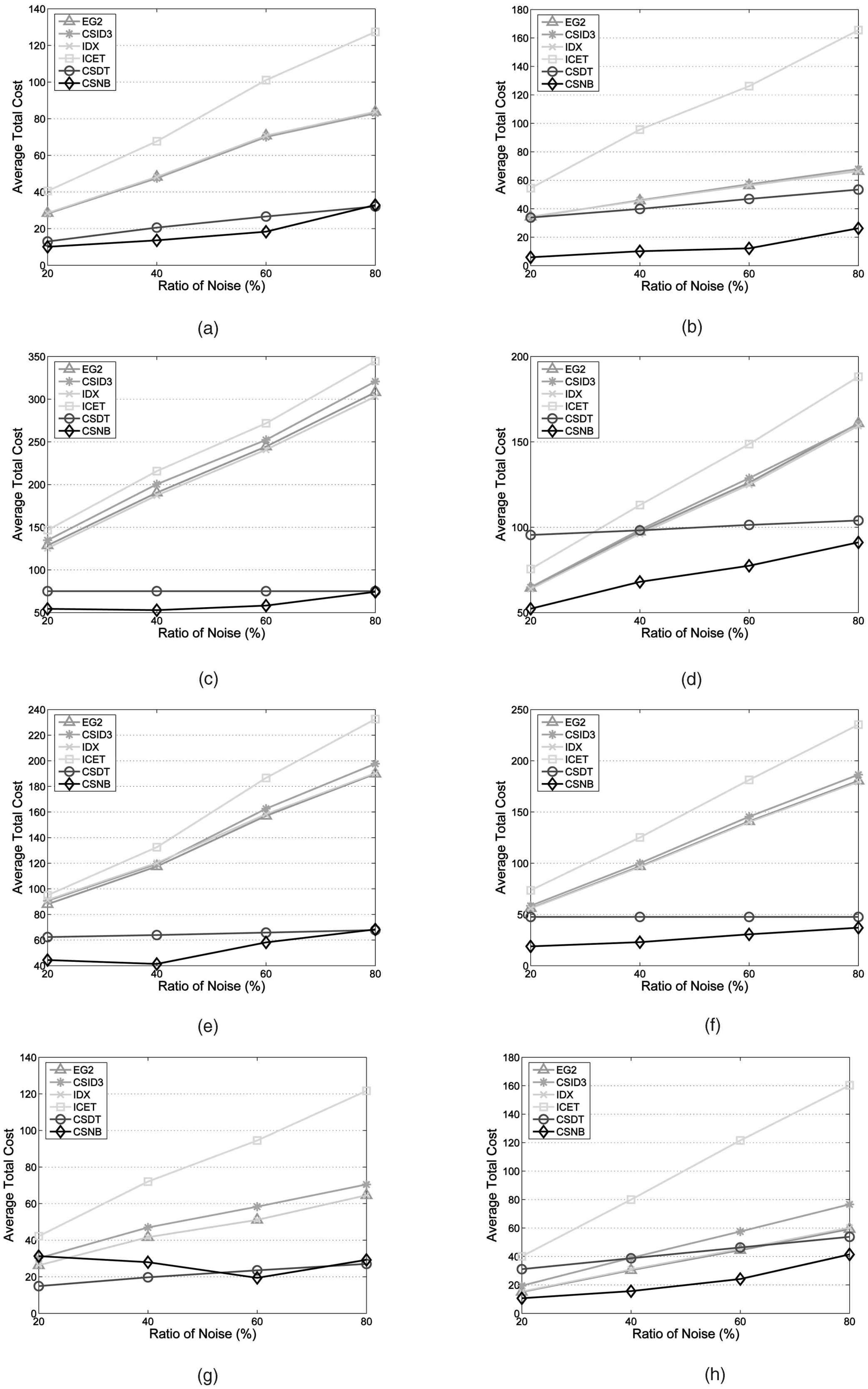

Fig. 6. Comparisons of CSDT and CSNB against other cost-sensitive algorithms: EG2, CS-ID3, IDX, and ICET on eight UCI data sets. (a) Ecoli. (b) Breast. (c) Heart. (d) Cars. (e) Australia. (f) Thyroid. (g) Voting. (h) Mushroom.

EG2 [20], CS-ID3 [21], and IDX [22]. In this section, we compare with these systems empirically.

EG2 [20] is a top-down decision-tree induction algorithm that uses the Information Cost Function ($ICF(A_i)$) for

TABLE 4
Attribute Names and Test Costs of the *Insurance* Data Set

Attr1 (Age)	Attr2 (Sex)	Attr3(Income)	Attr4 (No. of Cars)	Attr5	Attr6	Attr7	Attr8	Attr9	Attr10
16	7	23	17	11	28	31	5	9	14

selecting an attribute A_i, instead of the information gain used in C4.5. The selection score for the attribute A_i is defined as

$$ICF(A_i) = \frac{2^{\Delta I(A_i)} - 1}{(\text{cost}(A_i) + 1)^{\omega}}. \quad (6)$$

We implemented EG2 using this equation. In this equation, the value of ω needs to be determined. In the EG2 system, the value of ω is set to a constant value of one. In ICET [19], this value is learned through a genetic algorithm. We have implemented both EG2 and ICET for the comparison.

In addition, we have implemented and compared with the CS-ID3 system of [21]. Our implementation uses the following formula for attribute selection:

$$CSID3(A_i) = \frac{\Delta I(A_i)^2}{\text{cost}(A_i)}. \quad (7)$$

Finally, the IDX [22] system uses a similar but simpler heuristic

$$IDX(A_i) = \frac{\Delta I(A_i)}{\text{cost}(A_i)}. \quad (8)$$

We have done a comprehensive set of experiments to compare both our test-cost sensitive decision tree and naive Bayesian algorithms with these previous works. Furthermore, we have tested our system on a new real-world domain. The results are shown in Fig. 6.

Fig. 6 shows the comparison against previous methods. As can be seen, our methods csDT and csNB both preform much better on the UCI domains Ecoli, Heart, Australia, Thyroid, and so on, for increasing noise ratios. csNB performs the best in almost all the testing domains. For these tests, we used different settings of FP and FN costs: $FP = 400$ and $FN = 200$. The attribute costs are the same as that assumed in Section 4.1. Our conclusion is that both of our proposed algorithms work well on different domains with different levels of noise.

6.3 Evaluation on Real Data

In order to test the system's ability to handle real-world data, we have also tested all algorithms on an insurance data set collected with an insurance company in Canada. It consists of more than 900 records for customers who have the status of "stay" or "leave" the insurance company. Our target is to determine whether the customer will continue to stay with the insurance company or not. We will refer to them as positive and negative, respectively. The data set is described by more than 10 attributes that are associated with costs. The FP cost is 400 and the FN cost is 200. Some attributes and their costs are listed in Table 4. The idea is that, when some attributes have missing value, it costs the insurance company extra resources to obtain their values. Thus, in order to reduce costs, it is natural to apply test-cost-sensitive learning. In the test, we used three-fold cross validation.

The result of the empirical tests are shown in Fig. 7. As can be seen, both of our methods again wins by a large margin over the previous methods. This result demonstrates the effectiveness of csDT and csNB in their ability to reduce the total costs in a real-world setting.

6.4 Batch Test Strategy

Batch test is also investigated by conducting several experiments. In order to compare CSNB with CSDT in terms of their abilities in batch test strategies, set the parameters T_{total} given in Algorithm 5.

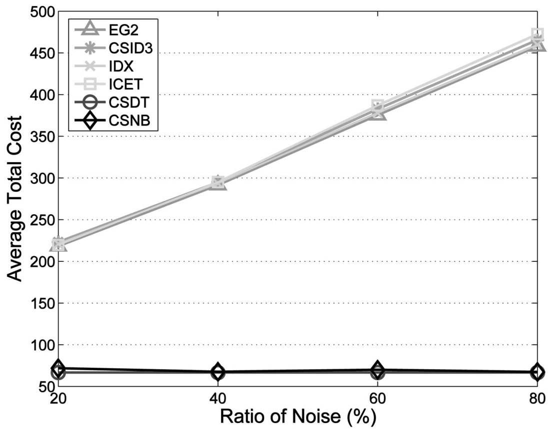

Fig. 7. Comparisons on the *Insurance* data set.

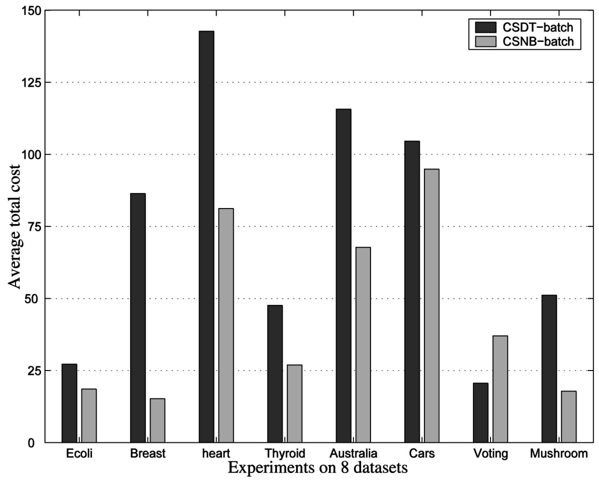

Fig. 8. Comparisons in batch test on all the eight data sets.

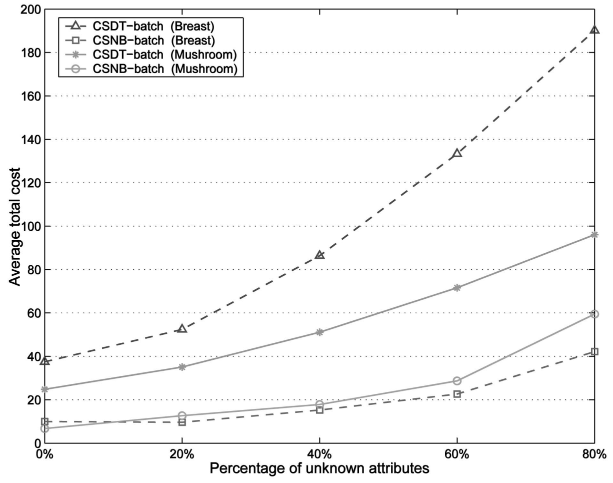

Fig. 9. Comparisons in batch test with varying missing rates.

The results of average total cost on the eight data sets with 20 percent missing rate are shown in Fig. 8. The resource constraint T_{total} is set to 100. Overall, csNB-batch outperforms csDT-batch greatly. This reveals that, although both algorithms aim to minimize the total cost, csNB trades off the misclassification cost and the test costs much better than csDT, especially when resource constraints are imposed.

Fig. 9 shows the two runs on the Breast and Mushroom data sets with the variation of percentage of unknown attributes. As we can see, csNB-batch is less sensitive to the missing-value rate and performs much better than csDT-batch.

7 Conclusions and Future Work

In this paper, we proposed a test-cost-sensitive earning framework for designing classifiers that minimize the sum of the misclassification cost and the test costs. In the framework of TCSL, attributes are selected for testing intelligently to get both the sequential test strategy and the batch test strategy. Experiments show that our method outperforms other competing algorithms. We observe that the decision tree algorithm we use here is a rather simple-minded one in that it simply follows the tree sequentially to obtain a next missing value. In the future, we plan to consider more sophisticated test strategies for decision trees. We will also consider the conditional test costs [3] in which the cost of a certain test is conditional on the other attributes. For example, in medical diagnosis, the cost of an exercise stress test on a patient may be conditional on whether the patient has heart disease or not. Another direction is to consider group tests where the cost of performing tests on a group of samples are cheaper than the sum of the costs spent individually.

Acknowledgments

The authors would like to thank Hong Kong RGC for their support under grant No. HKUST 6187/04E. They also thank the anonymous referees for their comments.

References

[1] T.M. Mitchell, *Machine Learning.* McGraw Hill, 1997.
[2] J.R. Quinlan, *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, 1993.
[3] P.D. Turney, "Types of Cost in Inductive Concept Learning," *Proc. Workshop Cost-Sensitive Learning at the 17th Int'l Conf. Machine Learning,* 2000.
[4] C. Elkan, "The Foundations of Cost-Sensitive Learning," *Proc. 17th Int'l Joint Conf. Artificial Intelligence,* pp. 973-978, 2001.
[5] P. Domingos, "Metacost: A General Method for Making Classifiers Cost-Sensitive," *Knowledge Discovery and Data Mining,* pp. 155-164, 1999.
[6] M.T. Kai, "Inducing Cost-Sensitive Trees Via Instance Weighting," *Principles of Data Mining and Knowledge Discovery, Second European Symp.,* pp. 139-147, 1998.
[7] M. Nunez, "The Use of Background Knowledge in Decision Tree Induction," *Machine Learning,* vol. 6, pp. 231-250, 1991.
[8] M. Tan, "Cost-Sensitive Learning of Classification Knowledge and Its Applications in Robotics," *Machine Learning J.,* vol. 13, pp. 7-33, 1993.
[9] C. Ling, Q. Yang, J. Wang, and S. Zhang, "Decision Trees with Minimal Costs," *Proc. 2004 Int'l Conf. Machine Learning,* 2004.
[10] 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.
[11] V.B. Zubek and T.G. Dietterich, "Pruning Improves Heuristic Search for Cost-Sensitive Learning," *Proc. 19th Int'l Conf. Machine Learning,* pp. 27-34, 2002.
[12] R. Greiner, A. Grove, and D. Roth, "Learning Cost-Sensitive Active Classifiers," *Artificial Intelligence J.,* vol. 139, no. 2, pp. 137-174, 2002.
[13] X. Chai, L. Deng, Q. Yang, and C.X. Ling, "Test-Cost Sensitive Naive Bayesian Classification," *Proc. 2004 IEEE Int'l Conf. Data Mining (ICDM '04),* Nov. 2004.
[14] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification,* second ed. Wiley and Sons, Inc., 2001.
[15] P. Domingos and M. Pazzani, "On the Optimality of the Simple Bayesian Classifier Under Zero-One Loss," *Machine Learning,* vol. 29, pp. 103-130, 1997.
[16] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms,* second ed. McGraw Hill and MIT Press, 2001.
[17] C.L. Blake and C.J. Merz, "UCI Repository of Machine Learning Databases," http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.
[18] U.M. Fayyad and K.B. Irani, *Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning,* pp. 1022-1027. Morgan Kaufmann, 1993.
[19] P.D. Turney, "Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm," *J. Artificial Intelligence Research (JAIR),* vol. 2, pp. 369-409, 1995.
[20] M. Núñez, "Economic Induction: A Case Study," *Proc. European Working Session on Learning,* pp. 139-145, 1988.
[21] M. Tan, "Cost-Sensitive Learning of Classification Knowledge and Its Applications in Robotics," *Machine Learning,* vol. 13, pp. 7-33, 1993.
[22] S.W. Norton, "Generating Better Decision Trees," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 800-805, 1989.

Qiang Yang received the PhD degree from the University of Maryland, College Park. He is a faculty member in the Hong Kong University of Science and Technology's 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*. His homepage is http://www.cs.ust.hk/~qyang.

Charles 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 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*, *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 *IEEE Transactions on Knowledge and Data Engineering* and a guest editor for several journals. He is also the director of 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. See http://www.csd.uwo.ca/faculty/cling for more info.

Xiaoyong Chai received the MPhil degree in computer science from the Hong Kong University of Science and Technology in 2005. Currently, he is a PhD student in the Department of Computer Sciences at Purdue University. His research interests include artificial intelligence and data mining.

Rong Pan received the BSc and PhD degrees in applied mathematics from Zhongshan University, China, in 1999, and 2004, respectively. He is a postdoctoral fellow at the Hong Kong University of Science and Technology. His research interest includes machine learning, data mining, and case-based reasoning. His homepage is http://www.cs.ust.hk/~panrong.