Test-Cost Sensitive Naive Bayes Classification

Xiaoyong Chai, Lin Deng and Qiang Yang
Department of Computer Science
Hong Kong University of Science and Technology
Clearwater Bay, Kowloon, Hong Kong, China
{carnamel, ldeng, qyang}@cs.ust.hk

Charles X. Ling
Department of Computer Science
The University of Western Ontario
London, Ontario N6A 5B7, Canada
cling@csd.wuo.ca

Abstract

*Inductive learning techniques such as the naive Bayes and decision tree algorithms have been extended in the past to handle different types of costs mainly by distinguishing different costs of classification errors. However, it is an equally important issue to consider how to handle the test costs associated with querying the missing values in a test case. When the value of an attribute is missing in a test case, it may or may not be worthwhile to take the effort to obtain its missing value, depending on how much the value will result in a potential gain in the classification accuracy. In this paper, we show how to obtain a test-cost sensitive naive Bayes classifier (*csNB*) by including a test strategy which determines how unknown attributes are selected to perform test on in order to minimize the sum of the misclassification costs and test costs. We propose and evaluate several potential test strategies including one that allows several tests to be done at once. We empirically evaluate the* csNB *method, and show that it compares favorably with its decision tree counterpart.*

1. Introduction

Inductive learning techniques such as the naive Bayes and decision tree algorithms, have met great success in building classification models with the aim to minimize the classification errors [9][12]. As an extension, much previous inductive learning research has also considered how to minimize the costs of classification errors, such as the cost of false positive (FP) and the cost of false negative (FN) in binary classification tasks. The misclassification costs are useful in deciding whether a learned model tends to make correct decisions on assigning class labels for new cases, but they are not the only costs to consider in practice. When performing classification on a new case, we often consider the "test costs" 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. Suppose that the doctors' previous experience has been compiled into a classification model such as a naive Bayes classifier. When diagnosing a new patient, it is often the case that certain information for this patent may not yet be known; for example, the blood test or the X-ray test may not have been done yet. Performing these tests will incur certain extra costs, but different tests may provide different informational values towards minimizing the misclassification costs. It is the balancing act of the two types of costs – namely the misclassification costs and the test costs – that determines which tests will be done.

Tasks that incur both misclassification and test costs abound in practice ranging from medical diagnosis to scientific research to drug design. One possible approach is to use the strategy in naive Bayes classification in dealing with missing values. That is, when a test case is classified by a naive Bayes classifier, and an attribute is found to have a missing value, no test will be performed to obtain its value; instead, the attribute is simply ignored in the posterior computation. The problem with this approach is that it ignores the possibility of obtaining the missing value with a cost, and thus reducing the misclassification cost and the total cost.

Inductive learning methods that consider a variety of costs are often referred to as *cost-sensitive learning* [15][5]. In this paper, we refer to cost-sensitive learning that specifically considers test costs as *test-cost sensitive learning*. We propose a test-cost sensitive naive Bayes algorithm (*csNB* for short) to minimize the sum of the misclassification costs and the test costs. The naive Bayes algorithm can be extended straightforwardly to incorporate the concept of "costs" by minimizing the risk instead of the classification error [4]. However, we observe that so far, few extensions have been made to consider naive Bayes classification with associated test costs for obtaining the missing values. In ad-

dition, different test strategies will result in different decisions on how the tests are performed. In this paper we consider two types of test strategies: the sequential test strategy and the batch test strategy. The former takes tests for missing values sequentially. Decisions on whether an additional test is needed or which unknown attribute should be tested next are made sequentially based on the outcome of the previous tests. The latter, the batch test strategy, requires several tests to be done at once rather than in a sequential manner. This scenario is more practical. For example, it is often the case that doctors need to have a number of test results all at once before making a diagnosis.

The novelty of our work can be seen as follows:

1. Previous work on naive Bayes classification has mostly considered how to reduce the misclassification costs by considering different classification risks. In our *csNB* framework, we additionally consider the test costs and aims to minimize the sum of them.

2. Previous work on test-cost sensitive learning has considered how to use decision trees, coupled with a sequential test strategy, to decide which attributes to perform test on one by one. In contrast, we consider a natural extension in *csNB* by which a batch test strategy can be easily derived and performed effectively.

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, and then minimizing the total misclassification costs. In Turney's survey article [15], a whole variety of costs in machine learning are analyzed, and the test cost is singled out as one of the least considered. In particular, two types of costs are considered:

- Misclassification costs: these are the costs incurred by classification errors. Works such as [2][5][7] considered classification problems with non-uniform misclassification costs.

- Test costs: these are the costs incurred for obtaining attribute values. Previous work such as [10][13] considered the test costs alone without incorporating misclassification cost. As pointed out in [15], it is obviously an oversight.

As far as we know, the only works that considered both misclassification and test costs include [8][16][6][14]. Of these works, [8] proposed a decision tree based method that explicitly considers how to directly incorporate both types of costs in decision tree building processes and in determining the next attribute to test, should the attributes contain missing values. Their method naturally extends the decision tree construction algorithm by using *minimal cost* as the splitting criterion and builds a sequential test strategy in a local search framework. Through experimentation with this method, however, we have also found some shortcomings. Because a decision tree places different levels of importance on the attributes by the natural organization of the tree, it cannot be easily fitted to make flexible decisions on selecting unknown attributes for tests. Furthermore, a decision tree is not well-suited for performing batch tests that involve a number of tests to be done together, since it is aimed at serializing attribute tests along its paths. In contrast, the naive Bayes based algorithms overcome these difficulties more naturally. As we will see, the performance offered by the test-cost sensitive naive Bayes is significant over its decision-tree counterpart.

In [16], the cost-sensitive learning problem is cast as a Markov Decision Process (MDP), and solutions are given as searches in a state space for optimal policies. For a given new case, depending on the values obtained so far, the resulting policy can suggest a best action to perform in order to minimize both the misclassification and test costs. However, it may take very high computational cost to conduct the search process. In contrast, we adopt the local search algorithm using the concepts of utility and gain, which is more efficient and also offers high quality solutions.

Similar in the interest in constructing an optimal learner, [6] studied the theoretical aspects of active learning with test costs using a PAC learning framework. [14] presented a system called ICET, which uses a genetic algorithm to build a decision tree to minimize the sum of both costs. In contrast, because our algorithm essentially adopts the conditional probability based framework, which requires only a linear scan through the dataset, our algorithm is expected to be more efficient than Turney's genetic algorithm based approach.

3. Test-cost sensitive naive Bayes

Naive Bayes classifier is shown to perform very well in practice to minimize classification errors, even in many domains containing clear attribute dependences [3]. 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 the Bayes' rule:

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

x is predicted to belong to the class c_{j^*} where $j^* = \arg\max_j P(c_j|x)$. 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.

However, classification errors are not the only criteria in evaluating a learned model. In practice, costs involved during classification are even more important in deciding whether the model is effective in making correct decisions. Therefore, a naive Bayes classifier should be extended to be cost-sensitive.

3.1. Costs in Naive Bayes Classification

In this paper, we consider two types of costs in naive Bayes classification: the misclassification cost and the test cost.

The misclassification cost is considered when there are different types of classification errors, and the costs they bring are different. The standard naive Bayes algorithm (NB) can be extended to take the misclassification cost into account. Suppose that C_{ij} is the cost of predicting a sample of class c_i as belonging to class c_j. The expected misclassification cost of predicting sample x as class c_j is also known as the *conditional risk* [4], which is defined as: $R(c_j|x) = \sum_i C_{ij} \times P(c_i|x)$, where $P(c_i|x)$ is the posterior probability given by a NB classifier. Sample x is then predicted to belong to class c_{j^*} which has the minimal conditional risk $R(c_{j^*}|x) = \min_j R(c_j|x)$.

To consider the test cost, take medical diagnosis as an example. Suppose that in diagnosing the disease of hepatitis, 21% of patients are positive (have hepatitis, c_1) and 79% are negative (healthy, c_2). Therefore, the priors are $P(c_1) = 21\%$ and $P(c_2) = 79\%$, respectively. The costs of different classification errors can be specified by setting the corresponding C_{ij}. Assume that the costs (conditional risk) are $C_{12} = 450$, $C_{21} = 150$, and $C_{11} = C_{22} = 0$.

Suppose there are four attributes to characterize a patient, and testing the value (*positive* or *negative*) of each attribute brings a certain amount of cost. The test costs of the four attributes and their likelihoods are listed in Table 1 below.

Attributes	Test Cost	Hepatitis (c_1)		healthy (c_2)	
		Pos	Neg	Pos	Neg
liver firm	18	51.7%	48.3%	61.8%	38.2%
spleen	24	56.9%	43.1%	81.9%	18.1%
spiders	26	29.0%	71.0%	75.6%	24.4%
ascites	32	54.8%	45.2%	95.0%	5.0%

Table 1. Likelihoods of attributes

When a patient first comes, values of these four attributes are unknown. To diagnose whether the patient has hepatitis or not, a doctor must decide whether a medical test is worthwhile to perform and if so, which one. Each test has its own discriminating power on disease and meanwhile, brings a certain amount of cost. Therefore, decisions must be made by considering both factors. After a test is selected and performed, based on its outcome, similar decisions are made subsequently on the unknown attributes left. As a consequence, during the process of diagnosis, the doctor adopts a sequential test strategy with the aim to minimize the sum of the misclassification cost and test costs.

In practice, the situation 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 these cases, for the sake of patients, doctors often ask for a batch of tests simultaneously. Therefore, a batch test strategy must consider the costs of the several tests done in one shot.

3.2. Problem Formulation

The classification problem of Test-Cost Sensitive Naive Bayes is formulated as follows:

Given: (D, C, T), where

- D is a training dataset 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.
- C is a misclassification cost matrix. Each entry $C_{ij} \triangleq C(i, j)$ specifies the cost of classifying a sample from class c_i as belonging to class c_j $(1 \leq i, j \leq P)$. Usually, $C_{ii} = 0$.
- T is a test-cost vector. Each entry $T_k \triangleq T(k)$ specifies the cost of taking a test on attribute A_k $(1 \leq k \leq M)$;

Build: a test-cost sensitive naive Bayes classifier *csNB* and for every test case, a test strategy (see Section 4) with the aim to minimize the sum of the misclassification cost C_{mc} and test cost C_{test}.

The above formulation provides a more general framework than the traditional naive Bayes does. Actually, the latter is just a special case of *csNB* where the test costs T_k are sufficiently large so that no test will be performed. Also, the conditional risk [4] can be equivalently implemented by setting the misclassification cost matrix C.

csNB classification consists of two procedures: First, a *csNB* classifier is learned from the training dataset D. Second, for each test case, a test strategy is designed to minimize the total cost based on the *csNB* obtained.

Learning a *csNB* classifier is basically the process of estimating the distribution parameters as in traditional NB. Let $c_j \in \{c_1, c_2, \cdots, c_P\}$ be the jth predefined class and $v_{m,k} \in \{v_{m,1}, v_{m,2}, \cdots, v_{m,|A_m|}\}$ be one of the possible

values attribute A_m can take. The learning procedure is exactly the estimation of prior probabilities $\hat{P}(c_j)$ and likelihoods $\hat{P}(A_m = v_{m,k}|c_j)$ from the training dataset D. In addition, when there are missing values in the training examples, the corresponding attributes are just ignored in the likelihood computation.

The real intriguing problem is how to design a test strategy for each test case with the aim to minimize the sum of the misclassification cost C_{mc} and test cost C_{test}. It is essentially an optimization problem that minimizes the total cost. However, to find an optimal test strategy given a test case is computationally difficult, since the problem can also be equivalently formulated as a MDP as in [16] which is shown to be NP-hard [11]. The problem is more complicated when different types of test strategies are demanded, such as the sequential test strategy (Section 4) and batch test strategy (Section 5). In this paper, we are interested in finding approximation solutions.

4. Prediction with Sequential Test Strategy

When a new test case with missing values comes, a *csNB* classifier need to design a test strategy as to how and which unknown attributes are selected to test. In this section, we consider sequential test strategies and leave batch test strategies to Section 5.

A sequential test strategy is as follows. During the process of classification, based on the results of previous tests, decisions are made sequentially on whether a further test on an unknown attribute should be performed, and if so, which attribute to select. More specifically, the selection of a next unknown attribute to test is not only dependent on all the values of initially known attributes, but also dependent on the values of those unknown attributes previously tested.

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

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

where $P(c_j|\widetilde{A}) = \frac{P(\widetilde{A}|c_j)P(c_j)}{P(\widetilde{A})}$ is the posterior probability obtained using Bayes' rule.

Prediction can be made based on $\widetilde{A}$. c_{j^*} with the minimum expected cost is predicted as the class label. Finally, the misclassification cost C_{mc} is C_{ij^*} if c_i is the actual class label of x. The test cost C_{test} is 0, since no test is performed. However, a sequence of tests on some unknown attributes may be more preferable to reduce the misclassification cost and thus to minimize the total cost. To decide whether a 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:

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

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

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

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

$$C_{mc}(\widetilde{A} \cup \overline{A}_i) = E_{\overline{A}_i}\left[\min_j \left(R(c_j|\widetilde{A} \cup \overline{A}_i)\right)\right] \quad (4)$$

$$= \sum_{k=1}^{||\overline{A}_i||} P(\overline{A}_i = v_{i,k}|\widetilde{A}) \times \min_j R(c_j|\widetilde{A}, \overline{A}_i = v_{i,k}) \quad (5)$$

In Equation (4), the expected minimum misclassification cost is conditional on the values of attributes $\widetilde{A}$ known so far. In the expended form (5), the minimum misclassification cost 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, an attribute $\overline{A}_i$ is worth testing on if testing it offers more gain than the cost it brings. Therefore, by using Equation (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 $\overline{A}_{i^*}$ to test ($i^* = \arg\max_i Util(\overline{A}_i)$).

After the attribute $\overline{A}_{i^*}$ is tested, its true value is revealed. The set of known attributes $\widetilde{A}$ is expanded to $\widetilde{A} \cup \{\overline{A}_{i^*}\}$ and correspondingly, $\overline{A}$ is reduced to $\overline{A}/\{\overline{A}_{i^*}\}$. Such a selection process is repeated until the utility of testing any unknown attribute is non-positive or there is no unknown attribute left. A class label is then predicted based on the expanded known attribute set $\widetilde{A}$.

Finally, the misclassification cost C_{mc} is C_{ij} if example x predicted as class c_j is actually from class c_i. All the costs brought by the attribute tests comprise the test cost C_{test}. Consequently, the total cost $C_{total} = C_{mc} + C_{test}$. The details of the csNB-sequential prediction are given in Algorithm 1. As the output, the algorithm gives the prediction of a test example x as well as the test cost C_{test} included.

Back to the example in Section 3.1, initially $\widetilde{A} = \phi$ and $\overline{A} = A$ since all the four attributes are unknown. At step 5, the utilities of the four attributes are calculated, which are -6.5, 8.6, 22.2 and 24.1, respectively. Consequently, at step 10, the attribute "ascites" with the maximum utility 24.1 is

selected to test and its true value is revealed. "ascites" is then removed from $\overline{A}$ to $\widetilde{A}$. Attribute selection for testing in the next round will be different depending on the outcome of the test on "ascites". If it is $positive$, the attribute "spleen" is chosen for testing; otherwise, the attribute "spiders" is selected. The same process continues and finally a sequential test strategy can be obtained during classification.

Algorithm 1 csNB-sequential-predict(x, cl)

Input: x — a test example, cl — a *csNB* classifier;
Output: Label — the predicted class, C_{test} — the test cost;
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 $Util(\overline{A}_i)$ using Equation (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 missing 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 expected misclassification costs $R(c_j|\widetilde{A})$ by Equation (1).
17: Label = $\arg\min_j R(c_j|\widetilde{A})$.

A desirable property is that even when all the test costs are zero, *csNB* may not do tests for all the missing attributes. One reason is that the gain from knowing the missing value of an attribute $\overline{A}_i$ is not always positive. According to Equation (3), if the expected misclassification cost $C_{mc}(\widetilde{A} \cup \overline{A}_i)$ is equal to or even larger than the original cost $C_{mc}(\widetilde{A})$, the gain is non-positive. This creates a paradox: adding new features (especially unrelated features) to a naive Bayes classifier may actually lead to more misclassification cost. The basic source can be traced back to the wrong independent assumption of naive Bayes [4]. For the same reason, adding these features to *csNB* can increase the misclassification cost and is therefore not preferred. Another possible reason is that the characteristics of the misclassification cost matrix C can affect the test strategy. As an example, suppose the entries $C_{\cdot j_0}$ in the j_0th column of matrix C is much smaller than other entries in C, so that the minimizing functions, $\arg\min_j R(c_j|\widetilde{A})$ and $\arg\min_j R(c_j|\widetilde{A} \cup \overline{A}_i)$, always have j_0 returned. In this case, the gain from any unknown attribute $\overline{A}_i$ may be zero and *csNB* will not do any test even if no cost is brought.

5. Prediction with Batch Test Strategy

A sequential test strategy is optimal in the sense that (1) it takes expectation over all possible outcomes of attribute tests, and (2) decisions are made in a sequential manner such that the next selection is dependent on the test results of the previous ones. However, in many situations, tests are required to be done all at once due to some practical constraints, such as time. In these situations, several unknown attributes are tested simultaneously and a batch test strategy is needed instead.

Specifically, while both batch test strategies and sequential test strategies aim to minimize the total cost, they are different in that: In batch test, tests on unknown attributes must be determined in advance before any one of them is carried out; therefore, strategies are designed beforehand. In sequential test, as discussed in Section 4, strategies are designed on the fly during prediction.

To find an optimal batch test strategy for a new test example, one possible way is to examine all possible subsets of unknown attributes $\overline{A}$ by calculating the utilities, and choose the one with the maximum utility. Let $\overline{A}'$ denote a subset of $\overline{A}$ ($\overline{A}' \subseteq \overline{A}$). $C_{mc}(\widetilde{A} \cup \overline{A}')$, the expected minimum misclassification cost is calculated to obtain its utility. To achieve it, expectation is taken over all possible value combinations of the unknown attributes in $\overline{A}'$. However, it is computationally difficult to do so.

By assuming the conditional independence among attributes, we can extend the sequential test algorithm using a greedy method. The idea is that in each round, after the best unknown attribute is selected (the one with maximum utility), its test cost is counted. However, its true value is not revealed. After that, this attribute is removed from $\overline{A}$ to $\widetilde{A}$ and the selection process continues. Equivalently, all unknown attributes with non-negative utility are selected.

Again, let $\overline{A}'$ denote the batch of attributes selected and C_{test} the test cost. They are computed as follows:

$$\overline{A}' = \{\overline{A}_i | Util(\overline{A}_i) > 0, \overline{A}_i \in \overline{A}\} \quad (6)$$

$$C_{test} = C_{test}(\overline{A}') = \sum_{\overline{A}_i \in \overline{A}'} C_{test}(\overline{A}_i) \quad (7)$$

In the above equations, the utility of an unknown attribute $\overline{A}_i$ is calculated as in Section 4, and the costs of testing the attributes in $\overline{A}'$ comprise the overall test cost C_{test}. The batch tests on $\overline{A}'$ are then taken, and the values of attributes in $\overline{A}'$ are revealed and added into $\widetilde{A}$. Finally, the class label is predicted as in the *csNB*-sequential prediction algorithm (Step 16-17).

Back to the example in Section 3.1, $\overline{A}'$ ={"spleen", "spiders", "ascites"}, since these three unknown attributes have non-negative utilities. The test cost C_{test} is $24 + 26 + 32 = 82$ and the batch test strategy is to perform tests on these three attributes in one shot.

6. Experiments

In order to evaluate the performance of *csNB* with both sequential and batch test strategies, experiments were carried out on eight datasets from the UCI ML repository [1]. For 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 [4]. 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 any unknown one. 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 termed Exacting Naive Bayes (ENB).

Comparisons were also made between *csNB* and the Cost-Sensitive Decision Tree (csDT) proposed in [8]. The latter is a novel and effective method for building and testing decision trees that also aims to minimize the sum of the misclassification cost and the test costs. The algorithm was shown to significantly outperform C4.5 and its variations.

In summary, four methods were examined: (1) Lazy Naive Bayes (LNB), (2) Exacting Naive Bayes (ENB), (3) Test-Cost Sensitive Naive Bayes (*csNB*), and (4) Cost-Sensitive Decision Trees (csDT).

The eight datasets used are listed in Table 2. These datasets were chosen because they have discrete attributes, binary class, and a sufficient number of examples. We only consider binary class problems in the following experiments to be consistent with the csDT algorithm, although our *csNB* algorithm can be used in multiple class problems naturally. Also, the numerical attributes in datasets were discretized using minimal entropy method as in [8].

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

Table 2. Datasets used in the experiments

We ran a 3-fold cross validation on these data sets. In the experiments, no missing value is assigned in the training examples and for the testing examples, a certain percentage (missing rate) of attributes are randomly selected and marked as unknown. If during classification, an algorithm decides to perform a test on an unknown attribute, its true value is revealed and the test cost is accumulated. Finally, the misclassification cost C_{mc} can be obtained by comparing the predicted label with the true class label, and C_{test} is the accumulated test cost. The performance of the algorithms is then measured in terms of the total cost $C_{total} = C_{mc} + C_{test}$. To the binary class problems, let c_1 be the positive class and c_2 the negative class. The misclassification matrix was set as $C_{12} = C_{21} = 600$ and $C_{11} = C_{22} = 0$, where C_{12} can be interpreted as false negative and C_{21} false positive. The test cost of each attribute is set randomly between 0 and 100.

6.1. Sequential Test Strategy

Figures 1 and 2 show the results of different algorithms with sequential test strategy on all the eight datasets. In these experiments, the percentage of unknown attributes is 40%. Each group of four bars represents the runs of four algorithms on one particular dataset. 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 standing for the average misclassification cost and the upper light portion standing for the average test costs.

There are several interesting observations from these experiments. First, although the misclassification costs of ENB are almost always the lowest among the four methods, the average total costs of it 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 of its lazy nature, LNB performs well, even better than csDT. This can be explained by the fact that, while csDT uses the splitting criterion of minimal costs for attribute selection in tree building, whenever trees are built, the test sequences are fixed. Only the attributes along a tree path are examined and the others are ignored. However, those attributes not examined can still be informative in classification. LNB, on the other hand, is capable of making use of these attributes.

Third, our *csNB* method performs the best overall because of its ability in selecting unknown attributes for testing. As we can see from the figures, *csNB* not only lowers the misclassification costs compared with LNB, but also maintains a low level of test costs compared with ENB.

To investigate the impact of the percentage of unknown attributes on the average total costs, experiments were carried out on the performance with the increasing percentage of unknown attributes. Figure 3 shows the results on the Mushroom dataset (other figures are spared for space). As we can see, when the percentage increases ($> 40\%$), the

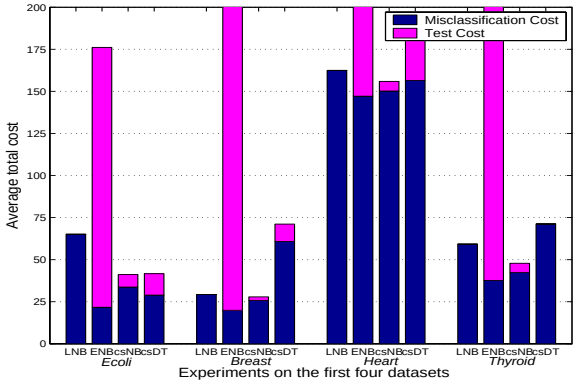

Figure 1. Average total cost comparisons of four methods on datasets: Ecoli, Breast, Heart and Thyroid.

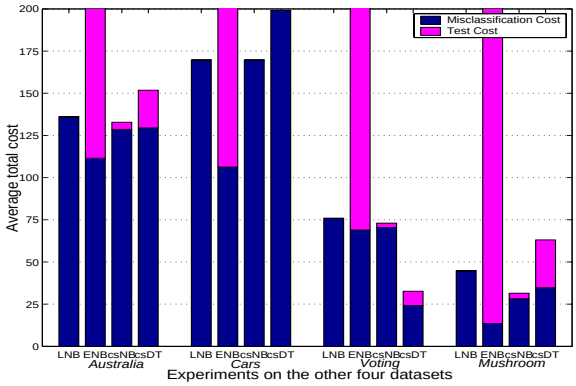

Figure 2. Average total cost comparisons of four methods on datasets: Australia, Cars, Voting and Mushroom.

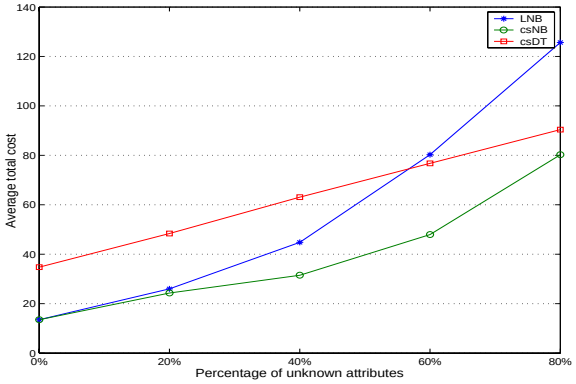

Figure 3. Comparisons with varying missing rates.

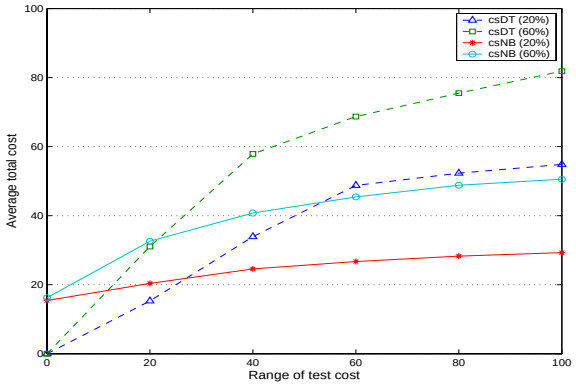

Figure 4. Comparisons with varying test costs.

average total cost of LNB increases significantly and surpasses that of csDT. Again, *csNB* is the best overall.

Another set of experiments was conducted to compare two cost-sensitive algorithms csDT and *csNB* in terms of varying test costs. Figure 4 shows the results on the Mushroom dataset with both the missing rates 20% and 60%. Still, *csNB* outperforms csDT overall. Also, as we can see, *csNB* is less sensitive to the test costs than csDT as the increasing of test costs. This reveals that the *csNB* method is effective at balancing the misclassification and test costs.

6.2. Batch Test Strategy

Batch test is another important scenario we want to investigate. In order to compare *csNB* with csDT in terms of their abilities with batch test strategy, we extended the csDT algorithm as suggested in [8]. The basic idea is that during classification, when a test case is stopped at the first attribute whose value is unknown, this attribute, together with all unknown attributes in the sub-tree will be tested.

The results of average total cost on the 8 datasets with 40% missing rate are shown in Figure 5. Again, the test cost of each attribute is set randomly between 0 and 100, and $C_{12} = C_{21} = 400$ while $C_{11} = C_{22} = 0$. Overall, *csNB* outperforms csDT greatly. On average, *csNB* incurs 29.6% less total cost than csDT. 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. Besides the same reasons as explained in sequential test (Section 6.1), in batch test, the advantage of *csNB* over csDT can also be explained as follows. By the nature of decision trees in tree-building, attributes are considered sequentially and conditionally one by one. Therefore, csDT is inflexible to derive batch test strategies. On the other hand, *csNB* has no such constraint and all the attributes can be evaluated at the same level.

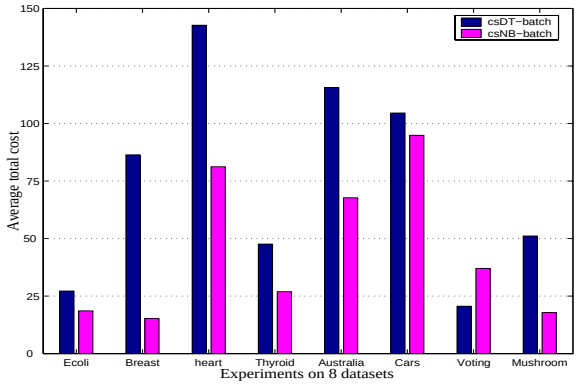

Figure 5. Comparisons in batch test on all the 8 datasets.

Figure 6 shows the two runs on the Breast and Mushroom datasets with the variation of percentage of unknown attributes. As we can see, *csNB* exhibits less sensitivity to the missing rate and performs mush better than csDT.

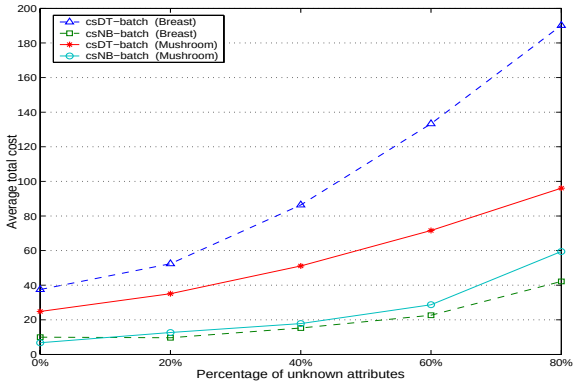

Figure 6. Comparisons in batch test with varying missing rates.

7. Conclusions and future work

In this paper, we proposed a test-cost sensitive naive Bayes algorithm for designing classifiers that minimize the sum of the misclassification cost and the test costs. In the framework of *csNB* , attributes are intelligently selected for testing to get both sequential test strategies and batch test strategies. Experiments show that our method outperforms other competing algorithms, including the cost-sensitive decision tree algorithm.

In the future, we plan to consider several extensions of our work. One direction to generalize the ideas to design testing strategies for other classifiers, such as Neural Nets, SVMs. It is also interesting to consider the cost of finding the missing values for training data. Another direction is to develop more effective algorithms for batch test strategies. In addition, it is worth considering the conditional test costs [15] 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.

8 Acknowledgments

This work is supported by Hong Kong Research Grant Committee (RGC) and Innovation and Technology Fund (ITF).

References

[1] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998.

[2] P. Domingos. Metacost: A general method for making classifiers cost-sensitive. In *KDD99*, pages 155–164, 1999.

[3] P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Willey and Sons, Inc., New York, 2nd edition, 2001.

[5] C. Elkan. The foundations of cost-sensitive learning. In *Proc. of the IJCAI01*, pages 973–978, 2001.

[6] R. Greiner, A. Grove, and D. Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence Journal*, 139(2):137–174, 2002.

[7] M. T. Kai. Inducing cost-sensitive trees via instance weighting. In Springer-Verlag, editor, *Principles of Data Mining and Knowledge Discovery, Second European Symposium*, pages 139–147, 1998.

[8] C. Ling, Q. Yang, J. Wang, and S. Zhang. Decision trees with minimal costs. In *Proc. of ICML04*, 2004.

[9] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[10] M. Nunez. The use of background knowledge in decision tree induction. *Machine Learning*, 6:231–250, 1991.

[11] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[12] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

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

[14] P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 1995.

[15] P. D. Turney. Types of cost in inductive concept learning. In *Workshop on Cost-Sensitive Learning at the 17th International Conference on Machine Learning*, 2000.

[16] V. B. Zubek and T. G. Dietterich. Pruning improves heuristic search for cost-sensitive learning. In *Proc. of ICML02*, pages 27–34, Sydney, Australia, 2002.