Feature Value Acquisition in Testing: A Sequential Batch Test Algorithm

Victor S. Sheng SSHENG@CSD.UWO.CA
Charles X. Ling CLING@CSD.UWO.CA
Department of Computer Science, The University of Western Ontario, London, Ontario N6A 5B7, Canada

Abstract

In medical diagnosis, doctors often have to order sets of medical tests in sequence in order to make an accurate diagnosis of patient diseases. While doing so they have to make a trade-off between the cost of the tests and possible misdiagnosis. In this paper, we use cost-sensitive learning to model this process. We assume that test examples (new patients) may contain missing values, and their actual values can be acquired at cost (similar to doing medical tests) in order to reduce misclassification errors (misdiagnosis). We propose a novel Sequential Batch Test algorithm that can acquire sets of attribute values in sequence, similar to sets of medical tests ordered by doctors in sequence. The goal of our algorithm is to minimize the total cost (i.e., the trade-off) of acquiring attribute values and misclassifications. We demonstrate the effectiveness of our algorithm, and show that it outperforms previous methods significantly. Our algorithm can be readily applied in real-world diagnosis tasks. A case study on the heart disease is given in the paper.

1. Introduction

Cost-sensitive learning considers a variety of costs in various components and process of learning (Turney, 2000), with the goal of minimizing the costs or the total cost. Two most important types of costs are identified as misclassification costs and attribute costs (also called test costs). Misclassification costs are an extension of error rate (or one minus accuracy) as different types of errors (such as false positive and false negative) can have different costs. Attribute costs reflect how expensive to obtain the missing attribute values. Recent works have considered methods of acquiring attribute values during training (Melville et al., 2004; 2005) and testing (Ling et al., 2004; Chai et al., 2004), for the purpose of reducing the misclassification cost. Learning algorithms that actively acquire extra information (such as missing attribute values) during learning is also called *active learning* (e.g., (Cohn et al., 1996; Saar-Tsechansky and Provost, 2004; Greiner et al., 2002)). Studying active learning in cost-sensitive framework has a unique advantage: learning becomes an optimization problem with a single goal: minimizing the sum of attribute acquisition cost and misclassification cost. In this paper, we propose a novel algorithm for attribute acquisition *during testing* based a model has already been built.

Assume that a cost-sensitive model has already been built. Given a test case with missing attribute values, the question is what attribute values should be acquired, at what order, such that the sum of attribute acquisition cost and expected misclassification cost is minimum. Answering to this question can make a significant impact to many real-world diagnosis applications. For instance, during medical diagnosis, doctors need to apply their existing knowledge (similar to a cost-sensitive model previously built) to predict the disease (class label) of new patents (test cases). As much is unknown (missing values) of a new patient, the doctors often have to order medical tests, such as blood tests (acquiring attribute values) before a final prediction (diagnosis) is made. Not only do medical tests have different costs (i.e., attribute costs, which include the actual cost as well as risks and uncomfortability to the patients), they also have different delay time in getting their results from the laboratories. Due to delays in getting the test results, doctors often order several tests to be done together as *in a batch* (that is, at the same time), and based on the results of the tests in the batch, decide what tests, if any, are to be performed next. When tests are done in a batch, the delay of the batch is (reasonably assumed to be) the maximum delay of all tests in the batch. Thus, doing tests in batch significantly reduces the delay compared to the total delay if these tests are done in sequence (i.e., the maximum of a set of numbers is always smaller than the sum of them). Clearly, doctors have to decide what tests should be performed in batches, at what order, in order to strike a good balance between reducing the test costs, reducing the total delay time, and making a good prediction on patient's disease (reducing misclassification cost).

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

In this paper we propose a novel algorithm that can decide, for each test example, what attribute values should be acquired in batch, at what order, in order to minimize the sum of attribute acquisition costs, delay costs, and misclassification costs. We call it *Sequential Batch Test* (SBT for short) algorithm. Our algorithm utilizes a cost-sensitive decision tree (Ling et al., 2004) built previously. We assume that test examples may have missing attribute values, and each attribute has a specific (given) cost, as well as a delay cost, which may be converted from the delay time (say, in hours). All costs (misclassification cost, attribute cost, and delay cost) are represented in the same unit (such as dollars). Our proposed algorithm SBT can suggest batches of attribute values to acquire, in a certain sequence, such that the sum of attribute acquisition costs, delay costs, and misclassification costs is minimal.

Our work is significantly different from previous work in attribute value acquisition during training. (Melville et al., 2004; 2005) proposed attribute value acquisition during training, instead of testing as in our paper. Their algorithm is also sequential in nature to query the missing values, instead of in batches. In addition, their goal is to reduce misclassification error, instead of total cost. (Ling et al., 2004) proposed attribute value acquisition during testing. However, their "test strategies" are simple: only one attribute is acquired at a time (the batch size is always 1). They did propose a "single batch" algorithm that suggests a batch of tests to be done at the same time. However, their single batch algorithm acquires all attributes in a sub-tree of the decision tree to be performed, and thus, too many attributes would be included in only one batch. In addition, no delay cost is considered. (Turney, 1995) included delay in the attributes, but only delayed or non-delayed. We quantify the amount of delay in each attribute. (Turney, 1995)'s algorithm is a genetic algorithm, which is more computationally expensive and was not studied in a true cost-sensitive framework as we do. Most previous work on active learning (such as (Cohn et al., 1996; Saar-Tsechansky and Provost, 2004; Greiner et al., 2002)) considers labeling costs while our algorithm acquires actively attribute values during the testing process.

Our paper is organized as follows. In Section 2, we introduce a real-world case on the heart disease to illustrate the problem we are studying. In Section 3, we describe our Sequential Batch Test (SBT) algorithm for test examples and its integration with the cost-sensitive decision tree learning. Then we demonstrate the effectiveness of our algorithm, and show experimentally that it outperforms previous methods. Finally we conclude the work in Section 5.

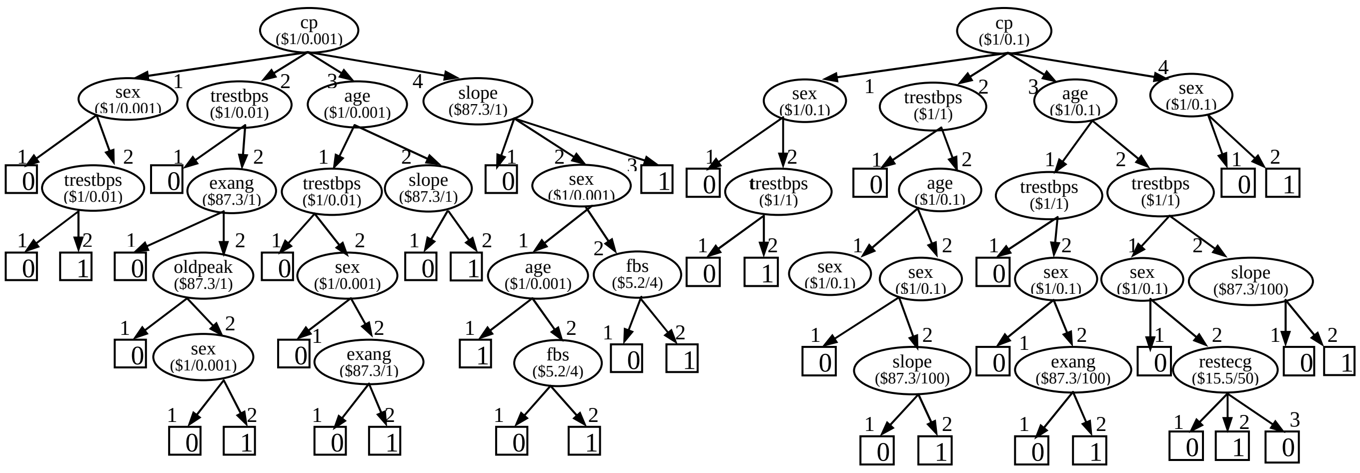
2. A Case Study of Heart Disease

Before we discuss the Sequential Batch Test algorithm, let us look at a concrete case first: the Heart Disease. The Heart Disease dataset was used in the cost-sensitive genetic algorithm by (Turney, 1995), and it involves the diagnosis of the heart disease. The class label is whether or not there is a coronary artery narrowing (positive/negative), and the attributes are 13 non-invasive tests including patient profile (they are simply regarded as "cheap tests"), as listed in Table 1. The misclassification costs (the values of false positive (FP) and false negative (FN)), attribute costs, and attribute delay time are obtained from medical experts in the local hospital and the insurance program in Canada: the misclassification costs FP = 600 and FN = 1,000, and the attribute costs and their delay time (in hours) are listed in Table 1. To convert the delay time to cost, an hourly rate is also given. The multiplication of the delay time and the hourly rate is attributes' delay cost. For real world applications, we can assume that test cases may have different hourly rate, just as different patients may have different tolerance for waiting; some people may be willing to wait while others cannot. The original dataset contains a total of 294 cases with 36.1% positive cases (106 positive cases) and the rest are negative cases. Note that in practice when a group of related tests (such as blood tests) are performed at the same time (in a batch), there is often a discount on the total test cost. This issue is not considered in this paper.

Table 1. Attribute costs (in Canadian Dollars) and delay time (in hours) for the Heart Disease dataset.

Tests (attributes)	Description	Attribute Costs (CN$)	Delay Time (hours)
Age	age of the patient	1.00	0.001
Sex	Sex	1.00	0.001
Cp	chest pain type	1.00	0.001
Trestbps	resting blood pressure	1.00	0.01
Chol	serum cholesterol in mg/dl	7.27	4
Fbs	fasting blood sugar	5.20	4
Restecg	resting electrocardiography results	15.50	0.5
Thalach	maximum heart rate achieved	102.90	1
Exang	exercise induced angina	87.30	1
Oldpeak	ST depression induced by exercise	87.30	1
Slope	slope of the peak exercise ST segment	87.30	1
Ca	number of major vessels colored by fluoroscopy	100.90	1
Thal	finishing heart rate	102.90	1

As we have discussed in the Introduction, we assume that a cost-sensitive decision tree has been built for the problem domain, and is used as a base learner for the SBT algorithm. In this paper, we re-implemented the cost-sensitive decision tree proposed by (Ling, et al., 2004), and thus we provide a quick review of it here. The cost-sensitive decision tree is very similar to C4.5 (Quinlan, 1993), except it uses total cost reduction, instead of entropy reduction, as the attribute split criterion. More specifically, the total cost before and after splitting on an attribute can be calculated, and the difference is the cost reduction "produced" by this attribute. The total cost before split is simply the misclassification cost of the set



(a) Hourly rate of $1/hr (b) Hourly rate of $100/hr

Figure 1. Cost-sensitive decision trees with two different hourly rates for the delay tolerance. The attribute cost and delay cost are displayed in each node separated by /. Continuous attributes have been discretized and are represented by 1, 2, and so on.

of examples. The total cost after split on an attribute is the sum of misclassification cost of subsets of examples split by the attribute values, plus the attribute cost and the attribute delay cost of these examples (the cost-sensitive decision tree proposed by (Ling, et al., 2004) does not consider the attribute delay cost). The attribute with the maximal positive cost reduction is chosen as the root, and the procedure recursively applies to the subsets of examples split by the attribute values.

We apply this cost-sensitive tree algorithm to the Heart Disease dataset with three different hourly rates of $1/hr, $10/hr, and $100/hr. Again the hourly rate multiplies the delay time equals to the delay cost. As one must wait for the result of attributes being tested, we include delay cost as a part of the attribute cost during the tree construction. The numerical attributes in datasets are discretized into discrete values (we use 1, 2, … to represent them) using the minimal entropy method of (Fayyad and Irani, 1993). Figures 1(a) and 1(b) show the two trees built with two different delay costs for the hourly rates of $1/hr and $100/hr. The tree with hourly rate of $10/hr is very similar to the one with $1/hr, except one different small sub-tree; therefore it is not shown.

Several interesting observations can be made from the two trees. First we can see that delay costs can affect the decision trees significantly. Compared Figures 1(a) and 1(b), the tree structures are quite different as the delay costs are varied. Specifically, the tree Figure 1(b) with a higher delay cost prefers to use the attributes with lower delay costs to split the training examples. The attributes with higher delay costs only appear in the bottom if they are used. Another observation is that the tree-building algorithm prefers to choose attributes with zero or small sum of attribute and delay costs at the top (or root) of the tree. This is because the attribute at the root will be tested by all test examples. Choosing an attribute with zero or small cost helps reduce the total cost. Of course attribute selection also depends on discriminative power of the attribute, as an attribute is selected if the sum of misclassification, attribute cost, and delay cost is minimal.

Given a test examples with missing values (even when all values are missing), and a cost-sensitive decision trees similar to the ones in Figure 1, the SBT algorithm will suggest batches of attribute values to be tested in a certain sequence, in order to minimize the total cost of attribute acquisition, delay, and misclassification. Note that when a batch of unknown attributes (tests) is being determined, their values are not known; only after the batch is determined, their true values will be obtained at cost. If too many attributes are included in a batch, many of them may be wasted and the total test cost can be high. If too few attributes are included, more batches of tests would be needed and the total wait cost would be high. SBT will then determine if the next batch of tests is needed or not, based on the revealed values of earlier tests. Clearly it is not trivial to make an optimal decision. We will describe SBT in the next section. After that we will revisit this case Section 4.4.

3. Sequential Batch Test (SBT) Algorithm

In this section, we describe our *Sequential Batch Test* (*SBT*) in detail. We will first describe the SBT algorithm in the generic form; that is, it is an A*-like search algorithm that can work with any cost-sensitive learning model, if the required components of the model can be implemented efficiently (see below). We will then describe *SBT* with cost-sensitive decision trees. The tree structure makes the implementation of *SBT* more efficient.

3.1 The Framework of SBT

The framework of the *SBT* algorithm is quite simple, and it can work with any cost-sensitive learning algorithms, such as ICET in (Turney, 1995), cost-sensitive decision tree in (Ling et al., 2004), and cost-sensitive naïve Bayes in (Chai et al., 2004), as long as the required components can be implemented efficiently. At the core of the *SBT* algorithm is an A*-like algorithm to search for one batch of tests to be performed. It has three components: initialization, evaluation, and update. The three components are directly dependent on the cost-sensitive learning model. The initialization component searches the initial candidates. Then the evaluation component evaluates the cost-effectiveness of each candidate in the list. The best candidate with the maximum expected cost reduction is chosen, and is added into the current batch if it is positive. The update component updates and maintains the potential candidates. After one batch is determined, the values of the attributes in the batch are obtained (after performing the tests). The *SBT* algorithm will continue to search the next batch until no more batches needed. The pseudo-code of the general *SBT* algorithm is shown in Figure 2.

Algorithm *SBT_General()*

Input: a cost-sensitive learning algorithm, a test example with missing values

```
1.  Loop
    a. $L$ = Get an initial list of candidates
    b. $B$ = empty   /* the current batch of tests */
    c. While $L$ is not empty do
       i.   Find $v$ with maximum cost reduction in $L$
       ii.  If $v$ has a positive cost reduction
       iii. Then add $v$ into $B$, delete $v$ from $L$, and update $L$
       iv.  Else exit the while loop /* No suitable candidate*/
    d. End of while
    e. If ($B$ is not empty) then
       i.  Output $B$ as the current batch of tests
       ii. Obtain values of attributes in $B$ at cost
    f. Else exit Loop at 1
2.  Predict the test case with all known values
```

Figure 2. The generic pseudo-code for *SBT*.

The first sub-step (*1a*) that initializes the candidate list *L* is dependent on the learning model. For decision trees, the initial candidate list contains only one node: the first unknown attribute encountered when a test example with missing values is classified by the tree. For cost-sensitive naïve Bayes, the initial list contains all unknown attributes of the test example. Similarly, the evaluation steps (*1c(i) and 1c(ii)*) are also dependent on the learning models. We will describe the integration of SBT with cost-sensitive decision trees and the corresponding evaluation method in the following section. The evaluation process for naïve Bayes is computationally expensive (involving combination of attributes in the current batch), thus we do not study it in this paper. Last, updating the candidate list *L* (*sub-step 1c(iii)*) also relies on the learning model. For decision trees, the update procedure will delete the best candidate from the list *L*, and then adds its children into the list *L*. For cost-sensitive naïve Bayes, the chosen candidate is simply deleted from the candidate list *L*.

3.2 SBT with Cost-sensitive Decision Tree

In this section we will describe SBT with cost-sensitive decision trees as the base learner. The tree structure (such as trees in Figure 1) makes the searching the initial candidate list and maintaining the candidate list easy and efficient. We have discussed the initialization and update steps in Section 3.1. In this section, we focus on the evaluation step of *SBT* with cost-sensitive decision trees.

With the cost-sensitive decision trees, *SBT* uses a heuristic utility measure to decide what attributes should be included in the batch. The utility *U(.)* of an attribute is the net profit brought by the attribute, described below. An attribute is added to a batch if the utility is positive and maximum (among other attributes). We keep on adding more unknown attributes into the current batch until no more unknown attributes can bring in net profit. The rationale behind this (heuristic) strategy is that attributes that bring in some net profit is worthwhile to be included in the current batch. The pseudo-code for *SBT* with cost-sensitive decision trees is given in Figure 3. The line numbers (such as *1a* and *1c(iii)*) in Figure 3 correspond to the same line numbers in Figure 2. The time complexity of this specific SBT algorithm is linear to the size of the tree, as each node in the tree would be considered at most once.

Algorithm *SBT_Tree()*

Input: the cost-sensitive decision tree learning algorithm CSDT, a test example with missing values

```
1.  Loop
    a. $L$ = the first unknown attribute when classifying a test
       case
    b. $B$ = empty   /* the batch of tests */
    c. While $L$ is not empty do
       i.   Calculate $U(i)$ for each $i \in L$, $U(t) = maxU(i)$
       ii.  If $U(t) > 0$
       iii. then add $t$ into $B$, delete $t$ from $L$, add $R(t)$ into $L$
       iv.  else exit the while loop  /* No positive utility */
    d. End of while
    e. If ($B$ is not empty) then
       i.  Output $B$ as the current batch of tests
       ii. Obtain values of attributes in $B$ at cost
    f. Else exit Loop at 1
2.  Predict the test case with all known values
```

Figure 3. The pseudo-code for SBT with cost-sensitive decision trees. The line numbers (such as *1a* and *1c(iii)*) correspond to the same line numbers in Figure 2.

The utility *U(.)* of an attribute (line *1c(i)* in the pseudo-code) is the net profit brought in by the attribute. As the attribute value is yet unknown when deciding the batch, *U(.)* of an attribute is estimated by the expected value of all possible values of the attribute. The probability of a possible value is estimated by the training examples falling under that node in the tree. More specifically, the utility *U(.)* of the attribute *i* is the difference of expected total cost before and after testing *i*, defined as follows:

$$U(i) = misc(i) - [c(i) + \sum_j p(i,j) \times misc(i,j) + W_{TB}]$$

where *misc(.)* is the total expected misclassification cost of a node in the tree before testing *i*, *c(.)* is the total attribute cost, and *p(i, j)* is the probability of the *j-th* value of the attribute *i*. W_{TB} is the current batch delay cost. It is defined as $W_{TB} = max(W_B, W_i)$, where W_B is the previous batch delay cost, and W_i is the delay cost of the attribute *i*.

When the attribute *t* is added into the batch, the update procedure (line *1c(iii)*) deletes the attribute *t* from the list *L*, and then adds its children *R(t)* into the list *L*.

After the current batch of attributes is determined, it will request the values of those attributes in the batch at cost (line *1e(ii)*). After these values are obtained, the test example can be classified further down in the tree, until it is stopped by another unknown attribute or at a leaf. In the former case, the same procedure is applied; in the latter case, the test example is classified with the class label of the leaf, and the testing process is completed.

The SBT algorithm described above is heuristic but it is close to the ideal one: it guarantees that if the delay cost of all attributes is 0, then the SBT would become an ideal, sequential process: it performs one attribute acquisition at a time (or the batch size is always 1). See Section 4.3 for experimental verification. This is also the myopia approach (Gorry and Barnett, 1968). On the other hand, if the delay cost of the first attribute in a batch is very large, the current batch would grow until the expected cost reduction of the remaining unknown attributes is no longer greater than 0. Thus, only one batch is recommended (see Section 4.3 for experimental verification).

4. Experiments

In this section, we empirically evaluate the SBT algorithm (from now on, when we mention the SBT algorithm, we mean SBT with cost-sensitive decision tree) on real-world datasets to show its effectiveness in finding batches of attributes to be acquired in sequence to minimize the sum of attribute costs, attribute delay costs, and misclassification cost.

4.1 Datasets and Experiment Setup

We choose 10 real-world datasets, listed in Table 2, from the UCI Machine Learning Repository (Blake and Merz, 1998). These datasets are chosen because they are binary class, have at least some discrete attributes, and have a good number of examples. The numerical attributes in datasets are discretized into integers (1, 2, …) using the minimal entropy method of (Fayyad and Irani, 1993). Each dataset is split into two parts: the training set (60%) and the test set (40%). To study the effect of various amount of missing values in the test data, a certain percentage of attributes (20%, 40%, 60%, 80%, and 100%) are randomly selected and marked as unknown (in test examples). If an attribute value is requested by the SBT algorithm, its original value is revealed.

Unfortunately, the detailed attribute costs and attribute delay costs of these datasets are unknown. To make the comparison possible, we simply assign random numbers in a certain range as these costs. This is reasonable because the same set of costs is used in the experimental evaluations and comparison. The attribute costs are assigned with random numbers between 0 and 100. In order to investigate the effect of various delay costs, we set up four levels for delay costs: (a) all delay costs are 0; (b) all delay costs are randomly assigned from 0 to 50, thus the delay costs are half of the range of attribute costs; (c) all delay costs are randomly assigned from 0 to 100; and (d) all delay costs are randomly assigned from 0 to 200 (twice the range of attribute costs). We also vary the misclassification costs (*FP/FN*) to be *1k/6k*, *2k/6k*, *4k/6k*, and *6k/6k*. (Due to symmetry, there is no need to experiment cases where FP > FN). The correct classification costs *TP* and *TN* are always set as 0.

Table 2. Datasets used in the experiments

	No. of Attributes	No. of Examples	Class dist. (N/P)
Ecoli	6	332	230/102
Breast	9	683	444/239
Heart	8	161	98/163
Thyroid	24	2000	1762/238
Australia	15	653	296/357
Tic-tac-toe	9	958	332/626
Mushroom	21	8124	4208/3916
Kr-vs-kp	36	3196	1527/1669
Voting	16	232	108/124
Cars	6	446	328/118

4.2 Delay Cost Varied while Misclassification Cost Fixed

In this section, we investigate the effects of delay costs whose range is varied under a fixed misclassification cost. For each misclassification cost (*FP/FN=1k/6k*, *2k/6k*, *4k/6k*, and *FP/FN=6k/6k*, we vary the three ranges of the delay costs ([0..50], [0..100], and [0..200]). We repeat this process 25 times, and the average number of batches for the 10 datasets is plotted in Figure 4.

Note that from Figure 4 we can see that the number of batches is often less than 1 especially when the ratios of missing attributes are small (such as 0.2). This is because with a small percentage of missing values, often the SBT algorithm does not need to acquire any missing values when classifying test examples by the decision tree.

From Figure 4, we can draw some interesting conclusions: First of all, the smaller the delay cost range, the more the number of batches (for the same ratio of unknown attributes). This meets our expectation: when the delay cost is small, it is preferable to have small batch sizes (thus more batches in sequence), as it is better to wait to obtain attribute values so that they can help to make decision for the next attribute(s) more accurately and save money from unnecessary attribute acquisition. The extreme case is zero (0) attribute delay cost. Under this case, it is preferable to do one test at a time. Our experiment results show that the average number of batches in this case is the highest. When the range of the delay cost is high, we can expect that SBT will reduce the number of batches to save the total cost, as shown in Figure 5.

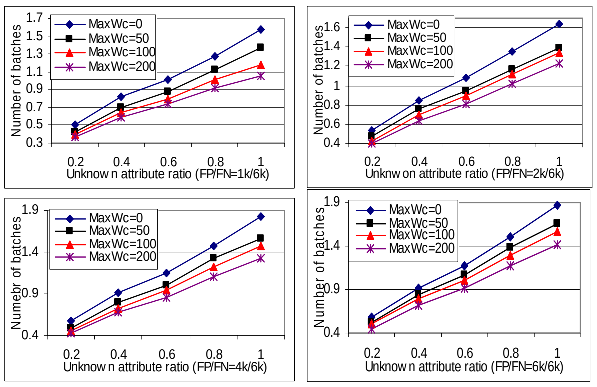


Figure 4. Number of batches of our SBT under different misclassification cost settings.

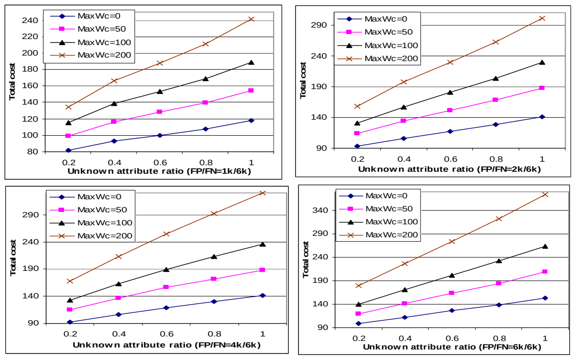


Figure 5. The average total costs (in $) for SBT under different misclassification cost settings.

Figure 5 shows the average total costs of SBT for the 10 datasets under the misclassification cost settings. We can see clearly that when the ratio of missing values increases, the average total cost also increases, as expected. What is more interesting is that when the range of the delay costs increases, the average total cost also increases. As the range of the delay costs increases, attribute delay costs become higher. This directly causes the batch delay cost to increase.

4.3 Delay Cost is Zero or Very Large

To further study the effect of the delay cost on the number of batches in SBT, we conduct two more experiments: setting one attribute delay cost (from some non-zero value) to zero or very large (greater than the maximum misclassification cost). When the delay cost of an attribute is set to 0, it is likely that this attribute will be single out to form a batch by itself, as knowing its value (without delay) would help to decide what other attributes should be acquired in the next step. On the other hand, if the delay cost of one attribute is set much larger than the misclassification cost, then this attribute will be "pushed out" of the decision tree (not chosen in the tree as the sum of attribute cost and delay cost is very large). Thus it will not be picked by SBT to be included in a batch. We study one specific case and one general case with the dataset (Kr-vs-kp), as shown below.

For the specific case, a particular test example is chosen. The attribute cost of all attributes is randomly assigned to be within 0 to 100, and the misclassification cost is set to be 4k/6k. In this case the SBT algorithm suggests 3 batches with total cost \$6,951.1 (sum of the attribute costs, delay costs, and misclassification cost of \$6,000). The three batches are: $\{A_{21}\}$, $\{A_1, A_6, A_{10}, A_{15}, A_{16}, A_{28}, A_{31}, A_{32}, A_{33}, A_{35}\}$, and $\{A_{20}, A_{22}, A_{23}\}$. Then one attribute, A_{31}, is chosen, and its delay cost is changed from 21.3 (randomly assigned) to be 0. The number of batches increases from 3 to 4: $\{A_{21}\}$, $\{A_1, A_6, A_{10}, A_{15}, A_{16}, A_{28}, A_{32}, A_{33}, A_{35}\}$, $\{A_{31}\}$, $\{A_{20}, A_{22}, A_{23}\}$, as the attribute A_{31} is singled out as a batch alone. On the other hand, when the delay cost of A_{31} is changed from 21.3 to a very large number (10,000 here), the number of batches decreases from 3 to 2: $\{A_{21}\}$, $\{A_1, A_6, A_{10}, A_{15}, A_{16}, A_{28}, A_{32}, A_{33}, A_{35}\}$, as A_{31} is not included in the tree.

To study the general case, we run our SBT 25 times for a set of test cases (40% randomly sampled the dataset Kr-vs-kp); other settings are the same as above. For each run, A_{31} is set to 0 or to 10,000. The average number of batches is listed in Table 3. From the table, we can clearly see the general trend: the an attribute's delay cost is set to 0, the number of batches increases by about 0.1 on average, indicating that the attribute is singled out from the batch, if it was in the batch. If the attribute's delay cost is set to a very large number, this attribute would be pushed out of the tree (if it was in the tree). The number of batches would decrease by about 0.1 on average. This indicates that the SBT algorithm, though heuristic, is close to the ideal one. When delay costs are small, the algorithm behaves like a sequential one, doing one test at a time. When delay costs are large, the algorithm just would suggest one single batch.

Table 3. The average number of batches under different delay cost of A_{31} from the dataset *Kr-vs-kp*.

Delay cost of A_{31}	[0..100]	0	10,000
Number of batches	1.85	1.93	1.76

4.4 Revisit the Heart Disease Dataset

With the cost-sensitive decision trees built under different hourly rates, we can apply our SBT algorithm to predict what attribute values should be acquired, at what order, in order to minimize the sum of attribute acquisition costs, delay costs, and misclassification costs. Here we assume that for all test examples, all attribute values are missing (as new patients). Under the different hourly rates ($1/hr, $10/hr, and $100/hr), the average total cost and the average of number of batches of our SBT are displayed in Figure 6.

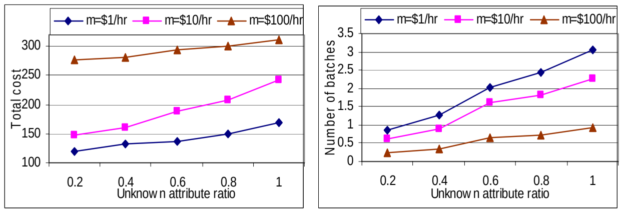


Figure 6. The average total cost and the average number of batches of our SBT on the dataset Heart Disease.

From Figure 6, we can see the number of batches changes dramatically when the delay costs are changed. When delay costs are increased (that is, the hourly rate is increased from 1 to 10 to 100), the number of batches decreases significantly. At the same time, the average total cost increases significantly as well.

We next choose a particular test example to study SBT under different hourly rates of 1, 10, and 100, simulating patients with different tolerance levels of waiting. When the hourly rate is set to be 1, SBT decides that two batches of tests are needed for the test case. The first batch of tests contains *cp*, *sex* and *trestbps*. The second batch contains *slope* and *fbs*. Thus, the total attribute costs for the test case is $95.5, and the total delay cost is $4.01. Thus the total cost is $99.51 (no misclassification cost). When the hourly rate is set to be 10, SBT decides that only one batch is needed, containing *cp*, *sex* and *trestbps*. The total attribute costs for this test example is $3.0, and the total delay cost is $0.1. Thus the total cost is $3.1 (no misclassification cost). Note that for this particular test example, the total cost under the hourly rate of 10 is lower than that under 1. However, in general the total cost under the hourly rate of 10 is more likely to be higher than that under 1. When the hourly rate is set to 100, SBT also decides that only one batch is needed, containing *cp*, *sex*, and *trestbps*, which is the same as the one with hourly rate of 10. Thus, the total attribute cost is same ($3.0). However, the total delay cost is $1.2 (higher than the delay cost under the hourly rate of 10). The total cost is $4.2 (no classification cost). In general, the higher the hourly rate for waiting, the few batches will be suggested.

4.5 Comparison with Previous Work

As we discussed in the Introduction, SBT is a generalization of the sequential test and single batch test in (Ling et al., 2004). When the delay cost of all attributes is 0, SBT behaves like sequential test (one test in a batch). When the delay cost of unknown attributes is large, SBT is much like single batch, forming one batch of tests. This section compares SBT to the sequential test (called SeqT here) and single batch (called SingB here) of (Ling et al., 2004).

First of all, SeqT must be modified slightly, so it can deal with attributes with delay cost. We add the delay cost into the attribute cost, when an attribute is needed to get its value. SingB is also modified to deal with delay cost. When the single batch is formed by SingB, the maximum delay cost of all attributes in the batch is added to the total cost. Attribute costs are set randomly in the range of [0..100], and the misclassification cost is set to be *4k/6k* in this comparison. All other settings in the experiment are the same as in Section 4.1. The experimental results of the three algorithms (SBT, SeqT, and SingB) are shown in Figure 7.

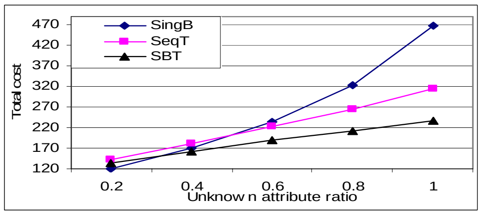


Figure 7. Comparing the average total cost of SBT to previous methods SeqT and SingB. The smaller the total cost, the better.

From Figure 7, we can see that SBT outperforms SeqT at any unknown attribute ratio. When the unknown attribute ratio is greater than 0.3 or so, SBT outperforms SingB significantly. SBT is much better than SeqT and SingB when the unknown attribute ratio increases. SingB outperforms slightly SeqT and SBT when the unknown attribute ratio is very low (less than 0.3), since SingB has only one batch delay cost (the maximum delay cost). However SBT has more batch delay costs and SeqT even has a delay cost for each acquiring attribute. When the unknown attribute ratio is greater than 0.5 or so, SeqT performs better than SingB, since each test in SeqT is used; however SingB performs many tests that are wasted. In all, SBT performs the best, SeqT is second, and SingB is the worst, especially when the unknown attribute ratio is high.

5. Conclusions and Future Work

In this paper, we use cost-sensitive learning to model medical diagnosis process in which doctors have to make a trade-off between the cost of the tests and possible misdiagnosis. In order to minimize the sum of misdiagnosis costs, test costs, and delay costs, we propose the sequential batch test strategy (SBT) that can acquire sets of attribute values in sequence, similar to sets of medical tests ordered by doctors in sequence. We empirically evaluate our SBT and investigate the effects of delay costs under different settings, and show that it outperforms previous methods. Our algorithm can be applied in real-world diagnosis tasks.

In our future work we plan to continue to work with medical doctors to apply our algorithms to medical data with real costs. We also plan to incorporate other types of costs in our decision tree learning and test strategies.

Acknowledgements

We thank very much Dr. Qingping Feng and Dr. Ray Guo at University Hospital for providing valuable medical information about the heart disease studied in the paper. The authors thank NSERC for supporting their research.

References

Blake, C.L. and Merz, C.J. 1998. *UCI Repository of machine learning databases (website)*. Irvine, CA: University of California, Department of Information and Computer Science.

Chai, X., Deng, L., Yang, Q., and Ling, C.X. 2004. Test-Cost Sensitive Naive Bayes Classification. *In Proceedings of The 2004 IEEE International Conference on Data Mining*.

Cohn, D., Ghahramani, Z., and Jordan, M. 1996. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4: 129-145.

Domingos, P. 1999. MetaCost: A General Method for Making Classifiers Cost-Sensitive. *In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 155-164. San Diego, CA: ACM Press.

Elkan, C. 2001. The Foundations of Cost-Sensitive Learning. *In Proceedings of the Seventeenth International Joint Conference of Artificial Intelligence*, 973-978. Seattle, Washington: Morgan Kaufmann.

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

Friedman, J., Yun, Y., and Kohavi, R. 1996. Lazy Decision Trees. *In proceedings of 13th National Conference Artificial Intelligence*.

Gorry, G.. and Barnett, G.. 1968. “Experience with a model of sequential diagnosis”, *Computers and Biomedical Research*.

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

Ling, C.X., Yang, Q., Wang, J., and Zhang, S. 2004. Decision Trees with Minimal Costs. *In Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, Alberta: Morgan Kaufmann.

Lizotte, D., Madani, O., and Greiner R. 2003. Budgeted Learning of Naïve-Bayes Classifiers. *In Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*. Acapulco, Mexico: Morgan Kaufmann.

Melville, P., Saar-Tsechansky, M., Provost, F., and Mooney, R.J. 2004. Active Feature Acquisition for Classifier Induction. *In Proceedings of the Fourth International Conference on Data Mining*. UK.

Melville, P., Saar-Tsechansky, M., Provost, F., and Mooney, R.J. 2005. Economical Active Feature-value Acquisition through Expected Utility Estimation. *UBDM Workshop, KDD 2005*.

Mitchell, T. 1997. *Machine Learning*, the McGraw-Hill Companies.

Nunez, M. 1991. The use of background knowledge in decision tree induction. *Machine learning*, 6:231-250.

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

Saar-Tsechansky, M. and Provost, F. 2004. Active sampling for class probability estimation and ranking. *Machine Learning*, 54(2): 153-178.

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

Ting, K.M. 1998. Inducing Cost-Sensitive Trees via Instance Weighting. *In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, 23-26. Springer-Verlag.

Turney, P.D. 1995. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research* 2:369-409.

Turney, P.D. 2000. Types of cost in inductive concept learning. *In Proceedings of the Workshop on Cost-Sensitive Learning at the Seventeenth International Conference on Machine Learning*, Stanford University, California.

Zadrozny, B. and Elkan, C. 2001. Learning and Making Decisions When Costs and Probabilities are Both Unknown. *In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, 204-213.

Zubek, V.B. and Dietterich, T. 2002. Pruning improves heuristic search for cost-sensitive learning. *In Proceedings of the Nineteenth International Conference of Machine Learning*, 27-35, Sydney, Australia: Morgan Kaufmann.