Accurate Decision Tree with Cost Constraints

Nan Wang[1], Jinbao Li[1(✉)], Yong Liu[1], Jinghua Zhu[1], Jiaxuan Su[2], and Cheng Peng[2]

[1] School of Computer Science and Technology, Heilongjiang University, Harbin 150080, China
lijbsir@126.com
[2] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

Abstract. A decision tree is a basic classification and regression method that uses a tree structure or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Decision tree is an effective approach for classification. At the same time, it is also a way to display an algorithm. It serving as a classical algorithm of classification has many optimization algorithms. Even though these approaches achieve high performance, the acquirement costs of attributes are usually ignored. In some cases, the acquired costs are very different and important, the acquirement cost of attributes in decision tree could not be ignored. Existing construction approaches of cost-sensitive decision tree fail to generate the decision tree dynamically according to the given data object and cost constraint. In this paper, we attempt to solve this problem. We propose a global decision tree as the model. The proper decision tree is derived from the model dynamically according to the data object and cost constraint. For the generation of dynamic decision trees, we propose the cost-constraint-based pruning algorithm. Experimental results demonstrate that our approach outperforms C4.5 in both accuracy and cost. Even though the attribute acquirement cost in our approach is much smaller, the accuracy gap between our approach and C4.5 is also small. Additionally, for large data set, our approach outperforms C4.5 algorithm in both cost and accuracy.

Keywords: Decision tree · Cost constraint · Machine learning
Algorithm of classification

1 Introduction

Decision tree [1] is an important approach for classification. Existing approaches, such as the LEGAL-tree using genetic algorithms [2], the HTILDE using incremental and anytime methods [3], trees with evaluation function [4], trees utilizing boosting and bagging [5], and some taking use of Cross-validation [6], focus on the selection for features that maximize the performance of the decision-tree-based classification.

Even though these approaches achieve high performance, the acquirement costs of attributes are ignored. In some cases, the acquired costs are very different and important, which should not be ignored. We consider a scenarios.

G. Sun and S. Liu (Eds.): ADHIP 2017, LNICST 219, pp. 154–165, 2018.
https://doi.org/10.1007/978-3-319-73317-3_19

In individual medical service, decision-tree- based classification could be used for assistant diagnosis. For assistant diagnoses, multiple medical indicators could be applied. Clearly, the costs of these indicators are very different. For example, the cost of body temperature is almost free. As a comparison, PAT-CT is much more expensive. Of course, the diagnosis abilities of these indicators are different. Some people prefer an assistant diagnose of a cold which need not an expensive acquirement cost. So how to select suitable indicators for diagnosis that could not only achieve high accuracy of diagnosis but be acquired within a reasonable cost is a problem.

For the wine dealers, it is important for them to know the origin of the wine so that they can estimate the quality of the wine accordingly. Hence approach of decision-tree-based classification could be used for assistant judgment. For assistant judgment, multiple chemical detections could be applied, and then different detection has different price and influence to the classification of origin. Thus, it is also a problem to select suitable detections that could be performed within a reasonable cost and also provide a high accuracy for the predicted value of origin.

From this scenario, construction techniques for optimal decision trees within the cost constraint are in demand. And such techniques brings following challenges.

- Without attribute costs, the ideal decision tree have two features, less leaf node and smaller tree depth. Since those features affect not only the efficiency but also the accuracy of classification, it has been proved that the construction of an idea decision tree is an NP-hard problem [7, 8]. With the constraint of cost, the problem becomes harder. The algorithm of construction for effective decision tree within tolerable time is the first challenge.
- Since different tasks and purposes of classification may have different cost constraint, a static decision tree does not satisfy those requirements. So the decision tree should change according to the cost constraints.
- In the practical cases, the indicators of zero cost could help to obtain a free decision tree. And users may want to obtain a free decision tree as a first baseline. Hence, those indicators should be used in high priority effectively.

With these challenges, existing approaches cannot be applied to solve this problem. They mainly concentrate on two aspects. One is to find the idea behind decision tree algorithm that aim to maximize accuracy, such as ID3, C4.5 and so on. The other is to induce decision tree that aim to minimize costs of misclassification or costs of obtaining the information in a way they adopt [9].

Even though they can generate effective decision trees with a relative small cost, they still cannot control the costs into a specified range, and also do not have a dynamic mechanism to get a decision tree in a tolerable time with different cost requirements. And as users tend to get the free information first, their methods cannot ensure the attributes with zero cost is located in the first place of the tree.

To achieve the goal of constructing optimal decision tree within the given cost, we develop a novel decision tree model. In this model, we design a novel classification criterion called cost-gain-ratio which combines the cost of attribute and the information gain ratio. And this ratio considers the cost first and attempts to select attribute that cost less.

To obtain a decision tree correctly, this ratio still considers information gain ratio [1] and construction criteria of our decision tree also makes the free indicators can be selected in a high priority during the construction of decision tree. Additionally, we can also choose more information gain when the cost is zero, and it also provides convenience for building the zero-cost-constraint decision tree in the dynamic building tree phase.

With the criteria, we use a top-down greedy approach to select the attributes with the least cost-gain-ratio as the tree node. After building the tree, we utilize a PEP post pruning to avoid problem of overfitting. Then, with the decision tree after PEP pruning as the basic decision tree, we develop a dynamic cost pruning strategy to make the decision tree satisfy different cost constraints. To increase the efficiency, we propose a cost-based pruning strategy based on cost constraint. We have proven that the worst complexity of the dynamic algorithm decreases from O(NK2) to O(N), where N is the size of training set and K is the number of attributes.

The contributions of this paper are summarized as follows.

- We develop a decision tree model. With the model, for a data object, we can get different decision trees with different requirement of cost constraints in a tolerable time.
- We propose a new method to induce a decision tree with a consideration of both the costs and information acquirement.
- We conduct extensive experiments to verify the effectiveness of the proposed methods. From the experimental results, the proposed method could achieve high accuracy within the cost constraint.

The remaining parts of this paper are organized as follows. Section 2 introduces some background knowledge of this paper. Section 4 describes the proposed algorithm in detail. We present the experiments and analysis in Sects. 5 and 6 draws the conclusions and give a future research directions.

2 Background

In this section, we introduce one of the based knowledge of decision tree, PEP (Pessimistic Error Pruning) post-pruning algorithm which is also a part of C4.5.

The main reason for pruning is to reduce the complexity of the decision tree and help prevent overfitting. However, among pruning methods, we choose PEP for the following reasons. Firstly, PEP does not use extra pruning set. Thus, it provides sufficient data for testing. Secondly, this method has valid statistics theoretical foundations. Thirdly, PEP is a powerful pruning strategy and quite efficient in terms of computational effort [10].

In the PEP, as the apparent error rate is optimistically biased, Quinlan [1] add an 0.5 adjustment to PEP pruning in order to provide a more practical error rate. Quinlan considered the event that whether a sample is classified correctly is in a Bernoulli distribution. 1 means correct classification and 0 means wrong classification. Thus, we can get the estimated error rate through samples statistics as

$$e_{subtree} = (\Sigma E_i + 0.5 * L)/\Sigma N_i$$

where E_i is error number of each leaf node, and L is the number of leaf nodes. Then we can estimate mean and variance of error count defined as follows.

$$E_{subtree\_\text{err}\_\text{count}} = N * e_{subtree} \quad \text{var}_{subtree\_err\_count} = \sqrt{N * e_{subtree} * (1 - e_{subtree})}$$

Then we let a leaf node with most frequent class label Ci in N replace the subtree. Hence the error count becomes E while sample size is still N and its error rate and mean of error count is calculated as follows.

$$e_{new\_leaf} = (E + 0.5)/N \quad E_{leaf\_err\_count} = N * e_{new\_leaf}$$

While the mean of subtree's error count is larger than the mean of new leaf node's error count by a variance, we replace this subtree with a leaf node. We get the following inequality.

$$E_{subtree\_err\_count} - \text{var}_{subtree\_err\_count} > E_{leaf\_err\_count}$$

We have introduced the main idea of the PEP pruning. We will utilize this method in the second step of our method.

3 Overview

3.1 Problem Definition

Definition 1 (Cost of a Decision Tree): The cost of a decision tree is defined as the maximal sum of the acquirement costs of the attributes from root to a leaf, denoted by cost(T). The cost means the maximal acquirement costs of the attributes required for the classification with T.
We denoted the optimal decision tree generated from an attribute set S as Acc(S). The problem is defined formally as follows.

Problem 1. Given a training set S with each tuple having attribute set $S_T = \{T1, T2, \ldots, Tn\}$, the goal is to find a decision tree model M, such that given a data object o and a cost C, the decision tree T is derived from M with attribute set $SR \subseteq ST$, such that cost ($T(SR)$) $\leq$ C and $\forall S' \subseteq$ ST with $cost(T(S')) \leq C$, Acc(S') $\leq$ Acc(SR).

3.2 The Overview of the Proposed Method

In this section, we show an overview of our solution. In our solution, the model is a global decision tree with all attributes. For a classification task with a data object and cost constraint, the global decision tree is pruned accordingly. The method contains three steps.

Step 1: We apply a branching strategy to generate a decision tree with all attributes as the model. For the sake of keeping the correctness of decision tree algorithm, we still take IGR (information gain ratio) as a part of choice criterion. It is also one of the standards to judge whether branching should stop, since if IGR of the attribute of class is equal to zero, it means there are only one class in the subset [11]. As we should consider of the cost of attributes. We develop a new branching criterion, CGR (cost gain ratio) as follows, while IGR of T_i is not equal to zero.

$$CGR(T_i) = \frac{\cos t(T_i)}{IGR(T_i)}$$

And In the premise that *IGR* of the class is not equal to zero and sample size is not lower than pre-threshold, then we recuse with the rest of conditional attributes and compared the value of $CGR(T_i)$ with each other, then we use the Greedy Algorithm to find the smallest one to generate the splitting attribute, which means the attribute with the lowest cost for each percentage point of info gain ratio.

For each discrete attribute, a branch is generated according to each value. For each continuous attribute, we attempt to find the best splitting point. The methods of calculating the best split point of continuous attribute and $IGR(T_i)$ is the same in C4.5 algorithm [11].

Step 2: After Step 1, we get a complete decision tree which fits the training data perfectly. Such tree is treated as the model. However, it may be overfitting and becomes inaccuracy on other data set. In order to avoid overfitting, we use re-pruning method which means that when the sample size is lower than a threshold, the decision tree stops to grow. Then, we use the PEP (Pessimistic Error Pruning) post-pruning algorithm of C4.5 since such algorithm does not need extra testing data and have a linear computational complexity. Thus, the pruning is exhaustive.

Step 3: Through Step 2, we get a basic decision tree as the model. All the decision trees are derived from it. Hence when a data object o is to be handled, we perform a second round of pruning according to the required cost constraint and generate a decision tree *T*. The classification of o is determined based on *T*.

In practice, we can combine PEP pruning with global decision tree generation, since they could be performed offline and will not affect the experience of individual users. Even though both Step 2 and Step 3 prune the global decision tree, we do not merge them since Step 3 has to be executed online and Step 2 could be performed offline.

4 Algorithms

In this section, we will discuss the algorithms based on the framework introduced in Sect. 3.2. Since Step 2 applies existing method introduced in Sect. 2, we discuss the algorithms in Step 1 and Step 3 in Sects. 4.1 and 4.2, respectively.

4.1 Branching Strategy

This step builds a global decision tree containing all attributes as the basic model and the cost-based pruning strategy will be applied on such model. For the convenience of the pruning in the following steps, the node with small cost tends to locate near to the root such that they are not easy to prune and we make the zero cost attributes on the top of the global decision tree.

The building of global decision tree always starts with the growth of a tree which means a branching strategy among all the attributes. Therefore, we need to deal with all attributes by a criterion called cost-gain-ratio to find the attribute with a smallest cost-gain-ratio as the best splitting attribute. By the best splitting attribute, the data set is divided into several parts. We handle the data set in such way recursively until all the instances belong to one class or the size of sample in each leaf is lower than the threshold. When the recursion finishes, we get the global decision tree.

At first, we introduce a new criterion called cost-gain-ratio *(CGR)* for the generation of global decision tree. Such criterion combines the cost of attributes and the information gain ratio as in C4.5 algorithm (Quinlan [11]). For an attribute T_i, we define that its *CGR* is in direct proportion to the cost (denoted by $cost(T_i)$) and inverse proportion to the information gain ratio. That is,

$$CGR(T_i) = \frac{\cos t(T_i)}{IGR(T_i)} \tag{1}$$

To make *CGR* well-defined, we need first check whether *IGR(Class)* equals to zero. If it equals to 0, such attribute is useless to the classification and could be discarded. An attribute T_i with a smaller $CGR(T_i)$ attempt to be cheaper in cost and more powerful in classification. During the usage *CGR*, the decision tree construction prefers the attribute with small acquirement cost and powerful ability of classification. Specially, for an attribute with zero cost, its *CGR* is 0 and could be selected directly.

The cost is defined according to the application and we need to calculate the information gain ratio. It is defined according to the entropy, which is common used in decision tree and defined as follows.

$$Entropy(C, S) = -\sum\nolimits_{i=1}^{m} p_i \log p_i \tag{2}$$

Let m denote the number of classes, C denote the attribute, and pi denote the proportion between the amount of *i-th* class instances to the amount of total sample size S.

Since we want to calculate the information gain ratio, which is a ratio of the gain of the attribute to the splitInfo of the attribute in the data set S. The gain of the attribute is the difference between the entropy of class and that of the attribute which is defined as $\sum_{v \in Value(S_t)} \frac{|s_{t,v}|}{|s_t|} Entropy(t_v)$, and the splitInfo of the attribute t is defined as $SplitInfo(t, S) = \sum_{v \in S} -\frac{|S_{t,v}|}{S_t} \log \frac{|S_{t,v}|}{S_t}$ Therefore, according to S, the gain of attribute t is defined as follows.

$$Gain(t,S) = Entropy(C,S) - \sum\nolimits_{v \in Value(S_t)} \frac{|S_{t,v}|}{|S_t|} Entropy(t_v) \tag{3}$$

where $Value(S_t)$ means all the value of attribute t in data set S, $|S_{t,v}|$ means the number of samples in data set S with attribute $T = v$.

Hence when we got the gain of the attribute, we need to get the information gain ratio, denoted as $GainRatio(T, S)$. For an attribute t in the data set S, $GainRatio(t,S) = \frac{Gain(t,S)}{splitInfo(t,S)}$.

The algorithm selects the attributes greedy according to *CGR* of all attributes. The pseudo code is shown in Algorithm 1. In this algorithm, the tree is generated recursively.

In this algorithm, Line 1 to Line 9 check recursion stopping condition, including that the dataset S is null (Line 1), no attribute is left (Line 2–3), single attribute is left (Line 4), and the size of the left dataset is smaller than a threshold (Line 5). The loop for selecting the best split attribute is in Line 8–20. Line 21–22 update parameters for next recursion.

In each loop, we compute the entropy of S according to (2) in Line 11, and in Line 12–14, we compute the entropy info and split info of attribute t. Then, in Line 15, we compute the gain info of attribute t according to (3). In Line 18, we compute the info gain ratio and cost gain ratio of attribute according to (1). With Line 16–17, we keep the attribute with the smallest cost-gain-ratio.

Since it is known that classification based on the entropy is reliable, similar as information gain ratio of C4.5 algorithm, CostGrain-Ratio is reasonable.

Note that calculating CostGrainRatio does not increase the time complexity of the algorithm, still linear complexity.

After building a complete decision tree, we can use the PEP method (introduced in Sect. 2) to optimize the tree, and generate the pruned global decision tree as the model.

4.2 Cost Pruning Strategy

In this part, we propose the algorithm for decision tree generation based on the cost constraint. To achieve this goal, we develop a pruning strategy on the global decision tree. The major idea is to traverse all the nodes in the global decision tree. For each node v, we calculate the sum cost of the nodes from the root to r, and judge that whether the sum cost is over the cost constraint. The pseudo code is shown in Algorithm 2.

This method handles the global decision tree nodes from top to bottom. For a specified cost constraint C, we calculate the total cost from root to node and then judge whether the total cost violates the cost constraint in Line 2. If so, we remove the subtree with *node* as the root and only *node* is left (Line 3). Otherwise, we move to the next *node* until all nodes in the tree are visited.

After this step we can get a decision tree with the cost of each root-to-leave path lower than the cost constraint. Hence the cost of the pruned decision tree is within the cost constraint.

Algorithm 1 Formtree(*S*, *T*, *C*, *P*, *Tree*)

Input: training dataset *S*, attributes *T*, class attribute *C*, costs *P*, the decision tree *Tree*
1: if *S* is null **then return** failure
2: if *T* is null **then**
3: **return** the node with the most frequent class label C_i in *S*
4: if $|T| \leq 1$ **then return** *T*
5: if $|S| < threshold$ **then**
6: **return** a leaf node with the most frequent class label C_i in *S*
7: ***Tree*=∅**
8: for each attribute $t \in T$ where the sample is *S* **do**
9: Info(*t*,*S*)=0, SplitInfo(*t*,*S*)=0;
10: MinGainRaio=100, node=null
11: $Entropy(C,S) = -\sum_{i=1}^{m} p_i \log p_i$
12: **for** $v \in$ value(*t*,*S*) **do**
13: set $S_{t,v}$ as the subset of *S* where attribute $t=v$
14: $Info(t,S) = \frac{|S_{t,v}|}{S_t} Entropy(t_v)$
15: $SplitInfo(t,S) = -\frac{|S_{t,v}|}{S_t} \log \frac{|S_{t,v}|}{S_t}$
16: Gain(*t*,*S*)=Entropy(*C*,*S*)-Info(*t*,*S*)
17: $GainRatio(t,S) = \frac{Gain(t,S)}{splitInfo(t,S)}$
18: **if** CostRatio(*t*,*S*)<MinGrainRatio **then**
19: MinGainRatio=CostGrainRatio(*t*,*S*); *node*=*t*
20: attach *node* to *Tree*;
21: update *S*, *T*, *C*, *P*
22: Formtree(*S*, *T*, *C*, *P*,*Tree*)

Algorithm 2 CostPruning(*Tree*, *n*, *c*)

Input: decision tree *Tree*, depth of tree *n*, cost constraint *c*;
1: for $node=nodes_1$ **to** *leaf_node* **do**
2: **if** TotalCostTo(*node*)>*c* **then**
3: turn subtree of the node to a leaf node with the most frequent class label in it

4.3 Algorithm Complexity

As discussed in Sect. 3.2, the branching and PEP pruning are executed at the same time. We denote the number of attribute and tuples in the training set as *K* and *N*, respectively. Since each layer only having one attribute, there are K layers in a tree totally. Since each layer of the tree needs to process *K* attributes and for each attribute, we need to traverse all the tuples in the training set. Hence the time complexity of these two steps is *O(NK2)*. Even though the cost in superlinear, these two steps are executed offline. During each cost pruning, we need to traverse the whole global decision tree a time and the number of nodes in the tree smaller than *K*. Therefore, the time complexity of online part (Step 3) is *O(K)*.

5 Experiment

In this section, we conduct extensive experiments to evaluate the performance of our algorithm. We compare our approach with C4.5 algorithm. All algorithms are implemented in python by with IDE Py2.7.9 IDLE 2.7.9 and running on the environment of Windows8.1 Pro with Inter CPU 2.39 GHz and 8.00 G RAM. We measure the

accuracy, which is ratio of the right classified records to the total records in the testing set. For the convenience of discussions, we define the decision tree after PEP as *PEP tree* and the maximum sum of the acquirement cost of attributes in all paths in the decision tree after PEP as *PEP cost*.

For experiments, we use three data sets from UCI Machine Learning Repository with various applications and properties. Their information is summarized in Table 1. To handle the missing values in the data set, we remove the tuples with 40% missing values and fill the blanks in other tuples with the mean value of other values in the corresponding attribute. For data set H and D, we use the attribute cost in the data set while for data set T, the cost of each attribute is set randomly from 0 to 1000.

Table 1. Information of data set

ID	Data set name	#Training tuples	#Testing tuples
H	Hepatitis	115	34
D	Heart disease	210	70 + 34
T	Thyroid	7980	1192

5.1 Comparisons

In this section, we compare the performance of the proposd algorithm with C4.5. The experimental results are shown in Figs. 1 and 2, respectively.

From Fig. 1, we have two observations. (1) The acquirement cost of attributes used in C4.5 algorithm is 27.8, while the acquirement cost used in our approach is at most 14.8, which is almost half of C4.5 cost. However, the largest gap of these two approaches is about 0.06. (2). The accuracy of our approach increase with cost decreasing when the cost is large and our approach outperforms C4.5 on both cost and accuracy when the cost is small. When the accuracy of our approach is smaller than that of C4.5 by no more than 15%, the cost of C4.5 is about ten times of ours, and sometimes we even get a better accuracy. Our approach outperforms C4.5 in accuracy as the cost is small.

From Fig. 2, we observe that with the data size increasing, the gap between our accuracy and C4.5 is smaller in general by almost 0.02, while the cost of C4.5 which is equal to 584 is nearly 3 times larger than our largest cost (222). And even though at the point of zero, the cost gap of these two approaches 0.12 in accuracy. When the data size increases from 114 to 280, the accuracy of our method increases as the average gap between C4.5 and our algorithm in accuracy is about 0.03 and the cost of C4.5 is 10 times larger than that of our approach.

5.2 Accuracy Vs. Cost Constraint

In this section, we test the relationship between the cost and accuracy. For D, we vary the cost constraint from 0 to 16, since the largest possible cost of decision trees after the cost pruning step is the cost of PEP tree (14.27).

The results are shown in Fig. 4. From Fig. 3, we observe that when the cost constraint is small, the accuracy changes significantly with the cost constraint and is better than those with larger cost constraint. As a comparison, with the cost increasing, the accuracy decreases but becomes stable. When the cost goes up to a point with cost constraint higher than PEP cost (14.27), the accuracy goes down to accuracy of the PEP tree (0.73529) and remains unchanged as the cost will not be over 14.27. Since the cost of any generated decision tree is smaller than PEP cost, the tree will be unchanged when cost constraint gets larger than PEP cost.

We vary the constraint from 0 to 230 step by 10, since the PEP cost is 222.17. And the results are shown in Fig. 4. From Fig. 5, we observe that the accuracy is stable with the changing of cost. We also observe that when the cost constraint is small, the accuracy changes significantly with the cost constraint, similar as that of dataset D.

From the discussion, Figs. 3 and 4 have similar trends and the accuracy converges that of the PEP tree. This is because all decision trees pruned according to the cost is based on the PEP tree. The pruning according to the cost constraint prevents the overfitting furthermore and thus the accuracy of cost-based-pruned decision tree is better than that of PEP tree in most of cases.

From the experimental results, the accuracy of our approach is comparable to C4.5 even though our approach has a pretty small cost constraint. It is caused by the use of new criterion cost gain ratio. Such that we keep all the zero cost attribute first, and the choice of splitting attribute has tend to low-cost attribute. The nodes far from the root tend to have high costs and will be pruned in high possibility.

5.3 The Impact of Attribute Costs

To test the impact of attribute costs, we set the costs to the attributes of T randomly for 5 times and the results are plotted in Fig. 5.

The cost of C4.5 is 200, which is larger than the maximal cost of our approach in the experiment. The cost of our approach and C4.5 is shown in the brackets in the accuracy and C4.5 in the legend, respectively.

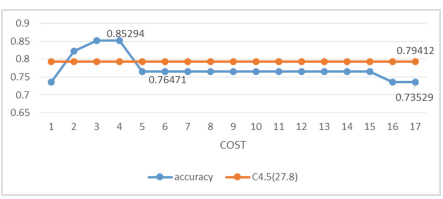


Fig. 1. Comparisons on H

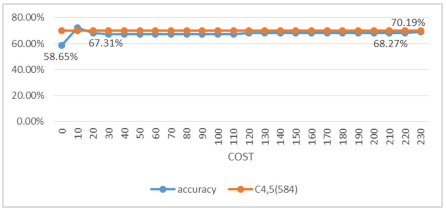


Fig. 2. Comparisons on D

Combing the experimental results in Figs. 1, 2 and 5, we can observe that the accuracy gap is about 11% in Fig. 1 with data size of 149 and 9% in Fig. 2 with data size of 280, and in Fig. 5 the largest gap is no more than 1% with data set of 9172. In Fig. 5(a) and (c), our approach outperforms C 4.5 in both accuracy and cost. Thus, we

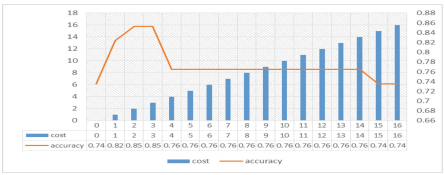


Fig. 3. Cost constraint vs. accuracy on H

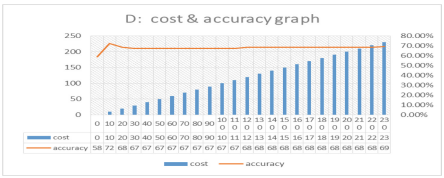
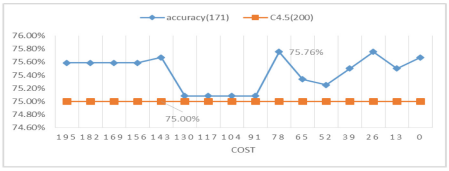
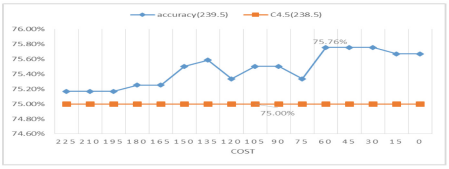


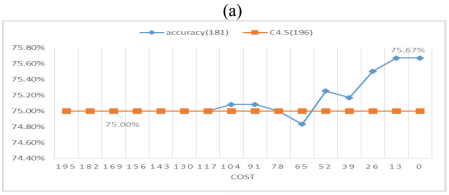
Fig. 4. Cost constraint vs. accuracy on D



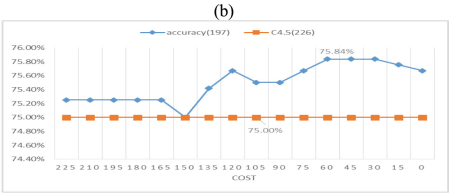
(a)



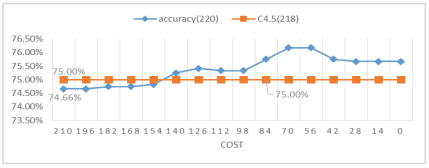
(b)



(c)



(d)



(e)

Fig. 5. Experimental results on T

have two conclusions. (1) When the cost constraint is smaller enough, our approach has a better accuracy than C4.5 in all the teams. (2) With the data size increasing, the gap of accuracy between C4.5 and our algorithm become smaller.

6 Conclusion

In this paper, we propose a generation algorithm of cost-based decision tree. The motivation is to solve classification problems with the non-ignorable attribute acquirement cost. To solve this problem, we develop an approach to generate the global decision tree with all attributes as the model. For a given object, the global decision tree is pruned according to the cost constraint. Experimental results demonstrate that for large data sets, our algorithm outperforms C4.5 in both cost and accuracy.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China (No. 61370222), the Natural Science Foundation of Heilongjiang Province (No. F201430), the Innovation Talents Project of Science and Technology Bureau of Harbin (No. 2017RAQXJ094), and the fundamental research funds of universities in Heilongjiang Province, special fund of Heilongjiang University (No. HDJCCX-201608).

References

1. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
2. Basgalupp, M.P., et al.: LEGAL-tree: a lexicographic multi-objective genetic algorithm for decision tree induction. In: ACM Symposium on Applied Computing, pp. 1085–1090 (2009)
3. Lopes, C.M., Zaverucha, G.: HTILDE: scaling up relational decision trees for very large databases. In: ACM Symposium on Applied Computing, pp. 1475–1479 (2009)
4. Rodriguez, J.J., Alonso, C.J.: Interval and dynamic time warping-based decision trees. In: ACM Symposium on Applied Computing, pp. 548–552 (2004)
5. Ren, C., King, B.R.: Predicting protein contact maps by bagging decision trees. In: International Conference on Bioinformatics, pp. 649–650 (2014)
6. Blockeel, H., Struyf, J.: Efficient algorithms for decision tree cross-validation. J. Mach. Learn. Res. **3**(1), 621–650 (2003)
7. Hong, J.: AE1: an extension matrix approximate method for the general covering problem. Int. J. Parallel Prog. **14**(6), 421–437 (1985)
8. Tu, P., Chung, J.: A new decision-tree classification algorithm for machine learning. In: International Conference on Tools with Artificial Intelligence, pp. 370–377 (1992)
9. Lomax, S., Vadera, S.: A survey of cost-sensitive decision tree induction algorithms. ACM Comput. Surv. **45**(2), 16–25 (2013)
10. Barros, R.C., et al.: Towards the automatic design of decision tree induction algorithms. In: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, pp. 567–574. ACM (2011)
11. Quinlan, J.R.: C4.5: Programs for Machine Learning. Elsevier, Amsterdam (2014)