Information Sciences 179 (2009) 967–979

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Building a cost-constrained decision tree with multiple condition attributes

Yen-Liang Chen [a,*], Chia-Chi Wu [a], Kwei Tang [b]

[a] *Department of Information Management, National Central University, No. 300, Jhongda Road, Chung-Li, Jhongli City 320, Taiwan, ROC*
[b] *Krannert School of Management, Purdue University, West Lafayette, IN 47907, USA*

A R T I C L E   I N F O

Article history:
Received 4 February 2008
Received in revised form 16 November 2008
Accepted 19 November 2008

Keywords:
Data mining
Decision analysis
Cost-sensitive learning
Classification
Decision tree

A B S T R A C T

Costs are often an important part of the classification process. Cost factors have been taken into consideration in many previous studies regarding decision tree models. In this study, we also consider a cost-sensitive decision tree construction problem. We assume that there are test costs that must be paid to obtain the values of the decision attribute and that a record must be classified without exceeding the spending cost threshold. Unlike previous studies, however, in which records were classified with only a single condition attribute, in this study, we are able to simultaneously classify records with multiple condition attributes. An algorithm is developed to build a cost-constrained decision tree, which allows us to simultaneously classify multiple condition attributes. The experimental results show that our algorithm satisfactorily handles data with multiple condition attributes under different cost constraints.

1. Introduction

Classification, which is a data mining task, necessitates the building of a model or a classifier for a given set of pre-classified examples in order to classify categories of new events [19]. Of the numerous approaches, decision trees are probably the most common of the classification models [1,7] and have been successfully used in various applications, including medicine, manufacturing, production, financial analysis, astronomy, and molecular biology [5]. Existing studies have identified several advantages to the use of decision trees: no domain knowledge is needed for classification, they are able to handle high dimensional data, they are intuitive and generally easy to comprehend, they are simple and fast, and they have good accuracy [5].

Basically, a decision tree is nothing but a directed acyclic graph containing a root, a set of nodes, and a set of edges. Within the decision tree, an internal node denotes the test of a decision attribute, a branch represents the outcome of this test, and a leaf node is associated with a condition attribute label. Fig. 1 shows a decision tree with the condition attribute “B_Car”, which indicates whether a customer will buy a car or not (1 = buy; 0 = not buy).

Most decision tree algorithms are designed for the classification of data with a single condition attribute. In many real world applications, however, we need more than one condition attribute per record. For example, a bank not only needs to evaluate a customer’s credit rating but also needs to be able to predict his/her likelihood to ask for a loan in the near future. In medical diagnosis, the doctor needs to diagnose many kinds of diseases based upon test results and patient symptoms. In both cases, it is necessary to predict the values of multiple condition attributes according to a given set of decision attributes.

To the best of our knowledge, no previous decision tree studies have ever addressed the issue of classifying multiple condition attributes. This is not surprising, because if no costs are associated with the decision attributes or if there are

* Corresponding author. Tel.: +886 3 4267266; fax: +886 3 4254604.
E-mail address: ylchen@mgt.ncu.edu.tw (Y.-L. Chen).

0020-0255/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2008.11.032

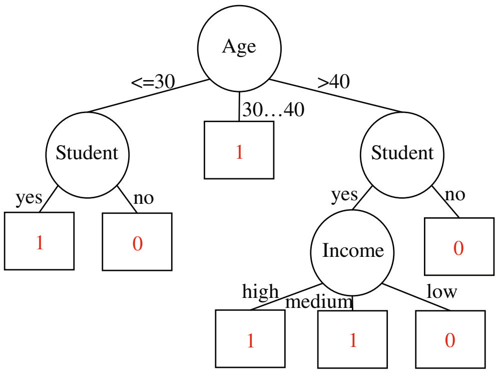


Fig. 1. Example of a decision tree.

no budgetary concerns, the multiple condition attributes problem can be solved independently by building a decision tree for each condition attribute. For example, if the physician needs to ascertain the potential for four different kinds of disease, there is no need to build a decision tree to determine all four disease potentials simultaneously. Instead, separate decision trees can be built to determine each of these four diseases.

Unfortunately, building independent decision trees for each condition attribute does not work when classifying multiple condition attributes under a total budgetary constraint. The difficulty lies in the fact that these decision trees may share some common decision attributes. For example, when determining multiple disease potentials there may be some common tests that can be used. The first time the test is done, we, of course, need to expend money to obtain it. Afterwards however, the previously obtained test results can be used to determine other potential diseases without accruing any additional cost. In other words, the cost of a test occurs only once, no matter how many times its results are used. In these types of situations, traditional approaches may encounter the following difficulties:

(1) It is difficult to determine how much of the budget should be allocated to classifying a condition attribute since there are numerous budget allocation combinations for the condition attributes.
(2) Since the test results can be shared by multiple decision trees, these trees are not independent. Due to their interdependency, they cannot be built independently.

To minimize these difficulties, we develop an algorithm which can be used to build cost-constrained decision trees to classify multiple condition attributes. Our algorithm can assign a future record a label for each condition attribute, without spending more on test costs. The structure and termination condition of the nodes in the decision tree are also altered to adapt to this context. Fig. 2 shows an example of a cost-constrained decision tree with two condition attributes, B_Car and Credit. In this tree, a label can be assigned to a condition attribute at an internal node, if the data in this node is discriminating enough to determine the label (e.g., n_3, n_5, n_6). A leaf node may include either one label (e.g., n_7, n_{12}, n_{17}) or two labels (e.g., n_{10}, n_{18}, n_{20}).

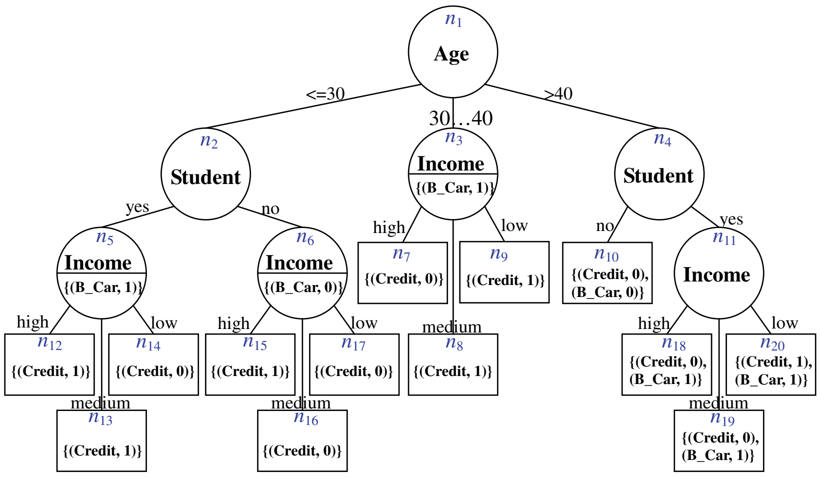


Fig. 2. Example of a cost-constrained decision tree with multiple condition attributes.

This study makes two significant contributions. First, we model a new decision tree problem, classifying multiple condition attributes under a cost constraint. Second, an algorithm is developed to solve the proposed problem. The remainder of this paper is organized as follows. We first review some related work in Section 2. In Section 3, we formalize the problem then introduce the algorithm in Section 4. The performance evaluation is presented in Section 5. Conclusions and suggestion for future work are discussed in Section 6.

2. Related work

Many algorithms have been developed for decision tree induction. In most of these methods, an attempt is made to maximize classification accuracy without any consideration to cost; see for example, ID3 [14], C4.5 [15], CART [2], Quest [10], and GATree [13]. Since classification in real world applications may involve different types of costs, such as misclassification costs or test costs, much attention has been given to the cost-sensitive classification issue in recent decades. Discussions of misclassification costs include [3,4,6]. Additionally, [11,12,16] have tried to balance classification accuracy and test costs. In recent years, more and more attention has been focused on both misclassification costs and test costs, such as in [8,9,17,18]. Instead of information gain, in the studies mentioned above, the sum of the misclassification and test costs has been used as the splitting criterion for attribute selection. In some of these studies cost has been taken as an adjustment function, in others cost has been regarded as a constraint, while in others, any kind of cost has been disregarded. All in all, whether cost-sensitive or not, all these algorithms only classified a target record with a single condition attribute. As a consequence, no previous algorithms have been able to solve problems where there are multiple condition attributes and where a total cost constraint is specified.

3. Problem definition

Decision tree induction involves the building of a decision tree from a training dataset *D*, which consists of a collection of records. A record d_k, which is the *k*th record in *D*, consists of a number of decision and condition attribute values, as indicated by the tuple in Fig. 3. It can be seen that there are *m* decision attributes and *n* condition attributes; $a_x\,(d_k)$ is the value of the *x*th decision attribute in d_k, while $c_y\,(d_k)$ is the label of the *y*th condition attribute in d_k. An example, training dataset *D* is shown in Table 1. Within the training dataset in Table 1, "Age", "Income", and "Student" are decision attributes that can be tested, while "Credit" and "B_Car" are condition attributes that must be determined.

A decision tree *T* is a directed acyclic graph that contains a root, a set of nodes, and a set of edges. Fig. 2 shows a decision tree built from the training dataset in Table 1. We use n_i to denote the *i*th node in decision tree *T*. An internal node n_i in the decision tree is associated with a decision attribute $s\,(n_i)$, which splits n_i. For example, in Fig. 2, n_1 and n_3 are internal nodes, where $s\,(n_1)$ = "Age" and $s(n_3)$ = "Income".

An important difference between our decision tree and traditional decision trees is that the labels for the condition attributes can be determined at an internal node if this internal node's data is discriminating enough. Let *CLpair* be a pair composed of a condition attribute and its assigned label. A *CLpair* takes the form of (c_y, l_y), where c_y is a condition attribute and l_y is a label of c_y. For example, consider node n_5 in Fig. 2 containing *CLpair* ("B_Car", 1). This means that label 1 is assigned to condition attribute B_Car at node n_5. For any node n_i in a decision tree, we use $L\,(n_i)$ to denote the set of *CLpair*s attached to n_i, and $O(n_i)$ to denote the set of condition attributes that have not yet been determined in node n_i. Semantically, $L(n_i)$ indicates

$a_1(d_k)$	…	$a_m(d_k)$	$c_1(d_k)$	…	$c_n(d_k)$

Fig. 3. Record format.

Table 1
Training data set ***D***.

Record	Age	Income	Student	Credit	B_Car
d_1	<=30	High	No	0	0
d_2	<=30	High	No	1	0
d_3	31···40	High	No	0	1
d_4	>40	Medium	No	0	1
d_5	>40	Low	Yes	0	1
d_6	>40	Low	Yes	1	0
d_7	31···40	Low	Yes	1	1
d_8	<=30	Medium	No	0	0
d_9	<=30	Low	Yes	0	1
d_{10}	>40	Medium	Yes	0	1
d_{11}	<=30	Medium	Yes	1	1
d_{12}	31···40	Medium	No	1	1
d_{13}	31···40	High	Yes	0	1
d_{14}	>40	Medium	No	1	0

which labels have been assigned to which condition attributes in node n_i, and $O(n_i)$ are the remaining condition attributes that still have no values in node n_i. For example, $O(n_1)$ = {"Credit", "B_Car"} and $L(n_1) = \phi$, $O(n_3)$ = {"Credit"} and $L(n_3)$ = {("B_Car", 1)}, $O(n_{10}) = \phi$ and $L(n_{10})$ = {("Credit", 0), ("B_Car", 0)}.

There are many edges in a decision tree. We use an ordered pair $\langle n_i, n_j \rangle$ to denote an edge that links nodes n_i and n_j, where n_i is the father node of n_j. Either a single value or an interval of values of $s(n_i)$ is assigned to edge $\langle n_i, n_j \rangle$ to represent the test conducted there. For example, $\langle n_1, n_3 \rangle$ in Fig. 2 is an edge that links n_1 and n_3. The interval "30· · ·40" is assigned to this edge to represent the condition that the age must be fall within the interval "30· · ·40".

As mentioned above, a decision tree is built from a training dataset D. Initially, all records in D are located at the root of the decision tree. Afterwards, the dataset is partitioned according to each node's tested attribute. The distribution of the records in Table 1, with respect to the decision tree in Fig. 2 is shown in Fig. 4.

We use D_{n_i} to denote the set of records d_k located in node n_i. In Fig. 4 for example, we have $D_{n_4} = \{d_4, d_5, d_6, d_{10}, d_{14}\}$ and $D_{n_6} = \{d_1, d_2, d_8\}$. We use $D_{n_i}(a_x = v)$ to denote the subset of records $d_k \in D_{n_i}$ satisfying a_x $(d_k) = v$. For example, $D_{n_6}(\text{"income"} = \text{"high"}) = \{d_1, d_2\}$. In addition, we use $D_{n_i}(c_y = l)$ to denote another subset of $d_k \in D_{n_i}$ satisfying c_y $(d_k) = l$. For example, $D_{n_4}(\text{"B\_Car"} = 0) = \{d_6, d_{14}\}$.

Conducting tests on decision attributes always creates costs. *TestCost*(a_x) represents the cost of conducting a test for decision attribute a_x. In Table 2, we set test costs for all the decision attributes in Table 1.

The total test cost for a record accumulates as this record goes through the internal node. Thus, we use *ArrCost*(n_i) to denote the total cost required for a record to travel from the root to node n_i. For example, in Fig. 2, the value of *ArrCost*(n_5) is 16.

Given a training dataset with multiple condition attributes, the problem of building a cost-constrained decision tree with multiple condition attributes can be defined as building a decision tree that maximizes the classification accuracy of all condition attributes. Additionally, the total cost of classifying a record cannot exceed a given threshold *maxcost*.

4. The algorithm

In this section, we propose a new algorithm to classify multiple condition attributes with a cost constraint. The outline of our algorithm is shown in Fig. 5. The framework looks like C4.5, however, to adjust to the context of the new problem, we propose a new splitting criterion, a new node structure, and an additional adjusting phase. In this section, we demonstrate our algorithm by using it with the training dataset shown in Table 1. The test cost of each attribute is listed in Table 2 and the total cost *maxcost* is set at 16.

The new splitting criterion, the multi-dimensional information gain, is the most important part of the new algorithm and will be introduced in Section 4.3. It stems from two other concepts, information need and distance, which are discussed in Section 4.2. Furthermore, to formally define the information need, we must transform a set of condition attributes into a

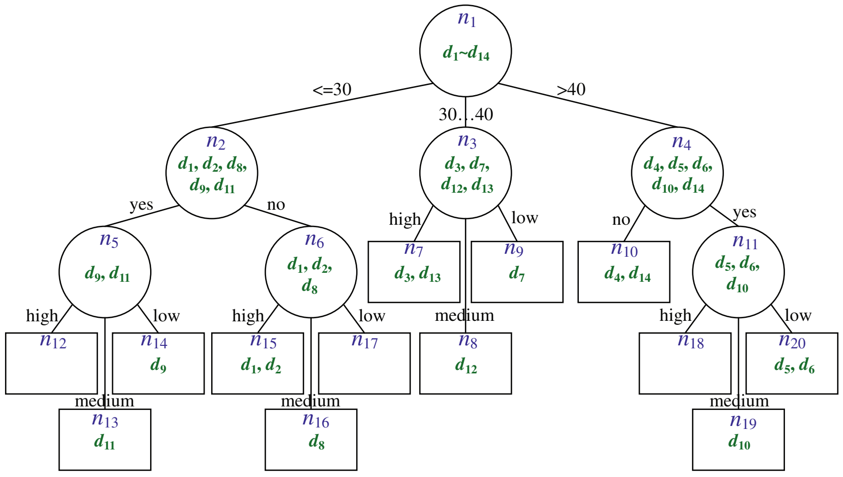


Fig. 4. Distribution of records in a decision tree.

Table 2
Test costs of decision attributes from Table 1.

a_x	*TestCost* (a_x)
Age	6
Income	5
Student	10

```
Input:
Training dataset $D$,
Test cost of each decision attributes $TestCost(a_x)$ ,
Threshold of total test cost $maxcost$,
Label assignment threshold $\theta$
Terminal node threshold $\varphi$

Output:
A cost constrained decision tree with multiple condition attributes.

Process:
1. Build an initial tree
    1.1. Starting with a single node, root
    1.2. For each non-leaf node, $n_i$
        1.2.1. Perform label assignment test to determine if there    /* see Section 4.5
               are any labels that can be assigned.
        1.2.2. Select an attribute according to splitting criterion to /* see Section 4.4
               further split $n_i$.
        1.2.3. If terminal condition is satisfied, stop splitting and /* see Section 4.6
               assign $n_i$ as a leaf node.
2. Update the bottom nodes in the tree built in step1.    /* see Section 4.7
```

Fig. 5. Outline of our algorithm.

multi-dimensional attribute. The transformation is described in Section 4.1. Finally, all other details are given in Sections 4.4–4.6 and 4.7.

4.1. Multi-dimensional attribute (MDA)

For node n_i, let $O(n_i) = \{c_1, c_2, \ldots, c_w\}$ be the set of condition attributes that have not yet been determined in node n_i. In this way, we can transform the attribute set $O(n_i)$ into a multi-dimensional attribute (*MDA*). For example, $O(n_i) = \{c_1, c_2, \ldots, c_w\}$ is transformed into $c_{MDA(n_i)} = (c_1, c_2, \ldots, c_w)$. Accordingly, the labels of a record d_k in n_i can be represented as an *MDA* label, which is a vector with w elements. Thus, we have $c_{MDA(n_i)}(d_k) = (c_1(d_k), c_2(d_k), \ldots, c_w(d_k))$, where $c_y\ (d_k)$ is the label of c_y in d_k. We use $MDL(n_i)$ to denote the set of all possible *MDA* labels in n_i, and let $mdl_z(n_i)$ be the zth *MDA* label in $MDL(n_i)$. In our example, since $O(root)$ = {“Credit”, “B_Car”}, we have $c_{MDA(root)}$ = (“Credit”, “B_Car”), $MDL(root) = \{(0,0),(0,1),(1,0),(1,1)\}$, $mdl_1(root) = (0,0)$, $mdl_2(root) = (0,1)$, $mdl_3(root) = (1,0)$, and $mdl_4(root) = (1,1)$. Therefore, the training dataset in *root* can be transformed into the form seen in Table 3.

4.2. Information need and distance

After transforming the condition attributes into an *MDA*, the information need proposed in ID3 can be applied. Therefore, the expected information for the labels in n_i can be given by $Info(D_{n_i}) = -\sum_{z=1}^{|MDL(n_i)|} p_z \log_2(p_z)$, where p_z is the probability of an arbitrary record in D_{n_i} with label $mdl_z(n_i)$, and is estimated by

$$\frac{|\text{records with label } mdl_z(n_i) \text{ in } D_{n_i}|}{D_{n_i}}.$$

Table 3
Transformed *root* records.

Record	Age	Income	Student	(Credit, B_Car)
d_1	<=30	High	No	(0, 0)
d_2	<=30	High	No	(1, 0)
d_3	31···40	High	No	(0, 1)
d_4	>40	Medium	No	(0, 1)
d_5	>40	Low	Yes	(0, 1)
d_6	>40	Low	Yes	(1, 0)
d_7	31···40	Low	Yes	(1, 1)
d_8	<=30	Medium	No	(0, 0)
d_9	<=30	Low	Yes	(0, 1)
d_{10}	>40	Medium	Yes	(0, 1)
d_{11}	<=30	Medium	Yes	(1, 1)
d_{12}	31···40	Medium	No	(1, 1)
d_{13}	31···40	High	Yes	(0, 1)
d_{14}	>40	Medium	No	(1, 0)

In Table 3, there are four different *MDA* labels (0,0), (0,1), (1,0), and (1,1) in *MDL*(*root*), and their ratios are 2/14, 6/14, 3/14, and 3/14, respectively. Therefore, the value of $Info(D_{root})$ is 1.877.

Traditional information need, however, does not reflect the distance between two *MDA* labels, which may cause bias. For example, let us look at the two datasets shown in Fig. 6. The information needs of the two datasets are identical. In D_1, however, all c_1 labels are the same, while in D_2, the ratio of c_1 labels for 1 and 0 is 50:50. It is clear that D_1 is better than D_2, because D_1 can more easily correctly classify c_1 than D_2. For this reason, the distance within a dataset should be considered for the traditional information need.

To estimate the average distance within a dataset, we first define the distance between two records. Suppose d_1 and d_2 are two records in n_i and $O(n_i) = \{c_1, c_2, \ldots, c_w\}$. $Dis_{n_i}(d_1, d_2)$, the distance between these two records in n_i, is defined as $\frac{\sum_{y=1}^{w}(c_y(d_1)\oplus c_y(d_2))}{w}$, where $c_y(d_1) \oplus c_y(d_2) = 0$ if $c_y(d_1) = c_y(d_2)$, else $c_y(d_1) \oplus c_y(d_2) = 1$, and w is the number of elements in $O(n_i)$.

Consider the sample dataset D_{n_i} shown in Fig. 7. If $O(n_i) = \{c_1, c_2\}$, the distance between d_1 and d_2 is $\frac{(1\oplus1)+(c\oplus d)}{2} = \frac{1}{2}$.

The average distance within a dataset $ADis(D_{n_i})$ is the average distance between any two records in this dataset. Let d_j and d_k be the jth and kth record in D_{n_i}, respectively. Now, we define the function of $ADis(D_{n_i})$ as

$$ADis(D_{n_i}) = \frac{\sum_{j=1}^{|D_{n_i}|}\sum_{k<j}(Dis_{n_i}(d_j, d_k))}{|D_{n_i}| \times (|D_{n_i}| - 1) \times \frac{1}{2}}.$$

Although the above formula is correct, the computation may be time-consuming for large datasets. Since many labels in the training dataset are the same, the number of different labels is usually much smaller than the total number of all records. Therefore, we can rewrite the original definition of $ADis(D_{n_i})$ based on the distance between two *MDA* labels.

The distance between two *MDA* labels $mdl_1(n_i)$ and $mdl_2(n_i)$ is defined as $Dis(mdl_1(n_i), mdl_2(n_i)) = Dis_{n_i}(d_1, d_2)$, where d_1 and d_2 are any two records in node n_i with *MDA* labels $mdl_1(n_i)$ and $mdl_2(n_i)$, respectively. Therefore, the original definition of $ADis(D_{n_i})$ can be rewritten as

$$ADis(D_{n_i}) = \sum_{j=1}^{|MDL(n_i)|}\sum_{k<j}(Dis(mdl_j(n_i), mdl_k(n_i)) \times w(mdl_j(n_i), mdl_k(n_i))), \tag{1}$$

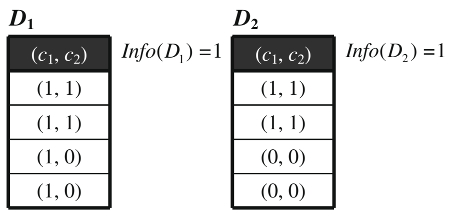
$w(mdl_j(n_i), mdl_k(n_i))$ is the weight of the distance between these two *MDA* labels and $w(mdl_j(n_i), mdl_k(n_i)) = \frac{|D_{n_i}(c_{MDA(n_i)}=mdl_j(n_i))|\times|D_{n_i}(c_{MDA(n_i)}=mdl_k(n_i))|}{|D_{n_i}|\times(|D_{n_i}|-1)\times\frac{1}{2}}$.

Consider the two datasets in Fig. 6. Although their information needs are identical, their average distances are different, $ADis(D_1) = 0.33$ and $ADis(D_2) = 0.67$.

4.3. Multi-dimensional information gain

To classify multiple condition attributes, we propose a multi-dimensional information function, *MDInfo*. *MDInfo* is a modified version of the information need that uses the average distance as an adjusting factor to reflect the distance within a dataset. $MDInfo(D_{n_i})$, the multi-dimensional information function of the dataset D_{n_i}, is defined as

$$MDInfo(D_{n_i}) = ADis(D_{n_i}) \times Info(D_{n_i}). \tag{2}$$



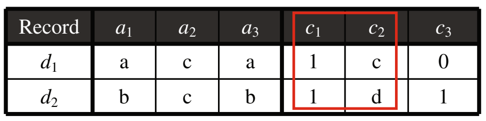
D_1 $Info(D_1) = 1$

(c_1, c_2)
(1, 1)
(1, 1)
(1, 0)
(1, 0)

D_2 $Info(D_2) = 1$

(c_1, c_2)
(1, 1)
(1, 1)
(0, 0)
(0, 0)

Fig. 6. Two sample datasets.



Record	a_1	a_2	a_3	c_1	c_2	c_3
d_1	a	c	a	1	c	0
d_2	b	c	b	1	d	1

Fig. 7. Sample dataset.

Returning to the training dataset in Table 1, we now have $Info(D_{root}) = 1.877$, and we can obtain the average distance within the records in *root* using Eq. (1). There are four different classes (0,0), (0,1), (1,0), and (1,1) in $MDL(root)$, and their numbers are 2, 6, 3, and 3, respectively. Therefore, we obtain $ADis(D_{root}) = 0.51$ and $MDInfo(D_{root}) = 0.96$.

After evaluating the *MDInfo* value of the dataset in a node with Eq. (2), we then split the node and partition the dataset using the decision attribute with the smallest *MDInfo* value. Suppose $V_x = \{v_{x1}, v_{x2}, \ldots\}$, which is the set of all possible values of decision attribute a_x. The dataset D_{n_i} can be partitioned into $|V_x|$ disjoint subsets. The *MDInfo* value of partitioning D_{n_i} with decision attribute a_x can be evaluated by $MDInfo_{a_x}(D_{n_i}) = \sum_{j=1}^{|V_x|} \frac{|D_{n_i}(a_x = v_{xj})|}{|D_{n_i}|} \times MDInfo(D_{n_i}(a_x = v_{xj}))$.

For example, let us partition the dataset in *root* D_{root} with the decision attribute “Age”. Three sub-datasets are formed after D_{root} is partitioned, as shown in Fig. 8. The values of *MDInfo* are 1.15, 0.33, and 0.58, respectively. Therefore, $MDInfo_{\text{“Age”}}(D_{root})$, the *MDInfo* value of partitioning D_{root} with “Age”, is 0.71.

We use a multi-dimensional gain function $MDGain_{a_x}(D_{n_i})$ to evaluate the gain of conducting a test on a_x in n_i, where $MDGain_{a_x}(D_{n_i})$ is defined as $MDGain_{a_x}(D_{n_i}) = MDInfo(D_{n_i}) - MDInfo_{a_x}(D_{n_i})$. Therefore, $MDGain_{\text{“age”}}(D_{root})$, the gain of conducting a test on “Age” in D_{root}, is 0.25.

4.4. Splitting criterion

The splitting criterion for our algorithm is shown in Fig. 9. For each non-leaf node, we select a decision attribute under the cost constraint *maxcost*. Since the total cost is limited, we select the decision attribute with the best gain/cost ratio.

In our example, for the training dataset in Table 1, in order to select a decision attribute to split *root*, we compute the multi-dimensional information gain for each decision attribute. As a result, $MDGain_{\text{“Age”}}(D_{root}) = 0.25$, $MDGain_{\text{“Income”}}(D_{root}) = -0.01$, and $MDGain_{\text{“student”}}(D_{root}) = 0.10$. Since $ArrCost(root)$=0 and the maximum cost threshold $maxcost = 16$, all these attributes satisfy the cost constraint. Next, we compute the gain and cost ratio and get $\frac{MDGain_{\text{“Age”}}(D_{root})}{TestCost(\text{“Age”})+1} \approx 0.04$, $\frac{MDGain_{\text{“Income”}}(D_{root})}{TestCost(\text{“Income”})+1} \approx -0.002$, and $\frac{MDGain_{\text{“Student”}}(D_{root})}{TestCost(\text{“Student”})+1} \approx 0.01$. Thus, “Age” is the most economical decision attribute to split *root* with. For each node n_j splitting from n_i, the initial states of n_j are $L(n_j) = \phi$ and $O(n_j) = O(n_i)$. Fig. 10 shows the temporary decision tree after splitting the *root*, where *root* is the first node n_1 in this decision tree.

We adopt the strategy used by C4.5 for dealing with the problem of numeric data and missing decision attribute values. Accordingly, we will split a numeric decision attribute into two branches by testing all possible split-points. Suppose there are n different values $v_1, v_2, \ldots, v_n$ for a numeric decision attribute, where $v_i > v_j$ for $i > j$. This means that $n - 1$ possible split-points $\frac{v_1+v_2}{2}, \frac{v_2+v_3}{2}, \ldots, \frac{v_{n-1}+v_n}{2}$ will be tested. The point which maximizes the gain/cost ratio is selected as the split-point of the numeric decision attribute. On the other hand, records which contain missing values for the tested attribute are partitioned and distributed into different branches according to the relative frequency of known values.

$D_{root}(\text{"}age\text{"}=\text{"}<=30\text{"})$

d_k	Credit	B_Car
d_1	0	0
d_2	1	0
d_8	0	0
d_9	0	1
d_{11}	1	1

$D_{root}(\text{"}age\text{"}=\text{"}31...40\text{"})$

d_k	Credit	B_Car
d_3	0	1
d_7	1	1
d_{12}	1	1
d_{13}	0	1

$D_{root}(\text{"}age\text{"}=\text{"}>40\text{"})$

d_k	Credit	B_Car
d_4	0	1
d_5	0	1
d_6	1	0
d_{10}	0	1
d_{14}	1	0

Fig. 8. Three datasets produced by partitioning D_{root}.

```
For each non-leaf node $n_i$
    If there exists any decision attribute $a_x$ where
    $ArrCost(n_i) + TestCost(a_x) \le maxcost$ and $MDGain_{a_x}(D_{n_i}) > 0$
        Select an attribute $a_x$' which has the maximum value of
        $MDGain_{a_x'}(D_{n_i}) / (TestCost(a_x') + 1)$ from decision attributes
        which satisfy the total cost constraint to further split node $n_i$.
    Else
        Assign $n_i$ as a leaf, $n_i$ will be labeled with the major labels of records in $n_i$.
```

Fig. 9. Splitting criterion.

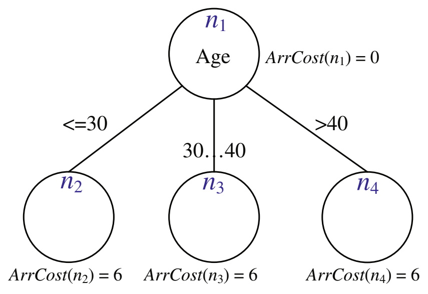


Fig. 10. Temporary decision tree after splitting the *root*.

4.5. Label assignment test

Table 4 shows $D(n_3)$ which is the dataset in node n_3. In $D(n_3)$, all labels of the condition attribute "B_Car" are 1. Thus, the records in n_3 can be classified as "B_Car=1" and the condition attribute "B_Car" can be eliminated from n_3.

Before selecting an attribute to further split a node, we first perform a label assignment test. For the records in node n_i, let $ml(n_i, c_y)$ be the majority label of condition attribute c_y. If the ratio of $ml\,(n_i, c_y)$ exceeds the label assignment threshold θ, $ml\,(n_i, c_y)$ can be assigned and c_y can be eliminated from n_i. That is, $L(n_i) = L(n_i) \cup (c_y, ml(n_i, c_y))$ and $O(n_i) = O(n_i) - c_y$. After the label assignment test, $L(n_3) = \{(\text{"B\_Car"}, 1)\}$ and $O(n_3) = \{\text{"Credit"}\}$.

4.6. Termination condition

A node n_i will stop splitting and be labeled a leaf node when one of the following conditions is satisfied: (1) after label assignment, no condition attributes are left in n_i. In other words, $O(n_i) = \phi$. (2) According to the splitting criterion in Section 4.4, no attribute can be selected for further splitting. (3) The number of records in n_i is less than the terminal node threshold φ. In this case, n_i is labeled with the records' majority labels in the parent node of n_i. Fig. 11 shows the output of the first step obtained using our algorithm. We use a square to represent a leaf node. There are seven leaf nodes and the budget does not allow any beneficial tests.

4.7. Updating the bottom nodes

In the initial decision tree, those nodes whose descendants are all leaves are called *bottom nodes*, such as n_2 and n_3 in Fig. 11. Originally, we used the best "*gain*/*cost*" ratio to pick a splitting attribute. Since no further tests will be conducted after

Table 4
Dataset in node n_3.

d_k	Age	Income	Student	Credit	B_Car
d_3	31···40	High	No	0	1
d_7	31···40	Low	Yes	1	1
d_{12}	31···40	Medium	No	1	1
d_{13}	31···40	High	Yes	0	1

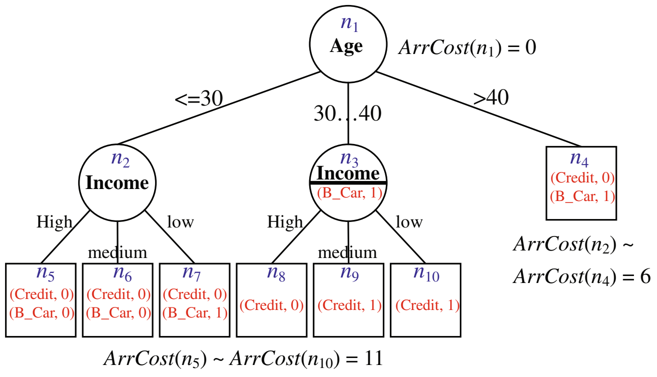


Fig. 11. Output of the first step of our algorithm.

splitting a *bottom node*, however, we should choose an attribute that provides the largest gain. Therefore, we update the *bottom nodes* in the tree built in Step 1 as follows. For each bottom node n_i, select a decision attribute a_x that has the maximum $MDGain_{a_x}(D_{n_i})$ from the decision attributes satisfying $ArrCost(n_i) + TestCost(a_x) \leqslant maxcost$ to further split node n_i. Each child node of n_i, n_j, is a leaf node. Condition attribute c_y, which has not yet been determined in n_j, will be determined according to the majority label of c_y in n_j. The final decision tree is shown in Fig. 12. After this step, we replace "Income" with "Student" in n_2 because $MDGain_{\text{“Student”}}(D_{n_2}) = 0.77$, which is larger than $MDGain_{\text{“Income”}}(D_{n_2}) = 0.55$.

5. Performance evaluation

The performance of our algorithm is evaluated by implementing four different methods, which are listed in Table 5. We evaluate the four methods under different cost constraints and different numbers of condition attributes.

We conducted experiments on five datasets obtained from the UCL Machine Learning repository. These datasets were chosen because they were used in [17] and their cost data was also donated in [17]. The datasets' properties are listed in Table 6. There is only one condition attribute in each of these datasets. Therefore, we randomly selected several decision

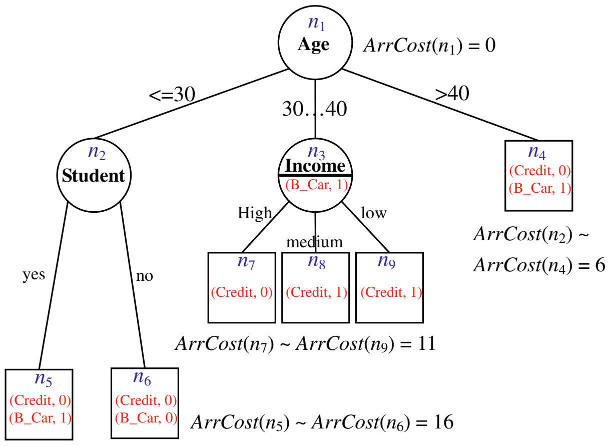


Fig. 12. Final decision tree built by our algorithm.

Table 5
Four experimental methods.

Methods	Description
M1	Sequentially build a decision tree for each condition attribute by C4.5 without a cost constraint
M2	Sequentially build a decision tree for each condition attribute by C4.5 with the budget equally allocated among all condition attributes. When a decision attribute has been tested in the preceding decision tree, the test cost of this attribute will be set to 0
M3	Same as M2 but with the difference that the condition attributes which have been predicted can be incorporated as decision attributes with test cost 0
M4	Use the proposed method to build a cost-constrained decision tree with multiple condition attributes

Table 6
Five datasets used in the experiments.

Dataset	Number of decision attributes	Number of instances	Class distribution
BUPA liver disorders	5	345	Drinks < 3: 169 (48.99%) Drinks ⩾ 3: 176 (51.01%)
Heart disease	13	303	<50% diameter narrowing: 164 (54.13%) >50% diameter narrowing: 139 (45.87%)
Hepatitis prognosis	19	155	Die: 32 (20.65%) Live: 123 (79.35%)
Pima indians diabetes	8	768	Healthy: 500 (65.10%) Diabetes: 268 (34.90%)
Thyroid disease	20	3772	Hypothyroid: 93 (2.47%) Hyperthyroid: 191(5.06%) Normal: 3488 (92.47%)

attributes and assigned them as condition attributes in the experiments. Numerical decision attributes which were selected as condition attributes were discretized into categorical variables. We also removed records which contain missing value in any condition attribute.

We ran a cross validation in each experiment. Datasets were randomly split into ten parts. Ten pairs of the training and test datasets were generated, where the test dataset involved one partition and the training dataset involved the other nine partitions. We built decision trees with the training datasets and tested the trees with the test datasets. The final result from each experiment is the average of the results obtained with the ten pairs of training and test datasets.

We propose two measures, *AvgAcu* and *MDAAcu*, to evaluate the decision trees. *AvgAcu* is the average accuracy of n condition attributes. Let $pc_y\ (d_k,T)$ be the predicted label of the yth condition attribute in decision tree T for d_k, and let $c_y\ (d_k)$ be the true label of the yth condition attribute in d_k. Then

$$AvgAcu(T) = \frac{\sum_{d_k \in \text{test dataset}} \sum_y \mathrm{cor}_y(T, d_k)}{|\text{test dataset}| \times n} \begin{cases} \mathrm{cor}_y(T, d_k) = 1, & pc_y(d_k, T) = c_y(d_k), \\ \mathrm{cor}_y(T, d_k) = 0, & pc_y(d_k, T) \neq c_y(d_k). \end{cases}$$

In many actual applications, however, all condition attributes have to be labeled correctly to make an appropriate decision. For example, even if a bank could accurately evaluate their customer's credit rating, it would be difficult to create a good marketing plan if the customer's likelihood to ask for a loan could not be predicted correctly. Therefore, we propose *MDAAcu*. *MDAAcu* is the ratio of records for which all condition attributes are correctly classified. The equation for *MDAAcu* is

$$MDAAcu(T) = \frac{\sum_{d_k \in \text{test dataset}} \mathrm{cor}_{MDA}(T, d_k)}{|\text{test dataset}|} \begin{cases} \mathrm{cor}_{MDA}(T, d_k) = 1, & \forall y (pc_y(d_k, T) = c_y(d_k)), \\ \mathrm{cor}_{MDA}(T, d_k) = 0, & \exists y (pc_y(d_k, T) \neq c_y(d_k)). \end{cases}$$

5.1. Performance evaluation under different cost constraints

To evaluate our method under different cost constraints, we set a different *maxcost* for each of the experiments. To estimate a reasonable upper boundary for the *maxcost* of a dataset, we first build a decision tree for each condition attribute by C4.5 without a cost constraint. For each condition attribute, let the largest test cost spent to reach a leaf node be called its *tcost*, and let *MaxTotalCost* be the sum of the *tcost* for all condition attributes. We can use this as a reasonable estimation of the upper boundary of the *maxcost* of a dataset. Accordingly, to observe how the scarcity of cost impacts performance, we set the *maxcost* at *MaxTotalCost*, 90% $\times$ *MaxTotalCost*, ..., and 20% $\times$ *MaxTotalCost*, successively.

For experiments on the first four datasets, the label assignment threshold θ is set to 90% and the terminal node threshold φ is set to 3. Because of the skewed class distribution in the Thyroid Disease dataset (92.47% of the records are in the "normal" class), the label assignment threshold θ is set to 95% and the terminal node threshold φ is set to 0 for experiments with this dataset. In addition to the original condition attribute, we randomly select a decision attribute to be another condition attribute. We implement the four methods with these two condition attribute datasets under different cost constraints. The results are shown in Table 7, where the columns show the levels of the cost constraint from 100% $\times$ *MaxTotalCost* to 20% $\times$ *MaxTotalCost*, the rows show the various alternatives, and the cell values are the accuracies expressed as a percentage.

The results for M1 are shown in the first row in Table 7, where the value before the parentheses indicates the accuracy while the value in parentheses indicates the total cost spent, i.e., *MaxTotalCost*. (Note that we first obtain the accuracy by applying C4.5 for each condition attribute without a cost constraint. Then, we accumulate the costs spent in all trees, to obtain the total cost spent to classify multiple condition attributes.) From the results of Table 7, we see that the proposed method M4 outperforms M2 and M3 for all datasets. It can also be seen that our method is very robust with respect to cost constraints even for a small budget. The results achieved with our method are near those obtained with an abundant budget. This proves that our method is very effective at using limited costs to achieve the best possible accuracy. Furthermore, we found that our method surpasses M1 for three datasets (Heart Disease, Hepatitis Prognosis and Pima Indians Diabetes) while achieving comparable accuracy for the other two datasets, but with much less cost. These results indicate that the multidimensional attribute, which our algorithm uses to find the best split-point and select the splitting attributes, has good discrimination ability.

M3, which incorporates the previously predicted condition attributes as decision attributes, is an extended version of M2. However, the performance of M3 is roughly the same as that of M2. This result indicates that a condition attribute classified in a prior decision tree may not be helpful to later ones. The reason may be that the predictions made previously may not always be correct.

Table 8 shows the budget utilization rates by the four methods for the five datasets, where the columns indicate the levels of the cost constraint from 100% $\times$ *MaxTotalCost* to 20% $\times$ *MaxTotalCost*, the rows indicate the various alternatives and the cells show the budget utilization rates expressed as a percentage.

According to the results in Table 8, the budget utilization rates of M2 and M3 are relatively low. The reason for this result may be that allocating the budget in advance is difficult. If too much budget is allocated to a condition attribute which does not need so much, the result may be waste. On the other hand, if too little is allocated to a condition attribute which needs a lot more, performance may be damaged. Since M2 and M3 allocate the budget equally among all condition attributes without considering their differences, there must be some condition attributes that get more than they really need. It is no wonder that the budget utilization rates of M2 and M3 are relatively low. In turn, the low budget utilization rate leads to the bad

Table 7
AvgAcu and *MDAAcu* obtained for each method in datasets: (a) BUPA liver disorders, (b) heart disease, (c) hepatitis prognosis, (d) pima indians diabetes, and (e) thyroid disease.

Dataset	Measurement	Method	*Maxcost* (*form*100% × *MaxTotalCost* to 20% × *MaxTotalCost*)								
			100%	90%	80%	70%	60%	50%	40%	30%	20%
BUPA liver disorders	*AdvAcu*	M1	72.05 (*maxcost*: 31.67)								
		M2	72.05	70.88	70.88	70.88	69.55	69.55	65.29	65.29	65.29
		M3	72.20	70.88	70.88	70.88	69.55	69.55	65.29	65.29	65.29
		M4	73.23	72.20	72.20	70.14	71.47	72.49	72.64	72.64	65.29
	MDAAcu	M1	48.52 (*maxcost*: 31.67)								
		M2	48.52	47.64	47.64	47.64	46.47	46.47	37.94	37.94	37.94
		M3	48.52	47.64	47.64	47.64	46.47	46.47	37.94	37.94	37.94
		M4	50.29	48.52	48.52	44.41	46.76	49.70	50.00	50.00	37.94
Heart disease	*AdvAcu*	M1	63.96 (*maxcost*: 469.70)								
		M2	62.75	62.75	62.06	62.24	62.24	62.75	58.79	58.96	58.96
		M3	62.58	62.58	62.06	62.06	62.06	62.24	57.93	58.44	58.44
		M4	65.68	65.68	65.68	65.68	65.68	65.68	66.03	65.86	65.34
	MDAAcu	M1	43.10 (*maxcost*: 469.70)								
		M2	43.10	43.10	43.79	44.13	44.13	44.82	40.68	39.65	39.65
		M3	44.13	44.13	44.48	44.48	44.48	44.82	42.75	43.10	43.10
		M4	47.24	47.24	47.24	47.24	47.24	47.24	47.93	47.93	45.86
Hepatitis prognosis	*AdvAcu*	M1	66.66 (*maxcost*: 50.38)								
		M2	66.99	66.99	68.33	66.99	66.33	65.99	65.66	62.66	61.66
		M3	68.66	67.99	69.66	67.99	66.33	65.99	65.66	62.66	61.66
		M4	71.99	71.99	71.99	71.99	71.99	71.99	71.99	71.66	70.99
	MDAAcu	M1	41.33 (*maxcost*: 50.38)								
		M2	42.00	40.66	43.33	40.66	42.66	40.66	40.00	34.66	32.66
		M3	44.66	42.66	45.33	42.00	42.66	40.66	40.00	34.66	32.66
		M4	49.33	49.33	49.33	49.33	49.33	49.33	49.33	48.66	47.33
Pima Indians diabetes	*AdvAcu*	M1	74.73 (*maxcost*: 45.39)								
		M2	74.21	75.72	75.98	72.17	72.17	72.17	72.17	72.17	72.17
		M3	74.34	75.32	75.65	72.17	72.17	72.17	72.17	72.17	72.17
		M4	75.85	76.44	76.31	76.31	76.31	76.31	72.82	72.82	72.82
	MDAAcu	M1	57.23 (*maxcost*: 45.39)								
		M2	56.05	58.81	59.47	53.81	53.81	53.81	53.81	53.81	53.81
		M3	56.71	58.02	58.55	53.81	53.81	53.81	53.81	53.81	53.81
		M4	60.13	60.65	60.39	60.39	60.39	60.39	55.39	55.39	55.39
Thyroid disease	*AdvAcu*	M1	83.96 (*maxcost*: 75.11)								
		M2	82.89	81.23	81.23	82.42	82.62	81.16	81.16	81.16	81.16
		M3	82.95	81.23	81.23	82.42	82.62	81.16	81.16	81.16	81.16
		M4	82.62	82.62	83.55	83.28	84.01	84.08	83.02	82.82	81.63
	MDAAcu	M1	68.06 (*maxcost*: 75.11)								
		M2	66.04	63.39	63.39	65.78	66.18	64.72	64.72	64.72	64.72
		M3	66.18	63.39	63.39	65.78	66.18	64.72	64.72	64.72	64.72
		M4	65.51	65.51	67.37	66.84	68.3	68.43	66.84	66.57	65.25

performance; so, as seen in Table 7 the performance of M2 and M3 is inferior to that of M4. To sum up, the advantage of our M4 method lies in the fact that it can fully utilize the budget without waste, which naturally results in a good performance.

5.2. Performance evaluation with different numbers of condition attributes

We now evaluate our method with different numbers of condition attributes. We first randomly drew four decision attributes from each dataset. Then, we put each of them back and treated each as an additional condition attribute. This set of experiments was conducted on only three datasets, Heart Disease, Hepatitis Prognosis, and Thyroid Disease, because the other datasets did not have enough decision attributes. The value of *maxcost* was set at 50% × *MaxTotalCost* and the settings of assignment threshold θ and terminal node threshold φ were the same as in Section 5.1. The results are shown in Table 9, where columns represent the number of condition attributes, the rows give the various alternatives and the cell values show the accuracies expressed as a percentage.

As seen in Table 9, as the number of condition attributes increases, the predicted accuracies decrease. This is true for all three methods. The performance of our method (M4), however, is better performance when the number of condition attributes is greater than one. A comparison of Tables 7 and 9 shows that our method (M4) greatly outperforms the others in terms of *MDAAcu* measurement. This may be because other methods, which necessitate the building of an independent tree for each condition attribute, ignore the interdependence of the condition attributes. For example, let us look at the dataset shown in Fig. 13. The majority labels for C_1, C_2, and C_3 are “1”, “1”, and “1”, respectively. In this dataset, however, the labels for these three condition attributes are never “1” at the same time. As observed from this example, classifying each condition

Table 8
Budget utilization rates obtained from the four methods for datasets: (a) BUPA liver disorders, (b) heart disease, (c) hepatitis prognosis, (d) pima Indians diabetes, and (e) thyroid disease.

Dataset	Method	*maxcost* (*form*100% × *MaxTotalCost* to 20% × *MaxTotalCost*)								
		100%	90%	80%	70%	60%	50%	40%	30%	20%
BUPA liver disorders	M1	100.00								
	M2	77.04	59.48	66.91	76.47	75.73	90.88	0.00	0.00	0.00
	M3	77.04	59.48	66.91	76.47	75.73	90.88	0.00	0.00	0.00
	M4	100.00	84.72	95.31	97.37	89.22	90.88	56.80	75.73	0.00
Heart disease	M1	100.00								
	M2	84.28	93.59	82.04	72.27	84.32	94.58	97.23	21.26	31.88
	M3	86.03	95.53	80.39	71.97	83.96	94.16	94.47	21.26	31.88
	M4	87.79	97.49	86.45	98.81	84.32	94.54	98.87	95.65	97.13
Hepatitis prognosis	M1	100.00								
	M2	82.14	87.97	95.86	98.35	97.42	76.63	81.23	95.03	0.00
	M3	82.14	87.97	95.86	98.35	97.42	76.63	81.23	95.03	0.00
	M4	98.02	98.87	98.24	94.71	97.52	96.24	99.36	100.00	98.04
Pima Indians diabetes	M1	100.00								
	M2	100.00	50.24	54.03	14.29	16.67	20.00	25.00	33.33	50.00
	M3	100.00	50.24	54.03	14.29	16.67	20.00	25.00	33.33	50.00
	M4	100.00	98.64	69.45	79.37	92.60	90.44	25.00	33.33	50.00
Thyroid disease	M1	100.00								
	M2	100.00	74.58	83.91	90.54	71.23	28.53	35.66	47.54	0.00
	M3	100.00	74.58	83.91	90.54	71.23	28.53	35.66	47.54	0.00
	M4	96.01	98.76	94.84	99.46	96.23	98.23	99.44	99.08	96.94

Table 9
AvgAcu and *MDAAcu* obtained from each method for datasets: (a) heart disease, (b) hepatitis prognosis, and (c) thyroid disease.

Dataset	Measurement	Method	Number of condition attributes				
			1	2	3	4	5
Heart disease	*AdvAcu*	M2	76.66	73.66	72.44	66.33	64.19
		M3	76.66	73.50	72.33	66.25	65.53
		M4	74.66	75.83	75.11	68.08	69.26
	MDAAcu	M2	76.66	55.33	39.00	18.66	11.00
		M3	76.66	54.99	39.00	18.66	15.33
		M4	74.66	58.99	46.00	25.33	21.33
Hepatitis prognosis	*AdvAcu*	M2	85.71	68.92	70.00	75.53	70.57
		M3	85.71	68.92	70.00	75.53	70.57
		M4	88.57	78.21	78.80	80.17	69.28
	MDAAcu	M2	85.71	42.85	32.14	34.28	19.28
		M3	85.71	42.85	32.14	34.28	19.28
		M4	88.57	59.28	51.42	45.00	19.28
Thyroid disease	*AdvAcu*	M2	96.81	90.14	93.14	94.46	95.41
		M3	96.81	90.14	93.14	94.46	95.41
		M4	93.76	92.52	94.60	95.55	96.28
	MDAAcu	M2	96.81	80.63	80.23	79.04	78.38
		M3	96.81	80.63	80.23	79.04	78.38
		M4	93.76	86.64	86.07	85.27	84.74

attribute independently may result in a low accuracy when classifying all condition attributes simultaneously. This phenomenon emphasizes the necessity of building a cost-sensitive decision tree to simultaneously classify multiple condition attributes.

According to the results of the experiments as discussed above, it can be seen that our method outperforms the others due to the following reasons:

(1) Multiple condition attributes are integrated into a single multi-dimensional attribute. In this way, all condition attributes are taken into account to compute the multi-dimensional information gain for finding the most efficient test under cost constraints. Therefore, when a numeric decision attribute is considered, we find the best split-point according to the multi-dimensional attribute rather than a single condition attribute. As a result, our method offers better performance than previous approach even when adopting the same strategy for dealing with numeric decision attributes.

C_1	C_2	C_3
1	0	1
1	0	1
1	0	1
1	1	0
1	1	0
0	1	1
0	1	1
0	1	1

Fig. 13. Sample dataset.

(2) There are numerous budget allocation combinations and strategies for the condition attributes. Unfortunately, it is difficult to efficiently allocate the budget in advance among these condition attributes. In view of this difficulty, the proposed method is able to classify all condition attributes within a single decision tree. This approach removes the difficulty of budget allocation, and the experiment results show that our method utilizes the budget efficiently.

6. Conclusions and future work

Cost-sensitive classification has become a popular topic in recent years. Although it has been studied extensively, existing studies have only classified a target record with a single condition attribute. In this study, we propose a new algorithm for building a cost-constrained decision tree with multiple condition attributes. Along with the algorithm, we propose a new splitting criterion, tree structure, and node termination condition. The results of the experiments show that our algorithm performs well under different cost constraints and for different numbers of condition attributes, especially when all condition attributes must be labeled correctly.

In future, we consider the following possible extensions. First, according to the experiment results, the overfitting problem may impair our algorithm's performance. Therefore, it may be necessary to add a tree pruning method to the algorithm as a post-processing phase. Second, we will consider batch testing, where a number of samples are tested at the same time. In this situation, the cost of conducting tests on a batch of samples may differ from the sum of test costs for many single samples. Therefore, we will attempt to propose new algorithms that can build decision trees for batch testing. Finally, decision trees, such as ID3 and C4.5, classify categorical labels, while regression trees, such as CART, CHAID and QUEST, predict continuous values. These two lines of research are closely related but distinct. In this paper, we only focus on decision trees. How to further extend our techniques to regression trees is worth further investigation in the future.

References

[1] F. Berzal, J.C. Cubero, N. Marin, D. Sanchez, Building multi-way decision trees with numerical attributes, Information Sciences 165 (2004) 73–90.
[2] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, 1984.
[3] P. Domingos, MetaCost: a general method for making classifiers cost-sensitive, in: Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, San Diego, CA, ACM Press, 1999, pp. 155–164.
[4] C. Elkan, The Foundations of Cost-Sensitive Learning, in: Proceeding of the Seventeenth International Joint Conference on Artificial Intelligence, 2001, pp. 973–978.
[5] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2006.
[6] M.T. Kai, Inducing cost-sensitive trees via instance weighting, in: Principles of Data Mining and Knowledge Discovery, Second European Symposium, Springer-Verlag, 1998, pp. 23–26.
[7] I. Koprinska, J. Poon, J. Clark, J. Chan, Learning to classify e-mail, Information Sciences 177 (2007) 2167–2187.
[8] C.X. Ling, V.C. Sheng, Q. Yang, Test strategies for cost-sensitive decision trees, IEEE Transactions on Knowledge and Data Engineering 18 (8) (2006) 1055–1067.
[9] C.X. Ling, Q. Yang, J. Wang, S. Zhang, Decision trees with minimal costs, in: Proceedings of the 21st International Conference on Machine Learning, 2004.
[10] W.Y. Loh, Y.S. Shih, Split selection methods for classification trees, Statistica Sinica 7 (4) (1997) 815–840.
[11] S.W. Norton, Generating better decision trees, in: Proceedings International Joint Conference Artificial Intelligence, MI, Morgan Kaufmann, San Mateo, CA, 1989, pp. 800–805.
[12] M. Nunez, The use of background knowledge in decision tree induction, Machine Learning 6 (1991) 231–250.
[13] A. Papagelis, D. Kalles, GATree: Genetically evolved decision trees, in: 12th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'00, 2000, pp. 203–206.
[14] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1986) 81–106.
[15] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.
[16] M. Tan, Cost-sensitive learning of classification knowledge and its application in robotic, Machine Learning Journal 13 (1993) 7–33.
[17] P.D. Turney, Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm, Journal of Artificial Intelligence Research 2 (1995) 369–409.
[18] Q. Yang, C. Ling, X. Chai, R. Pan, Test-cost sensitive classification on data with missing values, IEEE Transactions on Knowledge and Data Engineering 18 (5) (2006) 626–638.
[19] E. Yen, I.-W.M. Chu, Relaxing instance boundaries for the search of splitting points of numerical attributes in classification trees, Information Sciences 177 (2007) 1276–1289.