A Learning Algorithm for Multiple Rule Trees

Jiujiang An, Guoyin Wang, Yu Wu
Institute of Computer Science and Technology
Chongqing University of Posts and Telecommunications
Chongqing, 400065, P. R. China
anjiujiang@tom.com wanggy@cqupt.edu.cn wuyu@cqupt.edu.cn

Abstract

It is one of the key problems for web based decision support systems to generate knowledge from huge database containing inconsistent information. In this paper, a learning algorithm for multiple rule trees (MRT) is developed, which is based on ID3 algorithm and rough set theory. MRT algorithm can quickly generate decision rules from inconsistent decision information tables. Both space and time complexities of MRT algorithm are just polynomial, while those of Skowron’s default decision rule generation algorithm are exponential. With the increasing of the number of records and core attributes of an information table, Skowron’s default algorithm needs more memory and time for generating rules than MRT algorithm. In some cases, Skowron’s default decision rule generation algorithm could not generate rules due to the lack of memory. It’s proved by our simulation experiment results that MRT algorithm is effective and valid.

1. Introduction

Rough set theory has been applied successfully in such fields as machine learning, data mining and etc., since Prof. Z. Pawlak developed it in 1982 [1]. Generating rules from a decision table is one of the major research topics of rough set theory. Reduct is an important contribution of rough set theory for data mining, and its results are in accordance with rules. In uncertain or inconsistent cases, the default decision rule generation algorithm developed by Prof. Skowron (DDRG) can generate all rules, certainty factors of which are greater than a predefined threshold $\alpha_c$ [2]. At the same time, conflicts between these rules are solved by use of some block rules. It is proved that rules generated by this algorithm have high flexibility in processing unseen data. Based on DDRG algorithm, Prof. Wang developed a self-learning model under uncertain condition [3,4]. This model can automatically get the threshold $\alpha_c$ to generate rules without any prior domain knowledge. Simulation experiment results demonstrate that this model could generate a rather smaller number of rules than DDRG algorithm, and has high correct recognition rate for unseen data.

During the rule generation process of DDRG algorithm an inconsistent decision table will be projected onto many subtables by dropping one of its core attribute. Then DDRG generates rules with discernibility matrix from each subtable repeatedly. Thus, the cost for generating rules of this algorithm is determined by the number of records and core attributes of each subtable. It is illustrated in section 4 that its space and time complexities are exponential. When the number of records and core attributes of an inconsistent decision table are small, DDRG algorithm can quickly generate rules from it. However, with the increasing of the number of records and core attributes, it needs much more memory and time for generating rules, and even fails to generate rules due to the lack of memory.

ID3 is a classical machine-learning algorithm for generating rules from decision tables [5]. A typical tree generation process of ID3 algorithm is as follows. Firstly, a condition attribute with the minimal information entropy is chosen from a decision table. Then this attribute is used to divide the decision table into different classes. The above two steps are repeated until each record of the decision table only belongs to one of these classes. Unfortunately, ID3 algorithm can build a decision tree for a consistent decision table only. It cannot process an inconsistent decision table.

To solve the above problems, a learning algorithm for multiple rule trees (MRT) based on ID3 algorithm and rough set theory is proposed in this paper. MRT algorithm can generate multiple rule trees from an inconsistent decision table, and the certainty factor of each rule is greater than or equal to a predefined threshold. In the rule tree generation process, MRT algorithm computes and stores only a part of the original decision table to create an internal node of rule tree. With the predefined threshold, MRT algorithm judges whether an internal node of rule tree has a branch of child node or not. Thus, the depth and width
of rule tree is limited. Its space and time complexities are polynomial. Compared with DDRG algorithm, MRT algorithm generates decision rules from an inconsistent decision table at less memory and time cost. In addition, MRT algorithm generates multiple rule trees to overcome a drawback of ID3 algorithm that rules resulted from it centralize on a few condition attributes. MRT algorithm could be further used to deal with the problem of generating knowledge from huge databases containing inconsistent information for web based intelligent decision support systems.

In Section 2, some basic concepts of rough set theory and rule tree are discussed. MRT algorithm is presented in detail in Section 3. Its space and time complexities are analyzed and compared with DDRG algorithm in Section 4. In Section 5, some simulation experiments are done to test our results. In Section 6, we draw a conclusion for this paper.

2. Basic concepts

For the convenience of later discussion, some related concepts about rough set theory and rule tree are introduced here.

Def. 1 A decision table is defined as S=<U, R, V, f>, where U is a finite set of objects and R=C∪D is a finite set of attributes. C is the condition attribute set and D is the decision attribute set and D is the decision attribute set, V=U∪V, is a union of the domain of each attribute of R. Each attribute has an information function f: U×R→V.

Def. 2 Given a decision table S=<U, R, V, f>, C and D are its condition attribute set and decision attribute set separately. Condition class Ei is defined as Ei∈U/IND(C), where, i=1,⋯,m, and m=|U/IND(C)|. Decision class Xj is defined as Xj∈U/IND(D), where, j=1,⋯,n, and n=|U/IND(D)|.

Def. 3 Given a decision table S=<U, R, V, f>, C is its condition attribute set. A condition class Ei is called consistent if and only if its all objects have the same decision value. Otherwise, it is called inconsistent.

Def. 4 A decision table S is called certain if and only if its condition classes are all consistent. Otherwise, it is an uncertain one.

Def. 5 Given a decision table S=<U, R, V, f>, where R=C∪D, C is its condition attribute set and D={d} is its decision attribute set. The certainty factor of a decision rule A→B, CF(A→B), is defined as CF(A→B)=|X∩Y|/|X|, where X is the object set with condition attribute values satisfying formula A, while Y is the object set with decision attribute satisfying formula B.

Def. 6 Given a decision table S=<U, R, V, f>, PC⊂R is an attribute set, U/IND(P)={Xi, 1≤i≤n}, we define the entropy of P as H(P)=∑∑ p(Xi)log(p(Xi)) , where, p(Xi)=|Xi|/|U|.

Def. 7 The conditional entropy of an attribute set Q⊂R (U/IND(Q)=:{Y1,Y2,⋯,Ym}) with reference to another attribute set P⊂R (U/IND(P)=:{X1,X2,⋯,Xn}) is H(Q|P)=∑|Xj| j=1 n p(Yj|Xj)log(p(Yj|Xj)) , where, p(Yj|Xj)=|Yj∩Xj|/|Xj|, 1≤i≤n, 1≤j≤m.

Def. 8 A rule tree [6] is defined as follows:

(1) A rule tree is composed of one root node, some leaf nodes and some internal nodes.
(2) The root node represents the whole rule set.
(3) Each path from the root node to a leaf node represents a rule.
(4) Each internal node represents an attribute testing. A branch of rule tree represents each attribute class, and a new child node is created from each branch.

Def. 9 A collection consisting of some rule trees is called a rule tree set.

3. MRT algorithm

3.1 MRT algorithm description

The main idea of MRT algorithm is as follows. Firstly, a condition attribute with the minimal condition entropy with reference to the decision classes is chosen from the original decision table. Secondly, the chosen condition attribute is used to divide the original decision table into different classes. Each class is further processed in the following way. If its certainty factor with some decision class is 1, a new child node of current node is created and inserted into the rule tree. If its certainty factor is bigger than or equal to the predefined threshold, a new child node of current node is created and inserted into the internal node queue. Thirdly, the above two steps are repeated until the internal node queue becomes NULL. The whole rule tree is completely built and added into the rule tree set. At last, the corresponding condition attribute of the root node of the previous rule tree is deleted from the original decision table, the above three steps are done until the original decision table is empty. Consequently, the rule tree set is generated.

For the convenience of illustrating MRT algorithm, the class CTreeNode is firstly defined, which means a node of rule tree, contains a decision table and other memberships.

```
Class CTreeNode
{ Table; //A decision table
  NodeID; //Node’s ID
}
```
The other variables needed in describing MRT algorithm are illustrated in Table 1.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>RootNode</td>
<td>The root node of a rule tree</td>
</tr>
<tr>
<td>currentNode</td>
<td>Current node</td>
</tr>
<tr>
<td>ChildNode</td>
<td>The child node of the current node</td>
</tr>
<tr>
<td>InternalNodeQueue</td>
<td>An internal node queue with some internal nodes</td>
</tr>
<tr>
<td>Tree</td>
<td>A rule tree</td>
</tr>
<tr>
<td>ListTree</td>
<td>A rule tree set with some rule trees</td>
</tr>
<tr>
<td>NodeNumber</td>
<td>The sequence number of node in a rule tree.</td>
</tr>
</tbody>
</table>

Then, based on the above definitions, MRT algorithm is given in Algorithm 1.

**Algorithm 1: MRT algorithm**

Input: Original decision table and a predefined threshold

Output: Rule tree set

Step 1: Initialize InternalNodeQueue=NULL, Tree=NULL, ListTree=NULL, NodeNumber=0.

Step 2: If the original decision table is empty, then go to Step 7, else continue.

Step 3: Initialize RootNode and insert it into InternalNodeQueue, where,

- RootNode.Table is set to be the original decision table.
- RootNode.ParentID=-1;
- RootNode.NodeID=0;
- RootNode.IsLeafNode=FALSE.

Step 4: Repeat step 4.1 and 4.2 until InternalNodeQueue is NULL.

Step 4.1: Pop the first node from InternalNodeQueue and set it to be CurrentNode.

Step 4.2: Choose a condition attribute (Ci) with the minimal condition entropy with reference to the decision classes from CurrentNode.Table.

Step 4.3: According to the chosen condition attribute (Ci), divide the decision table of the current node into classes \( \{E_k|E_k @CurrentNode.Table | IND(C_i) \} \), where, \( k=1, 2, \ldots, m \) and \( m=|CurrentNode.Table | IND(C_i) | \). for \( k=1 \) to \( m \)

- If( \( \exists X_j @CurrentNode.Table | IND(D) \{ CF( |E_k \cap X_j|/|E_k|)==1 \} \) ), then create a new child node ChildNode and insert it into Tree, where,
  - ChildNode.Table is set to be NULL for leaf node.
  - ChildNode.IsLeafNode=TRUE// Leaf node.

- If( the predefined threshold \( \leq Max_{X_j @CurrentNode.Table | IND(D) \} \{ CF( |E_k \cap X_j|/|E_k|)\} <1 \) ), create a new child node ChildNode and insert it into InternalNodeQueue, where,
  - ChildNode.Table is set to be a decision table containing those records of \( E_k \) in the current node’s table except the column of the condition attribute \( C_i \).
  - ChildNode.IsLeafNode=FALSE// Internal node.

Step 4.4: Release the decision table of the current node, and insert CurrentNode into Tree.

Step 5: Insert Tree into ListTree, and Tree=NULL, NodeNumber=0.

Step 6: Delete the corresponding condition attribute of the root node of the previous rule tree from the original decision table, and go to Step 2.

Step 7: Output ListTree and stop.

**3.2 An example to illustrate MRT algorithm**

Table 2 is an inconsistent decision table used in several papers [2,3,4]. It contains three condition
attributes, A, B, C, a decision attribute D and 100 records. To illustrate the MRT algorithm more clearly, Table 2 is used as an example to explain the rule generation process of the MRT algorithm here.

Table 2 An inconsistent decision table

<table>
<thead>
<tr>
<th>U</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1(50x)</td>
</tr>
<tr>
<td>E2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2(5x)</td>
</tr>
<tr>
<td>E3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2(30x)</td>
</tr>
<tr>
<td>E4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>2(10x)</td>
</tr>
<tr>
<td>E5</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>3(4x)</td>
</tr>
<tr>
<td>E6</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>4(1x)</td>
</tr>
</tbody>
</table>

(1) From Table 2, choose condition attribute A, whose condition entropy with reference to the decision classes is minimal. According to A, Table 2 is divided into three condition classes U IND(A)= {{E1, E2}, {E3, E4}, {E5, E6}}. According to the decision attribute D, Table 2 is divided into four decision classes U IND(D)= {{E1}, {E2, E3, E4}, {E5}, {E6}}.

(2) Each condition class is processed in different ways according to the step 4.3 of MRT algorithm.

- Because CF({E1, E2} → {E1}) = |E1 ∩ E2|/|E1|=0.91, create a new child node (A=1) and insert it into the rule tree.
- Because CF({E3, E4} → {E2, E3, E4}) =|(E3 ∩ E4) ∩ E2, E3, E4)|/(E3 ∩ E4)|=1, create a new child node (A=2, D=2) and insert it into the rule tree. This new child node is a leaf node.
- Because CF({E5, E6} → {E1}) =|(E5 ∩ E6) ∩ E1|/(E5 ∩ E6)|=0.8, create a new child node (A=3) and insert it into the internal node queue.

The first rule tree gotten at this step is shown in Figure 1.

(3) Pop the first node from the internal node queue and set it to be the current node. Its decision table is shown in Table 3.

Table 3 A part of Table 2

<table>
<thead>
<tr>
<th>U'</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1'</td>
<td>2</td>
<td>3</td>
<td>1(50x)</td>
</tr>
<tr>
<td>E2'</td>
<td>2</td>
<td>1</td>
<td>2(5x)</td>
</tr>
</tbody>
</table>

(5) Delete the corresponding condition attribute (A) of...
the root node of the first rule tree from Table 2, and do step (1), (2), (3) and (4) until Table 2 is NULL. The rule tree set gotten at this step is shown in Figure 4.

![Rule tree set](image)

(6) Generate the following rules from Fig. 4. Note that the parameter behind “ | ” is the certainty factor of each rule.

- R1: A1C3 | D1 | 0.91
- R2: A1C1 | D2 | 0.91
- R3: A2 | D2 | 1
- R4: A3B5C1 | D3 | 0.80
- R5: B5C1 | D3 | 0.80
- R6: B3 | D2 | 1

4. Space and time complexities of MRT and DDRG algorithm

At first, suppose the number of record of a decision table is \( n \), and the number of condition attribute is \( m \). In a rule tree, the maximal number of leaf node is \( n \), the maximal depth of the path from the root node to a leaf node is \( m \). Thus, the maximal number of node is \( mn \).

4.1 Time complexity of MRT algorithm

Suppose the maximal number of condition class is \( b \), therefore, the time for computing the condition entropy of one condition class with reference to the decision classes is \( O(bmn) \), and the time complexity for generating a rule tree is \( O(bmn^2) \). MRT algorithm generates no more than \( m \) rule trees. So, the time complexity of MRT algorithm at worst case is \( O(bmn^2) \).

4.2 Time complexity of DDRG algorithm

If a decision table has only one core attribute, the maximal number of subtables projected from the original decision table is \( 2^m - 1 \). DDRG algorithm also has the same complexity of \( O(bmn^2) \) as MRT algorithm.

4.3 Space complexity of MRT algorithm

Suppose the maximal number of record in a condition class is \( q \), therefore, the memory for storing the internal node is \( O(qm) \), and the memory for a rule tree is \( O(qm^2mn) \). MRT algorithm generates no more than \( m \) rule trees. So, the space complexity of MRT algorithm at worst case is \( O(qmn^2) \).

4.4 Space complexity of DDRG algorithm

If the original decision table has \( m \) core attributes, the maximal number of subtables projected from the original decision table is \( 2^m - 1 \), and the memory for storing each subtable is \( O(mn) \). So, the space complexity of DDRG algorithm at worst case is \( O(2^m mn^2) \).

Thus, we can find that the space and time complexities of MRT algorithm are better than those of DDRG algorithm.

5. Simulation experiments

Our simulation experiments are done on a PC with 2.4GHZ CPU, 512MB memory. DDRG algorithm in RIDAS system is used for the comparison with MRT algorithm of this paper [8].

The simulation experiments have been conducted on 9 data sets from the UCI Machine Learning Repository [9]. At first, 50 percent of each data set is used to generate rules with these two algorithms separately. Then, the other 50 percent of each data set is used to test the recognition rate of rules by these two algorithms. In addition, the objects' combination based simple computation of attribute core algorithm is used to calculate attribute cores of each data set [10].

Simulation experiment results are shown in Table 4. The time cost for generating rules and recognition rate of these rules are compared in it.

Simulation experiment results show that when the number of records and core attributes of an inconsistent decision table are small, such as Experiments 1 and 2, the performance of MRT and DDRG algorithms are almost the same. In Experiments 3 to 9, we can find that MRT algorithm is more effective and valid than DDRG algorithm. Especially, in Experiments 6 to 9, DDRG algorithm fails to generate rules due to the lack of memory when the number of records and core attributes are high. It just proves that the cost for generating rules of DDRG algorithm is greatly affected by the number of records and core attributes of an inconsistent decision table.
6. Conclusion

MRT algorithm can generate multiple rule trees from inconsistent decision tables. On one hand, the certainty factor of each rule is higher than or equal to a predefined threshold. On the other hand, in the rule tree generation process of MRT algorithm, for creating an internal node of rule tree, only a part of an inconsistent decision table needs to be kept in memory and computed. It reduces the depth and width of rule tree with the predefined threshold. Therefore, MRT algorithm can quickly generates rules from an inconsistent decision table. Compared with DDRG algorithm, MRT algorithm needs less memory and time for generating rules. By our simulation experiments, MRT algorithm is proved to be more effective and valid than DDRG algorithm. As future research task, we will study various measures for selecting condition attribute to improve this algorithm, and further use MRT algorithm to solve the problem of generating knowledge from huge databases containing inconsistent information for web based intelligent decision support systems.

Table 4 Simulation experiment results

<table>
<thead>
<tr>
<th>Data set</th>
<th>Number of Record</th>
<th>Number of Attribute</th>
<th>Number of Core attribute</th>
<th>MRT algorithm</th>
<th>DDRG algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Time (S)</td>
<td>Time (S)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Recognition Rate (%)</td>
<td>Recognition Rate (%)</td>
</tr>
<tr>
<td>HAYES_BOTH</td>
<td>102</td>
<td>5</td>
<td>4</td>
<td>0.016</td>
<td>0.625</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>98.07</td>
<td>98.07</td>
</tr>
<tr>
<td>IRIS</td>
<td>150</td>
<td>5</td>
<td>3</td>
<td>0.016</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>98.68</td>
<td>98.68</td>
</tr>
<tr>
<td>LIVER_DISDORNE</td>
<td>1260</td>
<td>7</td>
<td>5</td>
<td>0.422</td>
<td>1431.812</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>99.68</td>
<td>100</td>
</tr>
<tr>
<td>AUSTRALIAN CREDIT APPROVA</td>
<td>345</td>
<td>15</td>
<td>5</td>
<td>0.375</td>
<td>57.437</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>WINE</td>
<td>148</td>
<td>14</td>
<td>3</td>
<td>0.250</td>
<td>0.359</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>POST_OPERATIVE</td>
<td>90</td>
<td>9</td>
<td>8</td>
<td>0.051</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>*</td>
</tr>
<tr>
<td>ZOO</td>
<td>71</td>
<td>16</td>
<td>8</td>
<td>0.045</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>100</td>
<td>*</td>
</tr>
<tr>
<td>BACT_T</td>
<td>6000</td>
<td>155</td>
<td>20</td>
<td>1009.813</td>
<td>44.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>44.13</td>
<td>44.13</td>
</tr>
<tr>
<td>LETTER_RECOGNITION</td>
<td>20000</td>
<td>17</td>
<td>15</td>
<td>84.469</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>92.62</td>
<td>*</td>
</tr>
</tbody>
</table>

Note: “*” means that DDRG algorithm cannot generate rules from data set due to the lack of memory.

Acknowledgements

This paper is supported by National Natural Science Foundation of P.R. China (No.60373111), PD Program of P.R. China, Application Science Foundation of Chongqing, and Science & Technology Research Program of the Municipal Education Committee of Chongqing of China.

References