An algorithm for induction of decision rules

Carine Halmenschlager¹ and Luis Otavio Campos Alvares²

¹ Universidade de Santa Cruz do Sul, Departamento de Informática, 
Av. Independência, 2293, 96815-900, Santa Cruz do Sul, Brasil 
carineh@inf.ufrgs.br

² Universidade Federal do Rio Grande do Sul, Instituto de Informática, 
Av. Bento Gonçalves, 9500, C.P. 15064, Porto Alegre, Brasil 
alvares@inf.ufrgs.br

Abstract. Several Data Mining techniques are used to extract implicit patterns out of databases, which constitute the useful knowledge of the stored data. Among them, decision rules are quite popular for their simplicity, interpretability and modularity. However, most of the algorithms for induction of decision rules partition recursively the data until a certain stop condition is satisfied. On the other hand, bases of real data for application in Knowledge Discovery are very large, involving millions of records, being necessary a way of quickly generalizing such data. Therefore, this paper presents a new model for the induction of decision rules, in that the main distinction of the proposed algorithm is the single pass performed in data during the induction process. This paper also presents the results of comparing the prototype with See5, using both repository data and real data.

1 Introduction

Every year, organizations accumulate thousands of records in their databases, what increases significantly the amount of data and the richness of its information. As a result of this effective increase, processing such information became more complex and difficult. Usually, data is kept stored in databases without being used in a truly efficient way [2].

Knowledge Discovery in Databases (KDD) appeared with the need for more powerful and automated methods of information recovery and use. KDD reveals the strategic information hiding in large databases, by the application of Data Mining techniques, the evaluation of the obtained patterns and the interpretation of the results [7].

Data Mining techniques operate on great amounts of data extracting implicit information and patterns in it, which cannot be discovered using conventional techniques of inference in databases [7]. Among the existent Data Mining techniques, one may mention Decision Trees, Decision Rules and Neural Networks as the most well-known.
Decision trees and rules are considered symbolic methods because they represent what is learned from the given attributes through expressions [8]. Neural networks, in turn, are connectionist methods, in that the learning consists of adjusting weights. Due to the high degree of legibility of symbolic methods, decision trees and rules are quite popular and used techniques.

Decision rules are structures of the type "If <condition> then <conclusion>.", where <condition> is a conjunction of feature tests – attributes and their values. The <conclusion> is a class of the dataset.

Decision rules are quite popular for their simplicity, modularity and interpretability, and may be generated directly from a set of data, using a rule generation algorithm, in a process called induction. They may also be generated from a decision tree, translating the concepts of this one in a more comprehensible and equivalent form, in a process known as derivation [3].

This way, decision rules solve a main disadvantage of decision trees: the difficulty of being understandable when applied on large databases, because the rules can be understood without needing to reference other rules. However, the main disadvantage of decision rules is that they don’t work very well with continuous attributes [1], besides being able to find many associations that are just noises. Such deficiencies can be resolved, respectively, with the addition of specific discretization techniques and of a minimum degree of support.

Support and confidence degrees are measures commonly used by algorithms to evaluate the quality of a rule [5]. Support indicates the percentage of cases covered by a certain rule and confidence indicates the success of the rule:

\[
\text{Support} = \frac{n_2}{n},
\]

\[
\text{Confidence} = \frac{n_2}{n_1}.
\]

Let us denote \( n \) as the total number of instances, \( n_1 \) as the number of instances that satisfy the conditions and \( n_2 \) as the number of instances that satisfy both the conditions and the class.

Besides, rules may also be evaluated for accuracy, comprehensibility, learning speed, storage requirements, compactness and any other desirable property that determines how good and appropriate they are for the task [9].

Algorithms for induction of decision rules are relatively simple. Most of these algorithms recursively partition the data into smaller subsets in an iterative way. Each iteration searches all the candidates’ conditions until a stop criterion is found. Such tests involve multiple passes in the dataset, making processing slower [14].

On the other hand, sets of real data for application in Knowledge Discovery may be very large, involving thousands or even millions of records. To carry out KDD in such large data requires the development of new techniques that limit secondary memory access, thus minimizing execution time in the induction process.
In that context, the main goal of this paper is to present a new model for the induction of decision rules whose main distinction is to perform a single pass in data. It is also presented the developed prototype and its evaluation with both repository and real data.

This work is organized as follows. Section 2 describes the proposed model and the developed prototype. Section 3 presents the evaluation of the prototype in both kinds of datasets, using the See5 tool for comparisons. Finally, concluding remarks are given in Section 4.

2 Proposed Model

This section presents the proposed model for induction of decision rules, whose main idea is to reduce the number of passes on the dataset. For bases of large and real data, it can involve a considerable reduction of processing time.

2.1 Induction Algorithm

The algorithm, that was proposed and detailed in Halmenschlager [10] [11], induces a list of rules using one of the attributes of the training set as the categorical attribute, i.e. as the conclusion of the rule. The algorithm relies on an evaluation function to choose the condition of the rule. Such function uses a table of occurrences, which consists in the main distinction of the proposed model.

The proposed model differs from most algorithms for induction of decision rules, like the C4.5rules [12] [4] and the Prism [6], which create the list of rules at the same time they partition the data. The proposed model passes through the data a single time, storing the attribute relation occurrences two by two in a table (Table 1). This generated table is used by the evaluation function to verify the best condition, without the need for partitioning or for new passes in the datasets. Thus, the best condition is given by the attribute-value pair with highest percentage of occurrences (probability).

The rules of each class are induced separately, in agreement with the values of the categorical attribute. So, the process begins inducing all the rules for a single class, followed by the induction of the other categories, until a list of the rules of all classes is formed.

For each rule, the evaluation function verifies the probability of all attribute-value pairs and chooses the best condition. The condition that has the highest probability given by the table of occurrences, will be added in the list of rules with the structure: "If <condition>". If such condition is considered a strong rule, in other words, has a probability in the table of occurrences for the class higher than the previously defined threshold, the rule will be finished by adding its conclusion with the structure "then <class>". Following, will be searched for the next attribute-value pair with highest probability.
Otherwise, if it isn’t a strong rule, the algorithm will continue searching the table of occurrences, looking for new conditions that will be added to the current condition, forming a more complex rule and, consequently, with more attribute-value pairs.

Table 1. Example of the occurrence table. For each value of attribute in each instance of the data set, an occurrence is added in the columns of the other attribute values in agreement with the class, in the respective line of the table of occurrences

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Weather</th>
<th>Humidity</th>
<th>Wind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Rainy</td>
<td>Sunny</td>
<td>Cloudy</td>
</tr>
<tr>
<td>Class</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>W. Rainy</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>W. Sunny</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>W. Cloudy</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>H. Low</td>
<td>0 1 2 0</td>
<td>0 0 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>H. High</td>
<td>3 1 0 3</td>
<td>3 3 0 3</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>W. No</td>
<td>3 0 1 2</td>
<td>2 2 0 2</td>
<td>2 0 4 2</td>
</tr>
<tr>
<td>W. Yes</td>
<td>0 2 1 2</td>
<td>0 1 1 2</td>
<td>1 1 1 2</td>
</tr>
</tbody>
</table>

Every time the algorithm concludes a new rule, the number of cases it covers is verified in the table of occurrences, without the need for new passes or queries in the datasets. Whenever the sum of the number of cases covered by the rules of a class is the same as the total number of cases of this class, in the training dataset, the algorithm stops the induction for this class, beginning the search for rules of the next class, and so on, until all the classes have their rules.

Core of the proposed algorithm for the induction of rules

Function InductRules (A: non-categorical attributes set, C: categorical attribute, S: training set) : RuleList

Begin
Create OccurrenceTable (OT)
RuleList := {}; For each class C_i in OT do
Cases := 0;
Repeat
ClassRule := `If’;
Repeat
BestCondition := EvaluationFunction (OT);
ClassRule := ClassRule + `BestCondition’;
Until EvaluationFunction (BestCondition) ? ;
ClassRule := ClassRule + `Then class = C_i’;
RuleList := RuleList + ClassRule;
Cases := Cases + Support(ClassRule);
Until Cases = Support(C_i);
End for;
End.
Initially, RuleList is empty. For each one of the values of the categorical attribute it is verified, by the evaluation function, the BestCondition, that is the attribute-value pair that has the largest probability in the table of occurrences. The rule of the $C_i$ class receives this condition and seeks for other conditions that will be also included in this rule, until a certain condition has its probability higher than or equal to the threshold $\theta$. When this happens, the current class is attributed to ClassRule. The list of rules receives this formed rule and the number of cases supported by the rule is verified in the table of occurrences.

At the end of the rule list induction, the support and confidence degrees of each rule are verified, to identify the most important and valuable rules. Those that don’t have a minimal support and confidence degree are considered weak rules. Such weak rules may be removed from the list of rules to simplify it. Also the double rules are eliminated, the ones that have same conditions, but in a different order. So, the final list of rules can be evaluated and used to verify the main associations among the attributes.

2.2 Midas Prototype

Based on the algorithm described above, a prototype, named Midas - Mining Data, was implemented using the Borland Delphi language (Fig. 1). After starting Midas, the dataset to be used in the mining should be informed. Next, Midas exhibits in a drop-down list the attributes of the training data. The selected attribute will be taken as the categorical attribute and, consequently, will be in the conclusion of the rules.
Fig. 1. Main interface of Midas. The results of mining can be seen through a list of rules. The support and confidence degrees obtained in each rule are presented at its end, in the format [support / confidence], in percentage.

Midas also allows the induction of decision trees. In this case, the decision rules are derived from the tree. To evaluate the error rate of the final model, Midas allows the use of the estimations by \textit{resubstitution}, in which the training dataset is used again to evaluate the final rule list. It also allows the use of an independent dataset for the evaluation, in which a set, apart from the test set, is used; and use of \textit{cross-validation}, in that several subsets of the data are used to generate and to evaluate the list of rules.

3 Evaluation

In this section results of the evaluation and comparison of Midas with See5\textsuperscript{1} [13] are presented, using datasets from the UCI Machine Learning Repository [15], besides real data from the health domain. Comparison among the algorithms is centered in the simplicity of the model, in processing time and in the error rate, using all the records of the database as training set.

In spite of the fact that the C4.5 algorithm is specific for induction of decision trees, the See5 tool also accomplishes the induction of rules, based on a variation of the same algorithm, known as C4.5rules [4], in that the rules are derived from the tree and, after that, generalized, ignoring superfluous conditions and/or rules.

The UCI datasets (Table 2) were chosen for they have a varied number of instances, attributes and classes. They also have discrete attributes and are automatic learning datasets, besides being of public domain.

Table 2. Characteristics of the repository datasets used in the tests

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Instances</th>
<th>Attributes</th>
<th>Values</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Balance</td>
<td>625</td>
<td>4</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>Breast Cancer</td>
<td>699</td>
<td>9</td>
<td>90</td>
<td>2</td>
</tr>
<tr>
<td>Car</td>
<td>1,728</td>
<td>6</td>
<td>21</td>
<td>4</td>
</tr>
<tr>
<td>Chess</td>
<td>28,056</td>
<td>6</td>
<td>40</td>
<td>17</td>
</tr>
<tr>
<td>Genetics</td>
<td>3,190</td>
<td>60</td>
<td>287</td>
<td>3</td>
</tr>
<tr>
<td>Mushrooms</td>
<td>8,124</td>
<td>22</td>
<td>117</td>
<td>2</td>
</tr>
<tr>
<td>Nursery</td>
<td>12,960</td>
<td>8</td>
<td>27</td>
<td>5</td>
</tr>
<tr>
<td>Voting</td>
<td>435</td>
<td>16</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>Zoo</td>
<td>101</td>
<td>16</td>
<td>36</td>
<td>7</td>
</tr>
</tbody>
</table>

By analyzing Table 3, it is noticed that the induction of decision rules by Midas almost always generates a smaller number of rules than See5. The Breast Cancer,

\textsuperscript{1} Data Mining tool that performs rules induction using the C4.5rules algorithm, a variation of C4.5 [9], one of most known and used in the Machine Learning community.
Car, Mushrooms, Voting, Balance, Chess and Nursery bases had the number of rules inferior in Midas. However, the Zoo base and the Genetics base both had a higher number of rules in the proposed model. This shows that, in general, the prototype generalizes well the bases of data and obtains simple models.

Table 3. Number of inducted rules for datasets of the UCI repository

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SEE5</td>
<td>15</td>
<td>6</td>
<td>77</td>
<td>2,450</td>
<td>45</td>
<td>9</td>
<td>155</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>Midas</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>47</td>
<td>53</td>
<td>2</td>
<td>13</td>
<td>2</td>
<td>16</td>
</tr>
</tbody>
</table>

The arithmetic average obtained by Midas was 16 rules, against 308 rules of See5. However, the Chess base distorted a little the results, once it had a value much above the average. Thus, by ignoring it to make the average of the induced rules again, it’s found an average of 40 rules induced by See5 and 12 induced by Midas, showing that, even so, Midas generates simpler models.

According to Halmenschlager [11], Midas has a cost of processing influenced mainly by the number of attributes, because its complexity is $O(m! \cdot n)$, where $m$ is the number of attributes and $n$ is the number of records, while the learning time in See5 is more influenced by the number of records in the base, for its complexity, according to [4], is $O(m \cdot n^2)$.

Thus, the Chess, Nursery, Genetics and Mushrooms bases were the ones that needed a longer time of induction (Table 4) in See5, since they have the largest number of records. In Midas, the Genetics base was the slowest one, followed by the Mushrooms, Nursery and Chess bases.

Table 4. Time of learning in each dataset, in seconds

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SEE5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>298</td>
<td>3</td>
<td>1</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Midas</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>24</td>
<td>91</td>
<td>34</td>
<td>32</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

It is interesting to observe that the Zoo base, in the induction in Midas, had an induction time over zero seconds, even having only 101 records. That happened because this base has a relatively high dimensionality (16 attributes).

Although Midas haven’t had a better rule induction time in some datasets, the main distinction and gain of the prototype are related to the bases with greater number of records, as it can be observed in the Chess base. In this dataset, See5 took 298 seconds for the induction of rules, while Midas took only 24 seconds.

Table 5 presents the support and confidence degree of the rules directly induced by Midas, in percentage. Those averages show that, despite the low number of rules and the high error rate apparently generated, the inducted rules have a quite high confidence degree. In the Car and Mushrooms bases, for example, the confidence degree reached 100%. So we can conclude that the induced rules for each base have
maximum confidence, in other words, all the instances covered by a rule are of the same class defined in the conclusion of that rule.

Table 5. Support and confidence average of the rules directly induced by Midas, in percentage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Support</td>
<td>14.48</td>
<td>30.32</td>
<td>33.33</td>
<td>0.50</td>
<td>1.00</td>
<td>16.84</td>
<td>6.98</td>
<td>48.74</td>
<td>6.62</td>
</tr>
<tr>
<td>Confidence</td>
<td>74.40</td>
<td>99.72</td>
<td>100.00</td>
<td>57.29</td>
<td>99.53</td>
<td>100.00</td>
<td>85.57</td>
<td>97.28</td>
<td>95.64</td>
</tr>
</tbody>
</table>

However, once the support degree didn’t reach 100% of the cases in some datasets, it is noticed that some instances of these bases were not covered by any rule. Thus, Midas was evaluated in two ways: (1) with the default class as the last rule in the list of rules and (2) not considering the default class and, consequently, not taking for errors the instances not covered by any rule (Table 6).

Table 6. Error rates found, using estimation by resubstitution, in percentage

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SEE5</td>
<td>22.10</td>
<td>3.10</td>
<td>3.90</td>
<td>29.50</td>
<td>4.00</td>
<td>0.24</td>
<td>1.40</td>
<td>2.80</td>
<td>1.00</td>
</tr>
<tr>
<td>Midas1</td>
<td>15.68</td>
<td>24.61</td>
<td>29.98</td>
<td>83.44</td>
<td>37.81</td>
<td>21.61</td>
<td>33.52</td>
<td>4.37</td>
<td>8.91</td>
</tr>
<tr>
<td>Midas2</td>
<td>8.00</td>
<td>0.14</td>
<td>0</td>
<td>2.95</td>
<td>0</td>
<td>2.41</td>
<td>3.68</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In Table 6, it is noticed that Midas has an average of apparent error rate higher than See5. From that, it can be concluded that the proposed algorithm is generalizing much the training set, not learning in detail the patterns present in the data. In other words, Midas isn’t overfitting the data, because it is generating few rules, but leading to an apparent error rate higher than See5.

On the other hand, when the default rule isn’t considered, those instances should not be taken for errors, but for non-classifications, since no rule was found for the instance, and so it was not classified. This is possible and perfectly appropriate in cases in which the objective of the induced model is to create a generic view of data, in order to obtain some new knowledge, but not in cases in which its objective is strictly to classify new instances.

In such case, the error rate decreases a lot in the induction of rules by Midas, because most of the instances covered by the rules are correct. However, as the error rate of Midas excludes the instances not classified by rules, while in See5 it was impossible not to consider such instances in the error rate calculation, it was difficult to do more detailed and precise comparison and evaluation.

Midas was also applied in databases from the Health Department of the State of Rio Grande do Sul (Secretaria da Saúde do Estado do Rio Grande do Sul), in Brazil, to verify the proposed model’s behavior in the generalization of real databases. These databases were chosen because they were easily accessible, they would make a greater approach with reality possible and the fact that it’s a domain rich in information.
The database chosen for the mining was the one containing AIHs – A Authorizations for Hospital Internment (Autorizações de Internação Hospitalar). The AIHs store the internments occurred in hospitals of Rio Grande do Sul through the Unified Health System (Sistema Único de Saúde). Each internment contains patient, hospital and doctor identification data. They also contain the respective diagnosis, the requested procedures, the procedures taken, the special procedures and the costs and materials needed. More details can be found in [11].

In a first mining, data of two months of internments in the year 2000 were selected, in a total of 96.145 records and 8 attributes. Midas took 55 seconds to find 217 rules, while See5 needed 93 seconds to the induction.

Also the data of the whole year 2000 was selected, in a total of 565.625 records and 5 attributes. In this case, Midas inducted 164 rules, taking 11 minutes and 39 seconds – See5 needed 28 minutes and 18 seconds to induct 423 rules.

These results confirm the simplicity of the model and the great modularity of Midas’ results, and also the agile processing, very important in the decision-making process. Midas inducted the rules very quickly, and generalized large databases with few rules, what facilitates the comprehension of the discovered knowledge.

4 Conclusions

Today, sets of real data for application in Data Mining are very large, containing an enormous amount of records (up to millions). To execute the induction of knowledge in such large data requests the development of new techniques that generalize these data and that limit secondary memory access, minimizing execution time.

Thus, the aim of this paper was to present a new model for induction of decision rules, the developed prototype and its evaluation with repository and real data, with the objective of quickly generalizing such datasets. The main idea was to reduce the number of passes on the datasets what, for real and large datasets, can involve a considerable reduction of processing time.

Midas generalizes data very well, leading to the discovery of few and simple rules, so the interpretation of the results becomes quite easy. However, this great generalization of data has been taking to a high error rate. But if the support and confidence degrees are considered – which are more important measures in rules than the error rate itself – it is verified that this error rate is related with instances that were not classified, and not with instances that were classified erroneously.

Yet, the prototype doesn’t accomplish treatment for unknown or continuous attributes. It always adds a condition for each value of the attribute, what is acceptable for discrete domains, but not for continuous ones. That way, all the continuous attributes of the training set should be firstly pre-processed.

On the other hand, the proposed model passes only once on the dataset. It doesn’t partition recursively the data in subsets (as it happens in See5). This is an interesting aspect concerning the processing time for the induction of the rules. In Midas, when the dataset has many attributes, the high dimensionality (measured by those values)
increases processing time, while in See5, processing time will be increased by the
great amount of records in the dataset.

This was proven by the complexities of See5 – $O(m.n^2)$ – and Midas – $O(n.m!)$, in
that the first has a computational cost given by the number of records, while in
Midas the computational cost is given by the number of attributes. That way, Midas
presents better results than See5 in databases that contain a larger amount of records
and a low dimensionality.

Thus, Midas is more appropriate and perfectly applied on real datasets, because
real datasets have exactly these characteristics: few attributes and thousands or
millions of records.

References

2. Ávila, B.C.: Data Mining. In: Anais da VI Escola Regional de Informática da SBC,
   Blumenau, SBC (1998) 87-106
3. Baranauskas, J.A. and Monard, M.C.: Reviewing some Machine Learning concepts and
5. Baranauskas, J.A. and Monard, M.C.: Extração automática de conhecimento por múltiplos
   induores. Tese, USP, São Carlos (2001)
   Association for Artificial Intelligence, California (1996)
   (1999)
    Decisão baseado em agentes. In: Anais da XXVII Conferencia Latinoamericana de
    Informatica, Mérida (2001)
11. Halmenschlager, C.: Um algoritmo de indução de árvores e regras de decisão. Dissertação,
    Conference on Data Mining, Rio de Janeiro (1998)