Multi-relational Decision Tree Induction

Arno J. Knobbe[1,2], Arno Siebes[2], Daniël van der Wallen[1]

[1]Syllogic B.V., Hoefseweg 1, 3821 AE, Amersfoort, The Netherlands,
{a.knobbe, d.van.der.wallen}@syllogic.com
[2]CWI, P.O. Box 94079, 1090 GB, Amsterdam, The Netherlands
arno@cwi.nl

Abstract. Discovering decision trees is an important set of techniques in KDD, both because of their simple interpretation and the efficiency of their discovery. One disadvantage is that they do not take the structure of the data into account. By going from the standard single-relation approach to the multi-relational approach as in ILP this disadvantage is removed. However, the straightforward generalisation loses the efficiency. In this paper we present a framework that allows for efficient discovery of multi-relational decision trees through exploitation of domain knowledge encoded in the data model of the database.

1 Introduction

The induction of decision trees has been getting a lot of attention in the field of Knowledge Discovery in Databases over the past few years. This popularity has been largely due to the efficiency with which decision trees can be induced from large datasets, as well as to the elegant and intuitive representation of the knowledge that is discovered. However, traditional decision tree approaches have one major drawback. Because of their propositional nature, they can not be employed to analyse relational databases containing multiple tables. Such databases can be used to describe objects with some internal structure, which may differ from one object to another. For example, when analysing chemical compounds, we would like to make statements about the occurrence of particular subgroups with certain features. The means to describe groups of such objects in terms of occurrence of a certain substructure are simply not available in propositional (attribute-value) decision trees.

In this paper, we present an alternative approach that does provide the means to induce decision trees from structural information. We call such decision trees multi-relational decision trees, in line with a previously proposed multi-relational data mining framework [4, 5]. In order to be able to induce decision trees from a large relational database efficiently, we need a framework with the following characteristics:

1. Both attribute-value and structural information are included in the analysis.
2. The search space is drastically pruned by using the integrity constraints that are available in the data model. This means that we are considering only the structural information that is intended by the design of the database, and we are not wasting time on potentially large numbers of conceptually invalid patterns.
3. The concepts of negation and complementary sets of objects are representable.

J.M. Żytkow and J. Rauch (Eds.): PKDD'99, LNAI 1704, pp. 378–383, 1999.

Decision trees recursively divide the data set up into complementary sets of objects. It is necessary that both the positive split, as well as the complement of that, can effectively be represented.

4. Efficiency is achieved by a collection of data mining primitives that can be used to summarise both attribute-value and structural information.

5. The framework can be implemented by a dedicated client/server architecture. This requires that a clear separation between search process and data processing can be made. This enables the data processing part (usually the main computational bottleneck) to be implemented on a scalable server.

Only two of these requirements are met by existing algorithms for inducing first order logical decision trees, as described in [1, 2]. Specifically the items 1. and 3. are addressed by this approach, but little attention has been given to efficient implementations. The concepts addressed in item 3. are partially solved by representing the whole decision tree as a decision list in Prolog that depends heavily on the order of clauses and the use of cuts. By doing so, the problem of representing individual patterns associated with internal nodes or leafs of the tree is circumvented.

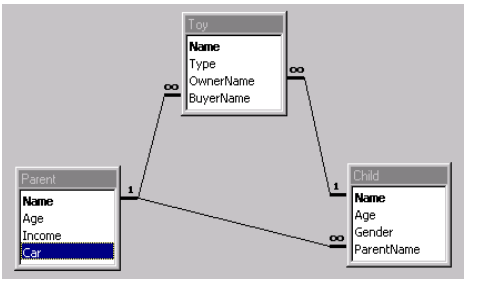
Decision trees typically divide some set of objects into two complementary subsets, in a recursive manner [8]. In propositional trees, this division is made by applying a simple propositional condition, and its complement, to the current set of objects. In an attribute-value environment complementary patterns are produced by simply negating the additional condition. In a multi-relational environment, producing such a complement is less trivial. Our previous work on a multi-relational data mining framework, described in [4, 5], does cover four of the five characteristics that were mentioned, but does not address the problem of handling negation and complementary sets of objects (item 3.). An extended framework that does allow the induction of multi-relational decision trees will be considered in more detail in this paper. We introduce a graphical language of selection graphs with associated refinement operators, which provides the necessary representational power. Selection graphs are graphical representations of the subsets of objects that are associated with individual nodes and leafs of the tree.

2 Multi-relational Data Mining

We will assume that the data to be analysed is stored in a relational database. A relational database consists of a set of tables and a set of associations (i.e. constraints) between pairs of tables describing how records in one table relate to records in another table. An association between two tables describes the relationships between records in both tables. The nature of this relationship is characterised by the *multiplicity* of the association. The multiplicity of an association determines whether several records in one table relate to single or multiple records in the second table. Also, the multiplicity determines whether every record in one table needs to have at least one corresponding record in the second table.

Example 1 The figure below shows an example of a data model that describes parents, children and toys, as well as how each of these relate to each other. The data model shows that parents may have zero or more children, children may have zero or more toys, and parents may have bought zero or more toys. Note that toys owned by a

particular child may not necessarily have been bought by their parents. They can be presents from other parents. Also note that children have one parent (for simplicity).



Even though the data model consists of multiple tables, there is still only a single kind of objects that is central to the analysis. You can choose the kind of objects you want to analyse, by selecting one of the tables as the *target table*. Each record in the target table, which we will refer to as t_0, will now correspond to a single object in the database. Any information pertaining to the object that is stored in other tables can be looked up by following the associations in the data model. If the data mining algorithm requires a particular feature of an object to be used as a dependent attribute for classification or regression, we can define a particular *target attribute* within the target table.

The idea of mining from multiple tables is not a new one. It is being studied extensively in the field of Inductive Logic Programming (ILP) [3]. Conceptually, there are many parallels between ILP and multi-relational data mining. However, ILP approaches are mostly based on data stored as Prolog programs, and little attention is given to data stored in relational database and to how knowledge of the data model can help to guide the search process [7, 10]. Nor has a lot of attention been given to efficiency and scalability issues. Multi-relational data mining differs from ILP in three aspects. Firstly, it is restricted to the discovery of non-recursive patterns. Secondly, the semantic information in the database is exploited explicitly. Thirdly, the emphasis on database primitives ensures efficiency.

In our search for knowledge in relational databases we want to consider not only attribute-value descriptions, as is common in traditional algorithms, but also the structural information which is available through the associations between tables. We will refer to descriptions of certain features of multi-relational objects as multi-relational patterns. We can look at multi-relational patterns as small pieces of substructure which we wish to encounter in the structure of the objects we are considering. As was explained in [4, 5], we view multi-relational data mining as the search for interesting multi-relational patterns. The multi-relational data mining framework allows many different top-down search algorithms, each of which are multi-relational generalisations of well-known attribute-value search algorithms. Each of these top-down approaches share the idea of a refinement operator. Whenever a promising pattern is discovered, a list of refinements will be examined. When we speak about refinement of a multi-relational pattern, we are referring to an extension of the actual description of the pattern, which results in a new selection of objects which is a subset of the selection associated with the original multi-relational pattern. Recursively applying such refinement operators to promising patterns results in a top-down algorithm which zooms in on interesting subsets of the database.

Taking into account the above discussion of multi-relational data mining and top-down approaches, we can formulate the following requirements for a multi-relational pattern language. In the following section we will define a language which satisfies these requirements. Descriptions of multi-relational patterns should

- reflect the structure of the relational model. This allows for easier

understanding of, and enforcing referential constraints on the pattern.

- be intuitive, especially where complementary expressions are considered.
- support atomic, local refinements.
- allow refinements which are complementary. If there is a refinement to a multi-relational pattern which produces a certain subset, there should also be a complementary refinement which produces the complementary subset.

3 Selection Graphs

In order to describe the constraints related to a multi-relational pattern, we introduce the concept of *selection graphs*:

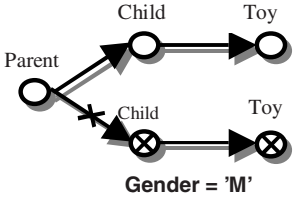
Definition (selection graph) A *selection graph G* is a directed graph (*N*, *E*), where *N* is a set of triples (*t*, *C, s*), *t* is a table in the data model and *C* is a, possibly empty, set of conditions on attributes in *t* of type *t.a operator c*; the *operator* is one of the usual selection operators, =, > etc. *s* is a flag with possible values *open* and *closed.*

E is a set of tuples (*p*, *q*, *a, e*) called *selection edges*, where *p* and *q* are selection nodes and *a* is an association between *p.t* and *q.t* in the data model. *e* is a flag with possible values *present* and *absent*. The selection graph contains at least one node n_0 that corresponds to the target table t_0.

Selection graphs can be represented graphically as labelled directed graphs. The value of *s* is indicated by the absence or presence of a cross in the node, representing the value *open* and *closed* respectively. The value of *e* is indicated by the absence or presence of a cross on the arrow, representing the value *present* and *absent* respectively.

A *present* edge combined with a list of conditions selects groups of records for which there is at least one record that respects the list of conditions. An *absent* edge combined with a list of conditions selects only those groups for which there is not a single record that respects the list of conditions. The selection associated with any subgraph is the combined result of all such individual constraints within the subgraph on groups of records. This means that any subgraph that is pointed to by an *absent* edge should be considered as a joint set of negative conditions. The flag *s* associated with nodes of the selection graph has no effect on the selection. Rather, it is used to indicate whether a particular node in a selection graph is a candidate for refinement. Selection graphs can easily be translated to SQL [5].

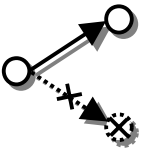
Example 2 The following selection graph and derived SQL statement selects those parents that have at least one child with a toy, but for whom none of such children are male.




```
select distinct T0.Name
from Parent T0, Child T1, Toy T2
where T0.Name = T1.ParentName
      and T1.Name = T2.OwnerName
      and T0.Name not in
         (select T3.ParentName
          from Child T3, Toy T4
          where T3.Name = T4.OwnerName
                and T3.Gender = 'M')
```

Refinements As was described earlier, the selection graphs will be used to represent sets of objects belonging to nodes or leafs in a decision tree. Whenever a new split is introduced in the decision tree, we are in fact refining the current selection graph in two ways. We will be using the following refinement operators of a selection graph G as potential splits in the multi-relational decision tree. The refinements are introduced in pairs of complimentary operations:

- add positive condition. This refinement will simply add a condition to a selection node in G without actually changing the structure of G.
- add negative condition. In case the node which is refined does not represents the target table, this refinement will introduce a new *absent* edge from the parent of the selection node in question. The condition list of the selection node will be copied to the new *closed* node, and will be extended by the new condition. If the node which is refined does represent the target table, the condition is simply negated and added to the current list of conditions for this node.



- add *present* edge and *open* node. This refinement will instantiate an associations in the data model as a *present* edge together with its corresponding table and add these to G.

- add *absent* edge and *closed* node. This refinement will instantiate an associations in the data model as an *absent* edge together with its corresponding table and add these to G.

4 Multi-relational Decision Trees

The induction of decision trees in first order logic has been studied by several researchers [1, 2, 6, 9]. Each of these approaches share a common Divide and Conquer strategy, but produce different flavours of decision trees. For example [6] discusses the induction of regression trees, whereas [1] discusses the induction of decision trees for classification. In [2] an overview is given of potential uses of decision trees within a single framework. However, these papers have largely focused on induction-parameters such as the choice of splitting criterion or stopping criterion. None of these papers provide a good solution for the representation of patterns associated with the leaves and internal nodes of the decision tree. In this section we give a generic algorithm for the top-down induction of multi-relational decision trees within the multi-relational data mining framework. It illustrates the use of selection graphs, and specifically the use of complementary selection graphs in the two

```
build_tree(T : tree, D : database, P : pattern)
        R := optimal_refinement(P, D)
        if stopping_criterion(R)
                T := leaf(P)
        else
                P_left := R(P)
                P_right := R_compl(P)
                build_tree(left, D, P_left)
                build_tree(right, D, P_right)
                T := node(left, right, R)
```

branches of a split.

Top-down induction of decision trees is basically a Divide and Conquer algorithm. The algorithm starts with a single node at the root of the tree which represents the set of all objects in the relational database. By analysing all possible refinements of the empty selection graph, and examining their quality by applying some interestingness measure, we determine the optimal refinement. This optimal refinement, together with its complement, is used to create the patterns associated with the left and the right branch respectively. Based on the stopping criterion it may turn out that the optimal refinement and its complement do not give cause for further splitting, a leaf node is introduced instead. Whenever the optimal refinement does provide a good split, a left and right branch is introduced and the procedure is applied to each of these recursively. For an extensive demonstration of how this works in practice, we refer to [5].

5 Conclusion

In this paper we have presented a framework that allows the efficient discovery of multi-relational decision trees. The main advantage above the standard decision tree algorithms is the gain in expressiveness. The main advantage above the ILP approach towards decision trees is the gain in efficiency achieved by exploiting the domain knowledge present in the data model of the database. One of the main remaining challenges is to extend this framework such that the selection graphs may contain cycles. Such an extension would, e.g., allow refining into parents that have not bought toys for their own children.

References

1. Blockeel, H., De Raedt, L. *Top-down induction of first order logical decision trees*, Artificial Intelligence 101 (1-2):285-297, June 1998
2. Blockeel, H., De Raedt, L., Ramon, J. *Top-down induction of clustering trees,* In Proceedings of ICML'98, 55-63, 1998
3. Dzeroski, S. *Inductive Logic Programming and Knowledge Discovery in Databases,* Advances in Knowledge Discovery and Data Mining, AAAI Press, 1996
4. Knobbe, A.J., Blockeel, H., Siebes, A., Van der Wallen, D.M.G. *Multi-Relational Data Mining,* technical report CWI, 1999, `http://www.cwi.nl`
5. Knobbe, A.J., Siebes, A., Van der Wallen, D.M.G. *Multi-Relational Decision Tree Induction,* technical report CWI, 1999, `http://www.cwi.nl`
6. Kramer, S. *Structural regression trees,* In Proceedings of AAAI'96, 1996
7. Morik, K., Brockhausen, P., *A Multistrategy Approach to Relational Knowledge Discovery in Databases,* in Machine Learning 27(3), 287-312, Kluwer, 1997
8. Quinlan, R.J., *C4.5: Programs for Machine Learning,* Morgan Kaufmann, 1993
9. Watanabe, L., Rendell, L. *Learning structural decision trees from examples,* In Proceedings of IJCAI'91, 770-776, 1991
10. Wrobel, S. *An algorithm for multi-relational discovery of subgroups,* In Proceedings of Principles of Data Mining and Knowledge Discovery (PKDD'97), 78-87, 1997