9 Logical Languages for Data Mining

Fosca Giannotti\(^1\), Giuseppe Manco\(^2\), and Jef Wijsen\(^3\)

\(^1\) ISTI, Institute of CNR  
   Via Moruzzi 1, 56124 Pisa, Italy  
\(^2\) ICAR, Institute of CNR  
   Via Bucci 41c, 87036 Rende (CS), Italy  
\(^3\) University of Mons-Hainaut, Institut d'Informatique  
   Avenue du Champ de Mars 6, B-7000 Mons, Belgium

Abstract. Data mining focuses on the development of methods and algorithms for such tasks as classification, clustering, rule induction, and discovery of associations. In the database field, the view of data mining as advanced querying has recently stimulated much research into the development of data mining query languages. In the field of machine learning, inductive logic programming has broadened its scope toward extending standard data mining tasks from the usual attribute-value setting to a multirelational setting. After a concise description of data mining, the contribution of logic to both fields is discussed. At the end, we indicate the potential use of logic for unifying different existing data mining formalisms.

9.1 Introduction

Data mining uses induction to infer understandable and useful knowledge (rules, patterns, regularities) from large data sets. As such knowledge inference is beyond the expressive power of classical query languages, stronger query paradigms are needed. Logic can contribute in several respects to fulfilling this need. From a database perspective, logic can be used as a unifying formalism for integrating input data sources with data mining tasks and discovered knowledge. From a machine learning perspective, the main challenge lies in upgrading existing “propositional” data mining techniques to first-order logic. We briefly look at each perspective in turn.

Database Perspective. The success of SQL relies mainly on a small number of primitives sufficient to support a large majority of database applications. Unfortunately, these primitives are not sufficiently expressive to support the emerging field of knowledge discovery in databases (KDD). Imielinski and Mannila [42] compare the current situation in KDD to the situation in database management systems in the early 1960s, when each application had to be built from scratch, without the benefit of dedicated database primitives. They note that today’s data mining techniques would more appropriately be described as “file mining” because of the loose coupling between data mining engines and databases. To remedy this situation, they propose to combine

J. Chomicki et al. (eds.), Logics for Emerging Applications of Databases  
© Springer-Verlag Berlin Heidelberg 2004
relational query languages with data mining primitives in an overall framework capable of specifying data mining problems as complex queries involving KDD objects (rules, clustering, classifiers, or simply tuples). In this way, the mined KDD objects become available for further querying. The principle that query answers can be queried further is typically referred to as closure and is an essential feature of SQL. KDD queries can thus generate new knowledge or retrieve previously generated knowledge. This allows for interactive data mining sessions, where users cross boundaries between mining and querying. Query optimization and execution techniques in such a query system will typically rely on advanced data mining algorithms.

These appealing ideas of closure and crossing boundaries in a KDD query system are often summarized by the term inductive database. Today, it is still an open question how to realize inductive databases. The route taken in this chapter is to use logic as a unifying framework to “glue” input data with KDD objects and queries.

**Machine Learning Perspective.** Most classical data mining techniques can work only on a single table and can discover only rules where all atoms are of the form “attribute=constant.” Such techniques are commonly called propositional because the problems solved can be expressed in propositional logic. It turns out, however, that many real-life data mining problems contain multiple tables and ask for first-order rules containing variables. One approach is to transform these problems first into a propositional format and then tackle them by propositional techniques. Nevertheless, “propositionalization” is often undesirable, because it may lose information or result in an awkward problem specification.

Rather than downgrading the problem, it is more attractive to upgrade existing techniques so that they can directly tackle first-order problems. As a consequence, there is currently a growing interest in upgrading propositional data mining tools, like classifiers, to first-order logic. The induction of first-order rules from examples is at the center of a subdomain of machine learning called inductive logic programming (ILP).

The chapter is organized as follows. Section 9.2 gives a general overview of the tasks and challenges in data mining. Section 9.3 takes a database perspective on the use of logic as a unifying framework for integrating data sources with data mining tasks and discovered knowledge. Section 9.4 explains the original foundations of ILP, and Sect. 9.5 indicates how ILP has recently broadened its scope from induction of logic programs to standard data mining tasks. Section 9.6 outlines a possible approach toward the integration of several concepts presented in this chapter. Finally, Sect. 9.7 contains concluding remarks. Throughout the chapter, we will use the eaters database of Fig. 9.1 as a running example. The three tables of this database capture their intuitive semantics.