1

Introduction

The field of logical and relational learning, which is introduced in this chapter, is motivated by the limitations of traditional symbolic machine learning and data mining systems that largely work with propositional representations. These limitations are clarified using three case studies: predicting the activity of chemical compounds from their structure; link mining, where properties of websites are discovered; and learning a simple natural language interface for a database. After sketching how logical and relational learning works, the chapter ends with a short sketch of the history of this subfield of machine learning and data mining as well as a brief overview of the rest of this book.

1.1 What Is Logical and Relational Learning?

Artificial intelligence has many different subfields. Logical and relational learning combines principles and ideas of two of the most important subfields of artificial intelligence: machine learning and knowledge representation. Machine learning is the study of systems that improve their behavior over time with experience. In many cases, especially in a data mining context, the experience consists of a set of observations or examples in which one searches for patterns, regularities or classification rules that provide valuable new insights into the data and that should ideally be readily interpretable by the user. The learning process then typically involves a search through various generalizations of the examples.

In the past fifty years, a wide variety of machine learning techniques have been developed (see Mitchell [1997] or Langley [1996] for an overview). However, most of the early techniques were severely limited from a knowledge representation perspective. Indeed, most of the early techniques (such as decision trees [Quinlan, 1986], Bayesian networks [Pearl, 1988], perceptrons [Nilsson, 1990], or association rules [Agrawal et al., 1993]) could only handle data and generalizations in a limited representation language, which was essentially

propositional. Propositional representations (based on boolean or propositional logic) cannot elegantly represent domains involving multiple entities as well as the relationships amongst them. One such domain is that of social networks where the persons are the entities and their social interactions are characterized by their relationships. The representational limitations of the early machine learning systems carried over to the resulting learning and data mining techniques, which were in turn severely limited in their application domain. Various machine learning researchers, such as Ryszard Michalski [1983] and Gordon Plotkin [1970], soon realized these limitations and started to employ more expressive knowledge representation frameworks for learning. Research focused on using frameworks that were able to represent a variable number of entities as well as the relationships that hold amongst them. Such representations are called *relational*. When they are grounded in or derived from first-order logic they are called *logical* representations. The interest in learning using these expressive representation formalisms soon resulted in the emergence of a new subfield of artificial intelligence that I now describe as logical and relational learning.

Logical and relational learning is thus viewed in this book as the study of machine learning and data mining within expressive knowledge representation formalisms encompassing relational or first-order logic. It specifically targets learning problems involving multiple entities and the relationships amongst them. Throughout the book we shall mostly be using logic as a representation language for describing data and generalizations, because logic is inherently relational, it is expressive, understandable, and interpretable, and it is well understood. It provides solid theoretical foundations for many developments within artificial intelligence and knowledge representation. At the same time, it enables one to specify and employ background knowledge about the domain, which is often also a key factor determining success in many applications of artificial intelligence.

1.2 Why Is Logical and Relational Learning Important?

To answer this question, let us look at three important applications of logical and relational learning. The first is concerned with learning to classify a set of compounds as active or inactive [Srinivasan et al., 1996], the second with analyzing a website [Craven and Slattery, 2001], and the third with learning a simple natural language interface to a database system [Mooney, 2000]. In these applications there are typically a variable number of entities as well as relationships amongst them. This makes it very hard, if not impossible, to use more traditional machine learning methods that work with fixed feature vectors or attribute-value representations. Using relational or logical representations, these problems can be alleviated, as we will show throughout the rest of this book.

1.2.1 Structure Activity Relationship Prediction

Consider the compounds shown in Fig. 1.1. Two of the molecules are active and two are inactive. The learning task now is to find a pattern that discriminates the actives from the inactives. This type of task is an important task in computational chemistry. It is often called *structure activity relationship prediction* (SAR), and it forms an essential step in understanding various processes related to drug design and discovery [Srinivasan and King, 1999b], toxicology [Helma, 2005], and so on. The figure also shows a so-called *structural alert*, which allows one to distinguish the actives from the inactives because the structural alert is a substructure (subgraph) that matches both of the actives but none of the inactives. At the same time, the structural alert is readily interpretable and provides useful insights into the factors determining the activity.

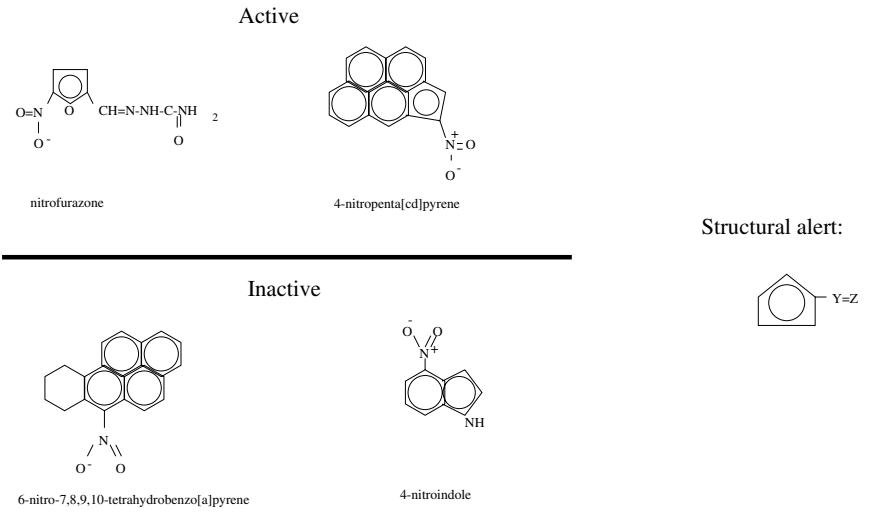

Fig. 1.1. Predicting mutagenicity. Reprinted from [Srinivasan et al., 1996], page 288, ©1996, with permission from Elsevier

Traditional machine learning methods employ the *single-tuple single-table assumption*, which assumes that the data can be represented using attribute-value pairs. Within this representation, each example (or compound) corresponds to a single row or tuple in a table, and each feature or attribute to a single column; cf. Table 1.1. For each example and attribute, the cells of the

table specify the value of the attribute for the specific example, for instance, whether or not a benzene ring is present. We shall call representations that can easily be mapped into the single-tuple single-table format *propositional*.

Table 1.1. A table-based representation

Compound	Attribute1	Attribute2	Attribute3	Class
1	true	false	true	active
2	true	true	true	active
3	false	false	true	inactive
4	true	false	true	inactive

From a user perspective, the difficulty with this type of representation is the mismatch between the graphical and structured two-dimensional representation in the molecules and the flat representation in the table. In order to use the flat representation, the user must first determine the features or attributes of interest. In structure activity relationship prediction, these are sometimes called *fingerprints*. Finding these features is in itself a non-trivial task as there exist a vast number of potentially interesting features. Furthermore, the result of the learning process will critically depend on the quality of the employed features. Even though there exist a number of specialized tools to tackle this kind of task (involving the use of libraries of fingerprints), the question arises as to whether there exist general-purpose machine learning systems able to cope with such structured representations directly. The answer to this question is affirmative, as logical and relational learners directly deal with structured data.

Example 1.1. The graphical structure of one of the compounds can be represented by means of the following tuples, which we call *facts*:

$$\mathsf{active}(\mathsf{f1}) \leftarrow$$
$$\mathsf{logmutag}(\mathsf{f1}, 0.64) \leftarrow$$
$$\mathsf{lumo}(\mathsf{f1}, -1.785) \leftarrow$$
$$\mathsf{logp}(\mathsf{f1}, 1.01) \leftarrow$$
$$\mathsf{atom}(\mathsf{f1}, \mathsf{f1}_1, \mathsf{c}, 21, 0.187) \leftarrow$$
$$\mathsf{atom}(\mathsf{f1}, \mathsf{f1}_2, \mathsf{c}, 21, -0.143) \leftarrow$$
$$\mathsf{atom}(\mathsf{f1}, \mathsf{f1}_3, \mathsf{c}, 21, -0.143) \leftarrow$$
$$\mathsf{atom}(\mathsf{f1}, \mathsf{f1}_4, \mathsf{c}, 21, -0.013) \leftarrow$$
$$\mathsf{atom}(\mathsf{f1}, \mathsf{f1}_5, \mathsf{o}, 52, -0.043) \leftarrow$$
$$\ldots$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_1, \mathsf{f1}_2, 7) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_2, \mathsf{f1}_3, 7) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_3, \mathsf{f1}_4, 7) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_4, \mathsf{f1}_5, 7) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_8, \mathsf{f1}_9, 2) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_8, \mathsf{f1}_{10}, 2) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_1, \mathsf{f1}_{11}, 1) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_{11}, \mathsf{f1}_{12}, 2) \leftarrow$$
$$\mathsf{bond}(\mathsf{f1}, \mathsf{f1}_{11}, \mathsf{f1}_{13}, 1) \leftarrow$$

In this encoding, each entity is given a name and the relationships among the entities are captured. For instance, in the above example, the compound is named $\mathsf{f1}$ and its atoms $\mathsf{f1}_1, \mathsf{f1}_2, \ldots$. Furthermore, the relation $\mathsf{atom}/5$ of arity 5 states properties of the atoms: the molecule they occur in (e.g., $\mathsf{f1}$),

the element (e.g., c denoting a carbon) and the type (e.g., 21) as well as the charge (e.g., 0.187). The relationships amongst the atoms are then captured by the relation bond/3, which represents the bindings amongst the atoms. Finally, there are also overall properties or attributes of the molecule, such as their *logp* and *lumo* values. Further properties of the compounds could be mentioned, such as the functional groups or ring structures they contain:

$$\begin{array}{l}\mathsf{ring\_size\_5}(\mathsf{f1},[\mathsf{f1}_5,\mathsf{f1}_1,\mathsf{f1}_2,\mathsf{f1}_3,\mathsf{f1}_4]) \leftarrow \\ \mathsf{hetero\_aromatic\_5\_ring}(\mathsf{f1},[\mathsf{f1}_5,\mathsf{f1}_1,\mathsf{f1}_2,\mathsf{f1}_3,\mathsf{f1}_4]) \leftarrow \\ \ldots\end{array}$$

The first tuple states that there is a ring of size 5 in the compound f1 that involves the atoms $\mathsf{f1}_5, \mathsf{f1}_1, \mathsf{f1}_2, \mathsf{f1}_3$ and $\mathsf{f1}_4$ in molecule f1; the second one states that this is a heteroaromatic ring.

Using this representation it is possible to describe the structural alert in the form of a rule

$$\mathsf{active}(\mathsf{M}) \leftarrow \mathsf{ring\_size\_5}(\mathsf{M},\mathsf{R}),\mathsf{element}(\mathsf{A1},\mathsf{R}),\mathsf{bond}(\mathsf{M},\mathsf{A1},\mathsf{A2},2)$$

which actually reads as[1]:

> Molecule M is active IF it contains a ring of size 5 called R and atoms A1 and A2 that are connected by a double (2) bond such that A1 also belongs to the ring R.

The previous example illustrates the use of logical representations for data mining. It is actually based on the well-known mutagenicity application of relational learning due to Srinivasan et al. [1996], where the structural alert was discovered using the inductive logic programming system Progol [Muggleton, 1995] and the representation employed above. The importance of this type of application is clear when considering that the results were published in the scientific literature in the application domain [King and Srinivasan, 1996], that they were obtained using a general-purpose inductive logic programming algorithm and were transparent to the experts in the domain. The combination of these factors has seldom been achieved in artificial intelligence.

1.2.2 A Web Mining Example

While structure activity relationship prediction involves the mining of a (potentially large) set of small graphs, *link mining* and discovery is concerned with the analysis of a single large graph or network [Getoor, 2003, Getoor and Dielh, 2005]. To illustrate link mining, we consider the best known example of a network, that is, the Internet, even though link mining is applicable in

[1] This form of description is sometimes called 'Sternberg' English in inductive logic programming, after the computational biologist Michael Sternberg, who has been involved in several pioneering scientific applications of logical learning.

other contexts as well, for instance, in social networks, protein networks, and bibliographic databases. The following example is inspired by the influential WebKB example of Craven and Slattery [2001].

Example 1.2. Consider the website of a typical university. It contains several web pages that each describe a wide variety of entities. These entities belong to a wide variety of classes, such as student, faculty, staff, department, course, project, and publication. Let us assume that for each such web page, there is a corresponding tuple in our relational database. Consider, for instance, the following facts (where the urls denote particular URLs):

faculty(url1, stephen) ←
course(url3, logic_for_learning) ←
department(url5, computer_science) ←
faculty(url2, john) ←
project(url4, april2) ←
student(url6, hiroaki) ←
...

In addition, there are relationships among these entities. For instance, various pages that refer to one another, such as the link from url6 to url1, denote a particular relationship, in this case the relationship between the student hiroaki and his adviser stephen. This can again be modeled as facts in a relational database:

adviser(stephen, hiroaki) ←
teaches(john, logic_for_learning) ←
belongsTo(stephen, computer_science) ←
follows(hiroaki, logic_for_learning) ←
...

Again, the structure of the problem can elegantly be represented using a relational database. This representation can easily be extended with additional background knowledge in the form of rules:

$$\mathsf{studentOf}(\mathsf{Lect}, \mathsf{Stud}) \leftarrow \mathsf{teaches}(\mathsf{Lect}, \mathsf{Course}), \mathsf{follows}(\mathsf{Stud}, \mathsf{Course})$$

which expresses that

> Stud is a studentOf Lect IF Lect teaches a Course and Stud follows the Course.

Further information could be contained in the data set, such as extracts of the text appearing on the different links or pages. There are several interesting link mining tasks in this domain. It is for instance possible to learn to predict the classes of web pages or the nature of the relationships encoded by links between web pages; cf. [Getoor, 2003, Craven and Slattery, 2001, Chakrabarti, 2002, Baldi et al., 2003]. To address such tasks using logical or relational learning, one has to start from a set of examples of relations that are known to hold. For instance, the fact that stephen is the adviser of hiroaki is a (positive) example stating that the link from hiroaki to stephen belongs to the relation adviser. If for a given university website, say the University of Freiburg, all

hyperlinks would be labeled (by hand) with the corresponding relationships, one could then learn general rules using logical and relational learning that would allow one to predict the labels of unseen hyperlinks. These rules could then be applied to determine the labels of the hyperlinks at another university website, say that of the University of Leuven. An example of a rule that might be discovered in this context is

$$\begin{aligned}\mathsf{adviser}(\mathsf{Prof},\mathsf{Stud}) \leftarrow\ & \\ & \mathsf{webpage}(\mathsf{Stud},\mathsf{Url}),\mathsf{student}(\mathsf{Url}),\\ & \mathsf{contains}(\mathsf{Url},\mathsf{advisor}),\mathsf{contains}(\mathsf{Url},\mathsf{Prof})\end{aligned}$$

which expresses that

> Prof is an adviser of Stud IF Stud has a webpage with Url of type student that contains the words adviser and Prof.

To tackle such problems, one often combines logical and relational learning with probabilistic models. This topic will be introduced in Chapter 8.

Link mining problems in general, and the above example in particular, cannot easily be represented using the single-tuple single-table assumption (as in Table 1.1) without losing information.

Exercise 1.3. Try to represent the link mining example within the single-tuple single-table assumption and identify the problems with this approach.

Exercise 1.4. Sketch other application domains that cannot be modeled under this assumption.

1.2.3 A Language Learning Example

A third illustration of an application domain that requires dealing with knowledge as well as structured data is natural language processing. Empirical natural language processing is now a major trend within the computational linguistics community [Manning and Schütze, 1999], and several logical and relational learning scientists have contributed interesting techniques and applications; cf. [Cussens and Džeroski, 2000, Mooney, 2000]. As an illustration of this line of work, let us look at one of the applications of Raymond Mooney's group, which pioneered the use of logical and relational learning techniques for language learning. More specifically, we sketch how to learn to parse database queries in natural language, closely following Zelle and Mooney [1996].[2] The induced semantic parser is the central component of a question-answering system.

Example 1.5. Assume you are given a relational database containing information about geography. The database contains information about various basic

[2] For ease of exposition, we use a slightly simplified notation.

entities such as countries, cities, states, rivers and places. In addition, it contains facts about the relationships among them, for instance, capital(C, Y), which specifies that C is the capital of Y, loc(X, Y), which states that X is located in Y, nextTo(X, Y), which states that X is located next to Y, and many other relationships.

The task could then be to translate queries formulated in natural language to database queries that can be executed by the underlying database system. For instance, the query in natural language

What are the major cities in Kansas?

could be translated to the database query

$$\mathsf{answer}(\mathsf{C}, (\mathsf{major}(\mathsf{C}), \mathsf{city}(\mathsf{C}), \mathsf{loc}(\mathsf{C}, \mathsf{S}), \mathsf{equal}(\mathsf{S}, \mathsf{kansas})))$$

This last query can then be passed on to the database system and executed. The database system then generates all entities C that are major, a city, and located in kansas.

Zelle and Mooney's learning system [1996] starts from examples, which consist of queries in natural language and in database format, from an elementary shift-reduce parser, and from some background knowledge about the domain. The task is then to learn control knowledge for the parser. Essentially, the parser has to learn the conditions under which to apply the different operators in the shift-reduce parser. The control knowledge is represented using a set of clauses (IF rules) as in the previous two case studies. The control rules need to take into account knowledge about the stack used by the parser, the structure of the database, and the semantics of the language. This is again hard (if not impossible) to represent under the single-tuple single-table assumption.

1.3 How Does Relational and Logical Learning Work?

Symbolic machine learning and data mining techniques essentially search a space of possible patterns, models or regularities. Depending on the task, different search algorithms and principles apply. For instance, consider the structure activity relationship prediction task and assume that one is searching for *all* structural alerts that occur in at least 20% of the actives and at most 2% of the inactives. In this case, a complete search strategy is applicable. On the other hand, if one is looking for a structural alert that separates the actives from the inactives and could be used for classification, a heuristic search method such as hill climbing is more appropriate.

Data mining is often viewed as the process of computing the set of patterns $Th(Q, D, \mathcal{L})$ [Mannila and Toivonen, 1997], which can be defined as follows (cf. also Chapter 3). The search space consists of all patterns expressible within a language of patterns $\mathcal{L}$. For logical and relational learning this will typically

be a set of rules or clauses of the type we encountered in the case studies of the previous section; the data set D consists of the examples that need to be generalized; and, finally, the constraint Q specifies which patterns are of interest. The constraint typically depends on the data mining task tackled, for instance, finding a single structural alert in a classification setting, or finding all alerts satisfying particular frequency thresholds. So, the set $Th(Q, D, \mathcal{L})$ can be defined as the set of all patterns $h \in \mathcal{L}$ that satisfy the constraint $Q(h, D)$ with respect to the data set D.

A slightly different perspective is given by the machine learning view, which is often formulated as that of finding a particular function h (again belonging to a language of possible functions $\mathcal{L}$) that minimizes a *loss* function $l(h, D)$ on the data. Using this view, the natural language application of the previous subsection can be modeled more easily, as the goal is to learn a function mapping statements in natural language to database queries. An adequate loss function is the accuracy of the function, that is, the fraction of database queries that is correctly predicted. The machine learning and data mining views can be reconciled, for instance, by requiring that the constraint $Q(h, D)$ succeeds only when $l(h, D)$ is minimal; cf. Chapter 3.

Central in the definition of the constraint $Q(h, D)$ or the loss function $l(h, D)$ is the *covers* relation between the data and the rules. It specifies when a rule covers an example, or, equivalently, when an example satisfies a particular rule. There are various possible ways to represent examples and rules and these result in different possible choices for the covers relation (cf. Chapter 4). The most popular choice is that of *learning from entailment*. It is also the setting employed in the case studies above.

Example 1.6. To illustrate the notion of coverage, let us reconsider Ex. 1.1 and let us also simplify it a bit. An example could now be represented by the rule:

$$\begin{aligned}\mathsf{active(m1)} \leftarrow\ & \\ & \mathsf{atom(m1, m11, c)}, \ldots, \mathsf{atom(m1, m1n, c)}, \\ & \mathsf{bond(m1, m11, m12, 2)}, \ldots, \mathsf{bond(m1, m11, m13, 1)}, \\ & \mathsf{ring\_size\_5(m1, [m15, m11, m12, m13, m14])}, \ldots\end{aligned}$$

Consider now the rule

$$\mathsf{active(M)} \leftarrow \mathsf{ring\_size\_5(M, R)}, \mathsf{atom(M, M1, c)}$$

which actually states that

> Molecule M is active IF it contains a ring of size 5 called R and an atom M1 that is a carbon (c).

The rule covers the example because the conditions in the rule are satisfied by the example when setting $\mathsf{M} = \mathsf{m1}$, $\mathsf{R} = [\mathsf{m15}, \mathsf{m11}, \mathsf{m12}, \mathsf{m13}, \mathsf{m14}]$ and $\mathsf{M1} = \mathsf{m11}$. The reader familiar with logic (see also Chapter 2) will recognize that the example e is a logical consequence of the rule r, which is sometimes written as $r \models e$.

Now that we know *what* to compute, we can look at *how* this can be realized. The computation of the solutions proceeds typically by searching the space of possible patterns or hypotheses $\mathcal{L}$. One way of realizing this is to employ a generate-and-test algorithm, though this is too naive to be efficient. Therefore symbolic machine learning and data mining techniques typically structure the space $\mathcal{L}$ according to *generality*. One pattern or hypothesis is *more general* than another if all examples that are covered by the latter pattern are also covered by the former.

Example 1.7. For instance, the rule

$$\mathsf{active(M)} \leftarrow \mathsf{ring\_size\_5(M,R)}, \mathsf{element(A1,R)}, \mathsf{bond(M,A1,A2,2)}$$

is more general than the rule

$$\begin{aligned}\mathsf{active(M)} \leftarrow\ & \\ & \mathsf{ring\_size\_5(M,R)}, \mathsf{element(A1,R)}, \\ & \mathsf{bond(M,A1,A2,2)}, \mathsf{atom(M,A2,o,52,C)}\end{aligned}$$

which reads as

> Molecule M is active IF it contains a ring of size 5 called R and atoms A1 and A2 that are connected by a double (2) bond such that A1 also belongs to the ring R and atom A2 is an oxygen of type 52.

The former rule is more general (or, equivalently, the latter one is more specific) because the latter one requires also that the atom connected to the ring of size 5 be an oxygen of atom-type 52. Therefore, all molecules satisfying the latter rule will also satisfy the former one.

The generality relation is quite central during the search for solutions. The reason is that the generality relation can often be used 1) to prune the search space, and 2) to guide the search towards the more promising parts of the space. The generality relation is employed by the large majority of logical and relational learning systems, which often search the space in a *general-to-specific* fashion. This type of system starts from the most general rule (the unconditional rule, which states that *all* molecules are active in our running example), and then repeatedly specializes it using a so-called refinement operator. Refinement operators map rules onto a set of specializations; cf. Chapters 3 and 5.

Example 1.8. Consider the rule

$$\mathsf{active(Mol)} \leftarrow \mathsf{atom(Mol,Atom,c,Type,Charge)},$$

which states that a molecule is active if it contains a carbon atom. Refinements of this rule include:

$\mathsf{active(Mol)} \leftarrow \mathsf{atom(Mol, Atom, c, 21, Charge)}$
$\mathsf{active(Mol)} \leftarrow \mathsf{atom(Mol, Atom, c, T, Charge), atom(Mol, Atom2, h, T2, Charge2)}$
$\mathsf{active(Mol)} \leftarrow \mathsf{atom(Mol, Atom, c, T, Charge), ring\_size\_5(Mol, Ring)}$
...

The first refinement states that the carbon atom must be of type 21, the second one requires that there be carbon as well as hydrogen atoms, and the third one that there be a carbon atom and a ring of size 5. Many more specializations are possible, and, in general, the operator depends on the description language and generality relation used.

The generality relation can be used to prune the search. Indeed, assume that we are looking for rules that cover at least 20% of the active molecules and at most 1% of the inactive ones. If our current rule (say the second one in Ex. 1.7) only covers 18% of the actives, then we can prune away all specializations of that rule because specialization can only decrease the number of covered examples. Conversely, if our current rule covers 2% of the inactives, then all generalizations of the rule cover at least as many inactives (as generalization can only increase the number of covered examples), and therefore these generalizations can safely be pruned away; cf. Chapter 3 for more details.

Using logical description languages for learning provides us not only with a very expressive representation, but also with an excellent theoretical foundation for the field. This becomes clear when looking at the generality relation. It turns out that the generality relation coincides with logical entailment. Indeed, the above examples of the generality relation clearly show that the more general rule logically entails the more specific one.[3] So, the more specific rule is a logical consequence of the more general one, or, formulated differently, the more general rule logically entails the more specific one. Consider the simpler example: $\mathsf{flies(X)} \leftarrow \mathsf{bird(X)}$ (if X is a bird, then X flies), which logically entails and which is clearly more general than the rule $\mathsf{flies(X)} \leftarrow \mathsf{bird(X), normal(X)}$ (only normal birds fly). This property of the generalization relation provides us with an excellent formal basis for studying inference operators for learning. Indeed, because one rule is more general than another if the former entails the latter, deduction is closely related to specialization as deductive operators can be used as specialization operators. At the same time, as we will see in detail in Chapter 5, one can obtain generalization (or inductive inference) operators by inverting deductive inference operators.

1.4 A Brief History

Logical and relational learning typically employ a form of reasoning known as *inductive inference*. This form of reasoning generalizes specific facts into gen-

[3] This property holds when learning from entailment. In other settings, such as learning from interpretations, this property is reversed; cf. Chapter 5. The more specific hypothesis then entails the more general one.

eral laws. It is commonly applied within the natural sciences, and therefore has been studied in the philosophy of science by several philosophers since Aristotle. For instance, Francis Bacon investigated an inductive methodology for scientific inquiry. The idea is that knowledge can be obtained by careful experimenting, observing, generalizing and testing of hypotheses. This is also known as empiricism, and various aspects have been studied by many other philosophers including Hume, Mill, Peirce, Popper and Carnap. Inductive reasoning is fundamentally different from deductive reasoning in that the conclusions of inductive reasoning do not follow logically from their premises (the observations) but are always cogent; that is, they can only be true with a certain probability. The reader may notice that this method is actually very close in spirit to that of logical and relational learning today. The key difference seems to be that logical and relational learning investigates *computational* approaches to inductive reasoning.

Computational models of inductive reasoning and scientific discovery have been investigated since the very beginning of artificial intelligence. Several cognitive scientists, such as a team involving the Nobel prize winner Herbert A. Simon (see [Langley et al., 1987] for an overview), developed several models that explain how specific scientific theories could be obtained. Around the same time, other scientists (including Bruce Buchanan, Nobel prize winner Joshua Lederberg, Ed Feigenbaum, and Tom Mitchell [Buchanan and Mitchell, 1978]) started to develop learning systems that could assist scientists in discovering new scientific laws. Their system META-DENDRAL produced some new results in chemistry and were amongst the first scientific discoveries made by an artificial intelligence system that were published in the scientific literature of the application domain. These two lines of research have actually motivated many developments in logical and relational learning albeit there is also a crucial difference between them. Whereas the mentioned approaches were *domain-specific*, the goal of logical and relational learning is to develop *general-purpose* inductive reasoning systems that can be applied across different application domains. The example concerning structure activity relationship is a perfect illustration of the results of these developments in logical and relational learning. An interesting philosophical account of the relationship between these developments is given by Gillies [1996].

Supporting the scientific discovery process across different domains requires a solution to two important computational problems. First, as scientific theories are complex by their very nature, an expressive formalism is needed to represent them. Second, the inductive reasoning process should be able to employ the available background knowledge to obtain meaningful hypotheses. These two problems can to a large extent be solved by using logical representations for learning.

The insight that various types of logical and relational representations can be useful for inductive reasoning and machine learning can be considered as an outgrowth of two parallel developments in computer science and artificial intelligence. First, and most importantly, since the mid-1960s a number

of researchers proposed to use (variants of) predicate logic as a formalism for studying machine learning problems. This was motivated by severe limitations of the early machine learning systems that essentially worked with propositional representations. Ranan Banerji [1964] was amongst the earliest advocates of the use of logic for machine learning. The logic he proposed was motivated by a pattern recognition task. Banerji's work on logical descriptions provided inspiration for developing logical learning systems such as CONFUCIUS [Cohen and Sammut, 1982] and MARVIN [Sammut and Banerji, 1986], which already incorporated the first inverse resolution operators; cf. Chapter 5. These systems learned incrementally and were able to employ the already learned concepts during further learning tasks.

Around the same time, Ryszard Michalski [1983] developed his influential AQ and INDUCE systems that address the traditional classification task that made machine learning so successful. Michalski's work stressed the importance of both learning readable descriptions and using background knowledge. He developed his own variant of a logical description language, the Variable Valued Logic, which is able to deal with structured data and relations. At the same time, within VVL, he suggested that induction be viewed as the inverse of deduction and proposed several inference rules for realizing this. This view can be traced back in the philosophy of science [Jevons, 1874] and still forms the basis for much of the theory of generalization, which is extensively discussed in Chapter 5.

Theoretical properties of generalization and specialization were also studied by researchers such as Plotkin [1970], Reynolds [1970], Vere [1975] and Buntine [1988]. Especially, Plotkin's Ph.D. work on θ-subsumption and relative subsumption, two generalization relations for clausal logic, has been very influential and still constitutes the main framework for generalization in logical learning. It will be extensively studied in Chapter 5. Second, there is the work on automatic programming [Biermann et al., 1984] that was concerned with synthesizing programs from examples of their input-output behavior, where researchers such as Biermann and Feldman [1972], Summers [1977] and Shapiro [1983] contributed very influential systems and approaches. Whereas Alan Biermann's work was concerned with synthesizing Turing machines, Phil Summers studied functional programs (LISP) and Ehud Shapiro studied the induction of logic programs and hence contributed an inductive logic programming system avant la lettre. Shapiro's MODEL INFERENCE SYSTEM is still one of the most powerful program synthesis and inductive inference systems today. It will be studied in Chapter 7.

In the mid-1980s, various researchers including Bergadano et al. [1988], Emde et al. [1983], Morik et al. [1993], Buntine [1987] and Ganascia and Kodratoff [1986] contributed inductive learning systems that used relational or logical description languages. Claude Sammut [1993] gives an interesting account of this period in logical and relational learning.

A breakthrough occurred when researchers started to realize that both problems (in automatic programming and machine learning) could be stud-

ied simultaneously within the framework of computational logic. It was the contribution of Stephen Muggleton [1991] to define the new research field of *inductive logic programming* (ILP) [Muggleton and De Raedt, 1994, Lavrač and Džeroski, 1994] as the intersection of inductive concept learning and logic programming and to bring together researchers in these areas in the inductive logic programming workshops [Muggleton, 1992b] that have been organized annually since 1991. A new subfield of machine learning was born and attracted many scientists, especially in Japan, Australia and Europe (where two European projects on inductive logic programming were quite influential [De Raedt, 1996]). Characteristic for the early 1990s was that inductive logic programming was developing firm theoretical foundations, built on logic programming concepts, for logical learning. In parallel, various well-known inductive logic programming systems were developed, including FOIL [Quinlan, 1990], GOLEM [Muggleton and Feng, 1992], PROGOL [Muggleton, 1995], CLAUDIEN [De Raedt and Dehaspe, 1997], MOBAL [Morik et al., 1993], LINUS [Lavrač and Džeroski, 1994]. Also, the first successes in real-life applications of inductive logic programming were realized by Ross King, Stephen Muggleton, Ashwin Srinivasan, and Michael Sternberg [King et al., 1992, King and Srinivasan, 1996, King et al., 1995, Muggleton et al., 1992]; see [Džeroski, 2001] for an overview. Due to the success of these applications and the difficulties in true progress in program synthesis, the field soon focused on machine learning and data mining rather than on automatic programming. A wide variety of systems and techniques were being developed that *upgraded* traditional machine learning systems towards the use of logic; cf. Chapter 6.

During the mid-1990s, both the data mining and the uncertainty in artificial intelligence communities started to realize the limitations of the key representation formalism they were using. Within the data mining community, the item-sets representation employed in association rules [Agrawal et al., 1993] corresponds essentially to a boolean or propositional logic, and the Bayesian network formalism [Pearl, 1988] defines a probability distribution over propositional worlds. These limitations motivated researchers to look again at more expressive representations derived from relational or first-order logic. Indeed, within the data mining community, the work on WARMR [Dehaspe et al., 1998], which discovers frequent queries and relational association rules from a relational database, was quite influential. It was successfully applied on a structure activity relationship prediction task, and motivated several researchers to look into graph mining [Washio et al., 2005, Inokuchi et al., 2003, Washio and Motoda, 2003]. Researchers in data mining soon started to talk about *(multi-)relational data mining* (MRDM) (cf. [Džeroski and Lavrač, 2001]), and an annual series of workshops on multi-relational data mining was initiated [Džeroski et al., 2002, 2003, Džeroski and Blockeel, 2004, 2005].

A similar development took place in the uncertainty in artificial intelligence community. Researchers started to develop expressive probabilistic logics [Poole, 1993a, Breese et al., 1994, Haddawy, 1994, Muggleton, 1996], and started to study learning [Sato, 1995, Friedman et al., 1999] in these frame-

works soon afterward. In the past few years, these researchers have gathered in the statistical relational learning (SRL) workshops [Getoor and Jensen, 2003, 2000, De Raedt et al., 2005, 2007a]. A detailed overview and introduction to this area is contained in Chapter 8 and two recent volumes in this area include [Getoor and Taskar, 2007, De Raedt et al., 2008]. Statistical relational learning is one of the most exciting and promising areas for relational and logical learning today.

To conclude, the field of logical and relational learning has a long history and is now being studied under different names: inductive logic programming, multi-relational data mining and (statistical) relational learning. The approach taken in this book is to stress the similarities between these trends rather than the differences because the problems studied are essentially the same even though the formalisms employed may be different. The author hopes that this may contribute to a better understanding of this exciting field.

http://www.springer.com/978-3-540-20040-6

Logical and Relational Learning
De Raedt, L.
2008, XV, 387 p., Hardcover
ISBN: 978-3-540-20040-6