From Data Mining to Knowledge Mining

Kenneth A. Kaufman and Ryszard S. Michalski

ABSTRACT

In view of the tremendous production of computer data worldwide, there is a strong need for new powerful tools that can automatically generate useful knowledge from a variety of data, and present it in human-oriented forms. In efforts to satisfy this need, researchers have been exploring ideas and methods developed in machine learning, statistical data analysis, data mining, text mining, data visualization, pattern recognition, etc. The first part of this chapter is a compendium of ideas on the applicability of symbolic machine learning and logical data analysis methods toward this goal. The second part outlines a multistrategy methodology for an emerging research direction, called *knowledge mining*, by which we mean the derivation of high-level concepts and descriptions from data through symbolic reasoning involving both data *and* relevant background knowledge. The effective use of background as well as previously created knowledge in reasoning about new data makes it possible for the knowledge mining system to derive useful new knowledge not only from large amounts of data, but also from limited and weakly relevant data.

1 INTRODUCTION

We are witnessing the extraordinary expansion of computer accessible data about all kinds of human activities. The availability of these large volumes of data and our limited capabilities to process them effectively creates a strong need for new methodologies for extracting useful, task-oriented knowledge from them. There is also a need for methodologies for deriving plausible knowledge from small and indirectly relevant data, as in many practical areas, only such data may be available, e.g., fraud detection, terrorism prevention, computer intrusion detection, early cancer diagnosis, etc. This chapter addresses issues and methods concerned with developing a new research direction, called *knowledge mining*, which aims at solving both types of problems.

Current tools for analyzing data and extracting from it useful patterns and regularities primarily use conventional statistical methods, such as regression analysis, numerical taxonomy, multidimensional scaling, and more recent data mining techniques, such as classification and regression trees, association rules, and Bayesian nets (e.g., Daniel and Wood, 1980; Tukey, 1986; Pearl, 1988, 2000; Morgenthaler and Tukey, 1989; Diday, 1989; Sharma, 1996, Neapolitan, 2003). While useful for many applications, these techniques have inherent limitations.

For example, a statistical analysis can determine distributions, covariances and correlations among variables in data, but is not able to characterize these dependencies at an abstract, conceptual level as humans can, and produce a causal explanation why these dependencies exist. While a statistical data analysis can determine the central tendencies and variance of given factors, it cannot produce a qualitative description of the regularities, nor can it determine a dependence on factors not explicitly provided in the data.

Similarly, a numerical taxonomy technique can create a classification of entities, and specify a numerical similarity among the entities assembled into the same or different categories, but it cannot alone build qualitative descriptions of the classes created and present a conceptual justification for including entities into a given category. Attributes and methods that are used to measure the similarity must be specified by a data analyst in advance. Popular classification (or decision) trees or forests can represent a relationship between the input and output variables, but their representation power is very modest. These methods may thus produce a very complex tree even for a conceptually simple relationship. Similarly, association rules, which are popular in data mining, have a limited representation power. Bayesian nets are very attractive for many applications, but typically rely on human input as to their structure, and can automatically determine only relatively simple interrelationships among attributes or concepts.

The above methods typically create patterns that use only attributes that are present in the data. They do not by themselves draw upon background domain knowledge in order to automatically generate additional relevant attributes, nor do they determine attributes' changing relevance to different data analysis problems. In cases where the goal is to address such tasks as those listed above, a data analysis system has to be equipped with a substantial amount of background knowledge, and be able to conduct symbolic reasoning involving that knowledge and the input data.

In efforts to satisfy the growing need for new data analysis tools that can overcome the above limitations, researchers have turned to ideas and methods developed in symbolic machine learning. The field of machine learning is a natural source of ideas for this purpose, because the essence of research in this field is to develop computational models for acquiring knowledge from facts and background knowledge.

The above and related efforts led to the emergence of a research area concerned with logical data analysis, and the development of methods for data mining and knowledge discovery (e.g., Lbov, 1981; Michalski, Baskin and Spackman, 1982; Zhuravlev and Gurevitch, 1989; Zagoruiko, 1991; Michalski et al., 1992; Van Mechelen et al., 1993; Fayyad et al., 1996; Evangelos and Han, 1996; Brachman et al., 1996; Fayyad, Haussler and Stolorz, 1996; Michalski and Kaufman, 1998; Han and Kamber, 2001; Hand, Mannila and Smyth, 2001; Alexe, Blackstone and Hammer, 2003).

A natural step in this progression appears to be the development of systems that closely integrate databases with inductive learning and data mining capabilities (e.g., Michalski et al., 1992, Khabaza and Shearer, 1995; Han et al, 1996; Imielinski, Virmani, and Abdulghani, 1996). Such systems would be able to, for example, automatically call upon a decision rule generator, regression analysis, conceptual clusterer or attribute generation operator, depending on the state of data analysis.

To achieve such a capability, a database needs to be integrated with a knowledge base and a wide range of data analysis, inductive learning, and data managment methods. We call such systems *inductive databases,* and consider them to be a technical basis for implementing knowledge mining (Michalski and Kaufman, 1998; Kaufman and Michalski, 2003). An inductive database can answer not only those queries for which answers are stored in its memory, but also those that require the synthesis of *plausible knowledge*, generated by inductive inference from facts in the database and prior knowledge in the knowledge base[1].

It should be mentioned that our meaning of the term "inductive database" is somewhat different from that of Imielinski and Mannila, (1996) and De Raedt et al., (2002), by which they mean a database that stores both original data and induced hypotheses. Because from any non-trivial dataset a very large number of inductive hypotheses can be generated, and it is not known a priori which of them may be the most useful for the future tasks of interest, it is better in our view to equip a database with inductive inference capabilities, rather than store only their amassed results. These capabilities should be tightly integrated with the query language, so that they are transparent to the user, who can apply them without having to invoke separate data analysis or inductive learning programs.

This chapter discusses selected ideas and methods for logical (or conceptual) data analysis, initiated by Michalski, Baskin, and Spackman (1982), which provide a basis for the development of a knowledge mining methodology and its implementation in an inductive database system. The chapter is an update and extension of our earlier work presented in (Michalski and Kaufman, 1998).

While this chapter presents many issues generally and makes many references to research done by others, its main focus is on the ideas and methods developed by the authors initially at the University of Illinois at Urbana-Champaign, and more recently at the George Mason University Machine Learning and Inference Laboratory. While the methods are presented in the context of analyzing numeric and symbolic data, they can also be applied to text, speech or image mining (e.g., Bloedorn, Mani and MacMillan, 1996; Umann, 1997; Cavalcanti et al., 1997; Michalski et al., 1998).

[1] It may be interesting to note that R.S. Michalski first introduced and taught this concept in his course on deductive and inductive databases in 1973 at the University of Illinois at Urbana-Champaign.

2 KNOWLEDGE GENERATION OPERATORS

This section discusses classes of symbolic learning methods that can serve as *knowledge generation operators* in logical data analysis and knowledge mining.

2.1 Discovering Rules and Patterns via AQ Learning

An important class of tools for knowledge discovery in databases stems from concept learning methods developed in machine learning. Given collections of examples of different concepts (or decision classes), the concept learning program hypothesizes a general description of each class. Some inductive methods use a fixed criterion for choosing the description from a large number of possible hypotheses, while others allow the user to define a criterion that reflects the problem at hand. A concept description can be in the form of a set of decision rules, a decision tree, a semantic net, etc. A decision rule can also take on many different forms.

In the AQ learning methodology, which we will discuss here, the general form of a decision (or classification) rule is:

$$\text{CONSEQUENT} \;\Leftarrow\; \text{PREMISE} \;|\_\; \text{EXCEPTION} \tag{1}$$

where CONSEQUENT is a statement indicating a decision, a class, or a concept name to be assigned to an entity (an object or situation) that satisfies PREMISE, provided it does not satisfy EXCEPTION; PREMISE is a logical expression (e.g., a product of logical conditions or a disjunction of such products), EXCEPTION (optional) defines conditions under which the rule does not apply; and $\Leftarrow$ denotes implication.

If PREMISE is a disjunctive description (a disjunction of products), then rule (1) can be transformed into several rules with the same CONSEQUENT, in which PREMISE is a conjunctive description (a single product of conditions). For example, Figure 1 shows two rules representing a disjunctive description of a computer user profile learned by the AQ21 learning program in a study on learning patterns in computer user behavior (Michalski et al., 2005). AQ21 is the most recent member of the family of AQ inductive learning programs (Wojtusiak, 2004). The rules characterize two different patterns of the behavior of User 1.

User1⇐ session_time_new_window ≤ 3hours (368, 8340) &
#characters_in_protected_words = 0 (110, 919) &
#processses_in_current_window ≤ 7 (203, 4240) &
#windows_opened ≤ 16 (369,7813): 65, 0

User1 ⇐ process_name = explorer (93, 875) &
#characters_in_protected_words = 9..24 (161, 2999) &
#processes_in_current_window ≤ 7 (203, 4240) &

#windows_opened ≤ 16 (369, 7813) &
#protected_words_in_window_title = 1 (109, 3304): 31, 0

Figure 1. Two rules in User 1's profile learned by AQ21.

The first rule says that *User 1* is indicated if the current window was opened less than 3 hours into the session, there are no protected words[2] in the window title, the number of active processes in the current window does not exceed seven, and the total number of windows opened during the session does not exceed 16.

The second rule says that User 1 is also indicated if the name of the active process is explorer, there are 9 to 24 characters in the protected words in the window title, the number of active processes in the current window does not exceed seven, the total number of windows opened during the session does not exceed 16, and there is only one protected word in the window title.

The pairs of numbers in parentheses after each condition in the rules indicate the number of positive and negative training events, respectively, that support (are covered by) the condition For example, the first condition in the first rule covers 368 training events of User 1's behavior, and 8340 training events of behavior of the other users. The pair of numbers at the end of each rule indicates the total number of positive and negative training events covered by the rule. For example, the first rule covers 65 events from the training data of User 1, but no events in the training data of other users. Because both rules cover no negative events, they are said to be *consistent* with the training data.

The rules in Figure 1 are examples of simple *attributional rules*, which are decision rules expressed in *attributional calculus*, a logic system used for representing knowledge in AQ learning (Michalski, 2004). A set of attributional rules with the same consequent (indicating the same decision) is called a *ruleset*. A collection of rulesets whose consequents span all values of the output (decision) variable is called a *ruleset family*, or a *classifier*. Rules, rulesets and classifiers are examples of *attributional descriptions*, as they involve only attributes in characterizing entities.

In contrast to attributional descriptions, *relational (*aka *structural)* descriptions employ not only attributes but also multi-argument predicates representing relationships among components of the entities. Such descriptions are produced, for example, by the INDUCE inductive learning programs (Larson, 1977; Bentrup. Mehler and Riedesel, 1987), and by inductive logic programs (e.g., Muggleton, 1992). Constructing structural descriptions requires a more complex description language that includes multi-argument predicates, for example, PROLOG, or Annotated Predicate Calculus (Michalski, 1983; Bratko, Muggleton and Karalic, 1997).

[2] In order to protect user privacy, during preprocessing, all words in the window title that were not names of system programs or functions were replaced with randomly selected numbers. The words that were not affected by this sanitization were called "protected."

For database exploration, attributional descriptions appear to be the most important and the easiest to implement, because most databases characterize entities in terms of attributes. As one can see, they are also easy to interpret and understand. A simple and popular form of attributional description is a decision or classification tree. In such a tree, nodes correspond to attributes, branches stemming from the nodes correspond to attribute values, and leaves correspond to individual classes (e.g., Quinlan, 1986). A decision tree can be transformed into a set of simple attributional rules (a ruleset family) by traversing all paths from the root to individual leaves. Such rules can often be simplified by detecting superfluous conditions in them, but such a process can be computationally very costly (e.g., Quinlan, 1993). The opposite process of transforming a ruleset family into a decision tree is simple, but may introduce superfluous conditions because a tree representation is less expressive than a rule representation (Imam and Michalski, 1993).

The attributional calculus distinguishes between many different types of attributes, such as nominal, rank, cyclic, structured, interval, ratio, absolute, set-valued, and compound (Michalski, 2004). By distinguishing so many attribute types, attributional calculus caters to the needs of knowledge mining. By taking into consideration different attribute types, a learning system can be more effective in generating inductive generalizations, because different generalization rules apply to different attribute types (Michalski, 1983). Thus, a specification of such attribute types constitutes a form of background knowledge used in knowledge mining.

The aforementioned AQ21 is a multipurpose learning system for generalizing cases into rules and detecting patterns in data. It is an updated version of AQ19, which was used as a major module of the INLEN system developed for testing initial ideas and methods for knowledge mining (see Section 5). The input to AQ21 consists of a set of training examples representing different concepts (classes, decisions, predictions, etc.), parameters defining the type of description to be learned and how it should be learned, a multi-criterion measure of description quality, and background knowledge, which includes a specification of domains and types of attributes, hierarchical structures defining structured domains, and arithmetic and logical rules suggesting ways to improve the representation space and/or define constraints on it. The measure of description quality (or preference criterion) may refer to computational simplicity of the description, its generality level, the cost of measuring attributes in the description, or an estimate of its predictive ability.

Many symbolic learning programs learn rules that are *consistent* and *complete* with regard to the input data. This means that they completely and correctly classify every distinct training example. Others select rules according to a description quality criterion that does not necessarily give maximal weight to a description's consistency. The AQ21 learning program, depending on the setting of its parameters, can generate either complete and consistent descriptions, or strong patterns that can be partially inconsistent and incomplete.

2.2 Types of Problems in Learning from Examples

Descriptions generated from examples by symbolic learning programs may take two forms, depending on whether the learning goal is to describe members of a particular concept (a group of entities), or contrast them against other groups. Descriptions that enumerate the common properties of the entities in each group are called *characteristic description*s. Descriptions that specify differences between groups are called *discriminant descriptions*.

Some methods for concept learning assume that examples do not have errors, that all attributes have a specified value in them, that all examples are located in the same database, and that concepts to be learned have a precise ("crisp") description that does not change over time. In many situations one or more of these assumptions may not hold. This leads to a variety of more complex machine learning and data mining problems and methods for solving them:

- *Learning from noisy data*, i.e., learning from examples that contain a certain amount of errors or noise (e.g., Quinlan, 1990; Michalski and Kaufman, 2001). These problems are particularly important for data and knowledge mining because databases frequently contain some amount of noise.

- *Learning from incomplete data*, i.e., learning from examples in which the values of some attributes are unknown (e.g., Dontas, 1988, Lakshminarayan et al., 1996).

- *Learning from distributed data*, i.e., learning from spatially distributed collections of data that must be considered together if the patterns within them are to be exposed (e.g., Ribeiro, Kaufman and Kerschberg, 1995).

- *Learning drifting or evolving concepts*, i.e., learning concepts that are not stable but changing over time, randomly or in a certain general direction. For example, the "area of interest" of a computer user is usually an evolving concept (e.g., Widmer and Kubat, 1996).

- *Learning concepts from data arriving over time*, i.e., incremental learning in which currently held hypotheses characterizing concepts may need to be updated to account for the new data (e.g., Maloof and Michalski, 2004).

- *Learning from biased data*, i.e., learning from a data set that does not reflect the actual distribution of events (e.g., Feelders, 1996).

- *Learning flexible concepts*, i.e., concepts that inherently lack precise definition and whose meaning is context-dependent; approaches concerned with this topic include *fuzzy sets* (e.g., Zadeh, 1965; Dubois, Prade and Yager, 1993), *two-tiered concept representations* (e.g., Michalski, 1990; Bergadano et al., 1992), and *rough sets* (e.g., Pawlak, 1991; Slowinski, 1992; Ziarko, 1994).

- *Learning concepts at different levels of generality*, i.e., learning descriptions that involve concepts from different levels of generalization hierarchies (e.g., Kaufman and Michalski, 1996). An example of such problem is learning the concept of a liver disease versus the concept of liver cancer.

- *Integrating qualitative and quantitative discovery*, i.e., determining sets of equations that fit a given set of data points, and qualitative conditions for the application of these equations (e.g., Falkenhainer and Michalski, 1990).

- *Qualitative prediction*, i.e., discovering patterns in sequences or processes and using these patterns to qualitatively predict the possible continuation of the given sequences or processes (e.g., Davis, 1981; Michalski, Ko and Chen, 1985; 1986; Dietterich and Michalski, 1986).

Each of these problems is relevant to the derivation of useful knowledge from a collection of data and knowledge. Therefore, it can be asserted that methods for solving these problems developed in the area of machine learning are directly relevant to logical data analysis and knowledge mining.

2.3 Clustering of Entities into Conceptually Meaningful Categories

Another class of machine learning methods relevant to knowledge mining concerns the problem of building a conceptual classification of a given set of entities. The problem is similar to that considered in traditional cluster analysis, but is defined in a different way, which allows for knowledge-based constructions. Given a set of attributional descriptions of entities, a description language for characterizing classes of such entities, and a classification quality criterion, the problem is to partition entities into classes that have a simple and meaningful description in the given description language and maximize the classification quality criterion. Thus, a conceptual clustering method seeks not only a classification structure of entities, but also an understandable description of the proposed classes (clusters). An important, distinguishing aspect of conceptual clustering is that, unlike in similarity-based cluster analysis, the properties of these class descriptions are taken into consideration in the process of determining the partition of entities into clusters (e.g. Michalski and Stepp, 1983).

To clarify the difference between conceptual clustering and conventional clustering, notice that a conventional clustering method typically determines clusters on the basis of a similarity measure that is a function solely of the properties (attribute values) of the entities being compared, and not of any other factors:

$$\textit{Similarity}(\mathrm{A}, \mathrm{B}) = \mathrm{f}(\mathrm{properties}(\mathrm{A}), \mathrm{properties}(\mathrm{B})) \qquad (2)$$

where A and B are entities being compared.

In contrast, a conceptual clustering program creates clusters based on *conceptual cohesiveness*, which is a function of not only properties of the entities, but also of two other factors: the *description language* L, which the system uses for describing the classes of entities, and of the *environment*, E, which is the set of neighboring examples:

$$\textit{Conceptual cohesiveness}(\text{A, B}) = \text{f}(\text{properties}(\text{A}), \text{properties}(\text{B}), \text{L}, \text{E}) \tag{3}$$

Thus, two objects may be *similar*, i.e., close according to some distance measure, while having a low conceptual cohesiveness, or *vice versa*. An example of the first situation is shown in Figure 2. The points (black dots) A and B are “close” to each other; in fact, they are closer than any other pair of points in the figure. They would therefore be placed into the same cluster by any technique based solely upon the distances between the points. However, these points have small conceptual cohesiveness, because they can be viewed as belonging to configurations representing different concepts.

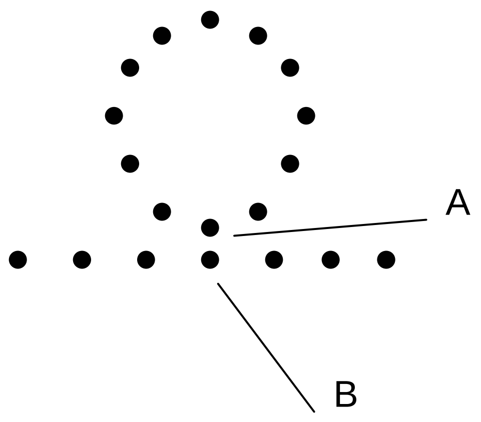

Figure 2. An illustration of the difference between closeness and conceptual cohesiveness.

A conceptual clustering method, if equipped with an appropriate description language, would cluster the points in Figure 2 into a circle and a horizontal line, as people normally would. A classification quality criterion used in conceptual clustering may involve a variety of factors, such as the *fit* of a cluster description to the data (called sparseness), the *simplicity* of the description, and other properties of the entities or the concepts that describe them (Michalski and Stepp, 1983; Stepp and Michalski, 1986). Ideas on employing conceptual clustering for structuring text databases and creating concept lattices for discovering dependencies in data are described by Carpineto and Romano (1995a; 1995b). The concepts created through the clustering are linked in lattice structures that can be traversed to represent generalization and specialization relationships.

Recent advances in traditional, similarity-based clustering have attempted to go beyond the limitations described above as well. Subspace clustering (e.g., Agrawal et al, 1998; Aggarwal et al, 1999; Wang et al, 2004) alters the description language and the environment before creating groups by projecting the event space on a subspace that produces better clusters. Various approaches to manifold learning (Tenenbaum, de Silva and Langford, 2000; Saul and Roweis, 2003; Belkin and Niyogi, 2004) also attempt to reduce the dimensionality of the problem; they differ from subspace clustering in that their focus is preserving the relationships among the datapoints, rather than compacting them.

2.4 Automated Improvement of the Search Space: Constructive Induction

Most methods for learning from examples assume that the attributes used for describing examples are sufficiently relevant to the learning problem at hand. This assumption does not always hold in practice. Some attributes used in the examples may not be directly relevant (e.g., if the target concept is based on density, and we only have mass and volume attributes), and others may be irrelevant or *nonessential* (e.g., if the target concept is predicting a student's performance in a class, and an attribute indicates the student's height). An important characteristic of logical analysis methods is that they can relatively easily determine irrelevant or nonessential attributes.

An attribute is *nonessential* if there is a complete and consistent description of the concepts to be learned that does not use this attribute. Thus, a nonessential attribute may be either irrelevant or relevant, but will by definition be dispensable. There may exist a set of attributes, each one by itself nonessential, yet some member of the set must be present in order to generate a complete and consistent attributional description. Inductive learning programs such as the rule-learning program AQ21, or the decision tree-learning C4.5, can cope relatively easily with a large number of nonessential attributes in their input data.

If there are very many nonessential attributes in the input data descriptions, the complexity of the learning process may significantly increase, along with the execution time, and the risk for generating spurious rather than relevant knowledge may increase. Such a situation calls for a method that can efficiently determine the most relevant attributes for the given problem from among all those given initially. Only the most relevant attributes should be used in the learning process.

Determining the most relevant attributes is therefore a useful data exploration operator. Such an operator can also be useful for the data analyst on its own merit, as it may be important to know which attributes are most discriminatory for a given learning task. By removing less relevant attributes, the representation space is reduced, and the problem becomes simpler. Thus, such a process is a form of improving the representation space. Some methods for finding the most relevant attributes are described in (Zagoruiko, 1972; Baim, 1982; Fayyad and Irani, 1992; Caruana and Freitag, 1994; Langley, 1994).

In applications in which the attributes originally given may only be weakly or indirectly relevant to the problem at hand, there is a need for generating new, more relevant attributes that may be functions of the original attributes. These functions may be simple, e.g., a product or sum of a set of the original attributes, or very complex, e.g., a Boolean attribute based on the presence or absence of a straight line or circle in an image (Bongard, 1970). Finally, in some situations, it will be desirable to abstract some attributes, that is, to group some attribute values into units, and thus reduce the attribute's range of possible values. A quantization of continuous attributes is a common example of such an operation (e.g., Kerber, 1992).

All the above operations—removing less relevant attributes, adding more relevant attributes, and abstracting attributes—are different means of improving the original representation space for learning. A learning process that consists of two (intertwined) phases, one concerned with the construction of the "best" representation space, and the second concerned with generating the "best" hypothesis in the found space is called *constructive induction* (Michalski, 1978; 1983; Bloedorn, Wnek, and Michalski, 1993; Wnek and Michalski, 1994; Bloedorn and Michalski, 1998). An example of a constructive induction program is AQ17 described in (Bloedorn, Wnek and Michalski, 1993), which performs all three types of operators for improving the original representation space. In AQ17, the process of generating new attributes is performed through the combination of existing attributes using mathematical and/or logical operators, and then selecting the "best" combinations.

2.5 Reducing the Amount of Data: Selecting Representative Examples

When a database is very large, determining general patterns or rules characterizing different concepts may be very time-consuming. To make the process more efficient, it may be useful to extract from the database the most representative or important cases (examples) of given classes or concepts. Even a random extraction should not be costly in the case of very large datasets, as the selected set will likely be quite representative. Most methods of heuristic selection of examples attempt to select those that are either most typical or most extreme (assuming that there is not too much noise in the data). A method for determining the most representative examples, called "*outstanding representatives*," is described by Michalski and Larson (1978).

2.6 Integrating Qualitative and Quantitative Methods of Numerical Discovery

In a database that contains numerical and symbolic attributes, a useful discovery could be an equation binding numerical attributes. A standard statistical technique for this purpose is regression analysis. This technique requires that the general form of the equation is provided to the system, as in multivariate linear regression. The application of machine learning to quantitative discovery produced another approach to this problem that does not require a specification of the form of the equation.

For instance, from a table of planetary data including planets' names, planet's masses, their densities, distances from the sun, periods of rotation, lengths of local years, and the number of moons, a quantitative discovery system would derive Kepler's Law, which states that the cube of the planet's distance from the sun is proportional to the square of the length of its year. The attributes such as the planet's name and the number of moons would be ignored.

Research on quantitive discovery was pioneered by the BACON system (Langley, Bradshaw and Simon, 1983), and then followed by many other systems, such as COPER (Kokar, 1986), FAHRENHEIT (Zytkow, 1987), and ABACUS (Falkenhainer and Michalski, 1990). Similar problems have been explored independently by Zagoruiko (1972) in Russia under the name of empirical prediction.

Some equations may not apply directly to data, because of an inappropriate value of a constant, or different equations may apply under different qualitative conditions. For example, in applying Stoke's Law to determine the velocity of a falling ball, if the ball is falling through a vacuum, its velocity depends on the length of time it has been falling and on the gravitational force being exerted upon it. A ball falling through some sort of fluid will reach a terminal velocity dependent on the radius and mass of the ball and the viscosity of the fluid.

The program ABACUS (Greene, 1988; Falkenhainer and Michalski, 1990; Michael, 1991) is able to determine quantitative laws under different qualitative conditions. It does so by partitioning the data into subsets, each of which adheres to a different equation determined by a quantitative discovery module. The qualitative discovery module can then determine conditions/rules that characterize each of these example sets. For example, given a table containing data on how fast different balls fall through different media, ABACUS can discover these patterns based on the medium of descent:

If Medium = vacuum then $v = 9.8175\, t$

If Medium = glycerol then $vr = .9556\, m$

If Medium = castor oil then $vr = .7336\, m$

2.7 Predicting Processes Qualitatively

Most programs for learning rules from examples determine them from examples of various classes of objects. An example of a concept represents that concept regardless of its relationship to other examples. Contrast that with a sequence prediction problem, in which a positive example of a concept is directly dependent on the position of the example in the sequence.

For example, Figure 3 shows a sequence of nine figures. One may ask what object plausibly follows in the tenth position. To answer such a question, one needs to search for a pattern in the sequence, and then use the pattern to predict a plausible sequence continuation. In *qualitative prediction*, the problem is not to predict a specific value of a variable (as in time series analysis), but rather to *qualitatively* characterize a plausible subsequent object, that is, to describe plausible properties of that future object.

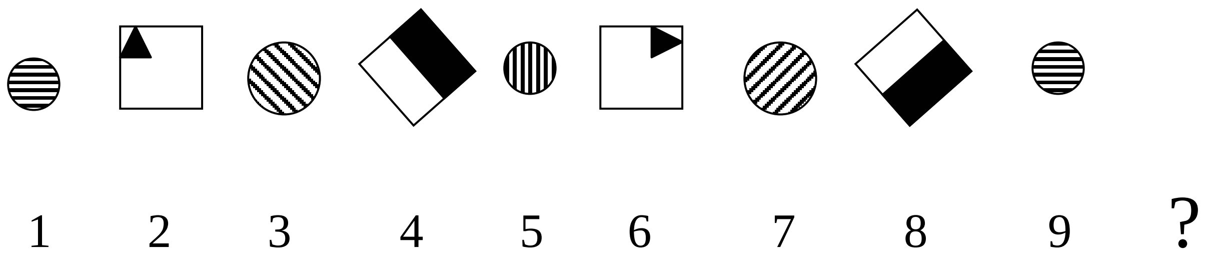

Figure 3 An example of a qualitative prediction problem.

In the example in Figure 3, one may observe that the sequence consists of circles with parallel shading and squares with dark shapes inside. The figures may be rotated in different orientations at 45-degree intervals. But is there a consistent pattern?

To determine such a pattern, one can employ different *descriptive models*, and instantiate the models to fit the particular sequence. The instantiated model that best fits the data is then used for prediction. Such a method was initially developed by Dietterich and Michalski, (1986), and then generalized in SPARC/G system to handle arbitrary sequences of entities described by attributes (Michalski, Ko and Chen, 1986). The method employs three descriptive models—periodic, decomposition, and DNF.

The *periodic model* is used to detect repeating patterns in a sequence. For example, Figure 3 depicts a recurring pattern that alternates round and square objects. In general, there can also be periodic subsequences within the periodic sequences. In the figure, the round objects form a subsequence in which individual objects rotate leftward by 45 degrees and alternate between small and large. The square objects have a subsequence alternating between those with a triangle in the corner and those half-filled. Each subsequence is rotating clockwise.

The second model, the *decomposition model*, is used to characterize a sequence by decision rules in the following general form: "If one or more of the previous elements of the sequence have a given set of characteristics, then the next element will have the following characteristics..." One such rule that applies to the sequence in Figure 3 would state that if an element in the sequence has a triangular component, then the next element in the sequence will have a diagonally shaded component; otherwise it will have no diagonal shading.

The third model, the DNF (disjunctive normal form) or "catch-all" model, tries to capture general properties characterizing the whole sequence. For example, for the sequence in Figure 3, it could instantiate to a statement such as “all elements in the sequence are round or square, their interiors are either shaded, or contain a dark rectangle or triangle, etc.

Given the problem in Figure 3, SPARC/G would find the following pattern based on the periodic model:

Period < [shape=circle] & [shading = parallel] [orientation(i+1)=orientation(i) + 45],
[shape = square] & [orientation(i+1)=orientation(i) + 45] >

The pattern can be paraphrased: there are two phases in a repeating period (their descriptions are separated by a comma). The first phase involves a circular figure, and the second phase a square figure. The cricular figure is shaded and rotates to the right and the square figure also rotates to the right by 45 degrees in relation to its predecessor. Based on this pattern, a plausible next figure in the sequence would be a square figure rotated clockwise 45 degrees in relation to the previous square figure. This rule does not specify the contents of that square.

The qualitative prediction capabilities described above can be useful for conceptual exploration of temporal databases in many application domains, such as agriculture, medicine, robotics, economic forecasting, computer intrusion detection, etc.

2.8 Knowledge Improvement via Incremental Learning

One of the very important aspects of the application of machine learning to logical data analysis is the existence of methods for incremental learning that can improve data generalizations when new data become available. This is analogous to Bayesian learning, but it is not the posterior probability of a description that being improved, but rather the description itself.

Incremental learning can take three forms, depending on how much data from which the prior knowledge was generated is available. Zero-memory learning, in which none of the earlier data is retained, is more economical, while full-memory incremental learning, in which all earlier training examples are retained, is likely to result in more accurate descriptions, provided that issues of concept drift can be accounted for. Partial-memory incremental learning is an attempt to strike a balance between these two extremes, by selecting for retention only the cases most likely to be of use later on.

The zero-memory algorithm is straightforward. New data that contradicts prior hypotheses is integrated into the prior hypotheses through specialization operators that reshape the hypotheses into ones consistent with the new data. The best of these modified hypotheses are then input to the learner along with the new data points (e.g., Michalski and Larson, 1983).

In the full-memory incremental learning, the new data that contradict the previous description are first filtered by removing any examples which were identical to earlier training examples.

The prior hypotheses are then appropriately specialized or generalized to account for new data, while preserving consistency and completeness of the description with regard to all past examples (e.g., Reinke and Michalski, 1988).

The partial memory method utilizes a selection of prior data for retention. Some partial memory systems select examples that are near the perceived boundaries of the concepts, either based on the incoming datastream (e.g., Kibler and Aha, 1987), or on induced rules (e.g., Maloof and Michalski, 2000, 2004). Others retain a seed positive example (e.g., Elio and Watanabe, 1991), or maintain only negative examples of a concept so as to define a boundary (e.g., Iba, Woogulis and Langley, 1988).

2.9 Summarizing the Logical Data Analysis Approach

To help the reader develop a rough sense of what is different and new in the above, let us consider operations typically performed by traditional multivariate data analysis methods. These include computing mean-corrected or standardized variables, variances, standard deviations, covariances and correlations among attributes; principal component analysis; factor analysis; cluster analysis; regression analysis; multivariate analysis of variance; and discriminant analysis. All these methods can be viewed as primarily oriented toward numerical characterizations of data.

In contrast, the logical data analysis approach, described above, focuses on developing symbolic logic-style descriptions of data, which may characterize data qualitatively, differentiate among classes, create a "conceptual" classification of data, qualitatively predict sequences, etc. These techniques are particularly well-suited for developing descriptions and seeking patterns in data that involve nominal (categorical), rank, and structured attributes (with hierarchically-ordered domains), although they can handle all types of attributes.

Another important distinction between the two approaches to data analysis is that purely statistical methods are particularly useful for globally characterizing a set of objects, but not so for determining a description for predicting class membership of individual objects (with some exceptions, e.g., classification trees). A statistical operator may determine, for example, that the average lifespan of a certain type of automobile is 7.3 years, but it may not provide conditions indicating the lifespan of an automobile with particular characteristics, nor the ability to recognize the type of a specific automobile from its description. A symbolic machine learning approach is particularly useful for such tasks. It may create a description such as "if the front height of a vehicle is between 5 and 6 feet, body color is silver or grey, and the driver's seat is 2 to 3 feet above the ground, then the vehicle is likely to be a minivan of brand X." Such descriptions are particularly suitable for classifying future, not yet observed entities based on their properties.

The knowledge mining methodology aims at integrating a wide range of strategies and operators for data exploration based on both machine learning research and statistical methods.

The reason for such a multistrategy approach is that a data analyst may be interested in many different types of information about the data, requiring different exploratory strategies and different operators.

3 STRONG PATTERNS VS. COMPLETE AND CONSISTENT RULES

In its early stages of development, machine learning was oriented primarily toward methods that produce consistent and complete descriptions of the training data, that is, descriptions that explain ("cover") all positive training examples of the target concepts, and none of the negative examples. In practical applications, however, data frequently contain some errors; therefore, a complete and consistent description will likely overfit the data, producing incorrect micro-patterns. Also, in practice, one may be more interested in determining a simple but not completely correct pattern than a complex but a correct one.

There have been several methods developed to determine such patterns using the symbolic learning approach. One method is through postprocessing of learned descriptions using ruleset optimization (e.g., Bergadano et al, 1992). The well-known decision tree pruning is a simple form of the same idea (e.g., Quinlan, 1993). In this method, an initially learned complete and consistent description is simplified by removing statistically insignificant components (subtree pruning in decision tree learning, or rule truncation in AQ learning), or optimizing some of its components (rule optimization in AQ learning).

Another method is to optimize descriptions during the rule generation process. Such a method employs a rule quality criterion, defined by the user, that specifies a tradeoff between completeness and consistency of a rule. At each stage of rule learning, candidate hypotheses are overgeneralized (introducing inconsistency, but increasing rule coverage), and then evaluated using the rule quality criterion. Whichever variant of the original hypothesis scores best is retained as input to the next iteration of rule learning. In this way, negative examples are ignored if the creation of a strong pattern requires it.

Such a method was implemented in AQ learning, as an additional option to rule truncation (Michalski and Kaufman, 2001). The method uses a rule quality measure Q(*w*), where *w* is a user-specified weight parameter controlling the relative importance of rule coverage in relation to rule consistency gain.

Specifically, given a training dataset consisting of *P* positive examples of a concept and *N* negative examples (examples of other concepts), and given a rule R that covers *p* positive examples and *n* negative examples, the rule's coverage (relative support) is defined as:

$$\mathrm{cov}(R) = p \,/\, P \qquad (4)$$

The consistency of rule R is defined as the fraction of covered examples that are positive (correctly classified), or:

$$\text{cons}(R) = p / (p + n) \tag{5}$$

However, without taking into account the distribution of training examples, a rule's consistency alone does not provide a strong indication of the predictive utility of the rule. Thus, we instead apply *consistency gain* (cgain), which measures the rule's improvement in performance over the expected performance of blindly guessing the positive class. A normalization factor ensures that this measure will be zero when the rule performs no better than such a blind guess, and 1 when the rule achieves 100% consistency.

$$\text{cgain}(R) = (p / (p + n)) - (P / (P + N)) * ((P + N) / N) \tag{6}$$

The Q(w) formula then combines the coverage and consistency gain terms through multiplication (so that Q will be 1 when both terms are 1, and Q will be 0 when either term is 0), and accordingly, the weight w is computed as an exponent in the equation. Specifically:

$$Q(w) = \text{cov}(R)^{w} * \text{cgain}(R)^{1-w} \tag{7}$$

It should be noted that both cov(R) and cgain(R) are functions of the rule's positive and negative support. Other programs typically also use various functions of rule's positive and negative support in evaluating the descriptions they generate.

Table 1 presents examples of how different methods can choose differently from the same set of candidate rules (Kaufman and Michalski, 1999). In the table, three separate data sets are assumed, each with total 1000 training examples. In Dataset A, 200 of the training examples are in the positive class; in Dataset B, 500 training examples are, and in Dataset C, 800 training examples are. For each data set, seven rules are hypothesized, each covering different numbers of positive and negative examples. The table shows how each set of seven rules would be ranked by information gain, by the programs PROMISE (Baim, 1982), CN2 (Clark and Niblett, 1989), and RIPPER (Cohen, 1995), and by Q(w) for w = 0, .25, .5, .75, and 1. In each column, "1" indicates the rule determined by the method to be the best (the highest rank), and "7" indicates the worst.

This is not meant to suggest that any of these ranking methods are superior or inferior to any other under.. Rather, it serves to indicate how by changing the rule quality criterion, one can often alter which rules will be selected, and demonstrates the flexibility of the Q(w) measure to emulate several different rule quality criteria through adjustment of its weight. This research thus shows that by controlling the w parameter in the AQ learning program, one can obtain rulesets representing different trade-offs between consistency and completeness, and approximate behavior of different learning programs.

Table 1 **How different methods rank different rules.**

Data Set	**Pos**	**Neg**	**RANKS** Inf. Gain	PROMISE	CN2	RIPPER	Q(0)	Q(.25)	Q(.5)	Q(.75)	Q(1)
A	50	5	7	7	4	7	4	7	7	7	6
	50	0	6	6	1	6	1	6	6	6	6
200	200	5	1	1	2	1	2	1	1	1	1
pos	150	10	2	2	3	2	3	2	2	2	2
	150	30	3	3	6	3	6	3	3	3	2
800	100	15	5	5	5	5	5	4	4	5	5
neg	120	25	4	4	7	4	7	5	5	4	4
B	50	5	7	7	3	7	3	7	7	7	7
	250	25	6	5	3	5	3	5	5	5	5
500	500	50	1	1	3	1	3	1	1	1	1
pos	500	150	2	3	7	3	7	6	4	2	1
	200	5	5	6	1	6	1	4	6	6	6
500	400	35	3	2	2	2	2	2	2	3	3
neg	400	55	4	4	6	4	6	3	3	4	3
C	50	5	7	–	3	7	3	6	6	6	7
	250	25	5	–	3	5	3	2	5	4	5
800	500	50	1	–	3	1	3	3	1	1	1
pos	500	150	6	–	7	3	7	7	7	7	1
	200	5	3	–	1	6	1	1	3	5	6
200	400	35	2	–	2	2	2	2	2	2	3
neg	400	55	4	–	6	4	6	5	4	3	3

Fürnkranz and Flach (2003) have studied the behavior of different rule quality measures, and present a means for showing graphically how these measures can be intuitively visualized and compared.

4 RULESET VISUALIZATION VIA CONCEPT ASSOCIATION GRAPHS

When working with symbolic knowledge as described above, it is desirable for a data analyst to be able to visualize the results of the learning process. The purpose of such visualization operators is to relate visually the input data to the rules that have been learned from them, to see which datapoints would corroborate or contradict these rules, to identify possible errors, etc. To this end, programs are needed that are specialized toward the visualization of data and attributional knowledge. Two such approaches are the *diagrammatic visualization* method implemented in the KV program (Zhang, 1997), and the *concept association graph* (e.g., Michalski and Kaufman, 1997; Kaufman and Michalski, 2000). The latter approach is particularly oriented toward problems of data and knowledge mining, due to a lack of complications arising from scaling up to many large attribute domains.

Concept association graphs were developed as a tool for visualizing attributional rulesets, or more generally the relationships between consequents and premises (Michalski and Kaufman, 1997)**.** Attributes, rules and their relationships are displayed in a graphical form using nodes and links. There are three different types of nodes: input nodes, output nodes and rule nodes.

Input nodes are nodes that represent components of the premise of a rule, while output nodes represent the consequent of a rule. Rule nodes represent the relationships between one or more input attributes and one or more output attributes. All of the conditions in the premise of a rule are linked to its rule-node, which is then linked to the output node(s). Input and output nodes appear as ovals in a concept association graph, and rule nodes appear as rectangles.

There are two types of links, presented as continuous links and dotted links. The dependency between input and output nodes is represented with continuous links of different thickness. The thicker the link, the stronger the relationship. The thickness of the link can be computed using many different methods, which may take into consideration for example, the completeness (what percentage of positive examples of the consequent class are covered by the condition) or the consistency (what percentage of the examples covered by the condition are of the target class). A third method combines completeness and consistency using the Q(*w*) measure (Section 3). These links are labeled by annotations that specify the values of the attribute represented by the associated input node that satisfy the condition represented by the link. This can be done either through a specification of attribute values (as seen, for example, in Figure 4 below) or, more simply, through one of four symbols that characterize those values (as seen, for example, in Figure 5 below).

The abstraction to four symbols can be used in the case of linear or binary attributes. The symbol '+' indicates a positive relationship between the attribute (or higher values of it) and the rule. The symbol ‘-‘ indicates a negative relationship; a rank attribute should have a low value, or a binary attribute should be false. Linear (rank, interval or ratio) attributes can also be characterized by the symbols ‘^’ and ‘v’, which indicate respectively that the attribute should have central or extreme values in the given condition.

Dotted links in concept association graphs are used to display generalization (is-a) relationship between nodes. For example, in the mushroom domain (see Figure 4), a dotted link shows that the output node [class=poisonous] is an instantiation of the classes-of-mushrooms node. Dotted links are optional, and are used primarily when output nodes can also serve as input nodes for another diagrammed rule.

The major advantage of a concept association graphs is that it can visualize multivariate relationships (rules) with a graphical indication of the strength of individual condition in these rules. The visualization method using concept association graphs has been implemented in program CAG1, which reads in a set of attributional rules learned by AQ-type learning program, and then displays a concept association graph. The program allows the user to modify the graph.

To illustrate different forms of concept association graphs we will use attributional rules learned from the “mushroom dataset” obtained from the data repository at the University of California at Irvine, and rules learned from a medical database representing patients with histories of different diseases.

The mushroom dataset contains the examples of more than 8000 different species of mushrooms, classified as edible or poisonous. Each mushroom is described in terms of 23 attributes, of which 22 are discrete input attributes (nominal or ordinal), and one is an output attribute, with the domain {edible, poisonous}. There were 3916 examples of poisonous mushrooms, and 4208 examples of edible mushrooms. The attributional rules learned from these examples are:

[class = poisonous]

$\Leftarrow$ [odor = creosote or fishy or foul or musty or pungent or spicy : 3796,0]
: p=3796, n=0

$\Leftarrow$ [cap_color $\neq$ cinnamon: 3904, 4176] & [gill_spacing = close or distant 3804,3008] & [stalk_root $\neq$ equal: 3660,2624] & [stalk_surface_above_ring = fibrous or silky or smooth: 3908,4192] & [ring_type $\neq$ flaring: 3916,4160] & [spore_print_color = green or purple or chocolate or yellow or white: 3468,672] & [habitat $\neq$ waste: 916,4016]: p=3440, n=24

[class = edible]

$\Leftarrow$ [odor = almond or anise or none : 4208,120]
: p=4208, n=120

Thus, there are two rules for poisonous mushrooms, and one rule for edible mushrooms. The pairs of numbers after ":" in each condition in each rule denote the number of positive and negative examples covered by this condition, respectively. Parameters p and n after each rule denote the total number of positive and negative examples covered by each rule, respectively. Thus, the first rule for the poisonous mushrooms covers 3796 examples of poisonous mushrooms and zero examples of edible mushrooms.

Given these rules, and numbers of positive and negative examples associated with each condition in each rule, CAG1 generated a concept association graph presented in Figure 4. The thickness of the lines connecting conditions with the class is proportional to the consistency of the conditions, measured by (p / p + n). Thus, in Rule 2 for the poisonous class, the link corresponding to the condition [spore_print_color = green or purple or chocolate or yellow or white] is much thicker than the others, as the consistency of this condition is approximately 84%, while the next strongest conditions, [stalk_root ≠ equal] and [gill_spacing = close or distant], have consistencies of 58% and 56%, respectively. The links associated with them are somewhat thicker than the links representing the rule's remaining conditions, which have consistencies around 50%.

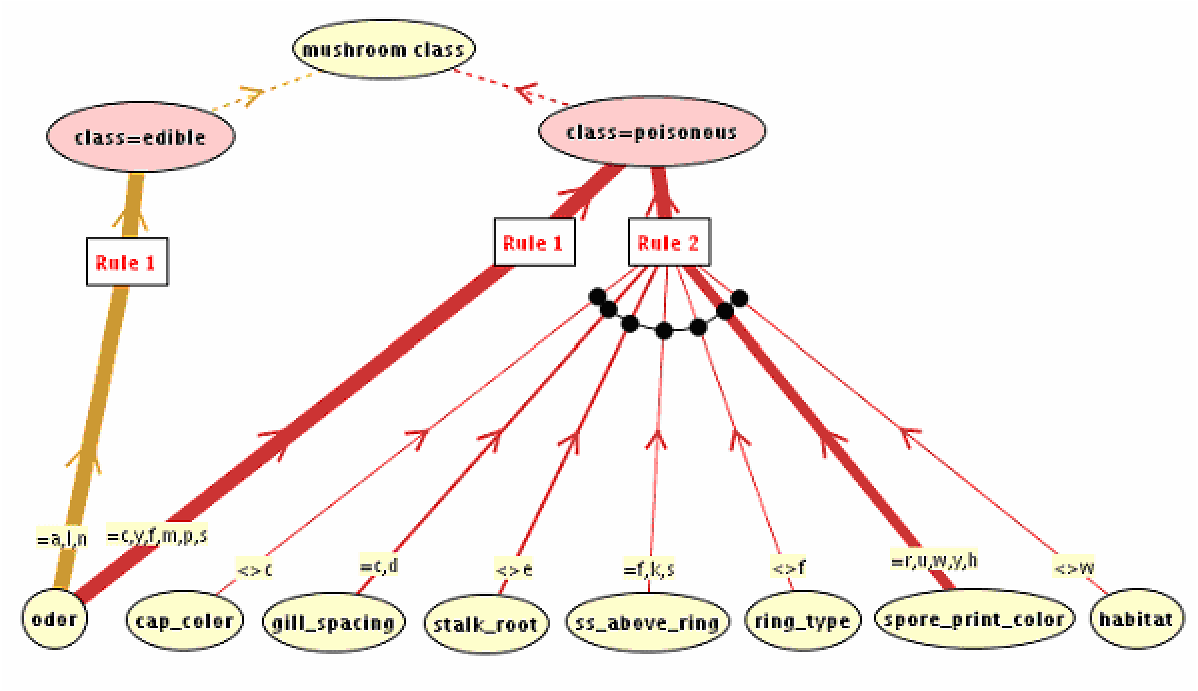

Figure 4. A concept association graph representing rules for distinguishing edible from poisonous mushrooms.

CAG1 was also used to visualize attributional rules learned from data collected by the American Cancer Society on lifestyles and diseases of nonsmoking men, aged 50-65. The data consist of over 73,000 records describing them in terms of 32 attributes; 25 are Booleans indicating the occurrence or non-occurrence of various classes of disease, and the other 7 describe elements of their lifestyles. Six of the seven are discrete attributes, with 2-7 linearly ordered values, and the seventh, representing how long the respondent had lived in the same neighborhood, is numeric. Among the discovered patterns were:

```
[Arthritis=present]
      ⇐   [High_Blood_Pressure = present] (432, 1765) &
          [Education < grad school] (940, 4529) &
          [Rotundity > very_low] (1070, 5578) &
          [Years_in_Neighborhood >= 1](1109, 5910): 325, 1156

[Colon_Polyps=present]
      ⇐   [Prostate_Disease = present] (34: 967) &
          [Sleep = 5, 9] (16, 515) &
          [Years_in_Neighborhood >= 8] (33, 1477) &
          [Rotundity = average] (58, 2693) &
          [Education < college degree] (83, 4146): 5, 0

[Diverticulosis=present]
      ⇐   [stroke = absent] (257, 7037) &
          [Arthritis = present] (70, 1033) &
          [Rotundity >= average] (170, 4202) &
          [Education >= some college] (176, 4412) &
          [Sleep = 7..9] (205, 5743) &
          [Years_in_Neighborhood > 10] (134, 3846): 24, 115

[Stomach_Ulcer=present]
      ⇐   [Arthritis = present] (107, 1041) &
          [Education <= college degree] (305, 5276) &
          [Exercise >= medium] (298, 5606): 79, 668
[Asthma=present]
      ⇐   [Hay_Fever = present] (170, 787): 170, 187
```

The first rule, for example, states that occurrence of arthritis is associated with high blood pressure, education below graduate school, rotundity (a relation of patient's weight to height) above "very low", and that patients moved into their current neighborhood at least a year ago.

When the above rules and several others were input to CAG1, it resulted in the generated graph shown in Figure 5. In this CAG, links are color-coded according to which rule they apply to for ease of viewing. The concepts are present at a higher level of abstraction than in Figure 4; as discussed above. The relationship between nodes is represented with symbols rather than with an exact "relation reference." That is, instead of the list of values for attributes shown on the input links of Figure 4, the links are instead annotated with the four symbols +, -, ^ and v.

In these graphs, the output nodes are also used directly as input nodes, without linking to an intermediate node to signify their values. This is possible because the output attributes are all binary, and the value true (or in this case "present") is understood.

In the graph in Figure 5, link thicknesses are based on completeness (support). For example, two of the links comprising the Stomach Ulcer rule in Figure 5 are noticeably thicker than the third, because the conditions involving Education and Exercise had approximately three times the support of the arthritis condition.

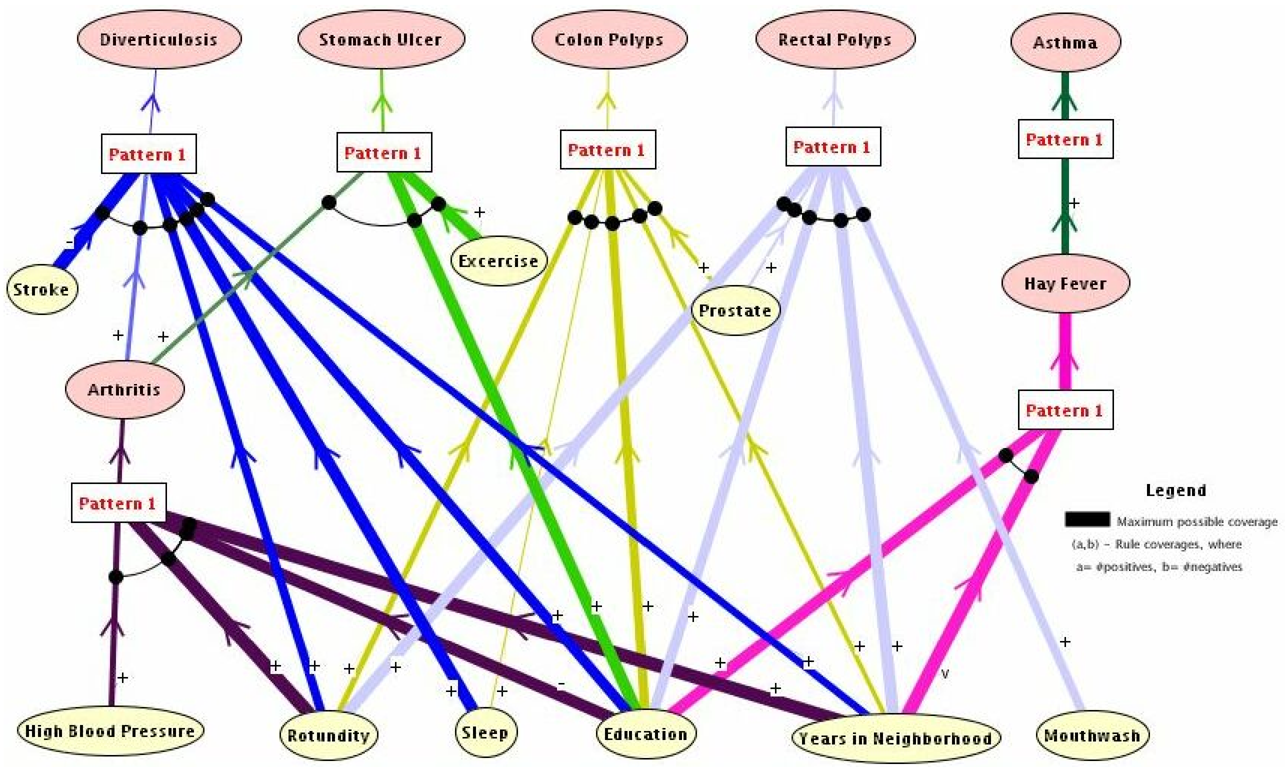

Figure 5 A concept association graph representing discovered multivariate relationships between diseases and lifestyles. Link thicknesses represent relative support.

5 INTEGRATION OF KNOWLEDGE GENERATION OPERATORS

To make the data exploration operations described above easily available to a data analyst, and applicable in sequences in which the output from one operation is an input to another one, programs performing these operations are best integrated into one system. This idea underlay the INLEN system (Michalski et al, 1992; Michalski and Kaufman, 1997), and its successor, VINLEN (Kaufman and Michalski, 2003), which is currently under development. The INLEN system integrates machine learning programs, simple statistical data analysis tools, data tables, a knowledge base, inference procedures, and various supporting programs under a unified architecture and graphical interface. The knowledge base is used for storing, updating and applying rules that may be employed for assisting data exploration, and for reporting results from it.

The general architecture of INLEN is shown in Figure 6. The system consists of *knowledge systems*, which maintain the data and knowledge relevant to a given application domains. Each knowledge system is associated with a database (DB) and a knowledge base (KB), both of which can be accessed by a set of operators. The operators are divided into three classes:

- *DBMOs:* Data Management Operators, which operate on the database. These are conventional data management operators that are used for creating, modifying and displaying relational tables.

- *KBMOs:* Knowledge Management Operators, which operate on the knowledge base. These operators play a similar role to the DBMOs, but apply to the rules and other structures in the knowledge base.

- *KGOs*: Knowledge Generation Operators, which operate on both the data and knowledge bases. These operators perform symbolic and numerical data exploration tasks. They are based both on various machine learning and inference programs and on conventional data exploration techniques.

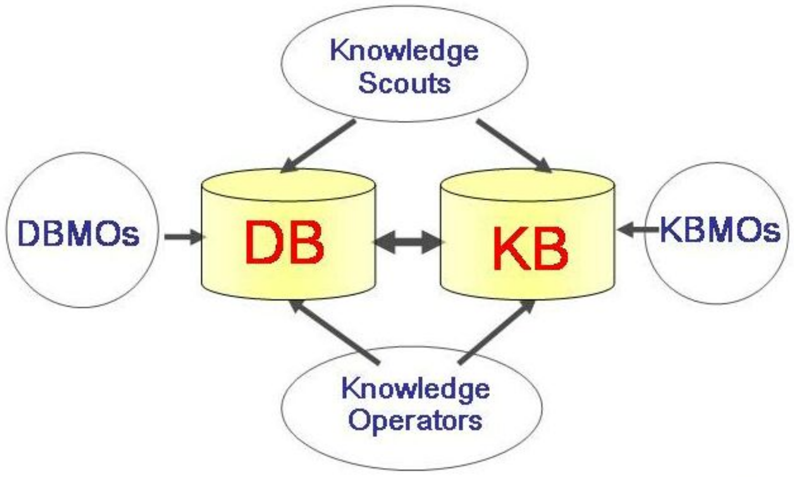

Figure 6 A general schema of the INLEN inductive database system.

The execution of a KGO usually requires some background knowledge, and is guided by control parameters (if some parameters are not specified, default values are used). The background knowledge may contain some general knowledge, previously discovered knowledge, and knowledge specifically relevant to a given application domain, such as a specification of the value sets and types of attributes, the constraints and relationships among attributes, initial rules hypothesized by an expert, etc. The KGOs can be classified into groups, based on the type of operation they perform, each of which includes a number of specific operators that are instantiated by a combination of parameters. For example one group consists of operators for learning decision rules, another for selecting attributes, another for applying knowledge, and so forth.

One INLEN operator extends beyond the traditional learning and discovery operators, and thus merits further discussion. Specifically, the *Scout Access* operator is used to build *knowledge scouts* – scripts to serve as intelligent agents performing discovery tasks in the inductive database (Michalski and Kaufman, 2000).

To explore the idea of a knowledge scout further, consider the task of data exploration and knowledge mining. Typically, the entire plan of discovery can not be determined in its entirety beforehand. Some results may require no action, others may require some action to be taken, and occasionally, some may warrant a new and completely unplanned course of action. Yet it is time-consuming and subject to errors for an analyst to stand over every stage of the discovery process and respond appropriately to the output from each of those steps. Thus, the idea of a knowledge scout is that of a mechanism that can encapsulate the user's knowledge of how to react to different contingencies.

For instance, an experiment based on 1993 World Factbook data found the following rule describing 25 of the 55 countries with low (<1%) population growth:

PopGrRate < 1% if:	pos	neg
1. BirthRate is 10..20 or 50..60	46	20
2. FertRate is 1..2 or > 7	32	17
3. Religion is Protestant or Roman_Catholic or Eastern_Orth or Shinto or Bulgarian_Orth or Russian_Orth or Romanian_Orth or Greek_Orth	38	32
4. NetMigRate <= +10	54	123

The first and strongest condition is surprising. Birth rates ranged in the data from 10 to 60, and while the low birth rate is intuitive, the very high one is not. Looking at the 25 countries that satisfy the rule, 24 of them had birth rates less than 20. Only one country, Malawi, had a birth rate above 50. Such a counterintuitive result could instigate a series of experiments to determine the cause of such behavior. In fact, subsequent investigation of Malawi compared to the rest of the countries quickly turned up an explanation: an outward net migration rate that dwarfs those of all the other countries.

Thus, a goal of knowledge scout would be to be able to specify in a script anomalies to be detected and what should be done in response to them (either logging them for a human, or calling upon new discovery operators). A knowledge scout needs the means to specify its plan of action, specifically, a language rich enough to specify the operators available to it and the means to select and execute actions. For instance, M-SQL extends the SQL data query language by adding to it the ability to query for certain types of rules and to invoke an association rule generating operator (Imielinski, Virmani, and Abdulghani, 1996). Thus, it has access to conventional database operators plus a data mining operator.

Built for the INLEN environment which contained a variety of learning and discovery operators, each of which offered a range of parameter settings, KGL was designed as a language for specifying detailed plans in such an environment (Michalski and Kaufman, 2000). By combining the means for specifying the steps to be taken and the means for specifying the control structure, KGL code, such as the set of instructions shown in Figure 7, could be provided to the program. In that figure, the program is asked to examine the PEOPLE data table extracted from the CIA's World Factbook, and to take action based on the rules it finds. Comments (bracketed and in italics) have been added to explain each line.

```
open PEOPLE                               {Select PEOPLE database}
do CHAR(decision=all, pfile=people1.lrn)  {Characterize concepts
                                          representing single values of
                                          all attributes, using parameters
                                           specified in file people1.lrn}
strongPGrules1 = #rules(PGR, compl >= 60)  {Count rules for Population}
strongPGrules2 = #rules(PGR, supp >= 25)   {Growth Rate that satisfy}
strongPGrules3 = #rules(PGR,               {three different conditions}
   num_conds(cons >= 50 and supp > 10) > 2){for threshold of strength}
```

```
print "Number of strong PGR rules:
   Type 1 = ", strongPGrules1, ",
   Type 2 = ", strongPGrules2, ",
   Type 3 = ", strongPGrules3
if #conditions(Fert) > 150                 {Is Fert ruleset too complex?}
  begin
  do SELECT(attributes, decision=Fert,
     thresh=4, out=PEOPLE2, criterion=max) {If so, find "thresh" best}
  do CHAR(pfile=people1.lrn, decision=Fert){independent attributes, then
  end                                      {recharacterize}
for i = 1 to 6
begin                                      {For each value of i, 1-6,}
print "Number of LE conditions with p/n    {count & display number of}
       ratio of at least", i, ":1 =",      {Life Expectancy conditions}
      #conditions(LE, cons >= i/(i+1))     {with consistency • i/(i+1)}
end
```

Figure 7 KGL code defining a knowledge scout for exploring a World Factbook data table.

The presented script learns rules for each possible decision attribute, then executes three tasks. First it counts the number of rules whose consequent predicts a country's population growth rate that are strong according to three criteria: high relative support, high absolute support, and containing at least two conditions that have both absolute support and standalone consistency above the given thresholds. Then it tests the ruleset that determines a country's likely fertility rate for complexity (based on the total number of conditions in the ruleset); if it is too complex, the four most relevant attributes to the task are chosen, and the learning is repeated on this streamlined dataset. Finally, it reports on the number of conditions in the life expectancy rule base that have confidence levels above different thresholds.

In summary, an inductive database integrates a database with a set of operators for performing various types of operations on the data base, on the knowledge base, or on the data and knowledge bases combined.

6 SUMMARY

The main thesis of this chapter is that modern methods for symbolic machine learning have a direct and important application to logical data analysis and the development of a new research direction, called knowledge mining. Knowledge mining has been characterized as a derivation of human-like knowledge from data and prior knowledge. It was indicated that a knowledge mining system can be implemented using inductive database technology that deeply integrates a database, a knowledge base, and operators for data and knowledge management and knowledge generation.

Among knowledge generation operators are operators for inductive learning of attributional rules or trees characterizing the relationship between designated output and input attributes, for creating conceptual hierarchies (conceptual clustering) from data, for selecting most relevant attributes, and for visualizing data and rules learned from the data. The learned rules represent high-level knowledge that can be of great value to a data analyst, and used for human decision-making or for automated classification. Other important operators include construction of equations along with logical preconditions for their application, determination of symbolic descriptions of temporal sequences of multi-attribute events, automated generation of new, more relevant attributes, and selection of representative examples.

The underlying theme of all these methods is the ability to generate knowledge that is easily understood and articulated. Visualization techniques such as concept association graphs facilitate the presentation of broad concepts to the user.

The knowledge generation capability was illustrated by presenting results from several application domains. In analyzing demographic data, the knowledge mining approach helped to discover the anomalous Malawi's population changes. In analyzing medical data, it showed in understandable terms relationships between the occurrences of diseases and the presence or absence of other diseases as well as factors in individuals' lifestyles. And in a computer intrusion detection domain, it created symbolic user models from process table data characterizing users' activities (Michalski et al, 2004).

In contrast to many data mining approaches, the methodology presented can utilize various types of background knowledge regarding the domain of discourse. This background knowledge may include, for example, a specification of the domain and the type of the attributes, the known relationships among attributes, prior concept descriptions, and other high-level knowledge. An important aspect of the methodology is its ability to take advantage of this knowledge.

We presented KGL as a language for developing knowledge scouts. KGL was designed for an environment in which the data was not stored in a full-scale DBMS. VINLEN, which works in conjunction with an SQL-accessible relational database, requires a knowledge scout language that is tailored to such an environment. Thus, one topic of ongoing research is the development of a *Knowledge Query Language* – a language of SQL-like form that extends its capabilities into a knowledge mining paradigm.

ACKNOWLEDGMENTS

The authors thank their past and current collaborators who provided research feedback, comments, and assistance at different stages of the development of methods and computer programs described in this chapter. We are grateful to the students and researchers who contributed in a constructive and helpful way in this research over the years, in particular, to Jerzy Bala, Eric Bloedorn, Michal Draminski, Mark Maloof, Jarek Pietrzykowski, Jim Ribeiro, Bartlomiej Sniezynski, Janusz Wnek, Janusz Wojtusiak, and Qi Zhang.

This research was conducted in the Machine Learning and Inference Laboratory of George Mason University. Support for the Laboratory's research related to the presented results has been provided in part by the National Science Foundation under Grants No. DMI-9496192, IRI-9020266, IIS-9906858 and IIS-0097476; in part by the UMBC/LUCITE #32 grant; in part by the Office of Naval Research under Grant No. N00014-91-J-1351; in part by the Defense Advanced Research Projects Agency under Grant No. N00014-91-J-1854 administered by the Office of Naval Research; and in part by the Defense Advanced Research Projects Agency under Grants No. F49620-92-J-0549 and F49620-95-1-0462 administered by the Air Force Office of Scientific Research.

The findings and opinions expressed here are those of the authors, and do not necessarily reflect those of the above sponsoring organizations.

REFERENCES

Aggarwal, C.C., Procopiuc, C., Wolf, J. Yu, P.S. and Park, J.S., "Fast Algorithms for Projected Clustering," *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pp. 61-72, 1999.

Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P., "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pp. 94-105, 1998.

Alexe, S., Blackstone, E, and Hammer, P., "Coronary Risk Prediction by Logical Analysis of Data," *Annals of Operations Research*, 119, pp. 15-42, 2003.

Baim, P.W., "The PROMISE Method for Selecting Most Relevant Attributes for Inductive Learning Systems," Report No. UIUCDCS-F-82-898, Department of Computer Science, University of Illinois, Urbana, 1982.

Belkin, M. and Niyogi, P., “Semi-supervised Learning on Riemannian Manifolds,” *Machine Learning*, **56**, 209-239, , 2004

Bentrup, J.A., Mehler, G.J. and Riedesel, J.D., “INDUCE 4: A Program for Incrementally Learning Structural Descriptions from Examples,” *Reports of the Intelligent Systems Group*, ISG 87-2. UIUCDCS-F-87-958, Department of Computer Science, University of Illinois, Urbana, 1987.

Bergadano, F., Matwin, S., Michalski, R.S. and Zhang, J., “Learning Two-Tiered Descriptions of Flexible Concepts: The POSEIDON System,” *Machine Learning*, 8, pp. 5-43, 1992.

Bloedorn, E., Mani, I. and MacMillan, T.R., “Machine Learning of User Profiles: Representational Issues,” *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, 1996.

Bloedorn, E. and Michalski, R.S., “Data-Driven Constructive Induction,” *IEEE Intelligent Systems, Special Issue on Feature, Transformation and Subset Selection,* pp. 30-37, March/April, 1998.

Bloedorn, E., Wnek, J. and Michalski, R.S., “Multistrategy Constructive Induction.,” *Proceedings of the Second International Workshop on Multistrategy Learning*, Harpers Ferry, WV, pp. 188-203, 1993.

Bongard, N., *Pattern Recognition*, Spartan Books, New York (a translation from Russian), 1970.

Brachman, R.J., Khabaza, T., Kloesgen, W., Piatetsky-Shapiro, G. and Simoudis, E., “Mining Business Databases,” *Communications of the ACM*, 39:11, pp. 42-48, 1996.

Bratko, I., Muggleton, S. and Karalic, A., “Applications of Inductive Logic Programming,” in Michalski, R.S., Bratko, I. and Kubat, M. (eds.), *Machine Learning and Data Mining: Methods and Applications*, London, John Wiley & Sons, 1997.

Carpineto, C. and Romano, G., “Some Results on Lattice-based Discovery in Databases,” *Workshop on Statistics*, *Machine Learning and Knowledge Discovery in Databases*, Heraklion, pp. 216-221, 1995a.

Carpineto, C. and Romano, G., “Automatic Construction of Navigable Concept Networks Characterizing Text Databases,” in Gori, M. and Soda, G. (eds.), *Topics in Artificial Intelligence*, LNAI 992-Springer-Verlag, pp. 67-78, 1995b.

Caruana, R. and Freitag, D., “Greedy Attribute Selection,” *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 28-36, 1994.

Cavalcanti, R.B., Guadagnin, R., Cavalcanti, C.G.B., Mattos, S.P. and Estuqui, V.R., “A Contribution to Improve Biological Analyses of Water Through Automatic Image Recognition,” *Pattern Recognition and Image Analysis,* 7:1, pp. 18-23, 1997.

Clark, P. and Niblett, T., “The CN2 Induction Algorithm,” *Machine Learning*, 3, pp. 261-283, 1989.

Cohen, W., “Fast Effective Rule Induction,” *Proceedings of the 12th International Conference on Machine Learning*, 1995

Daniel, C. and Wood, F.S., *Fitting Equations to Data*, New York, John Wiley & Sons, 1980.

Davis, J., “CONVART: A Program for Constructive Induction on Time-Dependent Data,” M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, 1981.

De Raedt., L., Jaeger, M., Lee, S.D. and Mannila, H., “A Theory of Inductive Query Answering,” *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM’02)*, pp. 123-130, 2002.

Dieterrich, T. and Michalski, R.S., “Learning to Predict Sequences,” in Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), *Machine Learning: An Artificial Intelligence Approach Vol. 2*, Morgan Kaufmann, pp. 63-106, 1986.

Diday, E. (ed.), *Proceedings of the Conference on Data Analysis, Learning Symbolic and Numeric Knowledge*, Nova Science Publishers, Inc., Antibes, 1989.

Dontas, K., “APPLAUSE: An Implementation of the Collins–Michalski Theory of Plausible Reasoning,” M.S. Thesis, Computer Science Department, The University of Tennessee, Knoxville, TN, 1988.

Dubois, D., Prade, H. and Yager, R.R. (eds.), *Readings in Fuzzy Sets and Intelligent Systems*, Morgan Kaufmann, 1993.

Elio, R. and Watanabe, L., “An Incremental Deductive Strategy for Controlling Constructive Induction in Learning from Examples,” *Machine Learning* 7: pp. 7-44, 1991.

Evangelos S. and Han, J. (eds.), *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR, 1996.

Falkenhainer, B.C. and Michalski, R.S., “Integrating Quantitative and Qualitative Discovery in the ABACUS System, in Kodratoff, Y. and Michalski, R.S. (eds.), *Machine Learning: An Artificial Intelligence Approach Vol. III*, San Mateo, CA, Morgan Kaufmann, pp. 153-190, 1990.

Fayyad, U.M., Haussler, D. and Stolorz, P., “Mining Scientific Data,” *Communications of the ACM*, 39:11, pp. 51-57, 1996.

Fayyad, U.M. and Irani, K.B., “The Attribute Selection Problem in Decision Tree Generation,” *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA, pp. 104-110, 1992

Fayyad, U.M. Piatetsky-Shapiro, G. Smyth, P. and Uhturusamy, R. (eds.), *Advances in Knowledge Discovery and Data Mining*, San Mateo, CA, AAAI Press, 1996.

Feelders, A., “Learning from Biased Data Using Mixture Models,” *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR, pp. 102-107, 1996.

Fürnkranz, J. and Flach, P., "An Analysis of Rule Evaluation Metrics," *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, pp. 202-209, 2003.

Greene, G., "The Abacus.2 System for Quantitative Discovery: Using Dependencies to Discover Non-Linear Terms," Reports *of the Machine Learning and Inference Laboratory*, MLI 88-4, Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, 1988.

Han, J., Fu,, Y., Wang, W., Chiang, J., Gong, W., Koperski, K., Li, D., Lu, Y., Rajan, A., Stefanovic, N., Xia, B. and Zaiane, O.R., "DBMiner: A System for Mining Knowledge in Large Relational Databases,". *Proceedings of the Second International Conference on Data Mining and Knowledge Discovery (KDD'96)*, pp. 250-255, 1996.

Han, J. and Kamber, M., *Data Mining: Concepts and Techniques*, San Francisco: Morgan Kaufmann, 2001.

Hand, D., Mannila, H. and Smyth, P., *Principles of Data Mining*, Cambridge, MA: MIT Press, 2001.

Iba, W., Woogulis, J. and Langley, P., "Trading Simplicity and Coverage in Incremental Concept Learning," *Proceedings of the Fifth International Conference on Machine Learning*, San Francisco: Morgan Kaufmann, pp. 73-79, 1988.

Imam, I.F. and Michalski, R.S., "Should Decision Trees be Learned from Examples or from Decision Rules?" *Proceedings of the Seventh International Symposium on Methodologies for Intelligent Systems (ISMIS-93)*, Trondheim, Norway, 1993.

Imielinski, T. and Mannila, H., "A Database Perspective on Knowledge Discovery," *Communications of ACM*, 39: pp. 58-64, 1996.

Imielinski, T., Virmani, A. and Abdulghani, A., "DataMine: Application Programming Interface and Query Language for Database Mining," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 256-261, 1996.

Kaufman, K.A. and Michalski, R.S., "A Method for Reasoning with Structured and Continuous Attributes in the INLEN-2 Multistrategy Knowledge Discovery System," *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR, pp. 232-237, 1996.

Kaufman K. and Michalski R.S., "Learning from Inconsistent and Noisy Data: The AQ18 Approach," *Proceedings of the Eleventh International Symposium on Methodologies for Intelligent Systems*, Warsaw, pp. 411-419, 1999.

Kaufman K. and Michalski, R.S., "A Knowledge Scout for Discovering Medical Patterns: Methodology and System SCAMP," *Proceedings of the Fourth International Conference on Flexible Query Answering Systems, FQAS'2000*, Warsaw, Poland, pp. 485-496, 2000.

Kaufman K. and Michalski R.S., "The Development of the Inductive Database System VINLEN: A Review of Current Research," *International Intelligent Information Processing and Web Mining Conference*, Zakopane, Poland, 2003.

Kerber, R., “Chimerge: Discretization for Numeric Attributes,” *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, AAAI Press, pp. 123-128, 1992.

Khabaza, T. and Shearer, C., “Data Mining with Clementine,” *Colloquium on Knowledge Discovery in Databases*, The Institution of Electrical Engineers, 1995.

Kibler, D. and Aha, D., “Learning Representative Exemplars of Concepts: A Case Study,” *Proceedings of the Fourth International Conference on Machine Learning*, San Francisco: Morgan Kaufmann, pp. 24-30, 1987.

Kokar, M.M., “Coper: A Methodology for Learning Invariant Functional Descriptions,” in Michalski, R.S., Mitchell, T.M and Carbonell, J.G. (eds.), *Machine Learning: A Guide to Current Research*, Kluwer Academic, Boston, MA 1986.

Lakshminarayan, K., Harp, S.A., Goldman, R. and Samad, T., “Imputation of Missing Data Using Machine Learning Techniques.” *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland, OR, pp. 140-145, 1996.

Langley, P., “Selection of Relevant Features in Machine Learning,” *AAAI Fall Symposium on Relevance*, pp. 140-144, 1994.

Langley, P., Bradshaw G.L. and Simon, H.A., “Rediscovering Chemistry with the BACON System,” in Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann, San Mateo, CA, pp. 307-329, 1983.

Larson, J.B., “INDUCE-1: An Interactive Inductive Inference Program in VL21 Logic System,” Report No. 876, Department of Computer Science, University of Illinois, Urbana, 1977.

Lbov, G.S., *Mietody Obrabotki Raznotipnych Ezperimentalnych Danych (Methods for Analysis of Multitype Experimental Data)*, Akademia Nauk USSR, Sibirskoje Otdielenie, Institut Matiematiki, Izdatielstwo Nauka, Novosibirsk, 1981.

Maloof, M.A. and Michalski, R.S, “Selecting Examples for Partial Memory Learning,” *Machine Learning*, 41, pp. 27-52, 2000.

Maloof, M. and Michalski R.S., “Incremental Learning with Partial Instance Memory,” *Artificial Intelligence*, 154, 95-126, 2004.

Michael, J., “Validation, Verification and Experimentation with Abacus2,” *Reports of the Machine Learning and Inference Laboratory*, MLI 91-8, Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, 1991.

Michalski, R.S., “A Planar Geometrical Model for Representing Multi-Dimensional Discrete Spaces and Multiple-Valued Logic Functions,” ISG Report No. 897, Department of Computer Science, University of Illinois, Urbana, 1978.

Michalski, R.S., “A Theory and Methodology of Inductive Learning,” *Artificial Intelligence*, 20, pp. 111-161, 1983.

Michalski, R.S., "Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-tiered Representation, in Kodratoff, Y. and Michalski, R.S. (eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. III,* San Mateo, CA, Morgan Kaufmann, pp. 63-102, 1990.

Michalski, R.S., "Inferential Theory of Learning: Developing Foundations for Multistrategy Learning," in Michalski, R.S. and Tecuci, G. (eds.), *Machine Learning: A Multistrategy Approach*, San Francisco, Morgan Kaufmann, pp. 3-61, 1994.

Michalski, R.S. "LEARNABLE EVOLUTION MODEL: Evolutionary Processes Guided by Machine Learning," *Machine Learning*, 38, pp 9-40, 2000.

Michalski, R.S., "Attributional Calculus: A Logic and Representation Language for Natural Induction," *Reports of the Machine Learning and Inference Laboratory*, MLI 04-2, George Mason University, Fairfax, VA, 2004.

Michalski, R.S., Baskin, A.B. and Spackman, K.A., "A Logic-based Approach to Conceptual Database Analysis," *Sixth Annual Symposium on Computer Applications in Medical Care* (SCAMC-6), George Washington University, Medical Center, Washington, DC, pp. 792-796, 1982.

Michalski, R.S. and Kaufman, K.A., "Multistrategy Data Exploration Using the INLEN System: Recent Advances," *Sixth Symposium on Intelligent Information Systems (IIS '97)*, Zakopane, Poland, 1997.

Michalski, R.S. and Kaufman, K.A., "Data Mining and Knowledge Discovery: A Review of Issues and a Multistrategy Approach," in Michalski, R.S., Bratko, I. and Kubat, M. (Eds.), *Machine Learning and Data Mining: Methods and Applications*, pp. 71-112, London: John Wiley & Sons, 1998.

Michalski R.S. and Kaufman K., "Building Knowledge Scouts Using KGL Metalanguage," Fundamenta Informaticae , 40, pp 433-447, 2000.

Michalski R.S. and Kaufman K., "Learning Patterns in Noisy Data: The AQ Approach," *Machine Learning and its Applications*, Paliouras, G., Karkaletsis, V. and Spyropoulos, C. (Eds.), pp. 22-38, Springer-Verlag, 2001.

Michalski, R.S., Kaufman, K., Pietrzykowski, J. Wojtusiak, J. and Sniezynski, B., "Learning User Behavior and Understanding Style: A Natural Induction Approach," *Reports of the Machine Learning and Inference Laboratory*, George Mason University, Fairfax, VA, 2005 (to appear).

Michalski, R.S., Kerschberg, L., Kaufman, K. and Ribeiro, J., "Mining for Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results," *Journal of Intelligent Information Systems: Integrating AI and Database Technologies*, 1, pp. 85-113, 1992.

Michalski, R. S., Ko, H. and Chen, K., "SPARC/E(V.2), An Eleusis Rule Generator and Game Player," *Reports of the Intelligent Systems Group*, ISG No. 85-11, UIUCDCS-F-85-941, Department of Computer Science, University of Illinois, Urbana, 1985.

Michalski, R.S., Ko, H. and Chen, K., “Qualitative Prediction: A Method and a Program SPARC/G,” in Guetler, C. (ed.), *Expert Systems*, London, Academic Press, 1986.

Michalski, R.S. and Larson, J.B., “Selection of Most Representative Training Examples and Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of Programs ESEL and AQ11,” Report No. 867, Department of Computer Science, University of Illinois, Urbana, 1978.

Michalski R.S. and Larson, J., “Incremental Generation of VL1 Hypotheses: The Underlying Methodology and the Description of Program AQ11,” *Reports of the Intelligent Systems Group, ISG 83-5, UIUCDCS-F-83-905*, Department of Computer Science, University of Illinois, Urbana, 1983.

Michalski, R.S., Rosenfeld, A., Duric, Z., Maloof, M. and Zhang, Q., “Application of Machine Learning in Computer Vision,” in Michalski, R.S., Bratko, I. and Kubat, M. (eds.), *Machine Learning and Data Mining: Methods and Applications*, London, John Wiley & Sons, 1998.

Michalski, R. S. and Stepp, R., “Learning from Observation: Conceptual Clustering,” in *Machine Learning: An Artificial Intelligence Approach*, R.S. Michalski, J. G. Carbonell, and T. M. Mitchell (eds.), Palo Alto, CA, Tioga Publishing, 1983.

Morgenthaler, S. and Tukey, J.W., “The Next Future of Data Analysis,” in Diday, E. (ed.), *Proceedings of the Conference on Data Analysis, Learning Symbolic and Numeric Knowledge*, Nova Science Publishers, Antibes, 1989.

Muggleton, S. (ed.), *Inductive Logic Programming*, Morgan Kaufmann, 1992.

Neapolitan, R.E., *Learning Bayesian Networks*, Prentice Hall, 2003.

Pawlak, Z., *Rough Sets: Theoretical Aspects of Reasoning about Data*, Kluwer Academic, Dordrecht, 1991.

Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* San Mateo, CA: Morgan Kaufmann, 1988.

Pearl, J., *Causality: Models, Reasoning and Inference*, Cambridge University Press, 2000.

Quinlan, J.R., “Induction of Decision Trees,” *Machine Learning*, 1, pp. 81-106, 1986.

Quinlan, J.R., “Probabilistic Decision Trees,” in Kodratoff, Y. and Michalski, R.S. (eds.), *Machine Learning: An Artificial Intelligence Approach, Volume III*, Morgan Kaufmann, San Mateo, CA, pp. 140-152, 1990.

Quinlan, J.R., *C4.5: Programs for Machine Learning*, Morgan Kaufmann, Los Altos, CA, 1993.

Reinke, R.E. and Michalski, R.S., “Incremental Learning of Concept Descriptions: A Method and Experimental Results,” *Machine Intelligence*, 11, pp. 263-288, 1988.

Ribeiro, J.S., Kaufman, K.A. and Kerschberg, L., “Knowledge Discovery From Multiple Databases,” *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, Montreal, PQ, pp. 240-245, 1995.

Saul, L.K. and Roweis, S.T., “Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifolds,” *Journal of Machine Learning Research*, 4, pp. 119-155, 2003.

Sharma, S., *Applied Multivariate Techniques*, London, John Wiley & Sons, 1996.

Slowinski, R. (ed.), *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Dordrecht/Boston/London, Kluwer Academic, 1992.

Stepp, R. and Michalski R.S., “Conceptual Clustering: Inventing Goal-Oriented Classifications of Structured Objects,” in Michalski, R.S., Carbonell, J. and Mitchell, T.M. (eds.), *Machine Learning: An Artificial Intelligence Approach, Vol. II*, Morgan-Kaufmann Publishers, 1986.

Tenenbaum, J.B., de Silva, V. and Langford, J.C., “A Global Geometric Framework for Nonlinear Dimensionality Reduction,” *Science*, 290, pp. 2319-2323, 2000.

Tukey, J.W., *The Collected Works of John W. Tukey, Vol. V, Philosophy and Principles of Data Analysis: 1965-1986*, Jones, L.V. (ed.), Wadsworth & Brooks/Cole, Monterey, CA, 1986.

Umann, E., “Phons in Spoken Speech: A Contribution to the Computer Analysis of Spoken Texts,” *Pattern Recognition and Image Analysis*, 7:1, pp. 138-144, 1997.

Van Mechelen, I., Hampton, J., Michalski, R.S. and Theuns, P. (eds.), *Categories and Concepts: Theoretical Views and Inductive Data Analysis*, London, Academic Press, 1993.

Wang, H., Chu, F., Fan, W. Yu, P.S. and Pei, J., “A Fast Algorithm for Subspace Clustering by Pattern Similarity,” *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM'04)*, Santorini Island, Greece, 2004.

Widmer, G. and Kubat, M., “Learning in the Presence of Concept Drift and Hidden Concepts,” *Machine Learning*, 23, pp. 69-101, 1996.

Wnek, J. and Michalski, R.S., “Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments,” *Machine Learning*, 14, pp. 139-168, 1994.

Wojtusiak, J., “AQ21 User’s Guide,” *Reports of the Machine Learning and Inference Laboratory*, MLI 04-3, George Mason University, Fairfax, VA, September, 2004..

Zadeh, L., “Fuzzy Sets,” *Information and Control*, 8, pp. 338-353, 1965.

Zagoruiko, N.G., *Recognition Methods and Their Application*, Sovietsky Radio, Moscow (in Russian), 1972.

Zagoruiko, N.G., “Ekspertnyie Sistemy I Analiz Dannych (Expert Systems and Data Analysis),” *Wychislitielnyje Sistemy*, N.144, Akademia Nauk USSR, Sibirskoje Otdielenie, Institut Matiematikie, Novosibirsk, 1991.

Zhang, Q., “Knowledge Visualizer: A Software System for Visualizing Data, Patterns and Their Relationships,” *Reports of the Machine Learning and Inference Laboratory*, MLI 97-14, George Mason University, Fairfax, VA, 1997.

Zhuravlev, Y.I. and Gurevitch, I.B., “Pattern Recognition and Image Recognition,” in Zhuravlev, Y.I. (ed.), *Pattern Recognition, Classification, Forecasting: Mathematical Techniques and their Application*. *Issue 2*, Nauka, Moscow, pp. 5-72 (in Russian), 1989.

Ziarko, W.P. (ed.), *Rough Sets*, *Fuzzy Sets and Knowledge Discovery*, Berlin, Springer-Verlag, 1994.

Zytkow, J.M., “Combining Many Searches in the FAHRENHEIT Discovery System,” *Proceedings of the Fourth International Workshop on Machine Learning*, Irvine, CA, pp. 281-287, 1987.