KNOWLEDGE DISCOVERY IN DATABASES : AN ATTRIBUTE-ORIENTED ROUGH SET APPROACH

A Thesis
Submitted to the Faculty of Graduate Studies and Research
In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy
in
Computer Science
Faculty of Graduate Studies
University of Regina

By
Xiaohua Hu
Regina, Saskatchewan
June, 1995

Abstract

Knowledge Discovery in Databases (KDD) is an active research area with the promise for a high payoff in many business and scientific applications. The grand challenge of knowledge discovery in databases is to automatically process large quantities of raw data, identify the most significant and meaningful patterns, and present this knowledge in an appropriate form for achieving the user's goal. Knowledge discovery systems face challenging problems from the real-world databases which tend to be very large, redundant, noisy and dynamic. Each of these problems has been addressed to some extent within machine learning, but few, if any, systems address them all. Collectively handling these problems while producing useful knowledge efficiently and effectively is the main focus of the thesis. In this thesis, we develop an attribute-oriented rough set approach for knowledge discovery in databases. The method adopts the artificial intelligent "learning from examples" paradigm combined with rough set theory and database operations. The learning procedure consists of two phases: data generalization and data reduction. In data generalization, our method generalizes the data by performing attribute-oriented concept tree ascension, thus some undesirable attributes are removed and a set of tuples may be generalized to the same generalized tuple. The generalized relation contains only a small number of tuples, which substantially reduces the computational complexity of the learning process and, furthermore, it is feasible to apply the rough set techniques to eliminate the irrelevant or unimportant attributes and choose the "best" minimal attribute set. The goal of data reduction is to find a minimal subset of interesting attributes that have all the essential information of the generalized relation; thus the minimal subset of the attributes can be used rather than the entire attribute set of the generalized relation. By removing those attributes which are not important and/or essential , the rules generated are more concise and efficacious.

Our method integrates a variety of knowledge discovery algorithms, such as DBChar for deriving characteristic rules, DBClass for classification rules, DBDeci for decision rules, DBMaxi for maximal generalized rules, DBMkbs for multiple sets of knowledge rules and DBTrend for data trend regularities, which permit a user to discover various kinds of relationships and regularities in the data. This integration inherit the advantages of the attribute-oriented induction model and rough set theory. Our method makes some contribution to the KDD. A generalized rough set model is formally defined with the ability to handle statistical information and also consider the importance of attributes and objects in the databases. Our method is able to identify the essential subset of nonredundant attributes (factors) that determine the discovery task, and can learn different kinds of knowledge rules efficiently from large databases with noisy data and in a dynamic environment and deal with databases with incomplete information. A prototype system DBROUGH was constructed under a Unix/C/Sybase environment. Our system implements a number of novel ideas. In our system, we use attribute-oriented induction rather than tuple-oriented induction, thus greatly improving the learning efficiency. By integrating rough set techniques into the learning procedure, the derived knowledge rules are particularly concise and pertinent, since only the relevant and/or important attributes (factors) to the learning task are considered. In our system, the combination of transition network and concept hierarchy provides a nice mechanism to handle dynamic characteristic of data in the databases. For applications with noisy data, our system can generate multiple sets of knowledge rules through a decision matrix to improve the learning accuracy. The experiments using the NSERC information system illustrate the promise of attribute-oriented rough set learning for knowledge discovery in databases.

Acknowledgments

This thesis has been researched and written under the supportive and helpful direction of my supervisor Dr. Nick Cercone. To whom I owe a debt of gratitude for the encouragement given in the undertaking of this work. I thank Nick Cercone for making these several years as his student enjoyable and challenging, and for his excellent guidance and financial support, and for the many conversations which brought the benefit of his wealth of knowledge in artificial intelligence and knowledge discovery in databases to my chosen research area.

I would like to thank all the members of my committee for their feedback and careful readings of the thesis, which lead to many improvements in the presentation: Thanks Christine Chan, Larry Saxton, Paitoon Tontiwachwuthikul, and Wojciech Ziarko. Thanks also to Dr. Randy Goebel as my external examiner.

I am grateful to Mr. Ning Shan for his friendship, valuable and fruitful discussions and very good corporation in our joint research work.

It should be mentioned that my study at the University of Regina was not only beneficial, but also very enjoyable. Special thanks go to Ms. Aijun An who took care of my mails while I was working in Ottawa. My sincere thanks are due to the many friendly and helpful people including Dr. Brien Maguire, Dr. S.K.M. Wong, Dr. Xiang Yang, Margaret Cooper, Zhiwei Wang, Lida Yang.

I am grateful to my wife, Michelle Shuet-yue Tsang who during the writing of this work helped, encouraged and supported me when times were tough. Without her love and support, I would not have had the ability to continue when things appeared to stop.

I wish to thank Jesus Cordoba for helpful comments in proofreading the work.

Financial support for the research in this thesis was gratefully received from the Natural Science and Engineering Research Council of Canada, Networks of Centres of Excellence of the Government of Canada, Saskatchewan Open Scholarship, Faculty of Graduate Studies of University of Regina, the Herzerberg Fellowship Foundation.

There are two supremely honourable people in this world, to whom all my writing has always been dedicated: to my parents, Ms. Chuanhui Wang, Mr. Zhikun Hu, this work is for you.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

Knowledge discovery is the process of mining a data source for information that one is unaware of prior to the discovery. This spans the entire spectrum from discovering information of which one has absolutely no knowledge to where one merely confirms a well known fact.

Knowledge Discovery in Databases (KDD) is an active research area with promise for high payoffs in many business and scientific applications. The corporate, governmental, and scientific communities are being overwhelmed with an influx of data that is routinely stored in on-line databases. Analyzing this data and extracting meaningful patterns in a timely fashion is intractable without computer assistance and powerful analytical tools. Standard computer-based statistical and analytical packages alone, however, are of limited benefit without the guidance of trained statisticians to apply them correctly and the domain experts to filter and interpret the results [MCP93]. Data mining has been ranked as one of the most promising topics for research for the 1990s by both database and machine learning researchers [SSU91].

William Frawley and his colleague [FPM91] give a definition of knowledge as follows:

"Given a set of facts (data) F, a language L, and some measure of certainty C, a *pattern* is defined as a statement S in L that describes relationships among a subset F_s of F with a certainty c, such that S is simpler (in some sense) than the enumeration of all facts in F_s. A pattern that is interesting (according to a user-imposed interest measure) and certain enough (again according to the user's criteria) is called *knowledge*."

This definition about the language, the certainty, and the simplicity and interestness measure are intentionally vague to cover a wide variety of approaches. Collectively, these terms encapsulate our view of the fundamental characteristics of discovery in databases.

Many machine-learning algorithms are readily applicable for KDD. An important machine learning paradigm, *learning from examples*, that is, learning by generalizing specific facts or observations [CoF83, DiM83], has been adopted in many existing induction learning algorithms. Real-world databases present additional considerations due to the nature of their contents which tend to be large, incomplete, dynamic, noisy and redundant. Each of these considerations have been addressed, to some extent, within machine learning, but few, if any, systems address all of them. Collectively handling these problems while producing useful knowledge is the challenge of KDD.

One of the major reasons that the machine learning systems do not integrate well with relational database systems is because of the inefficiency of current learning algorithms when applied to large databases. Most existing algorithms for *learning from examples* apply a tuple-oriented approach, an approach that examines one tuple at a time. In order to discover the most specific concept that is satisfied by all the training examples, the tuple-oriented approach must test the concept coverage after each generalization on a single attribute value of a training example [DiM83,Mic83]. Since there are a large number of possible combinations in such testing, the tuple-oriented approach is quite inefficient when performing learning from large databases. Moreover, most existing algorithms do not make use of the features and implementation techniques provided by database systems. To make learning algorithms applicable to database systems, highly efficient algorithms should be designed and explored in depth.

In many practical applications, during the data collection procedure, it is often difficult to know exactly which features are relevant and/or important for the learning task, and how they should be represented. So all features believed to be useful are collected into the database. Hence databases usually contain some attributes that are undesirable, irrelevant, or unimportant to a given discovery task, focussing on a subset of attributes is now common practice. Identifying relevant fields is the most common focussing technique.

In previous studies in [CCH91, HCC92, HCH93, HCH94], an attribute-oriented induction method has been developed for knowledge discovery in relational databases. The method integrates a machine learning paradigm, especially *learning from examples* techniques, with database operations. The general idea of basic attribute-oriented induction is performed attribute by attribute using attribute removal and concept ascension. As a result, undesirable attributes may be removed and different tuples may be generalized to identical ones, and the final generalized relation may consist of only a small number of distinct tuples. Then the method transforms the final generalized relation into logical rules. In the final generalized relation, all attributes are treated as equally important. But this is not true in actuality. The generalized relation normally will still contain some irrelevant, or unimportant attributes for a given discovery task. For example, to determine the mileage of a car, the weight and power of the car are much more important attributes while the number of doors of the car is not needed for consideration. So the important considerations are necessary to determine the most relevant attributes and eliminate the irrelevant or unimportant attributes according to the learning task without losing essential information about the original data in the database(s). These previous studies [CCH91, HCC92, HCH94] did not analyze the data dependency relation among the attributes, meaningful information about the data, such as data dependency among the attributes, are not explicitly analyzed by rule-generation algorithms; thus the rules generated in this way are not particularly concise and pertinent but contain some redundant information or unnecessary constraints in them.

Thus a technique is needed to perform a more comprehensive analysis of properties of data and identify relevant attributes prior to the generation of rules. Rough set techniques introduced by Pawlak [Paw82] provide the necessary tools to analyze the set of attributes globally. It is not feasible to apply rough set techniques directly to large database because of the computational complexity, which is NP-hard [Zir91, HuC94a]. Although these two approaches are apparently different, in both methods, objects are assumed to be characterized by attributes and attribute values. Our study shows that there is a close connection between attribute-oriented induction and the rough set approach. So a natural approach would combine the advantages of these two techniques. Based on this consideration, we present an attribute-oriented rough set based knowledge discovery system for large databases.

In this thesis, a framework for knowledge discovery in databases using rough set theory and attribute-oriented induction is proposed. Furthermore, the results from previous studies [CCH91, HCC92] are developed in two aspects. First our work [HCH93] expands the function of the previous system [CCH91, HCC92] and overcomes the "overgeneralization" problem of the previous studies. The previous method is further developed to find knowledge rules associated with different levels of the concepts in the concept hierarchy [HCH94]. If the concept hierarchy is unavailable, our method can construct a concept hierarchy automatically from the data and infer some knowledge rules based simply on the containment relationship between different clusters in the constructed concept hierarchy. This method combines our conceptual clustering technique [Hux94] with machine learning techniques.

The rough set technique is incorporated into the learning procedure. Using rough set theory, our method can analyze the attributes globally and identify the most relevant attributes to the learning task. It can handle databases with incomplete information.

The learning procedure consists of two phases: data generalization and data reduction. In data generalization, our method generalizes the data by performing attribute-oriented concept tree ascension to obtain a prime relation. The generalized prime relation only contains a small number of tuples and it is feasible to apply rough set techniques to eliminate the irrelevant or unimportant attributes and choose the best minimal attribute set. In the data reduction phase, our method finds a minimal subset of interesting attributes that have all the essential information of the generalized relation, thus the minimal subset of the attributes can be used instead of the whole attribute set of the generalized relation. Finally the tuples in the reduced relation are transformed into different knowledge rules based on different knowledge discovery algorithms. Some new knowledge discovery algorithms such as learning decision rules, maximal generalized rules, multiple sets of knowledge rules are designed by integrating attribute-oriented induction and rough set theory [Paw82].

We further propose a generalized rough set model to expand the application scope for rough set theory. The generalized rough set model can be applied to databases with noisy data. Moreover, the decision matrix method [SkR91] is combined into our method. The decision matrix approach has an incremental learning capability, which is essential for a large dynamic environment. Our system implements a number of novel ideas. It integrates a variety of knowledge discovery algorithms such as DBChar for characteristic rules, DBClass for classification rules, DBDeci for decision rules, DBMaxi for maximal generalized rules, DBTrend for data trend regularities and DBMkr for multiple sets of knowledge rules, which permit a user to discover relationships and regularities in the data. This integration allows it to exploit the strengths of diverse discovery programs.

The thesis contains nine chapters organized as follows:

An overview of the current knowledge discovery systems are discussed in Chapter 2 and several typical systems such as ID3, the AQ family, the KDW workbench, INLEN and ITRULE are briefly discussed. We describe in Chapter 3 an attribute-oriented induction system (DBLEARN) and our extension to this system. In Chapter 4, the general concept of a rough set is introduced and a general rough set model is proposed to handle uncertainty and vague information in databases. Chapter 5 is devoted to rough set based data reduction, along with some illustrative examples. Multiple sets of knowledge rules and a proposed decision matrix approach to constructing multiple sets of knowledge rules are the topic of Chapter 6. In Chapter 7 the experimental results of our system using the NSERC information system (Natural Science and Engineering Research Council of Canada) are presented and demonstrated and a discussion of our methods is given in Chapter 8. Some concluding remarks are presented in Chapter 9 with a summary of the major thesis findings and with suggestions about the directions for future progress.

Chapter 2

Overview: Knowledge Discovery in Databases

We survey some theoretical issues related to learning from examples, and some recent progress in knowledge discovery in database systems and knowledge base systems which adopt the *learning from examples* paradigm.

2.1 Concepts of Learning From Examples: An AI Approach

As a basic method in empirical learning, learning from examples has been studied extensively [CoF83, DiM83, HaM77, GeN87]. We review the basic components and the generalisation rules of learning from examples, the types of knowledge rules which can be learned, and the control strategies of the learning process.

2.1.1 Basic Components in Learning from Examples

Learning from examples can be characterised by a tuple $\langle$ P,N,C, Λ $\rangle$, where P is a set of positive examples of a concept, N is a set of negative examples of a concept, C is the conceptual bias which consists of a set of concepts to be used in defining learning rules and results, and Λ is the logical bias which captures particular logic forms [GeN87].

In most learning systems, the training examples are classified in advance by the tutor into two disjoint sets, the positive examples set and the negative examples set

[Mic83]. The training examples represent low-level, specific information. The learning task is to generalise these low-level concepts to general rules.

There could be numerous inductive conclusions derived from a set of training examples. To cope with this multiplicity of possibilities, it is necessary to use some additional information, *problem background knowledge*, to constrain the space of possible inductive conclusions and locate the most desired one(s) [Gen87]. The conceptual bias and the logical bias provide the desired concepts and the logic forms which serve as this kind of background knowledge. These biases restrict the candidates to formulas with a particular vocabulary and logic forms. Only those concepts which can be written in terms of this fixed vocabulary and logic forms are considered in the learning process.

Usually, the examples presented to the learning system consist of several attributes. Depending on the structure of the attribute domains, we can distinguish among three basic types of attributes [Mic83]:

(1) nominal attributes: the value set of such attributes consists of independent symbols or names.

(2) numerical attributes: the value set of such attributes is a totally ordered set.

(3) structured attributes: the value set of such attributes has a tree structure which forms a generalisation hierarchy. A parent node in such a structure represents a more general concept than the concepts represented by its children nodes. The domain of structured attributes is defined by the problem background knowledge.

2.1.2 Generalized Rules

Learning from examples can be viewed as a reasoning process from specific instances to general concepts. The following generalization rules are particularly useful in learning systems [CoF83, Mic83].

(1) Turning constants into variables

If the concept $F(v)$ holds for v when v is a constant a, and a constant b, and so on, then these concepts can be generalized into a statement that $F(v)$ holds for every value of v. This is the rule used most often in methods of inductive inference employing predicate calculus. As a logic formula, this can be expressed as (2.1), where the notation "$|<$" stands for "can be generalized to"

$$F(a) \wedge F(b) \wedge ... |< F(v). \qquad (2.1)$$

(2) Dropping conditions

Any conjunction can be generalized by dropping one of its conjuncts. A conjunctive condition can be viewed as a constraint on the set of possible instances that could satisfy the concept. By dropping a condition, one condition is removed and the concept is generalized. For example, the class of "red apple" can be generalized to the class of all "apples" of any colour by dropping the "red" condition. This can be written as:

$$red(v) \wedge apple(v) \ |< apple(v) \qquad (2.2)$$

(3) Adding options

By adding more conditions, the concept can be generalized because more instances may satisfy this concept. An especially useful form of this rule is when the alternative is added by extracting the scope of permissible values of one specific concept. For example, suppose that a concept is generalized by allowing objects to be not only *red* but also *blue*. This can be expressed as follows:

$$red(v) \ |< red(v) \vee blue(v) \qquad (2.3)$$

(4) Turning conjunction into disjunction

A concept can be generalized by replacing the conjunction by the disjunction operator. This process is analogous to the adding-option generalization rule. This rule can be written as follows:

$$red \wedge circle \; |< red \vee circle \qquad (2.4)$$

(5) Climbing a generalization tree

By ascending the generalization tree, the lower level concept is substituted by the higher level concept. This generalization rule is applicable only to the concept whose domain is a structure value set, (that is, concepts at different levels of generality). Formally, this rule can be expressed as:

$$\left.\begin{array}{l} L(u) \in a \\ L(v) \in b \\ .. \in .. \\ .. \in .. \\ L(z) \in i \end{array}\right\} |< \forall(x) L(x) \in s \qquad (2.5)$$

where L is a structure attribute; a, b,..., and i are the value of u,v,... and z in the attribute L, respectively; and s represents the lowest parent node whose descendants include nodes a, b,... and i.

(6) closing interval

$$\left.\begin{array}{l} L = a| < K \\ L = b| < K \end{array}\right\} L = [a..b]| < K \qquad (2.6)$$

The two premises are assumed to be connected by the logical conjunction. This rule states that if two descriptions of the same class (the premises of the rule) differ in the values of only one linear descriptor, then the descriptions can be replaced by a single description in which the reference of the descriptor is the interval linking these two values.

2.1.3 Types of Knowledge Rules

Given a learning-from-examples problem characterized as $\langle$ P,N,C, Λ $\rangle$, several different rules can be extracted. The learned concept is a *characteristic rule* if and only if it is satisfied by all of the positive examples. The learned concept is a *discriminant rule* if and only if it is not satisfied by any of the negative examples. The learned concept is an admissible rule if and only if it is both characteristic and discriminant [DiM83,GeN87].

Most learning algorithms are designed for learning admissible rules [DiM83,Mic83]. A few algorithms, such as INDUCE 1.2 [DiM81] and SPROUTER [HaM77], are designed for learning characteristic rules. DBROUGH [HuC94a, HuC94b, HSCZ94, HCH94, HCS94] can discover characteristic rules, discriminant rules and some other knowledge rules.

2.1.4 Control Strategies in Learning from Examples

Induction methods can be divided into data-driven (bottom-up), model-driven (top-down), and mixed methods depending on the strategy employed during the search for generalized concepts [DiM83]. All of these methods maintain a set, H, of the currently most plausible rules. These methods differ primarily in how they refine the set H so that it eventually includes the desired concepts.

In the data-driven methods, the presentation of the training examples drives the search. These methods process the input examples one at a time, gradually generalizing the current set of concepts until a final conjunctive generalization is computed. The typical examples of such control strategy include the candidate-elimination algorithm [Mit77, Mit79], the approach adopted in [HoM77,WaE87], the ID3 techniques of Quinlan [Qui86] and the Bacon learning system [Lan77].

In the model-driven methods, an a priori model is used to constrain the search. These methods search a set of possible generalisations in an attempt to find a few "best" hypotheses that satisfy certain requirements. Typical examples of systems which adopt this strategy are AM [Len77], DENDRAL and Meta-DENDRAL [BuM78], and the approach used in the INDUCE system [DiM81].

Data-driven techniques generally have the advantage of supporting incremental learning. The learning process can start not only from the specific training examples, but also from the rules which have already been discovered. The learning systems are capable of updating the existing hypotheses to account for each new example. In contrast, the model-driven methods, which test and reject hypotheses based on an examination of the whole body of data, are difficult to use in incremental learning situations. When new training examples become available, model-driven methods must either backtrack or restart the learning process from the very beginning, because the criteria by which hypotheses were originally tested (or schemas instantiated) have been changed [DiM83]. On the other hand, an advantage of model-driven methods is that they tend to have good noise immunity. When a set of hypotheses, H, is tested against noisy training examples, the model-driven methods need not reject a hypothesis on the basis of one or two counterexamples. Since the whole set of training examples is available, the program can use statistical measures of how well a proposed hypothesis accounts for the data. In the data-driven method, the set of hypotheses, H, is revised each time on the basis of the current training example. Consequently, a single erroneous example can cause a large perturbation in H (from which it may never recover) [DiM83].

2.2 Some Learning From Examples Models

Since the 1960's, many algorithms and experimental systems of *learning from examples* have been developed [Mit77], which demonstrated aspects of machine learning in science, industry and business applications [Hau87,Ren86]. In this section, we present several successful models which are related to our research.

2.2.1 The Candidate Elimination Algorithm

Mitchell developed an elegant framework, "*version space*", for describing systems that use a data-driven approach to concept learning [Mit82]. This framework can be described as follows. Assume we are trying to learn some unknown target concept defined on the instance space. We are given a sequence of positive and negative examples which are called samples of the target concept. The task is to produce a concept that is consistent with the samples. The set of all hypothesis, H, that are consistent with the sample is called the version space of the samples. The version space is empty in the case that no hypothesis is consistent with the samples.

Mitchell proposed an algorithm, called the candidate-elimination algorithm, to solve this learning task. The algorithm maintains two subsets of the version space: the set S of the most specific hypothesis in the version space and the set G of the most general hypotheses. These sets are updated with each new example. The positive examples force the program to generalise the S set, and the negative examples force the program to specialize the G set. The learning process terminates when $G = S$

A good feature of this method is that the incremental learning can be performed by the learning program. The sets S and G can easily be modified to account for new training examples without any re-computation.

However, as with all data-driven algorithms, the candidate elimination algorithm has difficulty with noisy training examples. Since this algorithm seeks to find a concept that is consistent with all of the training examples, any single bad example (that is, a false positive or false negative example) can have a profound effect. When the learning system is given a false positive example, for instance, the concept set becomes overly generalized. Similarly, a false negative example causes the concept set to become overly specialised. Eventually, noisy training examples can lead to a situation in which there are no concepts that are consistent with all of the training examples. The second and most important weakness of this algorithm is its inability to discover disjunctive concepts. Many concepts have a disjunctive form, but if disjunctions of arbitrary length are permitted in the representation language, the data-driven algorithm described above never generalises. Unlimited disjunction allows the partially ordered rule space to become infinitely "branchy".

There are two computational problems associated with this method. The first one is that in order to update the sets S and G we must have an efficient procedure for testing whether or not one hypothesis is more general than another. Unfortunately, this testing problem is NP-complete if we allow arbitrarily many examples and arbitrarily many attributes in the hypothesis [Hau86]. The second computational problem is that the size of the sets S and G can become unmanageably large. It has been shown that, if the number of attributes is large, the sizes of set S and set G can grow exponentially in the number of examples [Hau86].

To improve computational efficiency, Haussler proposed a one-sided algorithm in contrast to the two-sided approach of the candidate elimination algorithm. The one-sided algorithm computes only the set S using the positive examples and then checks to see if any negative examples are contained in the set S. If the rule in the set S is not satisfied by any negative examples, the rule is valid. Otherwise, there is no rule which can be discovered [Hau86,Hau87].

In some learning situations, it is possible for the user to select training examples and to acquire information about their classification. In this case, a common strategy to maximise the learning performance is to select an example that halves the number of candidate formulas, that is, one that satisfies one-half of the candidates and does not satisfy the other half. The advantage of this strategy is that, by getting the classification of such an example, we can eliminate one-half of the remaining candidates. However, the main problem with the halving strategy is computational expense. In the worst case, we need to compare each example with each concept to determine whether or not the example satisfies the concept. If there are m examples and n candidates, then in the worst case we need mn steps to select the best example. This is time consuming when either m or n is very large.

Subramanian and Feigenbaum proposed a method, *experiment generation*, to solve this problem [SuF86]. They proposed to partition an instance into several independent sub-instances and to factor the entire version space into multiple separate smaller version spaces. The test procedure for selecting the best training instance can be first performed in each factored version space, and then the resulting "sub-instance" can be combined into a single instance to be tested. The computational advantages of factoring are striking. Suppose that a version space can be factored into k factors, with p nodes each. Whenever this is the case, the size of the un-factored version space must be p^k. If we can factor the version space, then we can "factor" each instance into k parts, one for each factor of the version space. If there are q possibilities for each part, then there must be q^k instances. The total cost for selecting a training

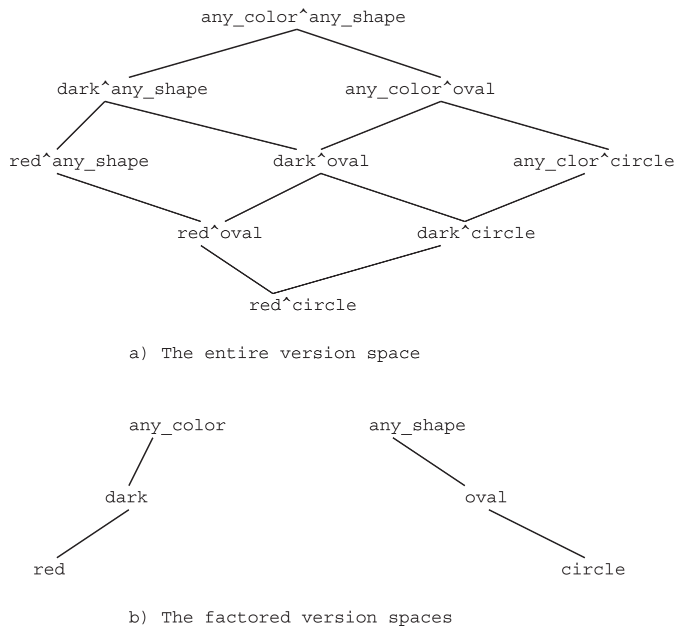


Figure 2.1: The version spaces for the positive example "red $\wedge$ circle"

instance without factoring is $p^k q^k$, whereas the total cost with factoring is just kpq, a substantial saving when p or q is large. Figure 2.1 shows the entire version space and the factored version spaces in which the training example "red $\wedge$ circle" is the sole positive example. While the entire version space contains 9 nodes, the factored version spaces consists of only 6 nodes.

2.2.2 AQ11 and AQ15 Systems

Michalski and his colleagues have developed a series of AQ learning systems. The AQ11 system [MiC80] is designed to find the most general rule in the rule space that discriminates training examples in a class from all training examples in all other classes. Michalski et al. call these types of rules *discriminate descriptions* or *discriminant rules* since their purpose is to discriminate one class from a predetermined set of other classes.

The language used by Michalski to represent discriminant rules is VL1, an extension of the propositional calculus. VL1 is a fairly rich language that includes conjunction, disjunction, and the set-membership operators. Consequently, the rule space of all possible VL1 discriminant rules is quite large. To search this rule space, AQ11 uses the AQ algorithm, which is nearly equivalent to the repeated application of the candidate-elimination algorithm. AQ11 converts the problem of learning discriminant rules into a series of single-concept learning problems. To find a rule for class A, it considers all of the known examples in class A as positive examples and all other training examples in all of the remaining classes as negative examples. The AQ algorithm is then applied to find a concept that covers all of the positive examples without covering any of the negative examples. AQ11 seeks the most general such concept, which corresponds to a necessary condition for class membership.

After developing the AQ11 system, Michalski et al. proposed another inductive learning system AQ15 in 1986 [MMHL86]. This system is an extended version of the AQ11 system, which is able to incrementally learn disjunctive concepts from noisy and overlapping examples, and can perform constructive induction in which new concepts are introduced in the formation of the inductive conclusions.

2.2.3 ID3, ID4, ID5

ID3 was developed by Quinlan [Qui83]. ID3 can discover classification rules in the form of a decision tree for a collection of instances. ID3 uses an information-theoretic approach aimed at minimizing the expected number of tests to classify the objects. The attribute selection part of ID3 is based on the plausible assumption that the complexity of the decision tree is strongly related on the amount of information conveyed by this message. It builds a decision tree by choosing a good test attribute that partitions the instance into smaller sets for which decision subtrees are constructed recursively. To determine which attribute should be the test attribute for a node, the algorithm applies an information-theoretic measure *gain*. An attribute with the maximal gain is selected as the test attribute.

The ability of ID3 to construct decision trees that are efficient classifier and that generalizes well is attractive. For learning problems in which the collection of instances is available and is not likely to change, ID3 is a good choice for building classification rules. However for problems in which new instances are expected to become available on a regular basis, it would be far more preferable to accept instances incrementally, without needing to built a new decision tree from scratch each time.

Schlimmer and Fisher constructed ID4 [ScF86], which incrementally builds a decision tree similar to that which ID3 would build. Instead of building a decision tree from a batch of instances, ID4 updates a decision tree based on each individual instance. This algorithm offers an approach to incremental learning of ID3-type decision trees. A potential drawback of the algorithm is that all or part of a decision tree will be discarded whenever it is determined that the test attribute should be replaced with a better attribute. To overcome this shortcoming, Utgoff [Utg88] developed the ID5 algorithm. ID5 builds on the idea of ID4 that one can maintain positive and negative instance counts of every attribute that could be a test attribute for the decision tree or subtree. ID5 differs from ID4 in its method for replacing the test attribute. Instead of discarding the subtree below the old test attribute, ID5 reshapes the tree by pulling the test attribute up from below. The advantage is that the positive and negative instance counts can be recalculated during the tree manipulations, without reprocessing the instances.

The algorithms ID3 and so on have been widely used for rule induction. However, such decision trees are essentially sequential decision algorithms which are quite different in nature from the data driven nature of expert systems or knowledge base systems. Rule bases are data driven in the sense that any set of input data can potentially be used to begin the inference. Decision trees must always begin with the attribute associated with the root node. In addition, rule bases can accommodate missing attribute information, whereas decision trees are not designed to do so. Decision trees can also be difficult to understand for the user [ArM85], a problem which should not be underestimated in light of the overall advantages of explicit knowledge representation inherent to "If ... then" rule. This is not to say that decision trees are not useful in problems areas, such as classification where a predetermined "hardwired" solution is sufficient [GoS88]. However, by their very definition, knowledge bases tend to be used for problems where variable inputs can be handled (incomplete, uncertain, or dynamic data), variable outputs (different goals) may be specified, and there is a need for an explicit representation of the system's knowledge for user interaction.

2.3 Concepts of Learning From Databases

Learning from databases can be characterized by a triple $\langle$ D,C, Λ $\rangle$ where D represents the set of data in the database relevant to a specific learning task, C represents a set of "concept biases" (generalization, hierarchies, etc.) useful for defining particular concepts, and Λ is a language used to phrase definitions.

Three primitives should be provided for the specification of a learning task: *task-relevant data*, *background knowledge*, and *the expected representations of learning results*. For illustrative purposes, we only examine relational databases, however, the results can be generalized to other kinds of databases.

2.3.1 Data Relevant to the Discovery Process

A database usually stores a large amount of data, of which only a portion may be relevant to a specific learning task. For example, to characterize the features of *mammal* in *animal*, only the data relevant to *mammal* in *animal* are appropriate in the learning process. Relevant data may extend over several relations. A query can be used to collect task-relevant data from the database. Task-relevant data can be viewed as examples for learning processes. Undoubtedly, *learning-from-examples* should be an important strategy for knowledge discovery in databases. Most *learning-from-examples* algorithms partition the set of examples into *positive* and *negative* sets and perform *generalization* using the positive data and *specialization* using the negative ones [DiM83]. Unfortunately, a relational database does not explicitly store negative data (even though the negative data can be derived based on the closed world assumption [Rei84]), and thus no explicitly specified negative examples can be used for specialization. Therefore, a database induction process relies mainly on generalization, which should be performed cautiously to avoid over-generalization.

The data relevant to the learning task can usually be classified into several classes based on the values of a specific attribute. For example, the data about animal may be classified into mammal and bird based on the value of the attribute "type". We introduce new concepts *target class* and *contrasting class*

Definition 2.1 *A target class is a class in which the data are tuples in the database consistent with the learning concepts.*

Definition 2.2 *A contrasting class is a class in which the data do not belong to the target class.*

For instance, to distinguish *mammal* from *bird*, the class of *mammal* is the target class, and the class of *bird* is the contrasting class.

2.3.2 Background Knowledge

The quality (or lack of) and vastness of the data in real-world databases represent the core problems for KDD. Overcoming the quality problem requires external domain knowledge to clean-up, refine, or fill in the data. The vastness of the data forces the use of techniques for focussing on specific portions of the data, which requires additional domain knowledge if it is to be done intelligently. A KDD system, therefore, must be able to represent and appropriately use domain knowledge in conjunction with the application of empirical discovery algorithms.

Concept hierarchies represent the necessary background knowledge which controls the generalization process. Different levels of concepts are often organized into a taxonomy of concepts. The concept taxonomy can be partially ordered according to a general-to-specific ordering. The most general concept is the null description (described by a reserved word "any"), and the most specific concepts correspond to the specific values of the attributes in the database [CCH91,Mit77]. Using a concept hierarchy, the rules learned can be represented in terms of generalized concepts and stated in a simple and explicit form, which is desirable to most users.

Concept hierarchies can be provided by knowledge engineers or domain experts. This is reasonable for even large databases since a concept tree registers only the *distinct* discrete attribute values or ranges of numerical values for an attribute which are, in general, not very large and can be input by a domain expert. But if the concept hierarchies are not available, in some case, it is possible to construct them based on the data in databases. This problem will be addressed in Chapter 3.

2.3.3 Representation of Learning Results

From a logical point of view, each tuple in a relation is a logic formula in conjunctive normal form, and a data relation is characterized by a large set of disjunctions of such conjunctive forms. Thus, both the data for learning and the rules discovered can be represented in either relational form or first-order predicate calculus.

The complexity of the rule can be controlled by the generalization threshold. A moderately large threshold may lead to a relatively complex rule with many disjuncts and the results may not be fully generalized. A small threshold value leads to a simple rule with few disjuncts. However, small threshold values may result in an overly generalized rule and some valuable information may get lost. A better method is to adjust the threshold values within a reasonable range *interactively* and to select the best generalized rules by domain experts and/or users.

2.3.4 Types of Rules

There are several types of rules, including characteristic rules, classification rules and decision rules which can be easily learned from relational databases.

Definition 2.3 *A characteristic rule is an assertion which characterizes the concepts satisfied by all of the data stored in the database.*

For example, the symptoms of a specific disease can be summarised as a characteristic rule.

Definition 2.4 *A classification rule is an assertion which discriminates the concepts of one class from other classes.*

For example, to distinguish one disease from others a classification rule should summarise the symptoms that discriminate this disease from others.

Definition 2.5 *A decision rule is an assertion which determines the cause-effect relationship between conditions and decision factors.*

Characteristic rules, classification rules and decision rules are useful in many applications. A characteristic rule provides generalized concepts about a property which can help people recognise the common features of the data in a class. The classification rule gives a discrimination criterion which can be used to predict the class membership of new data and the decision rules help people in decision making procedure.

In learning a characteristic rule, relevant data are collected into one class, the target class, for generalization. In learning a discrimination rule, it is necessary to collect data into two classes, the target class and the contrasting class(es). The data in the contrasting class(es) imply that such data cannot be used to distinguish the target class from the contrasting one(s), that is, they are used to exclude the properties shared by both classes. In learning decision rules, we need to organise the data into different group based on the value of the decision factors.

2.4 Knowledge Discovery in Large Databases

Currently, the steady growth in the number and size of large databases in many areas, including medicine, business and industry has created both a need and an opportunity for extracting knowledge from databases. Some recent results have been reported which extract different kinds of knowledge from databases.

Knowledge discovery in databases poses challenging problems, especially when databases are large. Such databases are usually accompanied by substantial domain knowledge to facilitate discovery. Access to large databases is expensive, hence it is necessary to apply the techniques for sampling and other statistical methods. Furthermore, knowledge discovery in databases can benefit from many available tools and techniques in different fields, such as, expert systems, machine learning, intelligent databases, knowledge acquisition, and statistics [CCH91,HCC92a, HCC92b].

2.4.1 INLEN System

The INLEN system was developed by Kaufman et al in 1989 [KMK91]. The system combines some database, knowledge-base, and machine learning techniques to provide a user with an integrated system of tools for conceptually analyzing data and searching for interesting relationships and regularities among data. It merges several existing learning systems and provides a control system to facilitate access. Figure 2.2 illustrates the general design of the system.

The INLEN system consists of a relational databases for storing known facts about a domain and a knowledge base for storing rules, constraints, hierarchies, decision trees, equations accompanied with preconditions and enabling conditions for performing various actions on the database or knowledge base. The knowledge base not only can contain knowledge about the contents of the database but also meta-knowledge for the dynamic upkeep of the knowledge base itself.

The motivating goal of the INLEN system is to integrate three basic technologies: databases, expert systems and machine learning and inference to provide a user with a powerful tool for manipulating both data and knowledge and extracting new or better knowledge from these data and knowledge. It is especially appropriate to apply INLEN to data systems that are constantly changing or growing; among the system's capabilities are the abilities to detect changes over time and explore the ramifications of the changes.

INLEN employs three sets of operators: data management operators (DMOs), knowledge management operators (KMOs), and knowledge generation operators (KGOs).

The DMOs are standard operators for accessing, retrieving and manually altering the information in the database. The KMOs are used to create, manipulate and modify INLEN's knowledge base, thereby allowing the knowledge base to be handled in a manner analogous to handling a database. The KGOs take input from both the database and knowledge base, and invoke various machine learning programs to perform learning tasks. For example, the operator CLUSTER creates the conceptual clustering algorithm developed in [MiC80]. The operator DIFF determines the discrimination rules, which can be executed in the AQ program [MiC80]. The operator CHAR discovers characteristic rules, which is also implemented in an AQ program

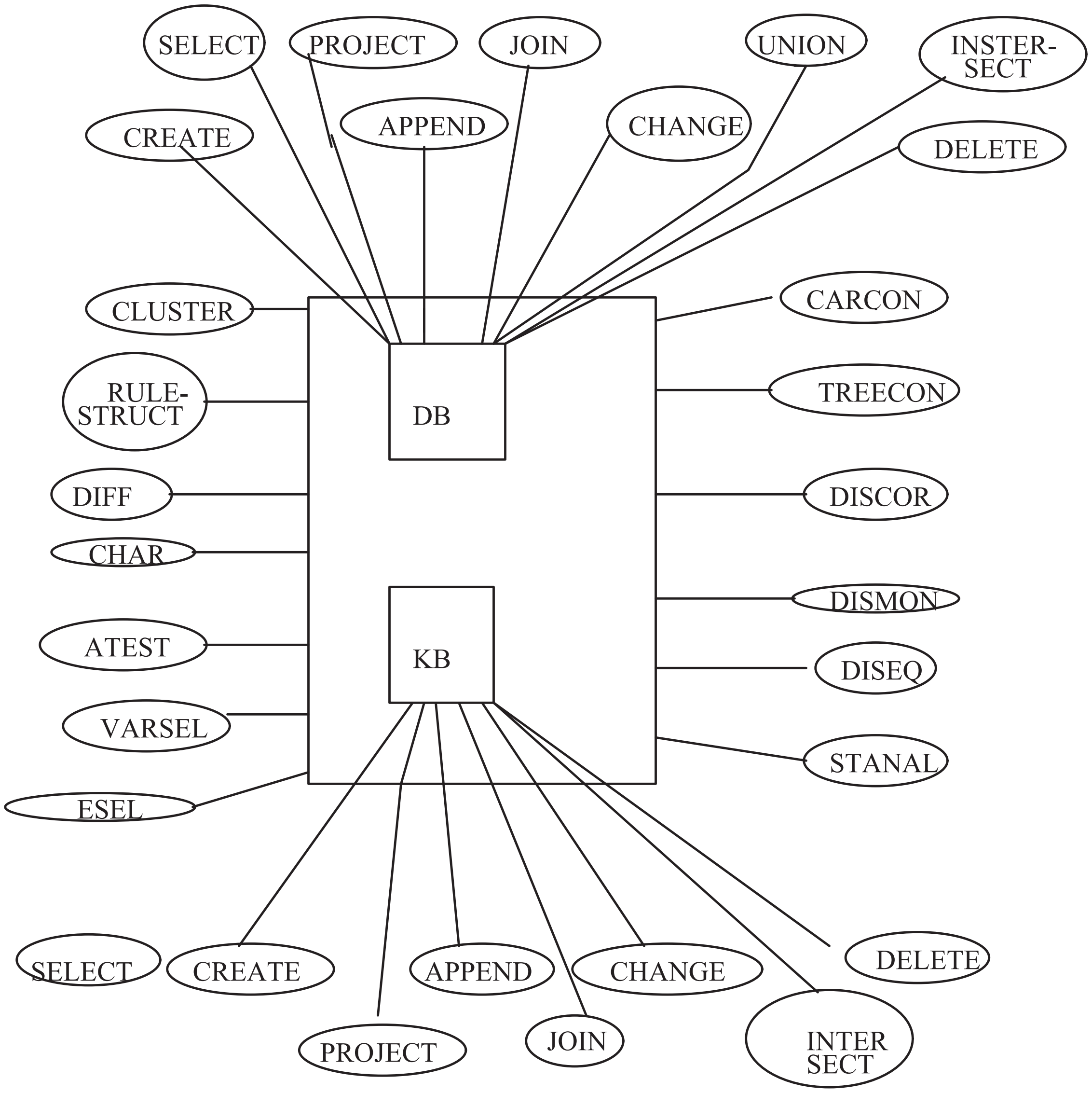


Figure 2.2: The architecture of INLEN

[MiC80]. The operator VARSEL selects the most relevant attributes and the operator ESEL determines the most representative examples. The operator DISEQ discovers equations governing numerical variables, which is based on the ABACUS-2 system for integrated qualitative and quantitative discovery [FaM86]. ABACUS-2 is related to programs such as BACON [LLBS83] and FAHRENHEIT [Zyt87]. Most of these machine learning programs invoked by KGOs are existing learning algorithms which have been well implemented.

As in the case of many machine learning systems, the major challenge to the INLEN system is computational inefficiency. Many learning algorithms included in this system adopt the tuple-oriented approach which examines the training examples tuple by tuple. In the learning process, these algorithms usually have a large search space and costly time complexity because they are not designed for large databases. Although this system integrates databases, knowledge-based and machine learning techniques, the database operations are applied only for retrieving data and storing knowledge rules. The algorithms in this system do not take advantage of database implementation techniques in the learning process.

2.4.2 KDW System

Like INLEN, the Knowledge Discovery Workbench (KDW) is a collection of tools for the interactive analysis of large databases [MCP93]. Its components have evolved through three versions (KDW, KDW II, and KDW ++), all of which provide a graphical user interface to a suite of tools for accessing database tables, creating new fields, defining a focus, plotting data and results, applying discovery algorithms and handling domain knowledge. The current version of the system is embedded with an extensible command interpreter based on *tcl* [Ous90], which enables the user to interactively control the discovery process or call up intelligent scripts to automate discovery tasks. The following extraction algorithms have been incorporated into one or more versions of the KDW: *clustering* for identifying simple linearly-related classes; *classification* for finding rules using a decision-tree algorithm; *summarisation* for characterizing classes or records; *deviation detection* for identifying significant differences between classes of records; *dependency analysis* for finding and displaying probabilistic dependencies.

The KDW has direct access to a DBMS through its SQL-based query interface. Its knowledge base contains information specific to a database regarding important field group, record group, functional dependencies, and SQL-query statement. Most of the domain knowledge is used to provide focus by guiding the access of information from the database. Control in the KDW is provided exclusively by the user, who may define scripts to automate frequently repeated operations.

The KDW itself is intended to be versatile and domain independent. As such, it requires considerable guidance from the user who must decide what data to access, how to focus the analysis, which discovery algorithms to apply, and how to evaluate and interpret the results. The "workbench" design is ideal for exploratory analysis by a user knowledgeable in both data and the operation of the discovery tools.

2.4.3 The ITRULE Algorithm

ITRULE is a database learning program based on information theory [SyG92]. Like ID3 [Qui83], CN2 [ClN89] and PRISM [Cen87], it searches for classification rules directly using a measure of rule goodness, J-measure. ITRULE takes sample data in the form of discrete attribute vectors and generate a set of K rules, where K is a user-defined parameter. The set of generated rules are the K most informative rules from the data as defined by the J-measure. The probabilities required for calculating the J-measures are estimated directly from the data using standard statistical point estimation techniques [SyG92].

The algorithm proceeds by first finding K rules, calculating their J-measures, and then placing these K rules in an ordered list. The smallest J-measure, that of the Kth element of the list, is then defined as the running minimum J_{min}. From that point onwards, new rules which are candidates for inclusion in the rule set have their J-measure compared with J_{min}. If greater than J_{min}, they are inserted in the list, the Kth rule is deleted, and J_{min} is updated with the value of the J-measure of whatever rule is now Kth on the list. The critical part of the algorithm is the specialization criterion since it determines how much of the exponentially large hypothesis space actually needs to be explored by the algorithm.

The number of possible rules is exponential in the number of attributes and the cardinality of their event space. For n *m-ary* attributes the number of possible rules in the data is R where

$$R = nm((2m+1)^{n-1} - 1)$$

since for each of the nm possible right-hand sides, the other $n-1$ attribute have $2m+1$ possible states, namely, a truth statement and its negation for each of the m propositions and a "do not care" state for the attribute as a whole (for the case of *binary* attribute $m = 1$ because the negation of a proposition is also a basic proposition). From a practical point of view, we do not have the computational resources to manage them. Hence in order to define a tractable algorithm we will need to "prune" the set of possible rule candidate considerably. The ITRULE produces the *set of best rules* rather than *best set of rules,* i.e., no attempt is made to evaluate the collective properties of the rules. It is conjectured that this problem is computationally intractable to solve optimally for arbitrary K.

Chapter 3

Extending DBLEARN

DBLEARN is a database learning system developed by Cai, Cercone and Han [CCH91, HCC92a, HCC92b]. It implements both LCHR (for Learning Characteristic Rules) and LCLR (for Learning Classification Rules) algorithms. The language of DBLEARN can be viewed as an extension to the relational language SQL for knowledge discovery in databases. The architecture of DBLEARN is presented in Figure 3.1.

DBLEARN [CCH91] was implemented in an Unix/C/Sybase environment. It can generate many interesting patterns, however, it sometimes tends to discover "over-generalized" patterns. A moderately large threshold may lead to a relatively complex rule with many disjuncts and the results may not be fully generalized. A small threshold value leads to a simple rule with few disjuncts. However, small threshold values may result in an overly generalized rule and some valuable information may get lost.

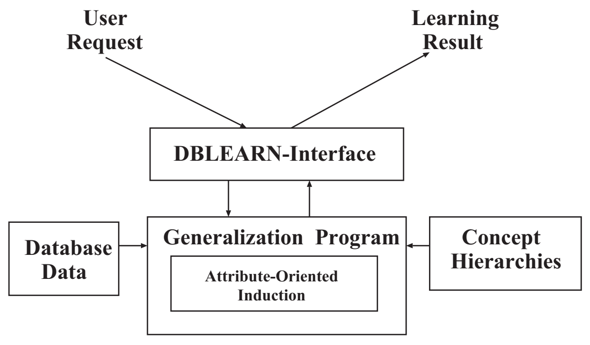


Figure 3.1: The architecture of DBLEARN

Furthermore, DBLEARN cannot derive the patterns that have a comparison in their bodies.

To overcome the "overgeneralization" problem, we introduced a new method, which first generalizes the primitive data into a prime relation. The prime relation contains the essential information of the original system. Then we generalize the prime table associated with different levels of the concept hierarchy. The attribute-oriented approach is further developed for learning different kinds of rules, including characteristic rules, classification rules, hierarchy rules, domain knowledge. Moreover, learning can also be performed with databases in some case while the concept hierarchies are not available.

3.1 Discovery of Knowledge Associated with Concept Hierarchies

In this section we propose a new method to overcome the "overgeneralization" problem of DBLEARN. Our method is performed in 4 steps. First, a set of data relevant to the learning task is collected by a database query. Second, the collected data is then generalized by removal of nondesirable attributes and by performing concept-tree ascension (replacing lower-level attribute values in a relation using the concept hierarchy) on each generalizable attribute until the attribute becomes desirable (i.e., containing only a small number of distinct values). The identical generalized tuples in the relation are merged into one with a special internal attribute, *vote*, created to register how many original tuples are generalized to this resultant tuple. The generalized relation obtained at this stage is called the *prime relation* and saved for later use. Third, we further simplify the generalized relation and map it into the feature table, then analyze the feature table and infer different kinds of rules. Finally, we examine the prime relation once more and infer the inheritance rules associated with the concept hierarchies.

A *prime relation* R_p for a set of data R stored in the relational table is an intermediate relation generalized from relation R by removing nondesirable attributes and generalizing each attribute to a *desirable level*. Let a *desirability threshold* be available for each attribute, which could be set by default or specified by the user or an expert, based on the semantics of the attributes and/or the expected forms of generalized rules. A prime relation maintains the relationship among generalized data in different attributes for a frequently inquired-of data set. It can be used for extraction of various kinds of generalized rules. The following algorithm extracts the prime relation R_p from a set of data R stored in relational table.

Algorithm 3.1 *Extraction of the prime relation from a set of data R*

Input: (i) A set of task-relevant data R (obtained by a relation query and stored in a relation table), a relation of arity n with a set of attributes A_i ($1 \leq i \leq n$); (ii) a set of concept hierarchies, H_i, where H_i is a hierarchy on the generalized attribute A_i, if available; and (iii) a set of desirability thresholds T_i for each attribute A_i
Output. The prime relation R_p
Method

```
1. $R_t := R$; /* $R_t$ is a temporary relation. */

2. for each attribute $A_i$ ( $1 \leq i \leq n$) of $R_t$ do {
       if $A_i$ is nondesirable then remove $A_i$;
       if $A_i$ is not desirable but generalizable then generalize $A_i$ to
       desirable level;

   /* Generalization is implemented as follows. Collect the distinct values in the
   relation and compute the lowest desirable level $L$ on which the number of dis-
   tinct values will be no more than $T_i$ by synchronously ascending the concept
   hierarchy from these values. Generalize the attribute to this level $L$ by substi-
   tuting for each value $A_i$'s with its corresponding concept $H_i$ at level $L$. */
   }

   /* Identical tuples in the generalized relation $R_t$ are merged
   with the number of identical tuples registered in vote */ .

3. $R_p := R_t$
```

Label	Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim
T11	tiger	Y	pointed	forward	N	claw	meat	Y	N	Y
HA1	cheetah	Y	pointed	forward	N	claw	meat	Y	N	Y
FT3	giraffe	Y	blunted	side	N	hoof	grass	Y	N	Y
HJ8	zebra	Y	blunted	side	N	hoof	grass	Y	N	Y
O9H	ostrich	N	N	side	Y	claw	grain	N	N	N
KJ2	penguin	N	N	side	Y	web	fish	N	N	N
OL2	albatross	N	N	side	Y	claw	grain	N	Y	N
LP1	eagle	N	N	forward	Y	claw	meat	N	Y	N
TT1	viper	N	pointed	forward	N	N	meat	N	N	N

Table 3.1: An animal world.

Observation 3.1: Algorithm 3.1 correctly extracts the prime relation R_p from a data relation R.

Rationale: An attribute-value pair represents a conjunct in the logical form of a tuple. The removal of a conjunct eliminates a constraint and thus generalizes the rule, which corresponds to the generalization rule *dropping conditions* in *learning from examples*. Thus if an attribute is nondesirable, the removal generalizes the relation. Moreover, if an attribute is not at the desirable level but generalizable, the substitution of an attribute value by its higher level concept covers more cases than the original tuple and thus generalizes the tuple. This process corresponds to the generalization rule, *climbing generalization trees* in *learning from examples*. Since all of the generalizable attributes are at the desired level, the generalized relation is the prime relation.

For example, suppose we have an animal relation for some zoo as depicted in Table 3.1 and the concept hierarchy for the attribute "Animal" as depicted in Figure 3.2:

In the initial relation, the first attribute "Label" is the key to the relation, the key value is distinct for each tuple in the relation . If there is no higher level concept provided for such an attribute in the concept tree, the value for the attribute cannot be

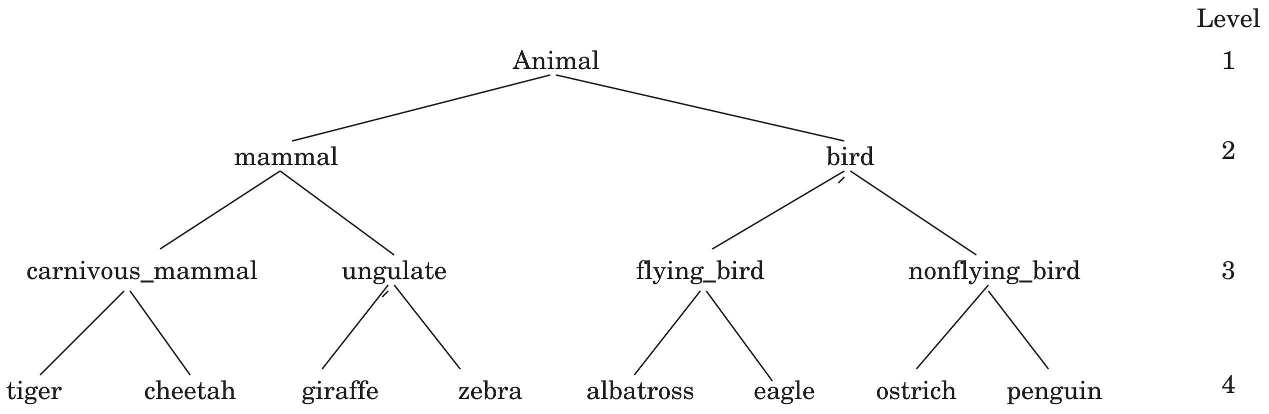


Figure 3.2: Conceptual hierarchy of the animal world

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	Vote
cmammal	Y	pointed	forward	N	claw	meat	Y	N	Y	2
ungulate	Y	blunted	side	N	hoof	grass	Y	N	Y	2
nonflyb	N	N	side	Y	claw	grain	N	N	N	1
nonflyb	N	N	side	Y	web	fish	N	N	N	1
flying	N	N	side	Y	claw	grain	N	Y	N	1
flying	N	N	forward	Y	claw	meat	N	Y	N	1
viper	N	pointed	forward	N	N	meat	N	N	N	1

Table 3.2: The prime relation table.

generalized and it should be removed in the generalization process. Other candidate key attributes or nonkey attributes can be eliminated under a similar condition. The next attribute "Animal", has 9 distinct values, which is greater than the threshold value for our desirable level (assume the desirability threshold is 6), the concept-tree ascension technique is applied; the attribute is generalized to the desirable level (level 3) $\{carnivorous\_mammal, ungulate, flying\_bird, nonflying\_bird\}$ in the conceptual hierarchy. We examine then the other attributes and since all of them are already at the desirable level, the prime relation is obtained as shown in Table 3.2.

The derivation and storage of prime relations for frequently inquired-of data sets may facilitate the extraction of different kinds of generalized rules from the prime relation. Further generalization can be performed on prime relations to derive characteristic or inheritance rules if there are still many tuples in the prime relation. Based upon different interests, a generalized relation can be directly mapped into different feature tables. We have the following algorithm for the extraction of a feature table from a generalized relation.

Algorithm 3.2 *Feature table T_A extraction for an attribute A from the generalized relation R'.*

Input: A generalized relation R' consists of (i) an attribute A with distinct values a_1,...,a_m, m is the number of distinct values for A (ii) j other attributes $B_1, ..., B_j$, j is the number of attributes in the relation R' except A (suppose different attributes have unique distinct values), and (iii) a special attribute, *vote*.
Output. The feature table T_A
Method.

1. The feature table T_A consists of $m+1$ rows and $l+1$ columns, where l is the total number of distinct values in all the attributes. Each entry of the table is initialized to 0.

2. Each slot in T_A (except the last row) is filled by the following procedure,

```
for  each row $r$ in $R'$ do {
for each attribute $B_j$ in $R'$ do
    $T_A[r.A, r.B_j] := T_A[r.A, r.B_j] + r.vote$;
$T_A[r.A, vote] := T_A[r.A, vote] + r.vote$; }
```

3. The last row p in T_A is filled by the following procedure:

```
for each column $s$ in $T_A$ do
for each row $t$ ( except the last row $p$) in $T_A$ do
    $T_A[p, s] := T_A[p, s] + T_A[t, s]$;
```

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	Vote
mammal	Y	pointed	forward	N	claw	meat	Y	N	Y	2
mammal	Y	blunted	side	N	hoof	grass	Y	N	Y	2
bird	N	N	side	Y	claw	grain	N	N	N	1
bird	N	N	side	Y	web	fish	N	N	N	1
bird	N	N	side	Y	claw	grain	N	Y	N	1
bird	N	N	forward	Y	claw	meat	N	Y	N	1
other	N	pointed	forward	N	N	meat	N	N	N	1

Table 3.3: A generalized relation.

Observation 3.2: Algorithm 3.2 correctly registers the number of occurrences of each general feature in the generalized relation R'.

Rationale: Following the algorithm, each tuple in the generalized relation is examined once with every feature registered in the corresponding slot in the feature table. Their column-wise summation is registered in the last row.

In our example, in order to obtain the feature table, the prime relation is further generalized by substituting the concept at level 3 by those at level 2, resulting in the generalized relation as shown in Table 3.3.

The feature table is then extracted from the generalized relation by using algorithm 3.2 based on the attribute "Animal" and the result is shown in Table 3.4 (since we are interested in learning for Animal). Different feature tables can be extracted from the generalized relation based on the interest in different attributes. The extracted feature table is useful for derivation of the relationships between the classification attribute and other attributes at a high level. For example, the generalized rule *All animals with hair are mammals* can be extracted from Table 3.4 based upon the fact the class *mammal* takes all the votes with *Hair* count.

We present two algorithms for discovering different kinds of rules, characteristic and equality, and inheritance rules from a database system.

Algorithm 3.3 *An attribute-oriented induction for discovering characteristic and equality rules associated with the concept hierarchy.*

Animal	Hair		Teeth			..	Feather		..	Swim		vote
	y	n	pointed	blunted	n	..	y	n	..	y	n	
mammal	4	0	2	2	0	..	0	4	..	4	0	4
bird	0	4	0	0	4	..	4	0	..	1	3	4
others	0	1	1	0	0	..	0	1	.	0	1	1
total	4	5	3	2	4	..	4	5	..	5	4	9

Table 3.4: The feature table for the attribute animal.

Input: (i) the prime relation obtained by Algorithm 3.1 (ii) a concept hierarchy table. (iii) a threshold N for the total number of tuples in the final generalized relation
Output: A set of characteristic rules and equality rules.
Method.

1. Generalize the prime relation further by performing an attribute-oriented concept ascension technique until the number of the tuples is equal or less than the threshold value N

2. Using the feature-table extraction algorithm (Algorithm 3.2), extract a feature table T_A from the prime relation based upon a certain attribute A.

3. Assume that there are in total J classes, i.e., there are J distinct values for attribute A, $A_1, \ldots, A_J$. Also, assume that there are I attributes: $C_1, \ldots, C_I$, for the data in the feature table. We use K_j to denote the number of distinct values for attribute J_j. According to the feature table, two probability values, $b_{i,j,k}$ and $c_{i,j,k}$, are associated with the k-th value ($k = 1, \ldots, K_j$) of the j-th attribute ($j = 1, \ldots, I$) in the i-th class ($i = 1, \ldots, J$). Notice that the number of tuples associated with the k-th value of the j-th attribute in the i-th class is denoted by $a_{i,j,k}$.

$$b_{i,j,k} = a_{i,j,k}/total.$$

$$c_{i,j,k} = a_{i,j,k}/vote.$$

where $b_{i,j,k}$ represents the probability of $a_{i,j,k}$ in the entire database and $c_{i,j,k}$ denotes the probability of $a_{i,j,k}$ in the i-th class.

4. Extract characteristic rules and equality rules based on the probability for each distinct value of every attribute in each class in the feature table T_A. This is performed as follows.

```
for each class do {
  if $b_{i,j,k} = c_{i,j,k} = 1$
  then the following rule is inferred.
      $A_j = T_A[i,j,k] \;\leftrightarrow\; Class = C_i.$
  if $b_{i,j,k} = 1 \; and \; c_{i,j,k} < 1$
  then the following rule is inferred.
      $A_j = T_A[i,j,k] \;\rightarrow\; Class = C_i.$
  if $b_{i,j,k} < 1 \; and \; c_{i,j,k} = 1$
  then include $A_j = T_A[i,j,k]$ as a component for the corresponding character-
      istic rule for the $i$-th class.
  if $b_{i,j,k} \neq 1 \; and \; c_{i,j,k} \neq 1 \; and \; b_{i,j,k} * c_{i,j,k} \leq r_{frequency}$
  then ignore this value
  else include the value as one of the characteristic values for the attribute.
```

{ /* Since data in a database may be distributed along the full spectrum of the possible values, it is impossible to obtain a meaningful rule for such kinds of data without using possible quantitative information. Various techniques can be developed for rule extraction using quantitative information. Our method treats data which occur rarely in the database as exceptional or noise data and filters it using $r_{frequency}$, where a small $r_{frequency}$ indicates that the data occurs with a very low frequency ratio. */ }.

5. Simplify the learned rules.

If the distinct data value set of an attribute covers the entire set of values for the attribute, remove this attribute and its associated values from the rule. Otherwise, compare the number of the values appearing as the characteristic values for the attribute with the total number of distinct values for the attribute. If the difference is larger than some pre-set number, the 'not' operator is introduced to the rules to simplify it.

6. Discover equality rules for different attributes based on the feature table.

   For each class C_i, for any two attributes j_1 and j_2 that relate the k_1-th value in the j_1-th attribute and k_2-th value in the j_2-th attribute, if $a_{i,j_1,k_1} = a_{i,j_2,k_2} = vote$, infer the following rule.

   $$A_{j_1} = T_A[i, j_1, k_1] \quad \leftrightarrow \quad A_{j_2} = T_A[i, j_2, k_2].$$

   * The next highest concept is the concept one level below the most generalized concept "*any*". □

Algorithm 3.4 *Attribute-oriented algorithm for discovering inheritance rules associated with concepts for different levels in the concept hierarchy.*

Input (i) the prime relation obtained by Algorithm 3.1, and (ii) the concept hierarchy tables. (iii) the attribute name ANAME (we intend to learn rules associated with the concept hierarchy for attribute ANAME)
Output A set of inheritance rules associated with concepts at different levels in the concept hierarchy of attribute ANAME.
Method.

1. Attach one class attribute to the prime relation (called E-attribute, E means extra).

2. Extract the concept hierarchy H for the attribute ANAME from the concept hierarchy tables

3. (Iterative Step) descend one level starting from the next highest generalized concept in the concept hierarchy H until reaching the desired level of the concept hierarchy. At each descent do the following:

(a) Fill the E-attribute with the higher concept value and the corresponding attribute (attribute ANAME) with the concept value one level down of the E-attribute value in the concept hierarchy H.

(b) Extract the related data, and store them in the temporary relation.

(c) Project off the corresponding attributes which have the same values for all the low level concepts within the same higher concept from the temporary relation.

(d) Find the inheritance rules: for each temporary relation, those remaining attributes which have different values for different lower level concepts but within the same higher concept category will be chosen as the component to form the corresponding inheritance rule. □

3.2 An Example

In this section, we use a data set from [WiH84] to demonstrate algorithm 3.3 and algorithm 3.4. Given the animal world relation shown in Table 3.1 and the concept hierarchy for the attribute "Animal" depicted in Figure 3.2, Algorithm 3.3 is demonstrated as follows:

First step: Applying algorithm 3.1 to Table 3.1, results in the prime relation of Table 3.2. Next, further generalize Table 3.2 to the generalized relation as shown in Table 3.3.

Second step: Extract the feature table based on the attribute "Animal" depicted in Table 3.4.

Third step: Examine the values in the feature table; there are three classes for animal category mammal, bird and other. For $Class = mammal$ and $Hair = yes$, we have $a_{1,1,1} = 4$, $b_{1,1,1} = c_{1,1,1} = 1$ because $Class = mammal$ appears four times, and the total tuples for $Class = mammal$ is four. However $Hair = yes$ appears only four times in the entire table, so a rule can be inferred as follows:
$Hair = yes \leftrightarrow Class = mammal.$

similarly we obtain:

$(Milk = yes) \leftrightarrow (Class = mammal)$
$(Class = mammal) \rightarrow (Feet = claw \vee hoof) \wedge (Eats = meat \vee grass)$

for Class=bird:
$(Feather = yes) \leftrightarrow (Class = bird)$
$(Class = bird) \rightarrow (Feet = claw \vee web) \wedge (Eats = grain \vee fish \vee meat)$

Fourth step: Simplify the above rules; count the number of values appearing as characteristic values for the attribute and compare them with the total number of distinct values for the attribute. If the difference is larger than some threshold (for example, 2) then the "not" operator is introduced to the rules to simplify the forms of the discovered rules.

Take the following rule as an example.
$(Class = bird) \rightarrow (Feet = claw \vee web) \wedge (Eats = grain \vee fish \vee meat)$.

Since there are four distinct values: *meat, grass, grain* and *fish* for the attributes *Eats* and *Eats* takes three values out of four in the above rule, we can use $(Eats \neq grass)$ instead of $(eats = grain \vee fish \vee meat)$ as a component for this rule. Thus the rule is simplified as
$(Class = bird) \rightarrow (Feet \neq hoof) \wedge (Eats \neq grass)$.

similarly, the rule:
$(Class = mammal) \rightarrow (Feet = claw \vee hoof) \wedge (Eats = meat \vee grass)$
can be simplified as
$(Class = mammal) \rightarrow (Feet \neq web) \wedge (Eats = meat \vee grass)$

The last step is to analyze the data between different attributes and find the relationship between them to infer equality rules: for example, for Hair=yes, Feather=no,
$(Hair = yes) \leftrightarrow (Feather = No)$
$(Hair = yes) \leftrightarrow (Milk = yes)$
:
$(Feathers = yes) \leftrightarrow (Milk = No)$

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	E
cmammal	Y	pointed	forward	N	claw	meat	Y	N	Y	mammal
ungulate	Y	blunted	side	N	hoof	grass	Y	N	Y	mammal
nonflyb	N	N	side	Y	claw	grain	N	N	N	bird
nonflyb	N	N	side	Y	web	fish	N	N	N	bird
flyingb	N	N	side	Y	claw	grain	N	Y	N	bird
flyingb	N	N	forward	Y	claw	meat	N	Y	N	bird
viper	N	pointed	forward	N	N	meat	N	N	N	other

Table 3.5: A temporary relation after the substitution

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	E
cmammal	Y	pointed	forward	N	claw	meat	Y	N	Y	mammal
ungulate	Y	blunted	side	N	hoof	grass	Y	N	Y	mammal

Table 3.6: A temporary relation for mammal

Next we demonstrate the usefulness of Algorithm 3.4. The prime relation table is illustrated in Table 3.2 and the concept hierarchy for "Animal" is shown in Figure 3.2.

Attach the E_attribute to the Table 3.2 as shown as the right most column in Table 3.5, we do this by putting the values of the next higher-level concept (level 2) in Figure 3.2 for attribute E and the corresponding animal value in level 3. For example, if the E attribute value is *mammal*, then the corresponding animal value in the animal attribute should be *carnivorous mammal* and *ungulate*, resulting in the temporary relation shown in Table 3.5.

From Table 3.5, the data related to *mammal* and *bird* are extracted, resulting in the temporary Tables 3.6 and 3.7. Observe that *Hair, Feather, Milk, Fly* and *Swim* do not distinguish *mammals* but *Teeth, Eye, Eat and Feet* do distinguish *mammals* in Table 3.6. Thus the following rules are generated.
$(Class = mammal) \wedge (Teeth = pointed) \rightarrow (Animal = carnivorous\_mammal)$
$(Class = mammal) \wedge (Teeth = blunt) \rightarrow (Animal = ungulate)$

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	E
nonflyb	N	N	side	Y	claw	grain	N	N	N	bird
nonflyb	N	N	side	Y	web	fish	N	N	N	bird
flyingb	N	N	side	Y	claw	grain	N	Y	N	bird
flyingb	N	N	forward	Y	claw	meat	N	Y	N	bird

Table 3.7: A temporary relation for bird

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	E
tiger	Y	pointed	forward	N	claw	meat	Y	N	Y	cmammal
cheetah	Y	pointed	forward	N	claw	meat	Y	N	Y	cmammal

Table 3.8: A temporary relation for carnivorous mammal

$(Class = mammal) \wedge (Eye = forward) \rightarrow (Animal = carnivorous\_mammal)$
$(Class = mammal) \wedge (Eye = side) \rightarrow (Animal = ungulate)$
$(Class = mammal) \wedge (Feet = claw) \rightarrow (Animal = carnivorous\_mammal)$
$(Class = mammal) \wedge (Feet = hoof) \rightarrow (Animal = ungulate)$
$(Class = mammal) \wedge (Eats = meat) \rightarrow (Animal = carnivorous\_mammal)$
$(Class = mammal) \wedge (Eats = grass) \rightarrow (Animal = ungulate)$

In a similar manner for bird, based on Table 3.7, we can derive the following rules:
$(Class = bird) \wedge (Fly = yes) \rightarrow (Animal = flying\_bird)$
$(Class = bird) \wedge (Fly = no) \rightarrow (Animal = nonflying\_bird)$

Then continue the process, descending one level of the concept hierarchy, for the animal category: carnivorous mammal, ungulate, flying bird and non-flying bird, Table 3.8, 3.9, 3.10, 3.11 are obtained

Nothing interesting can be found based on Table 3.8 and Table 3.9. Because the information stored in the database is not enough to distinguish between the animals: tiger and cheetah, giraffe and zebra. But some interesting inheritance rules about flying and non-flying birds are discovered based on Table 3.10 and 3.11.

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	E
giraffe	Y	blunted	side	N	hoof	grass	Y	N	Y	ungulate
zebra	Y	blunted	side	N	hoof	grass	Y	N	Y	ungulate

Table 3.9: A temporary relation for ungulate

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	E
ostrich	N	N	side	Y	claw	grain	N	N	N	nonflyb
penguin	N	N	side	Y	web	fish	N	N	N	nonflyb

Table 3.10: A temporary relation for non-flying bird

$(Class = nonflying\_bird) \wedge (Feet = claw) \rightarrow (Animal = ostrich)$

$(Class = nonflying\_bird) \wedge (Eat = grain) \rightarrow (Animal = ostrich)$

$(Class = nonflying\_bird) \wedge (Feet = web) \rightarrow (Animal = penguin)$

$(Class = nonflying\_bird) \wedge (Swim = yes) \rightarrow (Animal = penguin)$

$(Class = flying\_bird) \wedge (Eye = side) \rightarrow (Animal = albatross)$

$(Class = flying\_bird) \wedge (Eats = grain) \rightarrow (Animal = albatross)$

$(Class = flying\_bird) \wedge (Eye = forward) \rightarrow (Animal = eagle)$

$(Class = flying\_bird) \wedge (Eats = meat) \rightarrow (Animal = eagle)$

3.3 Knowledge Discovery by Conceptual Clustering

In last section, we discussed the method which can find knowledge rules associated with concepts in different levels in the concept hierarchy. The method integrates a machine learning paradigm, especially *learning from example* techniques, with

Animal	Hair	Teeth	Eye	Feather	Feet	Eat	Milk	Fly	Swim	E
albatross	N	N	side	Y	claw	grain	N	Y	N	flyingb
eagle	N	N	forward	Y	claw	meat	N	Y	N	flyingb

Table 3.11: A temporary relation for flying bird.

database operations and extracts generalized data from actual data in the databases.

It is often necessary to incorporate higher level concepts in the learning process [Mit77]; candidate rules are restricted to formula with a particular vocabulary, that is, a basis set called the *conceptual bias*, permitting the learned rules to be represented in a simple and explicit form. Different levels of concepts can be organized into a taxonomy of concepts. The concepts in a taxonomy can be partially ordered according to general-to-specific ordering. Such a concept tree is specified using an IS-A hierarchy and stored in a relational table, the conceptual hierarchy table.

Although data in a relational database are usually well-formatted and modelled by semantic and data models [CCH91], the contents of the data may not be classified. For example, a chemistry database may store a large amount of experimental data in a relational format, but knowledge and effort are needed to classify the data in order to determine the intrinsic regularity of the data. Clearly, schemas and data formats are not equivalent to conceptual classes. Observation of the cognitive process of human discovery shows that humans tend to cluster the data into different classes based on conceptual similarity and then extract the characteristics from these classes. For example, by clustering experimental data based on the knowledge of chemists, interesting relationships among data can be discovered.

Previous studies on the method assume that the *pre-existence* of concept hierarchy information (provided by users, experts or data analysts). However, such information may not be always available in many applications. It is important to discover data regularities in the absence of concept hierarchy information. In this section, we develop the method further. The algorithm presented here combines the techniques of conceptual clustering and machine learning. The new method can cluster the data automatically, extract characteristics for different classes and then derive some knowledge rules according to the relationships between different classes.

3.3.1 Review of the Related Work

Conceptual clustering, originally developed by Michalski and Stepp [MiS83] as an extension to the process of numerical taxonomy, groups objects with common properties into clusters and extracts the characteristic of each cluster over a set of data objects. Currently, there are two views regarding conceptual clustering: one represents an extension to techniques of numerical taxonomy, whereas the other is a form of *learning-by-observations* or *concept formation* as distinct from methods of *learning-from-examples* or *concept identification*. The clustering algorithms which have been framed as extensions to the numerical taxonomy techniques include CLUSTER/2 [MiS83] and COBWEB [Fis87]; whereas those which can be viewed as an extension of *learning-by-observations* include HUATAO [ChF83] and Thought/KD1 [HoM91].

3.3.2 An Approach to Concept Clustering

Our method is divided into three phases. Phase 1 uses a numerical taxonomy to classify the object set. Phase 2 assigns conceptual descriptions to object classes. Phase 3 finds the hierarchical, inheritance and domain knowledge based on different relationships among classes. For a numerical taxonomy, various measures of similarity have been proposed. Most of them are based on a Euclidean measure of distance between numerical attributes. Consequently, the algorithm works well only on numerical data. Many database applications use non-numerical data. A new measure is proposed using the number of common attribute values in two data sets S_1 and S_2 as a similarity measurement, called *sim_value*(S_1, S_2). Notice that for any data set S, we set *sim_value*$(S, S) = 0$.

Algorithm 3.5 *Conceptual Data Clustering [CDC]*

Input. A set of data stored in the relational table.
Output. A cluster hierarchy of the data set.
Method.

1. **Preliminary**: Generalize attributes to a "desirable form" [Hux94]. For example, for the attribute "age" in an employer database, the substitution of

different age values into a small number of distinct higher level concepts, such as "young", "middle-aged", "old", etc. will make the descriptions concise and meaningful.

2. **Concept clustering**:

```
candidate_set := the data set obtained at Step 1.
repeat
        for each pair of $S_1$ and $S_2$ in candidate_set, calculate $sim\_value(S_1, S_2)$.
        form clusters for the candidate_set based on a threshold for sim_value.
            (Note: The threshold varies for different candidate_sets and can be
            set by user/expert or determined by the analysis of sim_value distri-
            bution).
        remove redundant clusters.
        if there is a new cluster produced
        then form the hierarchy based on the new and untouched* clusters
        candidate_set := the new cluster $\cup$ the untouched clusters
until candidate_set = $\phi$.
```

*Note: An untouched cluster is a cluster which is not a component of any newly formed cluster.

Given a set of data, suppose that the data is clustered into a hierarchy as illustrated in Figure 3.3 after phase 1. In Figure 3.3, H's denote the clusters in the hierarchy, $H_{i,j}$ is a subclass of H_i ($1 \le i \le k$, where k is the number of clusters in level 2). Let the conceptual descriptions assigned to these classes be D_1, ..., D_k, $D_{1,1}$, $D_{1,l}$, ..., $D_{k,1}$, ..., $D_{k,m}$, ..., and so on. The values of $k, l, \ldots, m$ depend on the actual data set.

Three kinds of knowledge rules can be discovered from object classes: (1) *hierarchical knowledge rules*, (2) *the relationship between different attributes* and (3) *inheritance knowledge rules.*

For rule formation, there are three algorithms of knowledge discovery: *Hierarchical Knowledge Discovery (HKD), Attribute Knowledge Discovery (AKD)* and *Inheritance*

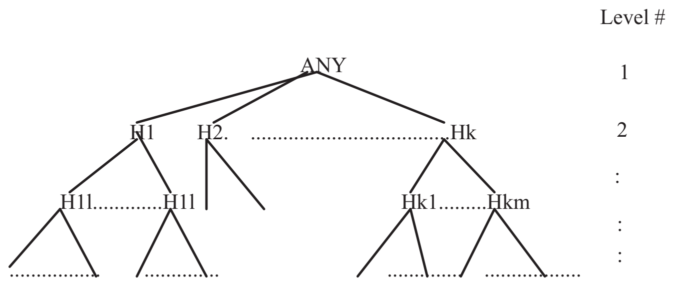


Figure 3.3: Conceptual hierarchy

Knowledge Discovery (IKD) [Hux94]. For HKD, new rules are discovered by finding all of the possible implications between the descriptions of clusters in a cluster and those in its father cluster, namely $D_{i,j} \rightarrow D_i$. For AKD, the algorithm just looks for the characteristic description for each cluster, based on the relationship on different attribute values, then gives the result in terms of a logically equivalent form. For IKD, which is a modification of HKD, labels are used, which are either explicitly defined by users/experts in terms of domain knowledge or labels are produced automatically by the system.

Cluster labelling plays an important role in knowledge discovery. The new rules discovered can be formed as

$D_1 \& D_{i,j} \& \ldots \& D_{i,j,\ldots,k,l} \rightarrow LABEL(H_{i,j,\ldots k,l})$, or

$LABEL(H_{i,j,..k}) \& D_{i,j,\ldots k,l} \rightarrow LABEL(H_{i,j,\ldots k,l})$

where the condition part of the rule consists of the conjunction of the description of the current cluster and the label of its father's cluster.

For example, given the animal world depicted in Table 3.12, which is viewed as the data set that was passed through the preliminary step.

The data in row 1 means that a tiger is a animal with hair, pointed teeth, forward eyes, claw feet, and no feather, it gives milk and cannot fly but can swim.

In Phase 1, the clustering algorithm CDC is applied to classify the data in Table 3.12. After the first iteration, the number of common attribute values between each pair of data is computed in Table 3.13. For example, the number "9" in row 1, column 2 is computed by counting the number of common attributes between the data set in row 1 and row 2 of Table 3.12.

#	Animal	Hair	Teeth	Eye	Feather	Feet	Eat	M	Fly	Swim
1	tiger	Y	pointed	forward	N	claw	meat	Y	N	Y
2	cheetah	Y	pointed	forward	N	claw	meat	Y	N	Y
3	giraffe	Y	blunt	side	N	hoof	grass	Y	N	N
4	zebra	Y	blunt	side	N	hoof	grass	Y	N	N
5	ostrich	N	N	side	Y	claw	grain	N	Y	N
6	penguin	N	N	side	Y	web	fish	N	N	N
7	albatross	N	N	side	Y	claw	grain	N	Y	Y
8	eagle	N	N	forward	Y	claw	meat	N	Y	N

Table 3.12: The animal world

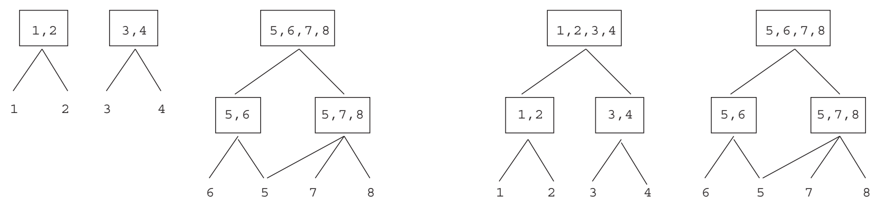
#	1	2	3	4	5	6	7	8
1	0	9	4	4	2	2	1	3
2	9	0	4	4	2	2	1	3
3	4	4	0	9	3	1	2	1
4	4	4	9	0	3	1	2	1
5	2	2	3	3	0	7	8	6
6	2	2	1	1	7	0	5	5
7	1	1	2	2	8	5	0	7
8	3	3	1	1	6	5	7	0

Table 3.13: Number of common attribute values after 1st iteration

Suppose 6 is chosen as the threshold sim_value, the algorithm CDC produces 8 clusters (1,2), (2,1), (3,4), (4,3), (5,6,7,8), (6,5), (7,5,8), (8,5,7). Thus, 5 distinct clusters (1,2), (3,4), (5,6,7,8), (5,6), (5,7,8) are formed after deleting redundant ones. A hierarchy is formed as depicted in Figure 3.4(a).

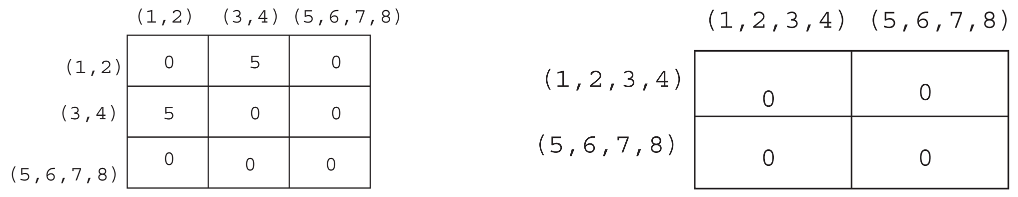
Next, the algorithm CDC is applied to (1,2), (3,4), (5,6,7,8). CDC calculates the similarity for the three clusters (1,2), (3,4), (5,6,7,8). The common attribute values are presented in Figure 3.5(a). Let 5 be the threshold value at this iteration. This results in the hierarchy shown in Figure 3.4(b).

Finally, the algorithm CDC is applied to (1,2,3,4), (5,6,7,8). After the third iteration, the common attribute values between these two clusters are presented in Figure 3.5(b) and the resultant conceptual hierarchy is illustrated in Figure 3.6. Notice that



(a) 1st iteration (b) 2nd iteration

Figure 3.4: Concept hierarchy



	(1,2)	(3,4)	(5,6,7,8)
(1,2)	0	5	0
(3,4)	5	0	0
(5,6,7,8)	0	0	0

(a) iteration 2

	(1,2,3,4)	(5,6,7,8)
(1,2,3,4)	0	0
(5,6,7,8)	0	0

(b) iteration 3

Figure 3.5: # of common attribute value

the characteristic descriptions of each cluster are the common values for all the data in the cluster.

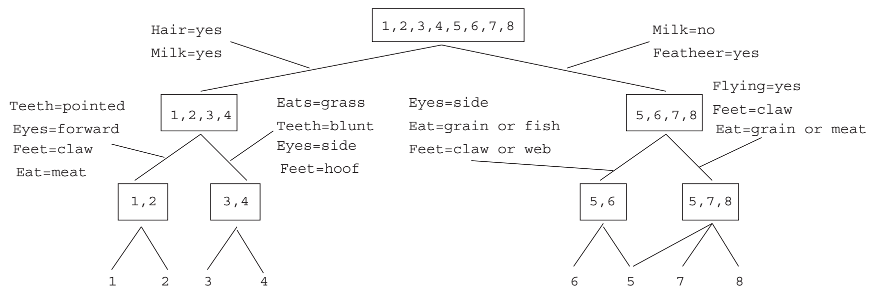


Figure 3.6: Conceptual hierarchy after 3rd iteration

In phase 3, the three Knowledge Discovery Algorithms HKD, AKD, and IKD are applied to the hierarchy depicted in Figure 3.6, respectively, resulting in three sets of rules as depicted in Tables 3.14(a), 3.14(b) & 3.15.

By substituting the labels by the names given by an expert as shown in Table 3.16, a set of meaningful rules can be obtained as shown in Table 3.17.

#	Knowledge Rules discovered by HKD
1	Feet=hoof $\rightarrow$ Milk=yes
2	Teeth=pointed $\vee$ blunt $\rightarrow$ Milk=yes
3	Eat=grass $\rightarrow$ Milk=yes
4	Feet =hoof $\rightarrow$ Hair=yes
5	Teeth=pointed $\vee$ blunt $\rightarrow$ Hair=yes
6	Eat=grass $\rightarrow$ Hair=yes

(a)

#	Knowledge Rules discovered by AKD
1	Hair=yes $\leftrightarrow$ Milk=yes
2	Feather=yes $\leftrightarrow$ Milk=no

(b)

Table 3.14: (a)Hierarchical knowledge rules; (b)Equivalence rules

#	Knowledge Rules discovered by IKD
1	Label(1,2,3,4,5,6,7,8) $\wedge$ (hair=yes $\vee$ Milk=yes) $\rightarrow$ Label(1,2,3,4)
2	Label(1,2,3,4,5,6,7,8) $\wedge$ (Feather=yes $\vee$ Milk=no) $\rightarrow$ Label(5,6,7,8)
3	Label(1,2,3,4) $\wedge$ (Teeth=pointed $\vee$ Eye=forward $\vee$ Feet=claw $\vee$ Eats=meat) $\rightarrow$ Label(1,2)
4	Label(1,2,3,4) $\wedge$ (Teeth=blunt $\vee$ Eye=side $\vee$ Feet=Hoof $\vee$ Eats=grass) $\rightarrow$ Label(3,4)

Table 3.15: Inheritance knowledge rules

Labels given by system	Names given by expert/user
Label(1,2,3,4,5,6,7,8)	Animals
Label(1,2,3,4)	mammal
Label(5,6,7,8)	bird
Label(1,2)	carnivorous mammal
Label(3,4)	ungulate
Label(5,6)	non-flying bird
Label(5,7,8)	meaningless cluster

Table 3.16: Names list

#	After renaming the labels by experts or users
1	(Thing=animal) $\wedge$ (hair=yes $\vee$ Milk=yes) $\rightarrow$ mammal
2	(Thing=animal) $\wedge$ (Feather=yes $\vee$ Milk=no) $\rightarrow$ bird
3	(Animal=mammal) $\wedge$ (Teeth=pointed $\vee$ Eye=forward $\vee$ Feet=claw $\vee$ Eats=meat) $\rightarrow$ carnivorous mammal
4	(Animal=mammal) $\wedge$ (Teeth=blunt $\vee$ Eye=side $\vee$ Feet=Hoof $\vee$ Eats=grass) $\rightarrow$ ungulate

Table 3.17: A set of meaningful rules after substitution

Chapter 4

Rough Sets and A Generalized Rough Set Model

Much attention has been paid recently by the expert systems research and machine learning community to the acquisition of knowledge and reasoning under vagueness and incompleteness [Paw91, Slo92, HCH93b]. Vagueness may be caused by the ambiguity of exact meaning of the terms used in the knowledge domain, uncertainty in data (e.g. due to noise), and uncertainty in knowledge itself (e.g. due to doubtful connection between the antecedent and the consequent in an inferred rule) [Zia91]. Incompleteness may be caused by the unavailability of data or the incompleteness of the knowledge of human beings. To deal with vagueness, expert systems require techniques other than classical logic. Statistics is the best tool for handling likelihood. However, many methods needed when using probability in an expert systems require an estimate of probabilities, sometimes without even recourse to relative frequencies. Estimates are likely to be very inaccurate. Expert systems based on statistical techniques have theoretical weaknesses cited by many authors [Zia93]. Another way to deal with uncertainty is to use fuzzy logic, based on Zadeh's theory of fuzzy sets [Zad65]. The basic tools of the theory are possibility measures. There is extensive literature on fuzzy logic which also discusses some of the problems with this theory. The basic problem of fuzzy set theory is the determination of the grade of membership or the value of possibility [Grz88].

In the past decade, Z. Pawlak [Paw82] introduced a new tool to deal with vagueness, called the "rough set model". Fuzzy set theory and rough set theory are independent and offer alternative approaches to uncertainty, as was shown in [Paw85]. The main advantage of rough set theory is that it does not need any preliminary or additional information about data (like probability in statistics, grade of membership, or the value of possibility in fuzzy set theory). Other advantages of the rough set approach include its ease of handling and its simple algorithms [Slo92].

Rough set theory has been successfully implemented in knowledge-based systems in medicine and industry [Grz88]. The rough set philosophy is based on the idea of *classification*. The most important issue addressed in the rough set theory is the idea of imprecise knowledge. In this approach, knowledge is imprecise if it contains imprecise concepts. It turns out that imprecise concepts can be however defined approximately in the available knowledge by employing two precise concepts called their *lower* and *upper approximation*. The lower approximation of a concept consists of all objects which surely belong to the concept whereas the upper approximation of the concept consists of all objects which possibly belong to the concept in question. The difference between the lower and upper approximation is a *boundary region* of the concept and consists of all objects which cannot be classified with certainty to the concept or its complement employing available knowledge. In this chapter we introduce the principal ideas of rough set from Pawlak [Paw82] and present a generalized model of rough set to handle uncertainty information.

4.1 Principal Concepts of Rough Set

4.1.1 Information System

By an information system S, we mean $S = \{U, A, V, f\}$, where U is a finite set of objects, $U = \{x_1, x_2, ..., x_n\}$, A is a finite set of attributes, the attributes in A is further classified into two disjoint subsets, *condition* attributes C and *decision* attributes D, $A = C \cup D$

$$V = \bigcup_{p \in A} V_p$$

and V_p is a *domain* of attribute p.

f: U $\times$ A $\rightarrow$ V is a total function such that $f(x_i, q) \in V_q$ for every $q \in$ A, $x_i \in$ U.

Let $IND \subset A$, $x_i, x_j \in$ U. We define a binary relation $\widetilde{IND}$, called an *indiscernibility relation*, as follow:

$$\widetilde{IND} = \{(x_i, x_j) \in U \times U : for\ every\ p \in IND\ p(x_i) = p(x_j)\}$$

We say that x_i and x_j are indiscernible by a set of attributes IND in S iff $p(x_i) = p(x_j)$ for every $p \in IND$. One can check that $\widetilde{IND}$ is an equivalence relation on U for every $IND \subset A$. Equivalence classes of relations are called IND-elementary sets in S. A-elementary sets are called atoms of S. Information system S is selective iff all atoms in S are one element sets, i.e. A is an identity relation.

An information system provides information about the real-world objects. However, information about objects may not be sufficient to characterize objects without ambiguity. Thus some objects are characterized by the same condition values. Two objects are indiscernible whenever they have the same values for all conditions. Objects can be characterized by some selected features represented by attributes. In general, information about objects expressed in this way is not sufficient to characterize objects uniquely, as any two objects are indistinguishable from one another whenever they assume the same values for all the attributes under consideration [Grz88].

A relational database may be considered as an information system in which columns are labelled by attributes, rows are labelled by the objects and the entry in column p and row x has the value $p(x)$. Each row in the relational table represents *information* about some object in U. The difference is that the entities of the information systems do not need to be distinguished by their *attributes* or by their relationship to entities of another type. In the relational database, one attribute is identified as a *decision* attribute (learning task), and the other attributes are the *condition* attributes. We adopt the view that a relational database is a selective information system and will use the term *relational database* and *information system* interchangeably in this work.

4.1.2 Approximation Space

For the information system $S = \{U, A, V, f\}$, and $IND \subset A$ derives an equivalence relation (*indiscernibility relation*) on U, an ordered pair $AS = (U, \widetilde{IND})$ is called an *approximation space*. For any element x_i of U, the equivalence class of x_i in relation $\widetilde{IND}$ is represented as $[x_i]_{IND}$. Equivalence classes of $\widetilde{IND}$ are called *elementary sets in AS* because they represent the smallest discernible groups of objects.

Any finite union of elementary sets in AS is called a *definable set in AS*.

Let $X \subset U$, we want to define X in terms of *definable sets* in AS, thus we need to introduce the following notions cited from [Paw82]:

(i) The lower approximation of X in AS is defined as:

$$\underline{IND}X = \{x_i \in U | [x_i]_{IND} \subset X\}$$

$\underline{IND}$X is the union of all those elementary sets each of which is contained by X. For any $x_i \in \underline{IND}$X, it is certain that it belongs to X.

(ii) The upper approximation of X in AS is defined as

$$\overline{IND}X = \{x_i \in U | [x_i]_{IND} \cap X \neq \emptyset\}$$

$\overline{IND}$X is the union of those elementary sets each of which has a non-empty intersection with X. For any $x_i \in \overline{IND}$X, we can only say that x_i is possible belong to X.

(iii) The set $\overline{IND}$X-$\underline{IND}$X is called the IND-doubtful region of IND in $(U, \widetilde{IND})$. For any $x_i \in$ U, if x_i in $\overline{IND}$X-$\underline{IND}$X, it is impossible to determine that x_i belong to X or not based on the descriptions of the elementary sets of $\widetilde{IND}$.

The following diagram Figure 4.1 illustrates the relationships among them.

The lower approximation of X in AS is the greatest definable set in AS, contained in X. The upper approximation of X in AS is the least definable set in AS containing X. Let X and Y be subset of U, lower and upper approximations in AS have the following properties [Paw82]:

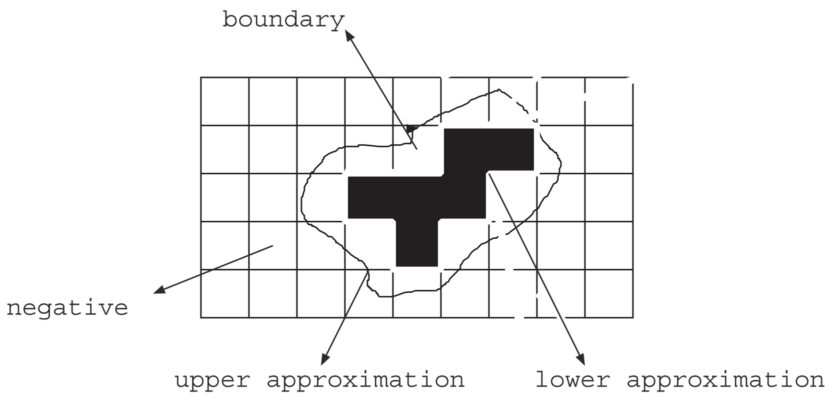


Figure 4.1: The diagram of rough set model

$\underline{IND}$X $\subseteq$ $\overline{IND}$X, $\underline{IND}$U=$\overline{IND}$U=U, $\underline{IND}\emptyset$=$\overline{IND}\emptyset$=$\emptyset$
$\underline{IND}$(X $\cup$ Y) $\supseteq$ $\underline{IND}$X $\cup$ $\underline{IND}$Y, $\overline{IND}$(X $\cup$ Y) = $\overline{IND}$X $\cup$ $\overline{IND}$Y,
$\underline{IND}$(X $\cap$ Y) = $\underline{IND}$X $\cap$ $\underline{IND}$Y, $\overline{IND}$(X $\cap$ Y) $\subseteq$ $\overline{IND}$X $\cup$ $\overline{IND}$Y,
$\underline{IND}$(-X)=-$\overline{IND}$X, $\overline{IND}$(-X)=-$\underline{IND}$X
$\underline{IND}$($\underline{IND}$X)=$\overline{IND}$($\underline{IND}$X)=$\underline{IND}$X, $\overline{IND}$($\overline{IND}$X)=$\underline{IND}$($\overline{IND}$X)=$\overline{IND}$X

Example 4.1 Let us consider a generalized car relation given by Table 4.1. $U = \{1, 2, 3, ..., 14\}$ is the collection of cars. Suppose we choose $IND = \{cyl, power, weight\}$ and $D = mileage$ is the decision attribute. Thus the decision attribute consists of two concepts D_{MEDIUM} = "$mileage = MEDIUM$" and D_{HIGH} = "$mileage = HIGH$",

$$D_{MEDIUM} = \{1, 2, 3, 4, 5, 6, 7\}$$

$$D_{HIGH} = \{8, 9, 10, 11, 12, 13, 14\}$$

we have the equivalence classes of $\widetilde{IND}$ as below

$E_1 = \{1, 6\}$, $E_2 = \{2\}$, $E_3 = \{3, 4, 10, 13, 14\}$, $E_4 = \{5, 7, 11\}$, $E_5 = \{8\}$, $E_6 = \{9\}$, $E_7 = \{12\}$

The corresponding lower approximation and upper approximation of D are as follows

obj#	Make_model	cyl	door	displace	compress	power	trans	weight	mileage
1	USA	6	2	MEDIUM	HIGH	HIGH	AUTO	MEDIUM	MEDIUM
2	USA	6	4	MEDIUM	MEDIUM	MEDIUM	MANUAL	MEDIUM	MEDIUM
3	USA	4	2	SMALL	HIGH	MEDIUM	AUTO	MEDIUM	MEDIUM
4	USA	4	2	MEDIUM	MEDIUM	MEDIUM	MANUAL	MEDIUM	MEDIUM
5	USA	4	2	MEDIUM	MEDIUM	HIGH	MANUAL	MEDIUM	MEDIUM
6	USA	6	4	MEDIUM	MEDIUM	HIGH	AUTO	MEDIUM	MEDIUM
7	USA	4	2	MEDIUM	MEDIUM	HIGH	AUTO	MEDIUM	MEDIUM
8	USA	4	2	MEDIUM	HIGH	HIGH	MANUAL	LIGHT	HIGH
9	JAPAN	4	2	SMALL	HIGH	LOW	MANUAL	LIGHT	HIGH
10	JAPAN	4	2	MEDIUM	MEDIUM	MEDIUM	MANUAL	MEDIUM	HIGH
11	JAPAN	4	2	SMALL	HIGH	HIGH	MANUAL	MEDIUM	HIGH
12	JAPAN	4	2	SMALL	MEDIUM	LOW	MANUAL	MEDIUM	HIGH
13	JAPAN	4	2	SMALL	HIGH	MEDIUM	MANUAL	MEDIUM	HIGH
14	USA	4	2	SMALL	HIGH	MEDIUM	MANUAL	MEDIUM	HIGH

Table 4.1: A generalized car relation

$$\underline{IND}(D_{MEDIUM}) = \{E_1, E_2\} = \{1, 6, 2\}$$

$$\overline{IND}(D_{MEDIUM}) = \{E_1, E_2, E_3, E_4\} = \{1, 6, 2, 3, 4, 10, 13, 14, 5, 7, 11\}$$

$$\underline{IND}(D_{HIGH}) = \{E_5, E_6, E_7\} = \{8, 9, 12\}$$

$$\overline{IND}(D_{HIGH}) = \{E_3, E_4, E_5, E_6, E_7\} = \{3, 4, 10, 13, 14, 5, 7, 11, 8, 9, 12\}$$

4.1.3 Core and Reducts of Attributes

In many applications, the set of objects is classified into a disjoint family of classes based on the values of the decision attribute, and we want to determine each class in terms of features of corresponding condition attributes belonging to each class. In most cases, classes are determined by several or even one attribute, not by small difference in all the attributes in the databases. This is also consistent with the cognitive process of human discovery, because people often have difficulty in taking more than a few attributes into account and tend to focus on a few important attributes. The rough set theory provides us the tool to deal with this problem. Core and reduct are the two fundamental concepts of rough set. A reduct is the essential part of an information system which can discern all objects discernible by the original information system. A core is the common parts of all the reducts.

Let $S = \{U, A, V, f\}$ be an information system, $A = C \cup D$, B $\subset$ C, a positive region B in $\tilde{D}$, $POS_B(D)$, is defined as

$$POS_B(D) = \cup\{\underline{B}X : X \in \tilde{D}\}$$

The positive region $POS_B(D)$ includes all objects in U which can be classified into classes of $\tilde{D}$ without error based on the classification information in $\tilde{B}$.

We say that the set of attributes D depends in degree k $(0 \leq k \leq 1)$ on the subset R of C in S if

$$k(R, D) = card(POS_R(D))/card(U)$$

The value $k(R, D)$ provides a measure of dependency between R and D.

Definition 4.1 *An attribute $p \in B$ is superfluous in B with respect to D if $POS_B(D) = POS_{B-\{p\}}(D)$; otherwise p is indispensable in B with respect to D.*

If an attribute is superfluous in the information system, it can be removed from the information system without changing the dependency relationship of the original system. While an indispensable attribute carries the essential information about objects of the information system. It should be kept if you do not want to change the dependency relationship of the original system.

Definition 4.2 *If every attribute of B is indispensable with respect to D, then B is orthogonal with respect to D.*

Definition 4.3 *$B \subset C$ is defined as reduct in S if B is orthogonal with respect to D and $POS_C(D) = POS_B(D)$*

The reduct of C is a nonredundant subset of attributes that discerns all object discernible by the entire set of attributes. Usually, C may have more than one reduct.

Definition 4.4 *The set of all attributes belonging to the intersection of all reducts of C with respect to D is called the core of C, denoted as $CORE(C, D)$.*

The concept of the core can be used as the starting point for computation of reducts.

4.2 A Generalized Rough Sets Model

The theory of rough sets, as proposed by Pawlak, provides a formal tool for dealing with imprecise or incomplete information. It has been successfully applied in machine learning, expert system design, and knowledge representation [Slo92]. Substantial progress has been achieved in understanding practical implications and limitations of this approach. In particular, the inability to model uncertain information was one limitation frequently emphasized by researchers. It may be inadequate to deal with situations in which the statistical information plays an important role. Consider, for example, two equivalence classes E_1, E_2 in the partition $\widetilde{IND}$ such that each has 100 elements. Suppose only a single element in E_1 belongs to X, and only a single element in E_2 does not belong to X. In the original rough set model, these two equivalence classes are treated in the same way and both will be included in the doubtful region. From a statistical point of view, such an identical treatment of E_1 and E_2 does not seem reasonable. Moreover, the observation that only one element in E_1 belongs to X may be a result of noise. Therefore, the original rough set model can be sensitive to noise often encountered in many real-world applications [WZY86]. This limitation severely reduces the applicability of the rough set approach to problems which are more probabilistic in nature. An attempt to overcome this restriction was reported in [PWZ88]. However, the proposed generalization was based on strong statistical assumptions and did not directly inherit all of the useful properties of the original model of the rough set.

In this section, a new generalized version of the rough set model is proposed. The generalized rough set model is introduced to overcome these shortcomings by incorporating the available statistical information. The generalized rough sets model is an extension of the concept of the variable precision rough sets model [Zia93a]. Our new approach will deal with the situations where uncertain objects may exist, different objects may have different importance degrees, and different classes may have different noise ratios. The standard rough set model and the VP-model of rough sets [Zia93b] become a special case of the GRS-model. The primary advantage of the GRS-model is that it modifies the traditional rough sets model to work well in a noisy environment.

4.2.1 Uncertain Information Systems (UIS)

In general, an information system represents objects crisply. That is, for a given object in the database, and a given property (attribute-value pair), there is no uncertainty whether or not the object has that property. This certainty is restrictive. Such a representation restricts our representation power in two ways. First, all objects in the universe must be represented by a uniform representation. Second, representative power is also restrictive because the object representation is crisp, i.e. there is no room for the expression of degree in an object's representation. That is, an object either has, or does not have a property.

To manage objects with uncertainty and different importance degrees, we introduce an uncertain information system (UIS) based on the information systems defined by Pawlak [Paw82]. In the uncertain information system, each object is assigned an uncertainty u and an importance degree d. The uncertainty u is a real number in the range from 0.0 to 1.0. If uncertainty u equals 1.0, it represents a completely positive object. If uncertainty u equals 0.0, it represents a completely negative object. The importance degree d represents the importance of the object in the information system. The $d \times u$ induces the positive class and $d \times (1 - u)$ induces the negative class in the uncertain information system. An example collection of classes (objects) of an uncertain information system is shown in Table 4.2. The uncertain information system (UIS) is defined as follows:

Definition 4.5 *$UIS =< U, C, D, \{VAL_a\}_{a \in C}, u, d >$ is an uncertain information system, where U is a non-empty set of object, C is an non-empty set of condition attributes, D is a decision attribute with uncertainty u. VAL_a is a domain of a condition attribute "a" with at least two elements. Each condition attribute $a \in C$ can be perceived as a function assigned a value $a(obj) \in VAL_a$ to each object $obj \in U$. $d(obj)$ is a function assigned an importance degree to each object $obj \in U$. Every object which belongs to U is therefore associated with a set of certain values corresponding to the condition attribute C, an uncertain value corresponding to the*

OBJ	**c1**	**c2**	**dec**	**d**
e_1	0	0	0.75	4
e_2	0	1	0.67	3
e_3	0	2	0.35	4
e_4	1	0	0.75	4
e_5	1	1	0.67	3
e_6	1	2	0.35	4

Table 4.2: An uncertain information system

decision attribute D and a real number corresponding to the importance degree d of the object.

Example 4.2 In the Table 4.2, we have a set of objects $U = \{e_i\}$, where $(i = 1, 2, ..., 6)$ are the rows of the table. The set of condition attributes is $C = \{c1, c2\}$ and the domains of condition attributes C are $V_{c1} = \{0, 1\}$, $V_{c2} = \{0, 1, 2\}$, and the decision attribute is $D = \{dec\}$ with uncertainty value $u_{dec_i} = \{0.75, 0.67, 0.35, 0.75, 0.67, 0.35\}$ $(i = 1, 2, ..., 6)$. For each object, an importance degree d is assigned and the set of importance degree is $d(obj_i) = \{4, 3, 4, 4, 3, 4\}$ $(i = 1, 2, ..., 6)$.

4.2.2 Noise Tolerance in Uncertain Information Systems

To manage noise in uncertain information systems, we adopt the concept of relative classification error which was introduced by Ziarko [Zia93a]. The main idea is to draw some boundary region between positive region and negative region, according to some classification factors. The goal is to generate some *strong* rules which are almost always correct. In the real world, each class (positive class and negative class) in the information system may contain different noise. Two classification factors P_β and N_β $(0.0 \leq P_\beta, N_\beta \leq 1.0)$ are introduced to solve this problem. P_β and N_β may be the same values and simultaneously exist, they can be determined by estimating noise degree in the positive region and the negative region respectively.

Let X be a non-empty subset of a finite universe U. The measure of the relative degree of misclassification of the set X with respect to the positive class P_{class} and negative class N_{class} defined as

$$C_P(X) = \frac{\sum(d_i \times (1-u_i))}{\sum d_i} \qquad if\ obj_i \in X, \quad X \subseteq OBJ$$

$$C_N(X) = \frac{\sum(d_i \times u_i)}{\sum d_i} \qquad if\ obj_i \in X, \quad X \subseteq OBJ$$

where $\sum d_i$ is the sum of importance degree of objects belonging to the set X, $\sum(d_i \times u_i)$ is the sum of inducing positive class degree of objects belonging to the set X, and $\sum(d_i \times (1-u_i))$ is the sum of inducing negative class degree of objects belonging to the set X.

$C_P(X)$ is defined as the ratio between the sum of inducing negative class degree of objects and the sum of importance degree of objects in the set X. $C_N(X)$ is defined as the ratio between the sum of inducing positive class degree of objects and the sum of importance degree of objects in the set X. If we classify objects belonging to the set X to the positive class, we may have an classification error rate $C_P(X)$. If we classify objects belonging to the set X to negative class, we may have an classification error rate $C_N(X)$.

Based on the measure of relative classification error one can define the set of objects X which belongs to the positive class if and only if the classification error $C_P(X)$ is less than or equal to given precision level P_β, or the negative class if and only if the classification error $C_N(X)$ is less than or equal to given precision level N_β. Thus,

$$P_{class} \supseteq X \qquad if\ only\ if\ C_P(X) \leq P_\beta$$

$$N_{class} \supseteq X \qquad if\ only\ if\ C_N(X) \leq N_\beta$$

otherwise, the set of situations X belongs to the boundary region.

Example 4.3 Assuming the same set of objects U as described by Table 4.2, and set $P_\beta = 0.30$, $N_\beta = 0.6$. The set of equivalence relation R is $R = \{X1, X2, ..., X6\}$, where $X1 = \{e1\}, X2 = \{e2\}$,..., and $X6 = \{e_6\}$. Thus

$$C_P(X1) = \frac{4 \times (1.0 - 0.75)}{4} = 0.25 \qquad C_N(X1) = \frac{4 \times 0.75}{4} = 0.75$$

Similarly,

$$C_P(X2) = \frac{3 \times (1.0 - 0.67)}{3} = 0.33 \qquad C_N(X2) = \frac{3 \times 0.67}{3} = 0.67$$

$$C_P(X3) = \frac{4 \times (1.0 - 0.35)}{4} = 0.65 \qquad C_N(X3) = \frac{4 \times 0.35}{4} = 0.35$$

$$C_P(X4) = \frac{4 \times (1.0 - 0.75)}{4} = 0.25 \qquad C_N(X4) = \frac{4 \times 0.75}{4} = 0.75$$

$$C_P(X5) = \frac{3 \times (1.0 - 0.67)}{3} = 0.33 \qquad C_N(X5) = \frac{3 \times 0.67}{3} = 0.67$$

$$C_P(X6) = \frac{4 \times (1.0 - 0.35)}{4} = 0.65 \qquad C_N(X6) = \frac{4 \times 0.35}{4} = 0.35$$

Now we can say

$$P_{class} = \{X1, X4\}$$

and

$$N_{class} = \{X3, X6\}$$

4.2.3 Set Approximation in the GRS-Model

In the original model of rough sets the approximation space is defined as a pair $A = (U, \widetilde{IND})$ which consists of a non-empty, finite universe of discourse U and the equivalence relation $\widetilde{IND}$ on U. The equivalence relation $\widetilde{IND}$, referred to as an indiscernibility relation, corresponds to a partitioning of the universe U into a collection of equivalence class or elementary sets $\widetilde{IND} = \{E_1, E_2, ..., E_n\}$. The elementary sets are the atomic components of given information systems. They correspond to the smallest groups of objects which are distinguishable in terms of the information used to represent them, e.g. in terms of object features and their values.

In the generalized rough set model objects which belong to an elementary set are perceived as identical, it may not be possible to determine set inclusion criteria for every subset of the universe U. We can consider some elementary sets in the upper approximation space with degree of classification error lower than given P_β and N_β factors. It means that this will draw some elementary sets of boundary area into the lower approximation space.

By using two classification factors P_β and N_β, we obtain the following generalization of the concept of rough approximation:

Let the pair $A = (U, \widetilde{IND}_{P,N})$ be an approximation space and $\widetilde{IND}_{P,N}$=$\{E_1$,E_2,...,$E_n\}$ be the collection of equivalence classes of the relation $\widetilde{IND}_{P,N}$. Let P_β and N_β be two real numbers as defined in previous section, such that $0.0 \leq P_\beta,\quad N_\beta \leq 1.0$. Given any arbitrary subset $X \subseteq OBJ$, its positive lower approximation $POS_P(X)$ is defined as a union of those elementary sets whose classification criteria guarantee that the relative error $C_P(E)$ of the set X will be less or equal to P_β,

$$POS_P(X) = \bigcup\{E \in \widetilde{IND}_{P,N} : C_P(E) \leq P_\beta\}$$

Its negative lower approximation $NEG_N(X)$ is defined as a union of those elementary sets whose classification criteria guarantee that the relative error $C_N(E)$ of the set X will be less or equal N_β,

$$NEG_N(X) = \bigcup\{E \in \widetilde{IND}_{P,N} : C_N(E) \leq N_\beta\}$$

Its upper approximation of the positive region $UPP_P(X)$ is defined as a union of those elementary sets whose classification criteria guarantee that the relative error $C_N(E)$ of the set X will be greater than or equal N_β,

$$UPP_P(X) = \bigcup\{E \in \widetilde{IND}_{P,N} : C_N(E) \geq N_\beta\}$$

Its upper approximation of the negative region $UPP_N(X)$ is defined as a union of those elementary sets whose classification criteria guarantee that the relative error $C_P(E)$ of the set X will be greater than or equal P_β,

$$UPP_N(X) = \bigcup\{E \in \widetilde{IND}_{P,N} : C_P(E) \geq P_\beta\}$$

The boundary region $BND_{P,N}(X)$ of the set X is the union of those elementary sets whose classification do not belong to the positive region and the negative region of the set X,

$$BND_{P,N}(X) = \bigcup\{E \in \widetilde{IND}_{P,N} : E \notin POS_P, NEG_N\}$$

Example 4.4 For the uncertainty information system in Table 4.2,

$$\begin{aligned} POS_P(D) = &\quad \{X1, X4\} \\ NEG_N(D) = &\quad \{X3, X6\} \\ UPP_P(D) = &\quad \{X1, X2, X4, X5\} \\ UPP_N(D) = &\quad \{X2, X3, X5, X6\} \\ BND_{P,N}(D) = &\quad \{X2, X5\} \end{aligned}$$

4.2.4 The Degree of Attribute Dependencies in the GRS-Model

To formally define the attribute dependency measure between the set of condition attributes $C \subset A$ and the set of decision attributes $D \subset A$ $(A = C \cup D)$, let $\tilde{C}$ denote the collection of equivalence classes of the relation $IND_{P,N}(C)$ and, similarly, let $\tilde{D}$ be a family of equivalence class of $IND_{P,N}(D) = \{P_{class}, N_{class}\}$. Given two classification factors P_β and N_β $(0.0 \leq P_\beta,\ N_\beta \leq 1.0)$ we say that the set of decision attributes D imprecisely depends on the set of condition attributes C to the degree $\gamma(C, D, P_\beta, N_\beta)$ if :

$$\gamma(C, D, P_\beta, N_\beta) = IMP(INT(C, D, P_\beta, N_\beta))/IMP(OBJ)$$

where $INT(C, D, P_\beta, N_\beta)$ is a union of positive and negative lower approximations of all elementary sets of the partition $\tilde{D} = \{P_{class}, N_{class}\}$ in the approximation space $(U,\ IND_{P,N}(C))$, and the $IMP(X)$ is an importance function assigning the sum of importance degree of objects in the set X, such that

$$IMP(OBJ) = \sum_{i=1}^{n} d_i \qquad\qquad obj_i \in OBJ$$

and

$$IMP(INT(C, D, P_\beta, N_\beta)) = \sum_{pos=1}^{a} d_{pos} + \sum_{neg=1}^{b} d_{neg},$$
$$obj_{pos} \in POS_P(X), obj_{neg} \in NEG_N(X)$$

We can transfer the above formula to:

$$\gamma(C, D, P_\beta, N_\beta) = \frac{\sum_{pos=1}^{a} d_{pos} + \sum_{neg=1}^{b} d_{neg}}{\sum_{i=1}^{n} d_i}$$

Informally speaking, the dependency degree $\gamma(C, D, P_\beta, N_\beta)$ of attributes D on the attributes C at the precision level P_β, N_β is the proportion of these objects $obj_i \in OBJ$ which can be classified into corresponding classes of the partition $\tilde{D}$ (positive class and negative class) with an error rate less than desired value (P_β, N_β) on the basis of the information represented by the classification $\tilde{C}$.

Example 4.5 Based on the uncertain information system given in Table 4.2, we can calculate the degree of dependency between condition attributes C and the decision attribute D with classification factors $P_\beta = 0.30$ and $N_\beta = 0.60$. From Example 4.4, we obtained the following:

$$POS_P(D) = \{X1, X4\}$$

$$NEG_N(D) = \{X3, X6\}$$

So that, the degree of dependency between C and D is,

$$\gamma(C, D, 0.30, 0.60) = \frac{4 + 4 + 4 + 4}{22} = 0.73$$

4.2.5 Attribute Reduct in the GRS-Model

Let $UIS =< U, C, D, \{VAL_a\}_{a\in C}, u, d >$ be an uncertain information system and $P \subseteq C$, and given classification factor P_β, N_β:

Definition 4.6 *An attribute $a \in P$ is redundant in P if $\gamma(P - \{a\}, D, P_\beta, N_\beta) = \gamma(P, D, P_\beta, N_\beta)$; otherwise the attribute a is indispensable*

Definition 4.7 *If all attribute $a_i \in P$ are indispensable in P, then P will be called* orthogonal

Definition 4.8 *A subset $P \subset C$ is called* reduct *of C in UIS iff P is* orthogonal *and $\gamma(P, D, P_\beta, N_\beta) = \gamma(C, D, P_\beta, N_\beta)$*

A relative reduct of the set of condition attributes will be defined as a nonredundant independent subset of condition attributes that discerns all objects which are discernable by the entire attribute set.

The GRS-reduct, or approximation reduct, of the set of condition attributes C with respect to a set of decision attributes D is a subset of $RED(C, D, P_\beta, N_\beta)$ of C which satisfies the following two criteria:

1. $\gamma(C, D, P_\beta, N_\beta) = \gamma(RED(C, D, P_\beta, N_\beta), D, P_\beta, N_\beta)$
2. no attribute can be eliminated for $RED(C, D, P_\beta, N_\beta)$ without affecting the first criteria

Example 4.6 Consider dropping the condition attribute $c1$ in Table 4.2 and set $P_\beta = 0.30$ and $N_\beta = 0.60$. The set of equivalence relation R is $\tilde{R} = \{X1, X2, X3\}$ where $X1 = \{e_1, e_4\}$, $X2 = \{e_2, e_5\}$ and $X3 = \{e_3, e_6\}$. So that,

$$C_P(X1) = \frac{2 \times 4 \times (1.0 - 0.75)}{8} = 0.25 \qquad C_N(X1) = \frac{2 \times 4 \times 0.75}{8} = 0.75$$

$$C_P(X2) = \frac{2 \times 3 \times (1.0 - 0.67)}{6} = 0.33 \qquad C_N(X2) = \frac{2 \times 3 \times 0.67}{6} = 0.67$$

$$C_P(X3) = \frac{2 \times 4 \times (1.0 - 0.35)}{8} = 0.65 \qquad C_N(X3) = \frac{2 \times 4 \times 0.35}{8} = 0.35$$

we obtain $POS_P(C^{'}) = \{X1\}$ and $NEG_P(C^{'}) = \{X4\}$ $(C^{'} = \{x2\})$. Thus, we can say

$$\gamma(C^{'}, D, 0.30, 0.60) = \frac{8+8}{22} = 0.73$$

From example 4.5, we know that $\gamma(C^{'}, D, 0.30, 0.60) = \gamma(C, D, 0.30, 0.60)$, so that $C^{'} = \{c2\}$ is a reduct of C on D.

The idea of reduct is most useful in those applications where it is necessary to find the most important collection of condition attributes responsible for a cause-effect relationship and also useful for eliminating irrelevant attributes from the information system. Given an information system, there may exist more than one reduct. Each reduct in the set of $RED(C, D, P_\beta, N_\beta)$ can be used as an alternative group of attributes which could represent the original information system with the classification factor P_β, N_β. An important problem is how to select an optimal reduct from the set of $RED(C, D, P_\beta, N_\beta)$. The selection can depend on the optimality criterion associated with attributes.

Chapter 5

Rough Set Based Data Reduction

In many practical applications, such as diagnosing unknown disease, identifying unknown objects, during the data collection phase, it is often difficult to know exactly which features are relevant and/or important for the learning task, and how they should be represented. So all features believed to be useful are collected into the database. Hence databases usually contain some attributes that are undesirable, irrelevant, or unimportant to a given discovery task, focussing on a subset of attribute is now common practice. Identifying relevant fields is the most common focussing technique. In Chapter 3, we discussed attribute-oriented induction of the DBLEARN system and its extensions. The general idea of the system is to extract the relevant data from the database, and then generalize the relevant data to the desirable level and transform the tuples in the generalized relation to logical rules. During the rule-generalization procedure, all the attributes in the generalized relation are treated in the same way, i.e., equally important. But this is not true in many real applications. In the generalized relation there are still some irrelevant, or unimportant attributes to a given discovery task. For example, to determine the (gas) mileage of a car, the weight and power of the car are important while the number of doors of the car is not needed for consideration. So one of the important issues need to be considered is to find out the most relevant attributes and eliminate the irrelevant or non-essential attributes according to the decision task without losing information about the data in the generalized relation. The goal is to find a minimal subset of interesting attributes that have the same power to distinguish different classes in the decision attributes as all the attributes in the generalized relation and thus simplify the generalized relation by removing those irrelevant or non-essential attributes and produce a set of much concise and meaningful decision rules for each class in the decision attribute. Rough set theory [Paw82] introduced in Chapter 4 provides one of the most powerful tools to analyze a set of attributes globally. Based on this consideration, we propose a new framework for knowledge discovery in databases, which combines database operations, machine learning techniques and rough set theory. In our system, the learning procedure consists of two phases: data generalization and data reduction. In data generalization, our method generalizes the data by performing attribute removal and attribute-oriented concept tree ascension, thus some undesirable attributes to the learning task are removed. Subsequently the primitive data in the databases are generalized to the high level concepts in the concept hierarchies and a set of tuples may be generalized to the same generalized tuple. The goal of data reduction is to find a subset of interesting attributes that have all the essential information of the generalized relation, so that the subset of the attributes can be used instead of the entire attributes set of the generalized relation. Finally the tuples in the reduced relation are transformed into different knowledge rules based on different knowledge discovery algorithms. Our method analyzes the cause-effect relationship among the condition and decision attributes, meaningful properties of data, such as data dependency among the attributes, are explicitly analyzed by rule-generation algorithms. The method is able to identify the essential subset of non-redundant attributes (factors) that determine the decision task, thus the rules generated in this way are very concise and strong with no redundancy information or unnecessary constraints in them. In this chapter we will discuss two algorithms: DBDeci and DBMaxi. One is to find a set of concise decision rules. The other is to compute all the maximal generalized rules from the generalized relation by using a decision matrix.

5.1 Reduction of the Generalized Relation

In the reduction of the generalized relation the basic role is played by two fundamental concepts – a reduct and a core. Intuitively, a reduct of the generalized relation is its essential part, which suffices to define all basic concepts occurring in the considered data, whereas a core is in a certain sense its most important part. Reducing generalized relations consists of removing superfluous partitions (equivalence relations) or / and superfluous attributes in such a way that the set of elementary categories in the generalized relation is preserved. This procedure enables us to eliminate unnecessary data from the generalized relation, preserving only that part of the data which is really useful.

5.1.1 Significant Value of Attributes

Different attributes may play different roles in determining the dependency relationship between the condition and decision attributes.

The significance of an individual attribute a added to the set R with respect to the dependency between R and D is represented by significance factor SGF, given by

$$SGF(a, R, D) = k(R + \{a\}, D) - k(R, D)$$

$SGF(a, R, D)$ reflects the degree of increase of dependency level between R and D as a result of the addition of the attribute a to R. In practice, the stronger the influence of the attribute a is on the relationship between R and D, the higher the value of the $SGF(a, R, D)$ is. For example, for the car relation in Table 4.1, if $R = \{Make\_model, trans\}$, $D = \{mileage\}$, then $SGF(cyl, R, D) = 0.07$, $SGF(displace, R, D)$ $= 0.07$, $SGF(compress, R, D) = 0.36$, $SGF(power, R, D) = 0.00$, $SGF(weight, R, D) =$ 0.07.

5.1.2 Criteria for the Best Reduct

It is quite often that an information system has more than one reduct. Each reduct can be used instead of the whole group of attributes in the original system in the decision making procedure without changing the dependency relation in the original system. So a natural question is which reduct is the best. The selection depends on the optimality criterion associated with attributes. If it is possible to assign a cost function to attributes, then the selection can be naturally based on the combined minimum cost criteria. For example, in the medical domain, some diagnostic procedures are much more expensive than others. By selecting the least expensive series of the tests represented by the minimum cost reduct, considerable saving can be accomplished without decreasing the quality of the diagnosis. In the absence of an attribute cost function, the only source of information to select the reduct is the contents of the table [Zia91]. Two approaches are possible in this case. In the first one, the reduct with the minimum number of attributes is selected. In the second approach, the reduct which has the least number of combinations of values of its attributes is selected. In this thesis we adopt the criteria that the best reduct is the one which has the minimum number of attributes and if there are two or more reducts with same minimal number of attributes, then the reduct with the least number of combinations of values of its attributes is selected.

Discernibility Matrix

In this subsection, we give a modified definition of a discernibility matrix based on [SkR91]. Using a *discernibility matrix*, we can compute the core of the information system easily.

Definition 5.1 *A* **discernibility matrix** *of C in S, $M(C) = \{m_{i,j}\}_{n \times n}$ is defined as*

$$(m_{ij}) = \begin{cases} \emptyset & x_i, x_j \in \textit{ the same equivalence class of } \tilde{D} \\ \{c \in C : f(c, x_i) \neq f(c, x_j)\} & x_i, x_j \in \textit{ different equivalence classes of } \tilde{D} \end{cases}$$

The entry m_{ij} contains the attributes whose values are not identical on both x_i and x_j (x_i, x_j belong to different classes of $\tilde{D}$, that is, x_i, x_j represent different concepts). In other words, m_{ij} represents the complete information to distinguish x_i, x_j. $M(S) = (m_{ij})$ is symmetric, we only need to compute the entries m_{ij} for $1 \leq j < i \leq n$.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1														
2														
3														
4														
5														
6														
7														
8	bfg	bd1d2eg	cefg	deg	dg	bd1d2fg	dfg							
9	abcefg	abcd1d2fg	acfg	acdeg	acdeg	abcd1d2efg	acdefg							
10	abdef	abd1	acdf	a	ae	abd1ef	aef							
11	abcf	abcd1d2e	aef	acde	acd	acd1d2f	acdf							
12	abcdef	abcd1e	adef	ace	ace	abcd1ef	acef							
13	bcef	abcd1d2	af	acdf	acdf	abcd1d2ef	acdef							
14	abcef	bcd1d2	f	cd	bcde	bcd1d2ef	cdef							

Abbreviations: a:Make_model b:cyl c:displace d1:door d2:compress e:power f:trans g:weight

Table 5.1: Discernibility matrix for the generalized car relation.

Example 5.1 For the generalized car relation in Table 4.1, the discernibility matrix is computed in Table 5.1. (Suppose the attribute "mileage" is the decision attribute, the other attributes are condition attributes)

5.1.3 Core and Discernibility Matrix

The Core is one of the most important concept of a rough set. A core has the common attributes of all the reducts. So a core can be used as a basis to compute a reduct. A core has a very close connection with the discernibility matrix. From the discernibility matrix, we can easily compute the core of the information system based on the following observation. (Note: a core of an information system may be empty)

For $S = \{U, A, V, f\}$, $A = C \cup D$, $M(S) = \{m_{ij}\}$, for any $c \in C$, $c \in CORE(C, D)$ iff there exists i, j, $1 \leq j < i \leq n$ such that $m_{ij} = \{c\}$.

For example, examine the discernibility matrix Table 5.1 for the generalized car relation in Table 4.1, $m_{10,4} = \{a\}$ and $m_{14,3} = \{f\}$, so the core of the attributes is $\{Make\_model, trans\}$.

Compute the best reduct or user minimal attribute subset

The general problem of finding all reducts is unsolvable [Zia91], but in most cases, it is usually not necessary to find all the reducts. The user is often more interested in finding the best reduct with respect to his problem, moreover some user usually knows better about the decision task and may prefer to emphasize some attributes in the decision making process and want to include these attribute values in the final decision rules. Based on the dependency relation and the significant values of attributes, it is very easy and efficient to find a "best" reduct or a "minimal" attribute subset (called *user minimal attribute subset*) which include the attributes the user emphasized and has the same discernibility as all the attributes in the original relation. In the latter case, the result may or may not be a reduct. If the attributes the user is emphasizing are superfluous with respect to D, then the result is not a reduct but still has the same discernibility to discern the objects as the original information system.

Here we present our algorithm to construct the "best" reduct or the user "minimal" attribute subset by using core as the starting point. The algorithm is very simple and straightforward. If the user does not have preference for any attribute, then the algorithm just finds the best reduct which consists of those attributes with the largest significant values in each step. If the user prefers some particular attributes, then our algorithm finds the user minimal attribute subset which includes the attributes the user emphasizes without losing any essential information from the original information system.

Algorithm 5.1 (Reduct Algorithm:) *Compute the best reduct or user minimal attribute subset.*

Input: (i) The task-relevant generalized relation R' (ii) a set of attributes AR for relation R', which is classified into condition attributes C, and decision attributes D (iii) the core CO of AR computed from the discernibility matrix of R' (CO may be empty) (iv) the attribute set UA user prefer to emphasize (UA may be empty, if UA is empty, that means the user does not have preference for any attribute)
Output. A set of attributes $REDU$
Method
Step 1: $REDU = CO \cup UA$;
Step 2: $AR' = AR - REDU$

Step 3: Compute the significant value for each attribute $a \in AR'$, sort the set of attributes AR' based on significant values

```
Step 4: While $K(REDU, D) \neq K(AR, D)$ Do /* Create a subset REDU of
            attributes AR by adding attributes */
             Select an attribute $a$ in $AR'$ with the highest significant value;
            (If there are several attributes $a_i$ (i=1,...,m) with the same maximal
            value $SGF(a, REDU, D)$, choose the attribute $a_j$ which has the least
            number of combination values with those attributes in $REDU$)
            $REDU = REDU \cup \{a_j\}$, $AR' = AR - \{a_i\}$ (i=1,...,m);
            compute the degree of dependency $K(REDU, D)$;
        Endwhile
Step 5: $|REDU| \rightarrow N$
Step 6: For i=0 to N-1 Do /* create a best reduct or user minimal attributes set
            by dropping redundant attributes */
             If $a_i$ is not in $CO \cup UA$ Then
                 remove it from $REDU$
            Endif;
             compute the degree of dependency $K(REDU, D)$;
             If $K(REDU, D) \neq K(AR, D)$ Then
                 $REDU \cup a_i \rightarrow REDU$
             Endif
        Endfor
```

The algorithm assigns a significant value on each attribute and sorts the attributes based on their significant values. A forward selection method is then employed to create a smaller subset of attributes with the same discriminating power as the original attributes. At the end of this phase, the attribute set $REDU$ contains the "good" performing attribute subset found thus far. Finally, to compute the reduct or user minimal attributes subset, a backward elimination method removes attributes one by one from the set $REDU$. The lower the significance value is, the earlier the attribute is processed. The degree of dependency is calculated at each step based on the remaining attributes in $REDU$; if the degree of dependency is changed, the attribute is restored to the set $REDU$, otherwise it is permanently removed. Attributes remaining in the set $REDU$ for the best reduct or user minimal attribute subset. For example, the best reduct of the generalized car relation in Table 4.1 is {*Make_model, compress, trans*} using this algorithm. On the other hand, if the user wants to find the effect of a car's weight on the mileage and prefer to emphasize the attribute *weight* in the derived rules, then the algorithm can find the user minimal attribute subset {*Make_model, display, trans, weight*} which satisfy the user's special preference. (In this case, the result happens to be a reduct). We can find the best reduct or user minimal attribute subset in $N_A \times O(N' \times N')$ in the worst case, where N_A is the number of attributes in the generalized relation R' and N' is the number of tuples in R'. Usually N' is not big in the generalized relation R'.

5.2 An Attribute-Oriented Rough Set Approach to Discover Decision Rules

In this section an example is used to illustrate the procedure of the attribute-oriented rough set approach to create decision rules from generalized relation. Suppose we have a collection of Japanese and America cars with the attributes plate number (**plate#**), **Make_model**, **colour**, number of cylinders (**cyl**), engine displacement (**displace**), compression ratio (**compress**), **power**, type of transmission (**trans**), **weight** of the car and **mileage** depicted in Table 5.2 and the concept hierarchy table for the car relation, the concept hierarchy tree for the attribute "Make_model" depicted in Figure 5.1:

{Honda_civic, Honda_acura,..., Honda_accord} $\subset$ Honda

{Toyota_tercel,...,Toyota_camry} $\subset$ Toyota

{Mazda_323, Mazda_626,..., Mazda_939} $\subset$ Mazda

{Toyota , Honda , ..., Mazda } $\subset$ Japan(Car)

{Ford_escort, Ford_probe,..., Ford_taurus } $\subset$ Ford

{Chevrolet_corvette, Chevrolet_camaro,...,Chervolet_corsica } $\subset$ Chevrolet

{Dodge_stealth, Dodge_daytona,..., Dodge_dynasty } $\subset$ Dodge

{Ford, Dodge, ..., Chevrolet } $\subset$ USA(Car)

Plate#	Make_Model	colour	cyl	door	displace	compress	power	trans	weight	mileage
BCT89U	Ford escort	silver	6	2	Medium	High	high	auto	1020	medium
UYT342	Chevrolet corvette	green	4	2	Small	High	medium	manu	890	high
LKIPO8	Chevrolet corvette	brown	4	2	Small	High	medium	auto	980	medium
IUTY56	Dodge stealth	green	4	2	Medium	Medium	medium	manu	1230	medium
DSA321	Toyota Paso	black	4	2	Small	Medium	low	manu	850	high
ERTW34	Ford probe	yellow	4	2	Medium	Medium	medium	manu	980	medium
9876T	Chrysler Le B	blue	6	4	Medium	Medium	high	auto	1180	medium
UYTHG7	Dodge sprite	light blue	6	4	Medium	Medium	high	auto	900	medium
RST45W	Dodge Stealth	red	4	2	Medium	Medium	high	auto	1180	medium
RGW45W	Dodge Dayton	light green	4	2	Medium	Medium	high	auto	1170	medium
78YUTE	Ford escort	black	4	2	Medium	High	high	manu	780	high
......	....	.....	..	....	....	....	....	...	....	...
7HGY65	Chevrolet corvette	black	4	2	Medium	High	high	manu	700	high
OPLSAD	Honda civic	pink	4	2	Small	High	low	manu	750	high
Ot76SAD	Mazda 323	red	4	2	Small	High	low	manu	700	high
UI89P0	Dodge shadow	red	6	4	Medium	Medium	medium	manu	890	medium
P0967H	Ford festival	brown	4	2	Small	High	medium	auto	850	medium
WEQ546	Toyota corolla	navy	4	2	Medium	Medium	medium	manu	900	high
PLMNH7	Mazda 323	yellow	4	2	Small	High	low	manu	700	high
QAS453	Dodge Dayton	green	4	2	Medium	Medium	medium	manu	1190	medium
PLMJH9	Honda accord	brown	4	2	Small	High	high	manu	870	high
PLMJH9	Honda prelude	yellow	4	2	Small	High	high	manu	970	high
KNM876	Chevrolet beretta	green	6	4	Medium	High	high	auto	1080	medium
IKLO90	Chevrolet cavalier	black	6	4	Medium	Medium	high	auto	1100	medium
OPL876	Mazda 626	purple	4	2	Small	High	medium	manu	980	high
TYUR45	Ford mustang	black	4	2	Medium	Medium	medium	manu	1090	medium
0987UO	Dodge dayton	orange	4	2	Medium	Medium	medium	manu	1170	medium
UYT789	Chevrolet Corvette	black	4	2	Small	High	Low	manu	1100	medium

Table 5.2: Car relation.

$$\{\text{Japan(Car), ..., USA(Car)}\} \subset \text{Any(Make\_model)}$$
$$\{0..800\} \subset \text{Light}$$
$$\{801..1200\} \subset \text{Medium}$$
$$\{1201..1600\} \subset \text{Heavy}$$
$$\{\text{Low, Medium, High}\} \subset \text{Any(Weight)}$$

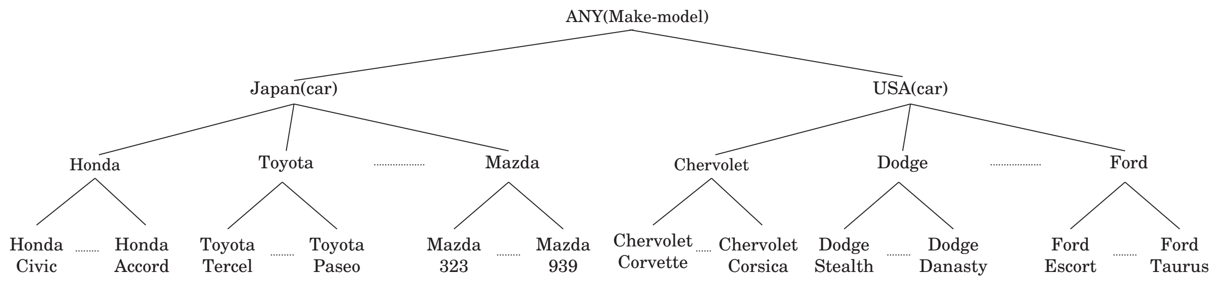


Figure 5.1: Concept hierarchy tree for make_model

Our objective is to learn the decision rule which tell which features of a car really determine the mileage. The request is specified as follows:

learn decision rule

for *Mileage*
from *Car_relation*

Notice in this learning request, the concept hierarchies and threshold are not specified, thus the default ones will be used.

First the user learning request is transferred to SQL, which extracts the data from the relation (**Car_relation**) and the result is obtained as shown in Table 5.2. Then we apply the generalization algorithm, and we get the generalized table as shown in Table 4.1.

After the generalization process, the rough set method is performed on the generalized relation table 4.1. First the core of the attributes is computed through the discernibility matrix, then the best reduct or the user minimal attribute subset of the attributes can be constructed by applying the reduct algorithm. The reduction of the generalized relation is performed further by removing those attributes which are not in the reduct or the user minimal attributes subset and thus simplify the generalized relation. Taking "mileage" as the decision attribute, we examine how to apply the reduct or user minimal attribute subset of the condition attributes with respect to "mileage" to reduce the generalized relation further.

Strategy 1 (find the desired reduct or user minimal attributes and reduce the generalized relation)

Using Algorithm 5.1, we can find the best reduct {*Make_model, compress, trans*} or any user minimal attribute subset based on the user's preference. (In the example above, our algorithm finds the user minimal attribute subset {*Make_model, display, trans, weight*} if the user has particular interest about the attribute *weight*). With the reduct or the user minimal attribute subset, we can remove those attributes which are not in the reduct or the user minimal attribute subset without changing the dependency relationship between the mileage and the condition attributes. The generalized car relation in Table 4.1 is further reduced, resulting in Table 5.3 using the best reduct and Table 5.4 using the user minimal attribute subset respectively. (In our later discussion, we only discuss Table 5.3)

Strategy 2 (combine the similar tuples)

Make_model	compress	trans	mileage
USA	HIGH	AUTO	MEDIUM
USA	MEDIUM	MANUAL	MEDIUM
USA	MEDIUM	AUTO	MEDIUM
USA	HIGH	MANUAL	HIGH
JAPAN	HIGH	MANUAL	HIGH
JAPAN	MEDIUM	MANUAL	HIGH

Table 5.3: Reduced table with best reduct

Make_model	display	trans	wight	mileage
USA	MEDIUM	AUTO	MEDIUM	MEDIUM
USA	MEDIUM	MANUAL	MEDIUM	MEDIUM
USA	SMALL	AUTO	MEDIUM	MEDIUM
USA	MEDIUM	MANUAL	LIGHT	HIGH
JAPAN	SMALL	MANUAL	LIGHT	HIGH
JAPAN	MEDIUM	MANUAL	MEDIUM	HIGH
JAPAN	SMALL	MANUAL	MEDIUM	HIGH
USA	SMALL	MEDIUM	MEDIUM	HIGH

Table 5.4: Reduced table with user minimal attributes subset

In the reduced table, as shown in Table 5.3, in the same class, two tuples can be combined into one if the values of the condition attributes differ in only one attribute; this corresponds to the *closing interval rule* in [Mic83]. If the data values appearing in the combined tuples cover all the possible values of the attribute in the corresponding generalization hierarchy, then this attribute should be dropped from the tuple. For example, in Table 5.3, the class with $mileage = Medium$, the first tuple {USA, HIGH, AUTO, MEDIUM} and third tuple {USA, MEDIUM, AUTO, MEDIUM} only differ in *compress*, then these two tuples can be combined into {USA, (HIGH, MEDIUM), AUTO, MEDIUM}, which can be further simplified to {USA, , AUTO, MEDIUM}. After examining the distribution of the values for each attribute, the reduced table Table 5.3 is further simplified to Table 5.5.

Strategy 3 (Transform the tuples in the reduced relation into decision rules for each class)

Make_model	compress	trans	mileage
USA		AUTO	MEDIUM
USA	MEDIUM		MEDIUM
	HIGH	MANUAL	HIGH
JAPAN		MANUAL	HIGH

Table 5.5: Reduced table after combination

According to Table 5.5, we can derive the following decision rules for the car class with $mileage = Medium$ or $mileage = High$ respectively:

(1) if (Make_model=USA $\wedge$ trans=AUTO) $\vee$ (Make_model=USA $\wedge$ compress=MEDIUM)
$then$ (mileage=MEDIUM)

(2) if (compress=HIGH $\wedge$ trans=MANUAL) $\vee$ (Make_model=JAPAN $\wedge$ trans=MANUAL)
$then$ (mileage=HIGH)

For example, rule (1) can be interpreted as: If a car is made in USA with *automatic transmission* , or made in USA with *medium compression*, then the *mileage* of the car is *medium*.

In summary, we present the algorithm below:

Algorithm 5.2 *DBDeci–An Attribute-Oriented Rough Set Approach for Learning Decision Rules in Databases*

Input: (i) A set of task-relevant data R (assume that they are obtained by a relation query and are stored in a relation table), a relation of arity n with a set of attributes $C = \{c_i\}$ ($1 \leq i \leq n-1$) and decision attribute D (ii) a set of concept hierarchies, H_i, where H_i is a hierarchy on the attribute c_i, if available; (iii) the class threshold value T

Output. A set of decision rules for each class of D.

Method

Step 1. *Attribute-oriented induction.* (Generalization Algorithm)

Step 2. *Find the best reduct or user minimal attribute subset with respect to D* (Reduct Algorithm).

Step 3. *Reduce the generalized relation by removing those attributes which are not in the reduct or user minimal attributes subset.*

Step 4. *Combine similar tuples in the reduced relation.*

Step 5. *Transform the tuples in the reduced relation into decision rules for each class in D.*

5.3 Computing Maximal Generalized Rules

In [ZiS93], Ziarko and Shan proposed a decision matrix to compute the minimal rules from a decision table. Based on their ideas, we propose a method which can find all the maximal generalized rules from databases by integrating attribute-oriented induction with decision matrix. It is shown that finding all the maximal generalized rules is reduced to the problem of simplifying a group of associated Boolean expressions. Below we first give the definitions of maximal generalized rules and decision matrix, and then discuss the algorithm DBMaxi.

5.3.1 Rules in Information System

As discussed in Chapter 4, a relational database may be considered as an information system in which columns are labelled by attributes, rows are labelled by the objects and the entry in column p and row e has the value $p(e)$. The collection of all tuples constitutes a set of training sample. Also, one of the attributes, say $d \in A$, is considered to be the learning target, or decision attributes representing the "concept" or "concepts" to be learned. The concept is simply a particular value V_d of the attribute d. The object of learning is to find a discriminating description of the subset $|V_d|$ of objects with the value of the attribute d equal to V_d that is as simple as possible, i.e., to learn the description of the set

$$|V_d| = \{e \in U : d(e) = V_d\}$$

The set V_d will be referred to as the target class (concept) or the set of possible cases.

For a value V_d of the decision attribute d (which is the "concept" we intend to learn), a rule r for V_d is defined as a set of attribute-value pair

$$r = \{(a_{i1} = V_{i1}), (a_{i2} = V_{i2}), ..., (a_{in} = V_{in})\}$$

such that

$$A_r = (a_{i1}, a_{i2}, ..., a_{in}) \subseteq A \qquad (5.1)$$

and

$$supp(r) = \{e \in U : A_r(e) = V_r\} \subseteq |V_d| \qquad (5.2)$$

where $V_r = (V_{i1}, V_{i2}, ..., V_{in})$.

That is, a rule is a combination of values of some attributes such that the set of all information vectors matching this combination is contained in the set of information vectors with the value of decision attribute equal to V_d. Traditionally, the rule r is denoted as a logical implication

$$r : (a_{i1} = V_{i1}) \wedge (a_{i2} = V_{i2}) \wedge ... \wedge (a_{in} = V_{in}) \rightarrow (d = V_d)$$

The set of attribute-value pairs occurring on the left hand side of the rule r is referred to as rule condition part $cond(r)$, and the right hand side is a decision part $dec(r)$, thus a rule can be simply expressed as $cond(r) \rightarrow dec(r)$. $supp(r)$ is called *rule support*, which contains all the objects in the universe U whose attribute values match the rule conditions r.

5.3.2 Maximal Generalized Rules

We say two rules r_1, r_2 with respect to the same concept V_d are comparable if either $cond(r_1) \subseteq cond(r_2)$ or $cond(r_1) \supseteq cond(r_2)$. In fact, the set of rules is partially ordered with regard to the relation of inclusion.

Definition 5.2 *A maximal generalized rule is a minimal element of the partially ordered rule set.*

The maximal generalized rules minimize the number of rule conditions and are in a sense better because their conditions are non-redundant

We use RUL to denote the collection of all maximal generalized rules for the decision V_d.

5.3.3 An Algorithm to Compute the Maximal Generalized Rules

Our algorithm computes the maximal generalized rules as follow: for large databases, first, the attribute-oriented induction algorithm is applied. After the generalization process, the rough set method is performed on the generalized relation. The decision matrix for the decision values of the decision attribute are constructed and the maximal generalized rules are computed from them.

Decision Matrix

For the selected decision attribute $d \in A$ and its particular value V_d, we will focus on the collection of objects e (the concept), for which $d(e) = V_d$, i.e., the set $|V_d|$. Before attempting to find discriminating rules for $|V_d|$ in terms of other attributes belonging to $A - \{d\}$, we will summarize all the attribute-value pairs distinguishing objects belonging to $|V_d|$ and $U - |V_d|$ in the matrix format defined as follows.

Definition 5.3 *Let e_i denote any object belonging to $|V_d|$, i.e., $i = 1, 2, ..., Card(|V_d|) = \rho$ and let $e_j \in U - |V_d|$, $j = 1, 2, ..., card(U - |V_d|) = \gamma$. The decision matrix $DM = (DM_{ij})_{\rho \times \gamma}$ is defined as*

$$DM_{i,j} = \{(a, a(e_i)) : a(e_i) \neq a(e_j)\}$$

The set $DM_{i,j}$ contains all pairs whose values are not identical on both e_i and e_j. In other words, $DM_{i,j}$ represents the complete information needed to distinguish e_i and e_j. The distinguishing attributes for different combinations of i and j can be represented in the form of a matrix $DM = [DM_{ij}]_{\rho \times \gamma}$.

Example 5.2 Suppose after data generalization, we have a simple car generalized relation in Table 5.6. In order to make our explanation simple, we introduce the numerical representation of the reduced form by replacing the symbolic value with numerical number. For example, for the $Make\_model$, 0 stands for USA, 1 for Japan, similar substitutions apply to other attributes. (Note that the same number in different columns denotes different symbolic value, e.g., 0 in column M denotes USA while

Make_Model	compress	power	trans	mileage
USA	HIGH	HIGH	AUTO	MEDIUM
USA	MEDIUM	MEDIUM	MANUAL	MEDIUM
USA	HIGH	LOW	MANUAL	MEDIUM
USA	HIGH	MEDIUM	AUTO	MEDIUM
USA	MEDIUM	HIGH	MANUAL	MEDIUM
USA	MEDIUM	HIGH	AUTO	MEDIUM
USA	HIGH	HIGH	MANUAL	HIGH
JAPAN	HIGH	LOW	MANUAL	HIGH
JAPAN	MEDIUM	MEDIUM	MANUAL	HIGH
JAPAN	HIGH	HIGH	MANUAL	HIGH
JAPAN	MEDIUM	LOW	MANUAL	HIGH
JAPAN	HIGH	MEDIUM	MANUAL	HIGH
USA	HIGH	MEDIUM	MANUAL	HIGH

Table 5.6: A simple generalized car relation

i	j	Obj	M	C	P	T	Mileage
1		e1	0	0	0	0	0
2		e2	0	1	1	1	0
3		e3	0	0	2	1	0
4		e4	0	0	1	0	0
5		e5	0	1	0	1	0
6		e6	0	1	0	0	0
	1	e7	0	0	0	1	1
	2	e8	1	0	2	1	1
	3	e9	1	1	1	1	1
	4	e10	1	0	0	1	1
	5	e11	1	1	2	1	1
	6	e12	1	0	1	1	1
	7	e13	0	0	1	1	1

Table 5.7: Numerical representation of Table 5.5

	j1	j2	j3	j4	j5	j6	j7
i1	(T,0)	(M,0),(P,0),(T,0)	(M,0),(C,0) (P,0),(T,0)	(M,0),(T,0)	(M,0),(C,0), (P,0),(T,0)	(M,0),(P,0) (T,0)	(P,0),(T,0)
i2	(C,1),(P,1)	(M,0),(C,1), (P,1)	(M,0)	(M,0),(C,1),(P,1)	(M,0), (P,1)	(M,0), (C,1)	(C,1)
i3	(P,2)	(M,0)	(M,0),(C,0) (P,2)	(M,0),(P,2)	(M,0), (C,0)	(M,0), (P,2)	(P,2)
i4	(P,1),(T,0)	(M,0),(P,1),(T,0)	(M,0),(C,0) (T,0)	(M,0), (P,1),(T,0)	(M,0),(C,0) (P,1),(T,0)	(M,0),(T,0)	(T,0)
i5	(C,1)	(M,0),(C,1),(P,0)	(M,0),(P,0)	(M,0),(C,1)	(M,0),(P,0)	(M,0),(C,1), (P,0)	(C,1),(P,0)
i6	(C,1),(T,0)	(M,0),(C,1),(P,0),(T,0)	(M,0),(P,0) (T,0)	(M,0),(C,1),(T,0)	(M,0), (P,0) (T,0)	(M,0),(C,1) (P,0),(T,0)	(C,1),(P,0),(T,0)

Table 5.8: Decision matrix for the class mileage=MEDIUM

0 in column C denotes $HIGH$, it is easy to distinguish from the context). Table 5.7 represents the numerical form of the information about cars given in Table 5.6. In this representation M is an abbreviation of "Make_model", C for "compress", and so on. Two extra index columns i, j are added to number the object belonging to the target class, $mileage = 0$ (i.e., mileage=MEDIUM) and its complement respectively.

Table 5.8 is a decision matrix derived for the decision class $mileage = MEDIUM$. Each cell (i, j) in this matrix is a collection of attribute-value pairs distinguishing row i of the target class from row j of its complements.

Decision Matrix and Maximal Generalized Rules

In this subsection, we will present the basic method to compute the maximal generalized rules from a decision matrix. Before discussing the main result, we will introduce the following notation cited from [ZiS93, HSCZ94a].

Let $e_i \in |V_d|$, we will use the symbol RUL_i to denote the set of all maximal generalized rules whose conditions match the features of object e_i, that is

$$RUL_i = \{r \in RUL : A_r(e_i) = V_r\}$$

Clearly, if the collection of rules RUL_i is known for each $e_i \in |V_d|$ then all the maximal generalized rules for target decision $|V_d|$ can be obtained by taking the union

$$RUL = \bigcup_i RUL_i$$

Consequently, in what follows we focus on the basis of the method to compute all maximal generalized rules matching an arbitrary object $e_i \in |V_d|$.

For the given decision matrix DM and fixed decision value V_d, let us consider the Cartesian product $F_i = DM_{i1} \times DM_{i2} \times ... \times DM_{ir}$ of sets of attribute-value pairs constituting the components of the decision matrix DM contained in the row i.

Since some components of the vectors belonging to F_i may be identical, we will consider the associated set

$$\tilde{F}_i = \{\{t\} : t \in F_i\}$$

where $\{t\}$ is a set of all distinct components contained in the vector t.

The elements of $\tilde{F}_i$ are all rules for $|V_d|$ since they match at least one object from $|V_d|$ (i.e., object e_i) and do not match any of the objects belonging to the complement of $|V_d|$, $(U - |V_d|)$. The rules in $\tilde{F}_i$ are partially ordered by the inclusion relation with the set of minimal elements in this denoted as MIN_i.

Theorem 5.1 (ZiS93) *Each maximal generalized rule in $\tilde{F}_i$ computed from the decision matrix DM is also minimal in the set of all rules for $|V_d|$ and each maximal generalized rule for $|V_d|$ is minimal in a certain set $\tilde{F}_i$.*

The above theorem states that, in essence, $RUL_i = MIN_i$ which in practice means that the decision matrix can be used to find all maximal generalized rules for the target concept $|V_d|$. A simple, systematic procedure described later can be used to produce the maximal generalized rules in the set $\tilde{F}_i$. For the proof, please refer to [ZiS93].

The maximal generalized rules in the set MIN_i can be computed by simplifying an associated Boolean function called the decision function which is inspired by the idea of the discernibility function introduced in [SkR91]. The decision function B_i is constructed out of the row i of the decision matrix, that is, $(DM_{i1}, DM_{i2}, ..., DM_{ir})$ by formally treating each attribute-value pair occurring in component DM_{ij} as a Boolean variable and then forming Boolean conjunction of disjunctions of the components belonging to each set DM_{ij} $(j = 1, 2, ..., \gamma)$. That is,

$$B_i = \bigcap_j \bigcup DM_{ij}$$

where $\bigcap$ and $\bigcup$ are respectively generalized conjunction and disjunction operators.

Example 5.3 Based on the decision matrix given in Table 5.8, we can construct the following decision function for row 1

$B_1 = ((T, 0)) \wedge ((M, 0) \vee (P, 0) \vee (T, 0)) \wedge ((M, 0) \vee (C, 0) \vee (P, 0) \vee (T, 0)) \wedge ((M, 0) \vee (T, 0)) \wedge ((M, 0) \vee (C, 0) \vee (P, 0) \vee (T, 0)) \wedge ((M, 0) \vee (P, 0) \vee (T, 0)) \wedge ((P, 0) \vee (T, 0))$

By applying the distribution and absorption laws of Boolean algebra, each decision function can be expressed in a simplified form of a disjunction of minimal conjunctive expressions.

Example 5.4 The decision function B_1 given in Example 5.3 can be easily simplified to $B_1 = (T, 0)$,

This corresponds to the rule:

$trans = AUTO \rightarrow mileage = MEDIUM$

Directly from the Theorem 5.1, we can derive the general procedure for computing all maximal generalized rules for the given target decision. The procedure requires the construction of the decision matrix for each target decision prior to computation of rules. The key steps to compute the rules are summarized in algorithm (DBMaxi).

Algorithm 5.3 *DBMaxi: Compute the maximal generalized rules*

Input: a relational system R
Output. the maximal generalized rules
Method
Step 1: Extract the generalized relation R' from R (Generalization Algorithm)
Step 2: Compute the decision matrix for the current decision category in R'
Step 3: For each positive case $e_i, (i = 1, 2, ..., \rho)$ compute the set of all maximal generalized rules MIN_i matching this case by evaluating and simplifying (using the absorption law) the associated decision function B_i.

Step 4: Compute the union $\cup MIN_i$ of maximal generalized rule sets to find all maximal generalized rules for the current decision category.

The central component of the above algorithm is the simplification of the decision functions associated with the positive cases of the information table. For example, to compute the maximal generalized rules for the decision class $mileage = MEDIUM$, decision functions have to be created and simplified for row 1-6 in Table 5.8. As can be verified from Table 5.8, the simplified functions yield the following complete set of maximal generalized rules for $mileage = MEDIUM$:

(1) If $trans = AUTO$ then $mileage = MEDIUM$

(1) If $make\_model = USA(car) \wedge compress = MEDIUM$ then $mileage = MEDIUM$

(3) If $make\_model = USA(car) \wedge power = LOW$ then $mileage = MEDIUM$

(4) If $compress = MEDIUM \wedge power = HIGH$ then $mileage = MEDIUM$.

Similarly, we can find the maximal generalized rules for $mileage = HIGH$:

(5) If $compress = HIGH \wedge power = HIGH \wedge trans = MANUAL$ then $mileage = HIGH$

(6) If $make\_model = JAPAN(car)$ then $mileage = HIGH$

(7) If $compress = MEDIUM \wedge power = LOW$ then $mileage = HIGH$

5.3.4 Complexity of Maximal Generalized Rules

In this subsection we give a quantitative analysis of the possible number of maximal generalized rules. Suppose after data generalization, there are N' tuples with K attributes left. For a particular learning task, the number of positive tuples is n and so the number of negative tuples is $N' - n$. Then we can construct a $n \times (N' - n)$ decision matrix, for each entry of the decision matrix, there are maximal K terms because that is the maximal number of different attributes number between the positive and negative tuples. Each row of the decision matrix corresponds a set of maximal generalized rules, so the maximal number of maximal generalized rules from each row is $K^{N'-n}$, there are total n row in the decision matrix, so the total number of possible maximal generalized rules are $n \times K^{N'-n}$. As a example, if we have 30 tuples with

5 attributes and 10 positive tuples, then the possible maximal generalized rules are 9.536E+14. From a practical point of view, we are not able to compute all these possible maximal generalized rules even using the fastest computer. Hence in order to define a tractable algorithm, we will need to "prune" the set of possible maximal generalized candidate rules considerably. We believe that using a good rule measure can help considerably when we are trying to learn rules from data. A feasible algorithm should learn the best set of rules rather than exhaustive learning all the possible rules. It is one of the topics for our future research.

Chapter 6

Multiple Sets of Knowledge Rules and Rough Sets

The importance of redundancy for coping with noise in communications is well known [ShW64]. A single knowledge base system which utilizes a single minimal set of decision rules to classify future examples may lead to mistakes, because each minimal set of decision rules represent different domain of the knowledge representation system and has different criteria. Recently, in order to enhance the accuracy of expert system, the subject of Multiple Sets of Knowledge Rules **(MSKR)** (also called multiple knowledge bases) and multiple experts have received considerable attention [KoK93, NgB92]. The idea is to generate several knowledge bases instead of one knowledge base for the classification of new object, hoping that the combination of answers of multiple knowledge bases results in better performance. Typically one object is classified with several rules in the multiple knowledge bases system, and the decisions are then combined to obtain the final conclusion. Many research results illustrated that such multiple rules, if appropriately combined during classification, can improve the classification accuracy [Kon91, KoK93, Gam89, CB88].

Some of the arguments raised in support this approach include: (1) in cases where expertise is diffused and a true expert in the domain of interest can not be identified, combining the insights of "competent people" could improve the application; (2) large complex domains which are generally not mastered by a single individual, requiring the use of multiple experts to ensure comprehensive coverage; (3) the acceptance of expert systems in the business world requires the consensus of organizational "experts", therefore, it is necessary to incorporate into Expert Systems (ES) the contributions of several experts; (4) large classes of problems could be more easily solved if we move away from the notion of a single expert as the basis of ES to the broader based on "community of experts" premise for ES applications [NgB92]; (5) to improve the classification accuracy in the presence of noise data in the database.

The informativity of the knowledge bases with redundant rules seems to be much better than without them. Redundant rules can be trimmed off and an "usual" knowledge base is obtained as a downgraded version. Since the user can define the number of redundant rules, the preference function and other parameters, this enables a thorough extraction of most valuable rules. The efficiency of the learning algorithms remains practically the same when using redundant knowledge [Gam89].

At this point, it seems essential to understand how and why redundant knowledge or multiple knowledge rules help. First, empirical tests [Kon91, KoK93] indicate that redundant knowledge is more helpful if it is as accurate and reliable as possible and at the same time as different from the other knowledge as possible. This also seems plausible in real life. Adding a novice is probably counterproductive and adding an expert whose knowledge is too similar to some other members will only give more importance to the previous expert. Another problem is the cooperation between redundant knowledge. Indeed this might be a more difficult problem than to determine whether to add another redundant method or not. Similarly, it is very difficult to analyze the cooperation between experts.

The phenomenon of importance of redundant knowledge in real life is empirically shown in [Gam89], several strategies for generating multiple knowledge bases or redundant knowledges from a data set and using multiple experts in expert system development have been proposed. Gams [Gam89] developed the inductive learning system GINESYS that generate multiple sets of decision rules. One set of rules consists of "main" rule and of several "confirmation" rule. Each instance is classified with one set of rules by combining the probability distribution returned by different rules. Although the combination rule used by Gams is rather ad-hoc, the reported results are encouraging. In the learning system YAILS [Tor93a], redundancy is used to deal with several types of uncertainty existing in real domains to achieve higher accuracy. YAILS uses a simple mechanism to control redundancy. This mechanism consists on splitting the learned rules into two sets by a user-definable parameter (minimal utility, which acts as a way of controlling redundancy) : *foreground* rules and *background* rules. YAILS uses only the foreground set of rules during classification. Only when it is not able to classify one example, it tries to find one rule in the background rule set. The experimental results showed that its classification accuracy is higher than CN2 [ClB91] and AQ-family algorithms [MMHL86] in most situations. Boose [Boo86] has proposed an approach for combining the expertise of several individuals by utilizing a common grid via the Expertise Transfer System (ETS). All these methods lack a theoretical formalism about the mechanism of redundant knowledge. The focus of this chapter is to make a theoretical model to explain the mechanism of multiple knowledge bases (or redundant knowledge) in the context of rough sets theory . In this chapter, we propose a rough set approach to construct multiple knowledge bases. A decision matrix is used to construct a multiple knowledge bases system in a dynamic environment. This approach combines the results of our previous works [Hu94,Zia93,Sh94]. The maintenance of knowledge base in a dynamic environment is an important problem in many applications. The current knowledge base would have to be changed when a new piece of information is delivered with a new object. Incremental learning system has the significant capability to change the knowledge base in a dynamic environment. The decision matrix method has the multiple learning and incremental learning capability. The method we propose here is more general and flexible: (1) it advocates the use of inductive-learning techniques to discover knowledge rules from the collected data in databases; (2) it can deal with development situations where more than one domain expert is used; (3) it can be used to merge two or more rules based KB into one comprehensive KB.

6.1 Multiple Sets of Knowledge Rules

In the decision making process, the Knowledge Representation System (**KRS**) must represent and generate a way of making decisions concerning the object class. The process of rule generation is an important part of data analysis in a knowledge base system. Different algorithms and approaches will generate different minimal decision trees or sets of decision rules (the different knowledge bases) which may or may not use the same condition attributes from the **KRS**. The word "minimal" means that each expert employs only the information necessary to represent the example data (or training data) without any loss of essential information. Depending on the criteria, one knowledge base can be more useful than another which employs different information.

By considering all the reduct tables of the experts in a **KRS**, the **KRS** can generate multiple sets of knowledge rules because it usually has more than one expert and there are many knowledge bases associated with each expert. The **KRS** could be partitioned into sub-systems based on the decision attributes. Each expert uses only the necessary condition attributes without changing the dependency relationship of the original **KRS**. A structure of the MSKR system is shown in Figure 6.1.

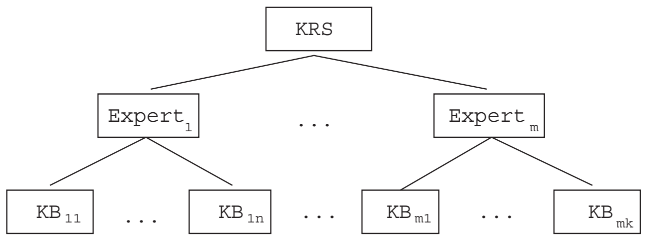


Figure 6.1: Structure of multiple sets of knowledge rules

In a **KRS**, it is possible that some condition attributes are superfluous, so it is very important to identify the essential subset of nonredundant attributes (factor) that determine the decision task.

6.2 A Decision Matrix Approach for Constructing Multiple Sets of Knowledge Rules

One can use different algorithms and systems to generate several different knowledge bases from a given knowledge representation system, and embed these knowledge bases into a expert system to form a multiple set of knowledge rules [ShH94,HuS94]. Different knowledge bases are taken into account in the problem solving phase. This method does not have an incremental learning capability. When new information is expected to become available on a knowledge representation system, it has to regenerate the knowledge bases from the newly organized knowledge representation system. The process of regeneration can be costly when the knowledge representation system is large. For knowledge discovery in a dynamic environment, it would be preferable to accept new information incrementally, without needing to regenerate from scratch.

In Chapter 5, we presented a decision matrix approach to compute all maximal generalized rules from a database. In this section the method is expanded further. Our extended method has an incremental learning capability and can be used to compute all maximal generalized decision rules and the reduct sets of a knowledge representation system S. It provides a way to generate the simplest set of decision rules, while preserving all essential information. The approach presented here is based upon the construction of a number of Boolean functions from decision matrices.

To make our explanation straightforward, we assume some notational conventions as used before. That is, we will assume that all positive and negative objects are separately numbered with subscript i (*i.e.*, $i = 1, 2, ...\gamma$) and j (*i.e.*, $j = 1, 2, ...\rho$) respectively. To distinguish positive from negative objects we will use superscripts V and $\sim V$, for instance, obj_i^V versus $obj_j^{\sim V}$ for the class "V" and class "$\sim V$".

Recall the definition of the decision matrix $M(S) = (M_{i,j})$ in Chapter 5. The set M_{ij} contains all attribute-value pairs $(attribute, value)$ which are not identical between obj_i^V and $obj_j^{\sim V}$. In other words, M_{ij} represents the complete information distinguishing obj_i^V from $obj_j^{\sim V}$.

The set of maximal generalized decision rules $|B_i|$ for a given object obj_i^V ($i = 1, 2, ...\gamma$) is obtained by forming the Boolean expression

$$B_i^V = \bigwedge_j \bigvee M_{ij}$$

where $\bigwedge$ and $\bigvee$ are respectively generalized conjunction and disjunction operators.

The Boolean expression called a decision function B_i^V is constructed from row i of the decision matrix, that is $(M_{i1}, M_{i2}, \ldots M_{i\rho})$, by formally treating each attribute-value pair occurring in the component M_{ij} as a Boolean variable and then forming a Boolean conjunction of disjunctions of components belonging to each set M_{ij} $(j = 1, 2, \ldots, \rho)$.

The decision rules $|B_i^V|$ are obtained by turning such an expression into disjunctive normal form and using the absorption law of Boolean algebra to simplify it. The conjuncts, or prime implicants of the simplified decision function correspond to the maximal generalized decision rules. By treating each of the classes as a target concept, a set of maximal generalized decision rules can be computed for each of the classes. Similarly, by treating the complement of the class "V" as a target concept, a set of decision rules can be computed for each object of the class "$\sim V$" using the same approach.

Once all the decision rule sets $|B_i^V|$ have been computed, a set of all maximal generalized decision rules $RUL(|V_d|)$ for the concept $|V_d|$ corresponding to the decision value V_d $(|V_d| = \{obj \in OBJ : d(obj) = V_d, d \in D, V_d \in VAL_d\})$ is given by

$$RUL(|V_d|) = \bigcup |B_i^V| \qquad (i = 1, 2, \ldots \gamma)$$

For computing the set of reducts of a knowledge representation system, we will introduce the concepts of the *phantom* decision function $\tilde{B}_i^V$ and the *reduct* function $F_{RED(V)}$. A phantom decision function $\tilde{B}_i^V$ is a Boolean expression defined by the conjunction of all Boolean expression $\vee\tilde{M}_{ij}$ of row i in the given decision matrix, where $\vee\tilde{M}_{ij}$ represents the disjunction of the only attribute names (does not contain the value of attributes) of the component M_{ij}. So that we have the following formula:

$$\tilde{B}_i^V = \bigwedge_j \bigvee \tilde{M}_{ij} \qquad (j = 1, 2, \ldots, \rho)$$

Informally speaking, a phantom decision function $\tilde{B}_i^V$ is a similarity of a decision function except for the elements of Boolean expression without the value of attributes.

One can directly derive the result of a phantom decision function $\tilde{B}_i^V$ from the result of a decision function B_i^V, it just eliminates the values of attributes in the prime implicants of the result.

The reduct function $F_{RED(V)}$ is a Boolean function constructed by the conjunction of all phantom decision function $\tilde{B}_i^V$ in the decision matrix. So that we have the following equivalence,

$$F_{RED(V)} = \bigwedge_i \tilde{B}_i^V \qquad (i = 1, 2, ..., \gamma)$$

or

$$F_{RED(V)} = \bigwedge_i (\bigwedge_j \bigvee \tilde{M}_{ij}) \quad (i = 1, 2, ..., \gamma; \quad j = 1, 2, ..., \rho)$$

The set of reducts, denoted as $RED(|V_d|)$, is obtained by performing the multiplications and applying the absorption law of Boolean algebra over the Boolean expression $F_{RED(|V_d|)}$. The conjuncts, or prime implicants of the result of the reduct function, are the whole set of reducts for the target concept V_d in a given knowledge representation system.

A minimized knowledge rule sets corresponding to a reduct is a set of decision rules which is *fully covered* by the attributes of a reduct. The fully cover means that all the condition attributes used by the decision rules is also the attributes of the reduct table.

Let $RUL_{max} = \{r_1, r_2, ..., r_k\}$ be the set of all maximal generalized decision rules generated by the decision matrix method and let $RED = \{RED_1, RED_2, ..., RED_i\}$ be the set of attribute reducts. A minimal knowledge base referred to RED_i ($RED_i \subseteq RED$) is denoted by $RUL_{max}(RED_i)$ and defined as

$$RUL_{max}(RED_i) = \bigcup \{Cond(r_k) \quad Cond(RED_i) : r_k \subseteq RUL_{max}\},$$

where $Cond()$ is the set of attribute names.

Example 6.1 Figure 6.2 depicts two decision matrices obtained from the knowledge representation system given in Table 6.1. Each cell (i, j) in a decision matrix is a collection of attribute-value pairs distinguishing row i of the target class from column j of its complement.

OBJ	S	H	E	C	CLASS
obj_1	0	0	1	0	0
obj_2	1	0	2	1	1
obj_3	1	1	1	0	0
obj_4	0	2	1	1	1
obj_5	1	2	1	0	1
obj_6	1	0	1	0	0
obj_7	1	2	2	1	1
obj_8	0	0	2	1	1

Table 6.1: A knowledge representation system.

	j	1	2	3	4	5
i	OBJ	obj_2	obj_4	obj_5	obj_7	obj_8
1	obj_1	(S,0)(E,1)(C,0)	(H,0)(C,0)	(S,0)(H,0)	(S,0)(H,0)(E,1)(C,0)	(E,1)(C,0)
2	obj_3	(H,1)(E,1)(C,0)	(S,1)(H,1)(C,0)	(H,1)	(H,1)(E,1)(C,0)	(S,1)(H,1)(E,1)(C,0)
3	obj_6	(E,1)(C,0)	(S,1)(H,0)(C,0)	(H,0)	(H,0)(E,1)(C,0)	(S,1)(E,1)(C,0)

(a) A decision matrix for class '0'

	j	1	2	3
i	OBJ	obj_1	obj_3	obj_6
1	obj_2	(S,1)(E,2)(C,1)	(H,0)(E,2)(C,1)	(E,2)(C,1)
2	obj_4	(H,2)(C,1)	(S,0)(H,2)(C,1)	(S,0)(H,2)(C,1)
3	obj_5	(S,1)(H,2)	(H,2)	(H,2)
4	obj_7	(S,1)(H,2)(E,2)(C,1)	(H,2)(E,2)(C,1)	(H,2)(E,2)(C,1)
5	obj_8	(E,2)(C,1)	(S,0)(H,0)(E,2)(C,1)	(S,0)(E,2)(C,1)

(b) A decision matrix for class '1'

Figure 6.2: Decision matrices for Table 6.1

Based on these decision matrices we can obtain the following decision functions B_i^0 (i = 1, 2, 3) from the class "0" decision matrix (and similarly, we can obtain B_i^1 (i = 1, 2, ...5) from the class "1" decision matrix).

Class "0" decision functions:

$$\begin{aligned}
B_1^0 = \quad & ((S,0) \vee (E,1) \vee (C,0)) \wedge ((H,0) \vee (C,0)) \wedge ((S,0) \vee (H,0)) \wedge ((S,0) \vee (H,0) \vee (E,1) \vee (C,0)) \\
& \wedge((E,1) \vee (C,0)) = ((S,0) \wedge (C,0)) \vee ((H,0) \wedge (E,1)) \vee ((H,0) \wedge (C,0)) \\
B_2^0 = \quad & ((H,1) \vee (E,1) \vee (C,0)) \wedge ((S,1) \vee (H,1) \vee (C,0)) \wedge ((H,1)) \wedge ((H,1) \vee (E,1) \vee (C,0)) \\
& \wedge((S,1) \vee (H,1) \vee (E,1) \vee (C,0)) = (H,1) \\
B_3^0 = \quad & ((E,1) \vee (C,0)) \wedge ((S,1) \vee (H,0) \vee (C,0) \wedge ((H,0)) \wedge ((H,0) \vee (E,1) \vee (C,0)) \\
& \wedge((S,1) \vee (E,1) \vee (C,0)) = ((H,0) \wedge (E,1)) \vee ((H,0) \wedge (C,0))
\end{aligned}$$

The $\bigcup |B_i^0|$ corresponds to all the maximal generalized decision rules RUL for the class "0" of the knowledge representation system shown in Table 6.1:

$$(S = 0) \wedge (C = 0) \rightarrow (CLASS =' 0')$$

$$(H = 0) \wedge (E = 1) \rightarrow (CLASS =' 0')$$

$$(H = 0) \wedge (C = 0) \rightarrow (CLASS =' 0')$$

$$(H = 1) \rightarrow (CLASS =' 0')$$

Similarly, we can obtain the set of all maximal generalized decision rules for the class "1":

$$(E = 2) \rightarrow (CLASS =' 1')$$

$$(C = 1) \rightarrow (CLASS =' 1')$$

$$(H = 2) \rightarrow (CLASS =' 1')$$

Now, let us compute the reduct function for the class "1" and class "0", such that

$$\begin{aligned} F_{RED(0)} &= \bigwedge \tilde{B}_i^0 \qquad\qquad (i = 1,2,3) \\ &= ((S \wedge C) \vee (H \wedge E) \vee (H \wedge C,)) \wedge (H) \wedge ((H \wedge E) \vee (H \wedge C)) = (H \wedge E) \vee (H \wedge C) \\ F_{RED(1)} &= \bigwedge \tilde{B}_j^1 \qquad\qquad (j = 1,2,3,4,5) \\ &= ((E) \vee (C)) \wedge ((H) \vee (C)) \wedge (H) \wedge ((H) \vee (E) \vee (C)) \wedge ((E) \vee (C)) = (H \wedge E) \vee (H \wedge C) \end{aligned}$$

So that we can obtain the sets of reducts for the class "0" and the class "1",

$$RED(0) = \{HE, HC\}; \qquad RED(1) = \{HE, HC\}$$

We have the set of reducts $RED = \{HE, HC\}$ with respect to the decision attribute. According to the above definition, the minimized knowledge bases corresponding to reducts "H, E" and "H, C" on the class "0", and to reducts "H, E" and "H, C" on the class "1" are the following sets of decision rules extracted from all maximal generalized decision rules:

The maximal generalized decision rules for reduct "H, E" on the class "0" is

$$(H = 0) \wedge (E = 1) \rightarrow (CLASS =' 0')$$

$$(H = 1) \rightarrow (CLASS =' 0')$$

The maximal generalized decision rules for reduct "H, C" on the class "0" is

$$(H = 0) \wedge (C = 0) \rightarrow (CLASS =' 0')$$

$$(H = 1) \rightarrow (CLASS =' 0')$$

The maximal generalized decision rules for reduct "H, E" on the class "1" is

$$(E = 2) \rightarrow (CLASS =' 1')$$

$$(H = 2) \rightarrow (CLASS =' 1')$$

The maximal generalized decision rules for reduct "H, C" on the class "1" is

$$(C = 1) \rightarrow (CLASS =' 1')$$

$$(H = 2) \rightarrow (CLASS =' 1')$$

6.3 Combination of Multiple Sets of Knowledge Rules

In last section, we presented a method to construct multiple sets of knowledge rules. The idea is to generate multiple set of knowledge rules instead of one set of knowledge rules for the classification of new objects, hoping that combining the answers of multiple knowledge rules will result in better performance. Typically one object is classified with several rules and the decisions are then combined to obtain the final decision. This strategy proved to be very efficient [CeB88, Gan89, ClB91]. Many studies showed that such multiple sets of knowledge rules if appropriately combined during classification can improve the classification accuracy [KoK93]. However, the problem of how to combine decisions of multiple knowledge bases remains.

Currently, there are four strategies for combining multiple sets of knowledge rule:

(1) Sum of distribution: Frequencies of covered training instances for all rules that cover a given testing instances are summed up and the instance is classified in the majority class of the resulting distribution [ClB91, Bun90].

(2) Voting: Each rule votes for one class. A training instance is classified into a class with maximal number of votes [Kon91].

(3) Naive Bayesian combination: For each class the probability is calculated with naive Bayesian formula [Kon89] where instead of simple conditions (attribute-value pairs) the conditions A_i of k rules, that covers a given testing example are used [SmG92]:

$$P(C|A_1, ..., A_k) = P(C) \prod_{i=1}^{k} \frac{P(C|A_i)}{P(C)}$$

Smyth and Goodman [SmG92] slightly modified the above formula as their ITRULE learning algorithm generates rules which are generated for each class separately.

(4) Decision Table method [NgB92]: This method is based on decision table approach to describe mathematically, analyze and merge knowledge rules (production rules) via matrix method. It focuses on rule inconsistency, logical incompleteness of rules and merging the rules of multiple knowledge bases. Three types of inconsistencies can be identified: (a) condition inconsistency—where two or more rules have equivalent action parts but different condition parts; (b) action inconsistency– two

or more rules have logically equivalent condition parts but different action parts; and (c) dynamic—–during processing of the rule-base, rules may develop any of the above types of inconsistencies. It consists of two phases: in Phase I, a 0–1 decision matrix is prepared and analyzed separately for each expert. The inconsistencies discovered are resolved by the knowledge engineer before the rule-sets are merged in Phase II. In Phase II, the rule-sets are merged and analyzed. Problems identified at this level are discussed and resolved in a group setting.

The above four strategies are complementary to each other, each has its strong and weak point depending on the domain. A deep analysis and comparison of these strategies and developing new methods for combining multiple sets of knowledge rules are one of our current research topics.

Chapter 7

Implementation and Experiments

To test and experiment on the database learning algorithms developed in the previous chapters, an experimental database learning system, **DBROUGH** [HuC94a, HCH93b, HSCZ94], has been constructed and some interesting experiments have been conducted in the learning system.

7.1 Architecture

DBROUGH is a descendant of DBLEARN [CCH91, HCC92a]. The architecture of the system is shown in Figure 7.1. The system can discover different kinds of knowledge rules from relational databases, including characteristic rules, discrimination rules, decision rules, maximal generalized rules, data trend regularities and multiple sets of knowledge rules for the discovery task. The system takes SQL-like database learning requests and performs different algorithms to find different rules. The background knowledge is stored in a concept hierarchy table. The provided concept hierarchies can be adjusted dynamically according to database statistics and specific learning requests.

DBChar: Find the characteristic rules for the target class

DBClass: Find the classification rules of the target class with other classes

DBDeci: Find the decision rules for the decision attributes

DBMaxi: Find all the maximal generalized rules

DBTrend: Find the data trend regularities for the target class

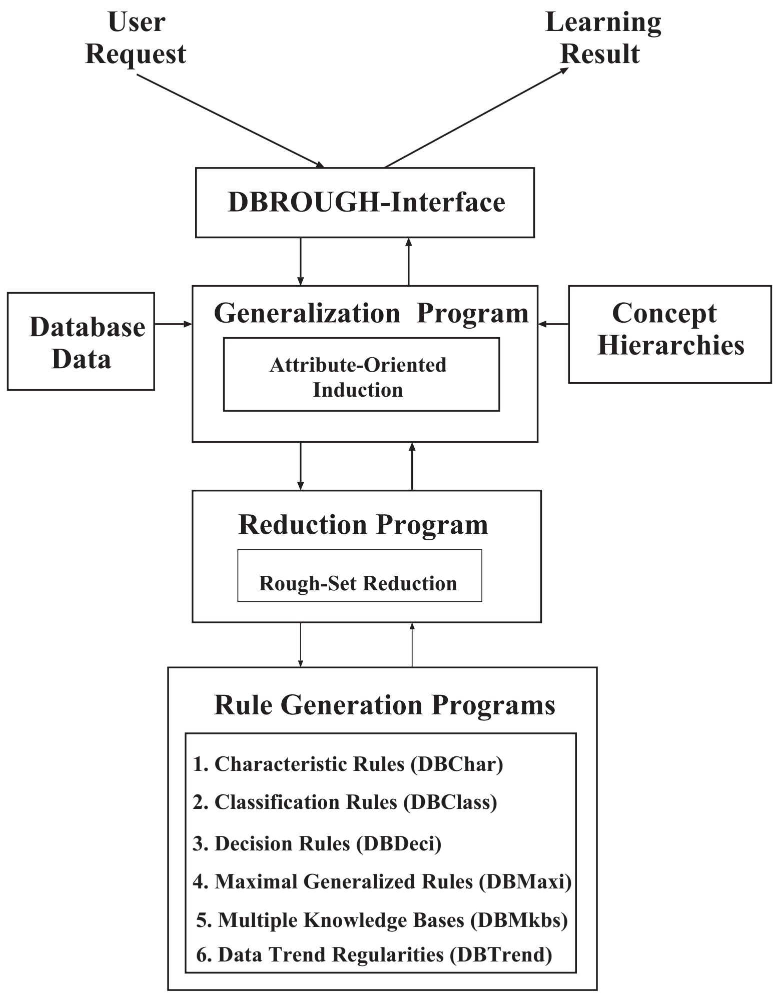

User Request
Learning Result
DBROUGH-Interface
Database Data
Generalization Program
Attribute-Oriented Induction
Concept Hierarchies
Reduction Program
Rough-Set Reduction
Rule Generation Programs
1. Characteristic Rules (DBChar)
2. Classification Rules (DBClass)
3. Decision Rules (DBDeci)
4. Maximal Generalized Rules (DBMaxi)
5. Multiple Knowledge Bases (DBMkbs)
6. Data Trend Regularities (DBTrend)

Figure 7.1: The architecture of DBROUGH

DBMkrs: Find multiple sets of knowledge rules for the target class

In order to constrain a knowledge discovery process to generalization on a particular set of data using a particular set of background knowledge, learning should be directed by specific requests. A database learning request should consist of (i) a database query which extracts the relevant set of data, (ii) the kind of rules to be learned, (iii) the specification of the target class and possibly the contrasting classes depending on the rules to be learned, (iv) the preferred concept hierarchies, and (v) the preferred form to express learning results. Notice that (iv) and (v) are optional since default concept hierarchies and generalization threshold values can be used if no preference is specified explicitly.

In our system DBROUGH, the learning procedure is initiated by a user learning request. The learning request can be viewed as an extension to relational language SQL for knowledge discovery in databases.

We have implemented DBROUGH using C under an Unix/Sybase environment. A high level interface has also been constructed with the assistance of UNIX software package LEX and YACC (for compiling the DBROUGH language interface) for the specification of learning tasks (either characteristic rules, classification rules, decision rules or maximal generalized rules and so on), conceptual hierarchies and thresholds as well as for communication with users in the learning process.

The syntax of the language is specified in Table 7.1 using extended BNF, where { } denotes one or more occurrences, *Target_Class_Name, Contrast_Class_Name, Rel_Name, Attr_Name, Concept_Hierarchy_Name* are the corresponding names specified by users, and *Int_Val* is a constant greater than 0.

```
<DBROUGH>          := learn <rule_type>
<rule_type>        := <charact_rule> | <class_rule> | <decision_rule> |
                      <maxi_gen_rule> | <mkr_tule> | <datatrend_rule>
<charact_rule>     := characteristic rule for <Class_name> <DB_name>
                      <Cond> <attr_list><tab_threshold> <con_hierarchy>
<class_rule>       := classification rule for Target_Clas_Name vs
                      {Contrasting_Class_Name} <DB_name><Cond>
                      <attr_list><tab_threshold> <con_hierarchy>
```

<decision_rule>	**:= decision rule for < Class_Name><DB_name><Cond> {<attr_list>}<tab_threshold> <con_hierarchy>**
<maxi_gen_rule>	**:= maximal generated rules for <Class_Name> <DB_name><Cond><attr_list><tab_threshold> <con_hierarchy>**
<mkr_rule>	**:= multiple knowledge rule for <Class_name> <DB_name> <Cond><attr_list><tab_threshold><con_hierarchy>**
<datatrend_rule>	**:= data_trend_regularities for <Class_name> <DB_name> <Cond><attr_list><tab_threshold> <con_hierarchy>**
<DB_name>	**:= from relation {Rel_Name}**
<Cond>	**:= where Condition_Sentence**
<attr_list>	**:= in relevant to attributes <attr>**
<attr>	**:= <attrs>, <attr>**
<attrs>	**:= Attr_Name**
<Class_Name>	**:= Attr_Name | Attr_Name=attribute_value**
<tab_threshold>	**:= using threshold Int_Val**
<con_hierarchy>	**:= using hierarchy hier_name**
<hier_name>	**:= Concept_Hierarcy_Name**

Table 7.1 Syntactic specification of DBROUGH.

7.2 Experimental Results of Some Algorithms

To test the effectiveness of our system DBROUGH, we present the experimental results of some discovery algorithms of DBROUGH on Canada's Natural Science and Engineering Research of Council (**NSERC**) Grants Information System and Car Relation as shown in Chapter 5.

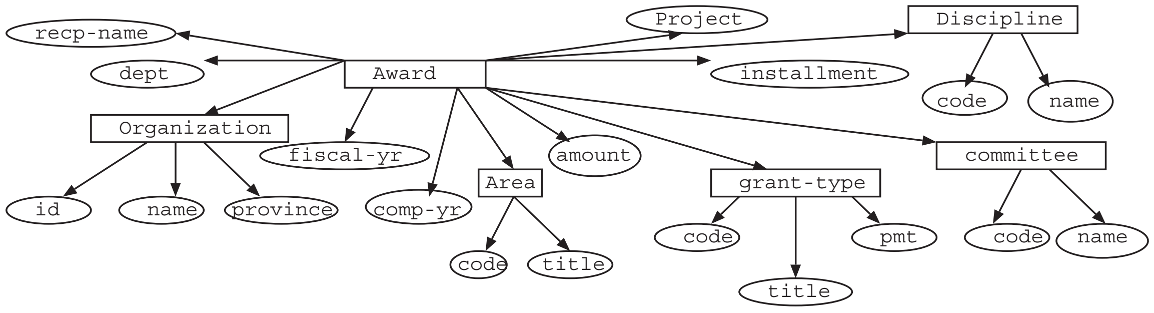


Figure 7.2: Schema diagram for NSERC grants information system

7.2.1 NSERC Grants Information System

The NSERC Grants Information System is a software package consisting of a database of information about the grants that are awarded by NSERC and a menu-based interface to that database. It is intended to be used by individuals in " universities, government agencies and industry... to search for grants that are of particular interest" [HCC92a].

The NSERC Grants Information System contains a database of information about the grants that are awarded by NSERC. The central table in the database has 10,087 tuples with 11 attributes currently. The central table in the database is made of rows each of which describes an award by NSERC to a researcher. The values constituting each row specify the different properties of the award, including the name of the recipient, the amount of the award and so on. In the schema diagram Figure 7.2, nodes representing the properties of awards are represented by nodes linked to the "Award" node. In the schema diagram, tables are specified by rectangular nodes.

The NSERC database can also be represented by the following relation-like schema.

Award(recp_name, dept, org_code, fiscal_yr, comp_yr, area_code, amount, grant_code, ctee_cde, installment, discipline_code, project)

Organization(org_code, org_name, province)

Area(area_code, area_title)

Grant_type (grant_code, grant_title, pmt)

Committee (ctee_code, cname)

Discipline (discipline_code, disc_title)

The task-specific concept hierarchies (shown in Figure 7.3) are constructed by both domain expert and knowledge discovery tools based on the statistics of data distribution in the database. The most general concept is the null description (described by a reserved word "ANY"), and the most specific concepts correspond to the specific values of attributes in the database.

$\{$0–20,000 $\} \subset$ 0-20Ks

$\{$20,000–40,000 $\} \subset$ 20Ks-40Ks

$\{$40,000-60,000 $\} \subset$ 40Ks-60Ks

$\{$60,000– $\} \subset$ 60Ks–

$\{$0–20Ks $\} \subset$ Low

$\{$20Ks–40Ks, 40Ks–60Ks $\} \subset$ Medium

$\{$60Ks– $\} \subset$ High

$\{$Low, Medium, High $\} \subset$ Any (amount)

$\{$0–15 $\} \subset$ Operating-Grants

$\{$150–165 $\} \subset$ Strategic-Grants

$\{$16–149, 166– $\} \subset$ Other

$\{$Operating-Grant, Strategic-Grants, Other $\} \subset$ Any(grant_code)

$\{$23000–23499 $\} \subset$ Hardware

$\{$23500–23999 $\} \subset$ System_Organization

$\{$24000–24999 $\} \subset$ Software

$\{$24500–25499 $\} \subset$ Theory

$\{$25500-25999 $\} \subset$ Database_Systems

$\{$26000–26999 $\} \subset$ AI

$\{$26500-26999 $\} \subset$ Computing_Method

$\{$0–22999, 27000– $\} \subset$ Other_Discipline

$\{$Hardware, System_Organization, Software, Theory, Database_Systems, AI, Computing_Method, Other_Discipline$\} \subset$ ANY(discipline_code)

$\{$British Columbia $\} \subset$ B.C.

$\{$Alberta, Manitoba, Saskatchewan $\} \subset$ Prairies

$\{$Ontario $\} \subset$ Ont.

$\{$Quebec $\} \subset$ Queb.
$\{$New Brunswick, Nova Scotia, Newfoundland, PEI $\} \subset$ Maritime
$\{$B.C., Prairies $\} \subset$ West_Canada
$\{$Ont., Queb.$\} \subset$ Central_Canada
$\{$Maritime$\} \subset$ East_Canada
$\{$West_Canada, Central_Canada, East_Canada$\} \subset$ Any(province)

Figure 7.3. A concept hierarchy table of the NSECR grants database

7.2.2 Some Test Results

Example 7.1 (DBChar)
The learning task "learning the characteristic rule for the operating grants awarded to computer science discipline from relation *award*, *organization*, and *grant_type* referring attributes: amount, province, with a table threshold value equal to 18 by using concept hierarchy file disc, amount, prov, and grant_type" can be specified as follows.

DBROUGH 1> **learn characteristic rule**
DBROUGH 2> **for** "CS_Op_Grants"
DBROUGH 3> **from** award A, organization O, grant_type G
DBROUGH 4> **where** O.org_code = A.org_code **AND** G.grant_order ="Operating_Grants" **AND** A.grant_code = G.grant_code **AND** A.disc_code ="Computer"
DBROUGH 5> **in relevance to** amount, province, prop(votes), prop(amount)
DBROUGH 6> **using table threshold** 18
DBROUGH 7> **using hierarchy** disc, amount, prov, grant_type

Notice that *prop(attribute)* is a built-in function which returns the percentage of the summation of the *attribute* value in the generalized tuple divided by the summation of the same *attribute* value in the whole generalized relation. The type of the *attribute* must be "int" or "float". *Votes* is a special attribute which registers the number of tuples in the original relation which are generalized to one tuple in the final generalized relation. *Prop(votes)* returns the percentage of tuples covered by a

generalized tuple in the final relation.
A default attribute threshold value, 5, is used in this query. Finally, you have to type "*go*" on a line by itself. It is the command terminator in DBROUGH, and let DBROUGH know that you are done typing and ready for your command to be executed.

DBROUGH first transforms the user learning request into High Level SQL query as below:

```
*******************************************
High level SQL query for task-relevant data
******************************************

   select amount, province
from award A,organization O,grant_type G
where ( O.org_code = A.org_code AND G.grant_order ="Operating_Grants"
AND A.grant_code = G.grant_code AND A.disc_code ="Computer" )
```

As one can see in the High Level SQL query, "Operating_Grants" and "Computer" are high level concepts in the concept hierarchies and are not the primitive data in the database, so DBROUGH replaces them by the primitive data (concept) stored in the database by consulting the corresponding concept hierarchies. For example , "Computer" (*discipline_code*) contains {Hardware, System_Organization, Software, Theory, Database_Systems, AI, Computing_Method, Other_Discipline}. Hence "Computer" in the query is replaced by the disc_code of the corresponding lower level concept, resulting in the primitive query for task-relevant data as follow:

```
***********************************************
Primitive level SQL query for task-relevant data
**********************************************
```

```
    select amount, province
from award A,organization O,grant_type G
where ( O.org_code = A.org_code) AND ( G.grant_order = 5 or G.grant_order = 10
or G.grant_order = 15 ) AND A.grant_code = G.grant_code
AND (( disc_code >= 23000 and disc_code < 23500 )
or ( disc_code >= 23500 and disc_code < 24000 )
or ( disc_code >= 24000 and disc_code < 24500 )
or ( disc_code >= 24500 and disc_code < 25500 )
or ( disc_code >= 25500 and disc_code < 26000 )
or ( disc_code >= 26000 and disc_code < 26500 )
or ( disc_code >= 26500 and disc_code < 27000 ) ) )
```

Then DBROUGH extracts the task-relevant data from the NSERC grants information system, after attribute-oriented generalization and rough set based reduction, the resultant relation is shown in Table 7.1, hence the characteristic rules for "CS_Op_Grants" is derived as:

```
*******************************************
The characteristic rule for"CS_Op_Grants" is:
******************************************

    For all x, CS_Op_Grants(x) —->
( ( amount = 0-20Ks ) and ( province = [ Ont. , Queb. ] ) [38.272%])
or ( ( amount = 20Ks-40Ks ) and ( province = [ Ont. , Prairies ]) [18.107%])
or ( ( amount = [ 40Ks-60Ks , 0-20Ks ] ) and ( province = B.C. ) [ 8.642%])
or ( ( amount = 20Ks-40Ks ) and ( province = [ Queb. , B.C. ] ) [10.494%])
or ( ( amount = 40Ks-60Ks ) and ( province = [ Ont. , Prairies ] ) [ 5.350%])
or ( ( amount = 0-20Ks ) and ( province = [ Prairies , Maritime ] ) [15.021%])
or ( ( amount = [ 40Ks-60Ks , 60Ks- ] ) and ( province = Queb. ) [ 1.235%] )
or ( ( amount = 60Ks- ) and ( province = [ Ont. , Prairies ] ) [ 1.646%] )
or ( ( amount = 20Ks-40Ks ) and ( province = Maritime ) [ 0.010%])
```

amount	province	prop(votes)	prop(amount)
0-20Ks	Ont.	24.49%	3.88%
0-20Ks	Queb.	13.79%	2.92%
20Ks-40Ks	Ont.	12.76%	2.22%
20Ks-40Ks	Prairies	5.35%	9.69%
40Ks-60Ks	B.C.	1.23%	4.58%
0-20Ks	B.C.	7.41%	4.24%
20Ks-40Ks	Queb.	5.14%	5.22%
40Ks-60Ks	Ont.	5.14%	5.54%
0-20Ks	Prairies	8.23%	11.20%
0-20Ks	Maritime	6.79%	4.61%
20Ks-40Ks	B.C.	5.35%	7.01%
40Ks-60Ks	Prairies	0.21%	3.32%
40Ks-60Ks	Queb.	1.03%	4.23%
60Ks-	Ont.	1.23%	10.66%
60Ks-	Prairies	0.41%	4.99%
20Ks-40Ks	Maritime	1.03%	6.37%
60Ks-	Queb.	0.21%	3.78%
60Ks-	B.C.	0.21%	5.54%

Table 7.1: The final generalized relation

disc_code	grant_order	amount	votes
Computer	Operating_Grants	20Ks-40Ks	62
Computer	Operating_Grants	40Ks-60Ks	25
Computer	Other	60Ks-	7
Computer	Other	40Ks-60Ks	5
Computer	Strategic_Grants	60Ks-	8
Computer	Operating_Grants	60Ks-	6
Computer	Strategic_Grants	40Ks-60Ks	1

Table 7.2: The final generalized relation

or ((amount = 60Ks-) and (province = B.C.) [0.002%])

Example 7.2 (DBCLass)
Similarly, the following learning request learns the discrimination rule that can distinguish the computer science grants awarded to Ontario from those awarded to Newfoundland.

DBROUGH 1> **learn discrimination rule**
DBROUGH 2> **for** "Ontario_CS_Grants"
DBROUGH 3> **where** O.province = "Ontario"
DBROUGH 4> **in contrast to** "Newfoundland_CS_Grants"
DBROUGH 5> **where** O.province = "Newfoundland"
DBROUGH 6> **from** award A, organization O, grant_type G
DBROUGH 7> **where** A.grant_code = G.grant_code **AND** A.org_code = O.org_code
AND A.disc_code = "Computer"
DBROUGH 8> **in relevance to** disc_code, amount, grant_order

Notice that both attribute and table threshold value are default ones. All the concept hierarchy information required is stored in a default file *concept*

****

The classification rule for“Ont_Grants” vs “Newfoundland_Grants” is:
****

For all x, Ont_Grants(x) <—-
((disc_code = Computer) and (grant_order = Operating_Grants) and
(amount = [20Ks-40Ks , 40Ks-60Ks]) [34.387%])
or ((disc_code = Computer) and (grant_order = Other) and
(amount = [60Ks- , 40Ks-60Ks]) [4.743%])
or ((disc_code = Computer) and (grant_order = [Strategic_Grants ,
Operating_Grants]) and (amount = 60Ks-) [5.534%])
or ((disc_code = Computer) and (grant_order = Strategic_Grants) and
(amount = 40Ks-60Ks) [0.004%])

Example 7.3 (DBDeci)
This experiment shows how the decision rules were used to analyze the possibility of bankruptcy for a firm based on five financial indicators. The data were based on E.L. Altman's [Alt68].

The data set contains 66 collected records which represent either *bankrupt* or *non-bankrupt* firms. The five numerical attributes correspond to five financial ratios: W: *working capital/total assets*; R:*retained earnings/total assets*; E:*earning before interest and taxes/total assets*; M:*market value of equity/book value*; and S:*sales/total assets*.

The objective of this test is to analyze the data and to compute a set of decision rules. This is a set of rules can be used to predict a firm's potential of bankruptcy on the basis of its previous performance. The decision rules for both bankrupt and non-bankrupt companies are as follows:

$(M \leq 35.00) \rightarrow bankrupt$

$-100.30 < W \leq -13.60) \rightarrow bankrupt$

$(R \leq 7.20) \vee (E \leq 1.60) \rightarrow bankrupt$

$(E \leq 1.60) \vee (35.00 < M \leq 72.00) \rightarrow bankrupt$

$(W \leq -100.30) \rightarrow bankrupt$

Concepts	Decision Matrix Predictions	MDA Prediction
Banrupt	97.0	96.0
Non-Bankrupt	97.0	78.8
All (Concepts)	97.0	83.5

Table 7.3: Comparision of decision matrix method to MDA method

$(R \leq 7.20) \vee (35.00 < M \leq 72.00) \rightarrow bankrupt$

$(R \leq 7.20) \vee (35.00 < S \leq 72.00) \rightarrow bankrupt$

$(R \leq 1.60) \vee (3.00 < S \leq 6.77) \rightarrow bankrupt$

$(1.60 < E \leq 34.40) \vee (72.00 < M \leq 779.00) \rightarrow Non\_Bankrupt$

$(7.20 < R \leq 69.30) \vee (35.00 < M \leq 72.00) \rightarrow Non\_Bankrupt$

$(1.60 < E \leq 34.40) \vee (35.00 < M \leq 72.00) \rightarrow Non\_Bankrupt$

$(7.20 < R \leq 69.30) \vee (1.60 < E \leq 34.40) \rightarrow Non\_Bankrupt$

$(7.20 < R \leq 69.30) \vee (72.00 < M \leq 779.00) \rightarrow Non\_Bankrupt$

$(7.20 < R \leq 69.30) \vee (3.00 < S \leq 6.77) \rightarrow Non\_Bankrupt$

$(1.60 < E \leq 34.40) \vee (3.00 < S \leq 6.77) \rightarrow Non\_Bankrupt$

$(72.00 < M \leq 779.00) \vee (3.00 < S \leq 6.77) \rightarrow Non\_Bankrupt$

The rule demonstarted very good prediction capabilities when validated using cross-validation procedures. The rules were correct 97.00% of the time using the Leave-One-Out method. The results were then compared to the multiple discriminant analysis (MDA) reported by Altman [Alt68]. The performance of each method is depicted in Table 7.3.

Example 7.4 (DBMaxi)

Experimental Results of Three Test Data Sets: IRIS Data, Appendicitis Data, Thyroid Data.

Fisher's [Fis36] IRIS Flower data base is a well-known data set used as a standard benchmark example in today's rule discovery research. Three classes of iris type, *i.e. virginica, versicolor* and *setosa* are described by four numerical attributes, *i.e., sepal length, sepal width, petal length* and *patal width*. The data set consists of 150 cases,

Methods	Iris		Appendictis		Thyroid	
	Training	Testing	Training	Testing	Training	Testing
Decision Matrix	100.0	95.33	99.06	91.05	100	98.86
Linear	98.00	98.00	88.70	86.80	93.85	93.85
Quadratic	98.00	97.3	78.30	73.60	89.69	88.39
Nearest Neighbour	100.0	96.00	100.0	82.10	100.0	95.27
Bayes Indendence	95.30	93.3	88.70	83.00	97.03	96.06
Bayes 2nd Order	96.00	84.00	95.30	81.10	97.72	92.44
Neural Net (BP)	98.30	96.67	90.20	85.80	99.5	98.54
PVM Rule	97.30	96.00	91.50	89.60	99.79	99.33
CART Tree	96.00	95.33	90.6	84.90	99.79	99.36

Table 7.4: The comparative performance

50 cases for each class.

The Appendicitis Data set is from a published study on the assesment of eight lab tests to confirm the diagnosis of appendicitis [MAG80]. Following surgery, only 85 of 106 patients was confirmed by biopsy to have had appendicitis. Thus, the ability to discriminate the true appendicitis patients by lab tests prior to surgery would prove extremely valuable. The sample consisted of 106 patients and eight diagnostic tests.

The thyroid data is used to determine whether a patient referred to the clinic was hypothyroid. There are three classes: *normal (not hypothyroid), hyperfunction* and *subnormal functioning*. The training data set consisted of 3772 cases and the testing data consisted of 3428 cases. There were 22 symbolic and numeric attributes. Over 10% of the values were missing because some lab tests were deemed unnecessary. The data set used here are the same as described in [WeK89].

Table 7.4 shows the results of decision matrix method and the comparision results reported by Weiss [WeK89]

Chapter 8

Discussion

8.1 A Comparison with Other Learning Methods

Our learning procedure consists of two phases: data generalization and data reduction. Our method uses attribute-oriented induction for generalization, which provides an efficient way to generalize the database and greatly reduce the computational complexity. The efficiency of the attribute-oriented generalization can also be demonstrated by analyzing its worst case time complexity. Suppose there are N tuples in the database which are relevant to the learning task, A attributes for each tuples, and H levels for each concept tree, the time complexity in the worst case is analyzed as follows. For each attribute, the time for substituting the lower level concepts by the higher level concepts is N, and the time for checking redundant tuples is $NlogN$. Since the height of the concept tree is H, the time spent on each attribute is at most $H * (N + NlogN)$. Obviously, the upper bound of the total time for processing A attributes is $A * H * (N + NlogN)$. In general, A and H are much smaller than N in a large database. Therefore, the time complexity of our approach is $O(NlogN)$ in the worst case, which is more efficient than the tuple-oriented generalization.

In data reduction, suppose there are only N' tuples with A' attributes left in the generalized relation, to construct the discernibility matrix, it only takes $O(N' \times N')$ steps. To search the core attributes in a discernibility matrix, it costs $O(N' \times N')$. To find the reduct for the condition attributes, in the worst case, the complexity is $A' \times O(N' \times N')$. Since A' is usually much less than N', the worst case in the reduction process is $O(N' \times N')$.

Then we examine other learning methods. Most learning algorithms in the literature [DiM83] are tuple-oriented algorithms. A tuple-oriented method examines data in the database tuple by tuple and performs generalization based on the comparison of tuple values with the intermediate generalization results. Since the number of the possible tuple combinations is exponential to the number of tuples in the relevant data set, the worst case complexity of the generalization process is exponential to the size of the relevant data sets.

8.2 Search Space

A concept tree ascending technique is the major generalization techniques used in both attribute-oriented generalization and tuple-oriented generalization. However, the tuple-oriented approach performs generalization tuple by tuple, but the attribute-oriented approach performs generalization attribute by attribute. We compare the search spaces of our algorithms with that of a typical method of *learning from examples*, the *candidate elimination* algorithm [DiM83]

In the candidate elimination algorithm, the set of all concepts which are consistent with the training examples is called the version space of the training examples. The learning process is the search in this version space to induce a generalization concept which is satisfied by all of the positive examples and none of the negative examples.

Since generalization in an attribute oriented approach is performed on an individual attribute, a concept hierarchy of each attribute can be treated as a factored version space. Factoring the version space significantly improves the general efficiency. Suppose there are p nodes in each concept tree and there are k concept trees (attributes) in the relation, the total size of a k factorized version space is pk. However, the size of the unfactorized version space for the same concept tree should be p^k.

8.3 Utilizing Database Facilities

Relational database systems provide many attractive features for machine learning, such as the capacity to store a large amount of information in a structured and organized manner and the availability of well developed implementation techniques. However most existing algorithms do not take advantage of these database facilities [CCH91]. An obvious advantage of our approach over many other learning algorithms is the integration of the learning process with database operations. Most of the operations used in our approach involve traditional relational database operations, such as selection, join, projection (extracting relevant data and removing attributes), tuple substitution (ascending concept trees), and intersection (discovering common tuples among classes). These operations are set-oriented and have been efficiently implemented in many relational systems. While most learning algorithms suffer from inefficiency problems in a large database environment [CCH91,HCC92a,HCC92b], our approach can use database facilities to improve the performance.

8.4 Dealing with Different Kinds of Concept Hierarchies

In our examples, all of the concept hierarchies are represented as balanced concept trees and all of the primitive concepts reside at the same level of a concept tree. Hence generalization can be performed synchronously on each attribute to generalize the attribute values at the same lower level to the ones at the same higher level. However, we may encounter other kinds of concept hierarchies or we may encounter the case where the primitive concepts do not reside at the same level of a concept tree.

Generalization of the Concepts at Different Levels of a Hierarchy

The concept hierarchies may be organized as unbalanced concept trees. For example, the left branch of a tree may have fewer levels of leaves than the right branch. In these cases, synchronous tree ascension may reach the same level at different stages, which may result in an incorrect generalization at that level. A similar problem

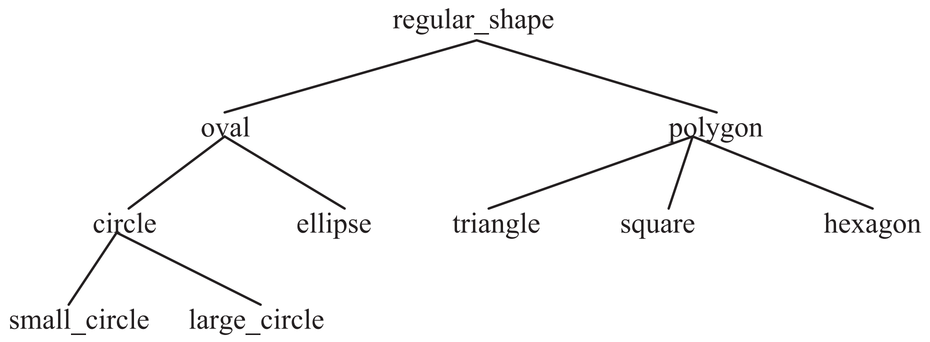


Figure 8.1: An unbalanced concept tree

may occur when the primitive concepts reside at the different levels of a concept tree. These problems can be solved by checking whether one generalized concept may cover other concepts of the same attribute. If one generalized concept covers a concept several levels down the concept tree, the covered concept is then substituted for by the generalized concept, that is, ascending the tree several levels at once.

Figure 8.1 shows an unbalanced concept tree. Based on the discussion above, as long as the attribute value "ellipse" has been generalized to "oval", those attribute values, "small_circle", "large_circle" and "circle", can be substituted by "oval" at once.

This idea can be used for incremental learning as well. Relational databases are characterized by frequent updating. As new data become available, it will be more efficient to amend and reinforce what was learned from previous data than to restart the learning process from scratch [HCC92]. Our algorithms are able to be extended to perform incremental learning. When new data are presented to a database, an efficient approach to characterization and classification of data is to first generalize the concepts of the new data up to the level of the rules which have been learned, then the learning algorithms can be used to merge the generalized concepts derived from the old data and the new data.

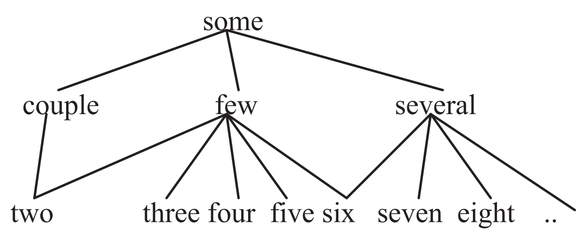


Figure 8.2: A concept tree with lattices

Generalization of Concepts in the Hierarchies with Lattices

In all of our previous examples, the concept hierarchies are trees, that is, every node has only one parent node. For any concept, therefore, there is only one direction to perform the generalization. In some cases, however, the concept hierarchy may be a lattice. Figure 8.2 illustrates this case.

As illustrated in Figure 8.2, the concept "two" can be generalized either to "couple" or "few". Both generalized concepts should be considered. Our method is to put all possible generalized concepts into intermediate generalized relations when a lattice is encountered, and then perform further generalization on all those tuples. In this example, after the tuple containing attribute value "two" is generalized, two new tuples, containing attribute values "couple" and "few", respectively, should be generalized. For the concept "six", the same technique should be applied. As a consequence, the size of the generalized relation table may increase at some stage of the generalization process because of the effect of a lattice. However, since the generalization is controlled by the specified value, the generalized relation will eventually shrink in further generalization.

8.5 Discovery of Knowledge by Conceptual Clustering

Most conceptual classification algorithms in the literature [MiS83, Fi87a] are tuple-oriented algorithms. A tuple-oriented algorithm examines data in the database tuple by tuple and performs generalization and classification based on the comparison of tuple values with the intermediate generalization results. Since the number of possible tuple combinations is exponential to the number of tuples in the relevant data set, the worst case complexity of the generalization and classification process is exponential to the size of the relevant data sets. But our method uses a new method to classify the data set based on the common attribute values between different tuples. At each iteration, a matrix is constructed in $O(n^2)$ where n is the number of the tuples of the data set. According to the distribution of the values in the matrix, a suitable value is chosen which is a similarity measure for classification.

The advantages of our method include:

(1) Our algorithm can automatically find a hierarchy table without assistance. The number of clusters and the levels of the hierarchy are determined by the algorithm; it is unlike the famous CLUSTSER/2 in which the user must specify the number of final clusters and the initial seeds in the beginning.

(2) Objects are not assigned to clusters absolutely.

Our method calculates the similarity between each pair of objects, providing a more intuitive classification than absolute partitioning techniques. Our method aggregates objects from bottom to top based on the similarity between them and if an object has the same number of common attribute value to two clusters, then the object is assigned to both clusters.

(3) The threshold value has a big influence on whether or not an instance is admitted to a class. We can vary the threshold, get different hierarchy tables so the algorithm can generate different sets of rules to meet the needs of varied applications.

8.6 Reduction of Databases

In DBROUGH, the learning procedure is initiated by a learning request submitted from the user. The query condition determines what data should be retrieved from the DBMS. This is accomplished by specifying which tables need to be accessed, which fields should be returned, and which or how many records should be retrieved. Learning task are those tuples which satisfying the query conditions and the specified fields, which greatly reduce the search space of the data. Using rough set theory, the minimal attribute set or reduct of the attribute in the databases can be computed and each reduct can be used instead of the whole attribute set without losing any essential information. By removing those attributes which are not in the reduct, the generalize table can be further reduced.

8.7 Data Evolution Regularity

One of the big challenge facing KDD is that the content of data is constantly changing. There are a lot of algorithms developed to find rules from databases directly [FrP91, CeT93], but all these algorithms assume that the data and the data scheme are stable and most of the algorithms focus on discovering the regularities about the current data in the databases. The reality is that the contents of databases and database scheme may change over time and users are often interested in finding the general trends of data evolution to predict the future. So it is important to discover data evolution regularities in a dynamic evolving database. Since the data for the future is usually not available at the current time, we have to learn the data trend regularities for the future data based on the current data in the databases. Machine learning technology should be adopted to extract such regularities in databases. In this section we use an example to illustrate how to expand the attribute-oriented rough set approach to learn data evolution regularities.

One of the key issues to learn from data in a dynamic environment is how the relationships between the instance in different states are defined. In our method, we combine the concept hierarchy with the transition constraints to model the relationship between the instances in different states.

We say that an entity which is an instance of one class (called the source class) undergoes a transition when it becomes an instance of another class (called target class). There are two types of transition **evolution** and **extension** [HaG90], based on whether or not the entity undergoing the transition is preserved as an instance of the source class or not. In other words, an evolution occurs when the transition entity ceases to be an instance of the source class. For example, when an entity representing

name	sex	birthday	employer	salary	dependents
Sam	M	Dec. 5, 1954	NCR	70k	3
Janet	F	Aug. 4, 1965	BNR	53k	3
Mary	F	June 23, 1945	NT	60k	3
Tom	M	July 17, 1963	Gov.	36k	0
...	..	....	..	..	..
Jay	M	Oct. 24, 1970	MPE	40k	1
Mark	M	Jan. 29, 1940	NGE	100k	2

Table 8.1: **Adult** relation

an applicant changes to reflect the acceptance of the applicant, it undergoes an evolution; that is, it ceases to be an instance of the applicant and becomes an instance of the student. An extension is a transition with the negative of the additional condition associated with evolution. In other words, an extension occurs when the entity remains an instance of the source class with the negation of the additional condition associated with evolution. For example, when an alumnus with a Master's degree applies to the Ph.D program, the transition of the entity representing the alumnus into an instance subclass is an extension.

Note that some of the transition events are triggered solely by time whereas others are triggered by other events in the dynamic system. To make our explanation simple, we assume only evolution occurs in our dynamic environment model and all the transitions are triggered by time.

Consider a simple version of the social security database in some social benefit office in Canada as shown in Table 8.1, 8.2 (a), (b). Figure 8.3 is the concept hierarchies for attributes **age, salary** and **pension.** Figure 8.4 is the corresponding concept hierarchy and transition network. Citizen may start as a child. When children reach the age of 18, they become an instance of Adult. Later, at age 65, they retire (senior citizen) and eventually die. The transition from senior citizen to death is weak because some people may live older than 85 while some other may not. We use $\not\longrightarrow$ to represent weak transition.

{0–4} : children; {4–14}: teenages; {14–20} : young

{20-29}: twenties; {30–39}: thirties; {40–49}: forties

{50–64}: late_mid; {65-}: old

{children , teenages}: child_age; {young, twenties}: young_age

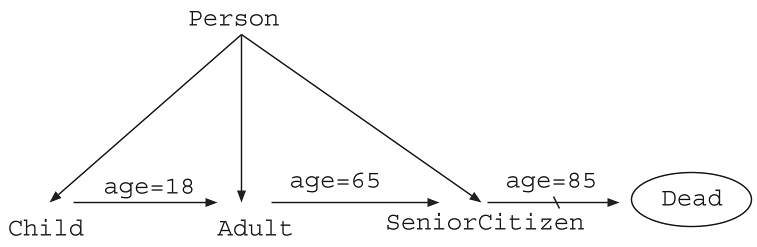
name	sex	birthday	school	guardian
Jane	F	Oct. 5, 1984	No. 1	Sam
Janet	F	June. 4, 1986	No.1	Mary
Mary	F	June 6, 1983	No. 2	Tom
Peter	M	July 17, 1979	Bran	Mark
.....	....	..	..	
John	M	Feb 24, 1980	MMM	Jay
Frank	M	Jan. 29, 1982	PCC	Janet

(a)

name	sex	birthday	pension
Woope	F	Oct. 5, 1925	17k
Jason	M	July 14, 1929	23k
Rose	F	Jan. 28, 1913	60k
..	...	....	..
Codoba	M	Aug.,24, 1910	40k
Clark	M	Feb. 23, 1914	10k

(b)

Table 8.2: (a) **Child** relation; (b) **Senior citizen** relation



```
SeniorCitizen.pension=Adult.salary when retired * 65%
Child.name=Adult.Name=SeniorCitizen.name
age=current date-birthday
```

Figure 8.4: The class hierarchy and transition network for people

{thirties, forties, late_mid}: mid_age; { old}: old_age

{child_age, youth_age, mid_age, old_age}: Any(age)

{0–20k}: low_income; {20K–34k}: low_middle_income; {35k-45k}: mid_income

{46–65k}: high_income; {66k–}:very_high_income;

{low_income, low_mid_income, mid_income, high_income, very_high_income}: Any(income)

Figure 8.3: The concept hierarchy for age, salary, pension

To discover data evolution regularities in the future, the evolving data should be identified first and be extracted from the database. For example, if the city administrator wants to know the general situation about the senior citizen 5 years later, the query may be submitted as below:

DBROUGH 1> **learn data evolution regularities for** *seniorcitizen S*

DBROUGH 2> *5 years* **later**

DBROUGH 3> **in relevant to** *S.name, S.sex, S.pension*

name	sex	pension
Woope	F	17k
Jason	M	23k
Rose	F	60k
..	..	...
Codoba	M	40k
Clark	M	10k

Table 8.3: Instance of **senior citizen**

The evolving data may have two kinds of attributes: stable attributes and evolving attributes. The stable attributes, in which the data values do not change over time, can be generalized by attribute-oriented induction in the same way as those discussed in Chapter 3. The evolving attributes, in which the data values change over time, can be generalized according to a generalized time slots when appropriate. For example, adult's salary keeps changing yearly and so we need to update the salary based on the time value. Once we get the value for the salary, then we can still apply attribute-oriented induction. The data extraction procedure is performed in two steps (1) extract the target class entities based on the query; (2) examine the class hierarchy and transition network to check whether there are any source class entities which can transform to the current learning class as time goes by. For example, for the above query, the first step is to extract all the citizens from the current senior citizen relation except those who are 80 years old (because we assume that a senior citizen dies at 85). Then we examine the concept hierarchy and transition network and find an Adult becomes a senior citizen when he reaches 65. Hence we have to look through the Adult relation and extract those adults who are older than 60 and derive the corresponding attributes values, e.g. replace salary by pension. (We can assume that adult salary increases 4% each year, first compute the adult salary when he retires, and then apply the procedure: seniorcitizen.pension=adult salary when retired * 65 %). As a result, we get a set of task-relevant instances objects as shown in Table 8.3. After we get the task-relevant data, the data generalization and data reduction procedure can be applied in the same way as discussed in previous chapters and interesting data trend regularities can be found [HCX94].

Chapter 9

Conclusion and Future Directions

9.1 Conclusion

The rapid growth of data in the world's databases is one reason for the recent interest in KDD. The vastness of this data also creates one of KDD's greatest challenges. Exhaustive, empirical analysis is all but impossible on the megabyte, gigabytes or even terabytes of data in many real-world databases. In these situations, a KDD system must be able to focus its analysis on samples of data by selecting specific fields and/or subsets of records.

In this thesis, we proposed a framework for knowledge discovery in databases using rough sets and attribute-oriented induction. Our system implements a number of novel ideas. In our system, attribute-oriented induction is applied in the generalization process to remove undesirable attributes and to generalize the primitive data to the desirable level. In the data reduction process, rough set theory is used to compute the minimal attribute set, or reduct of the attribute in the databases and each reduct can be used instead of the entire attribute set, without losing any essential information. By removing those attributes which are not in the reduct, the generalized relation can be further reduced. The rules generated after data generalization and reduction are much more concise and efficacious.

Our method integrates a variety of knowledge discovery algorithms such as DBChar for characteristic rules, DBClass for classification rules, DBDeci for decision rules, DB-Maxi for maximal generalized rules, DBMkr for multiple sets of knowledge rules and

DBTrend for data trend regularities, which permit a user to discover various kinds of relationships and regularities in the data. This integration allows our method to exploit the strengths of diverse discovery programs. Our systems inherit the advantages of the attribute-oriented induction model and rough set theory and make some contribution to the KDD, such as handling large volume data (millions of tuples), redundancy data, uncertainty information, multiple sets of knowledge rules, discover data trend regularities and so on.

KDD systems face challenging problems from real-world databases which tend to be dynamic, incomplete, redundant, noisy and very large. Each of these problems has been addressed to some extent within machine learning, but few, if any, systems address all of them. In this thesis, our system collectively handles these problems while producing useful knowledge rules efficiently and effectively. In our system, we use attribute-oriented induction rather than tuple-oriented induction, thus greatly improving the learning efficiency. By integrating rough set techniques into the learning procedure, the derived knowledge rules are particularly concise and pertinent, since only the relevant and/or important attributes (factors) to the learning task are considered. In our system, the combination of transition network and concept hierarchy provides a nice mechanism to handle dynamic characteristic of data in the databases. For applications with noisy data, our system can generate multiple sets of knowledge rules through a decision matrix to improve the learning accuracy. The experiments using the NSERC information system demonstrate the promise of our method.

9.2 Future Direction

The realization of a general purpose, fully-automated knowledge discovery system is still far from reach. The attribute-oriented rough set approach represents a promising direction to follow in the development of efficient and effective learning strategy for knowledge discovery in databases. There are many issues which should be studied further. The following are some interesting topics for future research.

9.2.1 Applications of Knowledge Rules Discovered from Relational Databases

The knowledge rules learned from relational databases are very useful in many applications, some of which are listed below:

(1) Discovery of knowledge rules from knowledge-base systems and expert systems [ASC95].

Since rules are derived from a huge number of data stored in a relational database, they represent important knowledge about data in the database. Thus our approach is an important method to obtain knowledge rules for knowledge-base systems and expert systems

(2) Processing of queries which involve abstract concepts

In general, relational databases can only answer queries which involve the concepts in the database, but they cannot handle queries like "What are the major characteristic of mammals?" and "How can we describe the major differences between mammals and birds?". Such queries involve concepts which are at a higher level than the primitive data stored in relational databases. By applying the knowledge rules obtained by our learning algorithms, it is possible to answer such learning-requests.

(3) Semantic query optimization using the learned rules.

Learning query-transformation rules are vital for the success of semantic query optimization in domains where the user cannot provide a comprehensive set of integrity constraints. Some queries can be answered more efficiently by the learned knowledge rules without searching databases. For example, the query, "Is there any mammal who has feathers?", usually indicates that the relation must be searched. However, if the characteristic rule indicates that there is no mammal who has feathers, this query can be answered immediately without any search. Learned rules may speed up or optimize database query processing as previously studied in semantic query optimization. Notice that when there is a large number of learned rules, it is nontrivial to search such a rule space. In such a case, there is a trade-off between performing such semantic optimization versus searching the database directly.

9.2.2 Construction of An Interactive Learning System

As illustrated in our learning system, the database learning process is guided by experts or users. Experts and users must specify the learning task and define the threshold value. It is important to obtain such information by interaction with users and experts because:

(1) the system should have a user-friendly interface to facilitates users' communication with the learning system. A more flexible database learning language should be developed for such an interface; and

(2) the entire learning process should be monitored and controlled by users. For example, at some stage of the learning process, users may terminate the generalization on some selected attributes but continue the process on other attributes. In order to obtain multiple rules, users may influence the learning process using different threshold values.

9.2.3 Integration of Multiple Types of Discovery Strategy

Most research in knowledge discovery in databases has been thus far primarily concerned with the development of single-strategy learning approaches. Such approaches include empirical induction from examples, explanation-based learning, learning by analogy, cased-based learning, and abductive learning. Single-strategy approach has specific requirements as to the kind of input information from which they can learn, and the amount of background knowledge needed prior to learning. They also produce different kinds of knowledge. Consequently, they apply to relatively narrow classes of problems.

Real-world problems rarely satisfy all the requirements of single-strategy learning methods. However, they usually satisfy partially the requirements of several strategies. In this context, there is a need for systems that can apply different strategies in an integrated fashion. The method is based on the idea of "understanding" the input through an explanation of system's background knowledge, and an employment of different inference type-deduction, analogy and induction.

A major advantage of the method is that it enables the system to learn in situations in which single-strategy learning methods, or even previous integrated learning methods were insufficient. Therefore, the proposed method reduces to a single-strategy whenever the applicability conditions for such a method are satisfied. In this respect, the multiple strategy method may be regarded as a generalization of these single-strategy methods.

References

[Alt68] E.L. Altman, (1968). Discriminant Analysis and the Prediction of Corporate Bankruptcy, *The Journal of Finance*

[ASC95] A. An, N. Shan, C. Chan, N. Cecone, W. Ziarko, (1995). Discovering Rules from Data for Water Demand Prediction, *accepted in the IJCAI workshop on Machine Learning and Expert System*, Montreal, Canada, Aug. 21-23, 1995.

[ArM86] B. Arbab and D. Michie, (1985). Generating Rules from Examples, *Proc. Ninth Int. Joint Conf. on Artificial Intelligence*, 631-633

[BKM91] C. Baral, S.Kraus, and J. Minker, (1991). Combining Multiple Knowledge Bases, *IEEE Trans. on Knowledge and Data Engineering,* Vol. 3, 208-220

[Boo86] J. Boose, (1986), Rapid Acquisition and Combination of Knowledge from Multiple Experts In The Same Domain. *Future Computing Systems,* 1/2, 191-216

[BuM78] B.G. Buchanan and T. M. Mitchell, (1978). Model-Directed Learning of Production Rules, *Pattern-Directed Inference System, Academic Press*, Waterman et. al. (eds), 291-312.

[CCH91] Y. Cai, N. Cercone and J. Han, (1991). Attribute_Oriented Induction in Relational databases, in *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds), 213-228.

[CeT93] N. Cercone, M. Tsuchiya, (eds), (1993). Special Issue on Learning and Discovery in Knowledge_Based Databases, *IEEE Transaction on Knowledge and Data Engineering,* Vol. 5(6).

[CHH95] N. Cercone, H. Horward, X. Hu and N. Shan, (1995). Data Mining Using Attribute-Oriented generalization and Information Reduction, *invited paper in the Second Annual Joint Conf. on Information Sciences Workshop on Rough Set Theory*, Wrightville Beach, NC, USA

[Cen87] J. Cendrowska, (1987). PRISM: An Algorithm for Inducing Modular Rules, *Int. J. Man-Machine Studies*, Vol. 27, 349-370

[CeB88] B. Cestnik, I. Bratko, (1988). Learning Redundants Rules in Noisy Domains, *Proc. Europe Conf. on Artificial Intelligence*, Munich, Italy 348-351

[Ces90] B. Cestnik, (1990). Estimating Probabilities: A Crucial Task in Machine Learning, *Proc. Europe Conf. on Artificial Intelligence.*

[ClN89] P. Clark, T. Niblett, (1989). The CN2 Induction Algorithm, *Machine Learning Journal*, 3(4): 261-283

[ClB91] P. Clark, R. Boswell, (1989). Rule Induction with CN2: Recent Improvement, *Proc. EWSL 91*, Porto, 151-163

[ChF85] Y. Cheng, K.S. Fu, (1985). Conceptual Clustering in Knowledge Organization, *IEEE Transaction on Pattern Analysis and Machine Intelligence, 9(5)* 592-598.

[CKS88] P. Chessman, J. Kelly, M. Self, J. Stutz, W. Taylor, D. Freeman, (1988). AutoClass: A bayesian Classification System, *Proc. of the Fifth Internatioal Workshop on Machine Learning*, Morgan Kaufmann, San Mateo, CA, 230-245.

[CoF83] P. Cohen and E. A. Feigenbaum, (1983). *The Handbook of Artificial Intelligence* Vol. 3, Heuristic Press and William Kaufmann Inc.

[DiM81] T.G. Dietterich and R.S. Michalski, (1981). Inductive Learning of Structural Descriptions: Evaluation Criteria and Comparative Review of Selected Methods, *Artificial Intelligence*, Vol. 16, 251-294.

[DiM83] T.G. Dietterich and R.S. Michalski, (1983). A Comparative Review of Selected Methods for Learning from Examples, *Machine Learning: An Artificial Intelligence Approach, Vol. 1, Morgan Kaufmann* 41-82

[FaM86] B.C. Falkenhainer and R.S. Michalski, (1986). Integrating Quantitative and Qualitiative Discovery: the ABACUS system, *Machine Learning*, Vol. 1, No.4, 367-401.

[FaI92] U. M. Fayyd, K. B. Irani, (1992). The Attribute Selection Problem in Decision Tree Generation, *Proc. of 1992 AAAI Conf.,* 104-110

[Fi87a] D. Fisher, (1987). Improving Inference Through Conceptual Clustering, *Proc. of 1987 AAAI Conf.*, Seattle, Washington, 231-239.

[Fi87b] D. Fisher, (1987). A Computational Account of Basic Level and Typicality Effects, *Proceedings of 1987 AAAI Conf.*, Seattle, Washington, 461-465.

[Fis36] R. Fisher, (1936). The Use of Multiple Measurements in Taxonomic Problems, *Annals of Eugenics* , Vol. 7, pp 179-188

[FPM91] W. J. Frawley, G. Piatetsky and C.J. Matheus, (1991). Knowledge Discovery in Database : An Overview, *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds) 1-27.

[Gam89] M. Gams, (1989). New Measurements Highlight the Importance of Redundant Knowledge, *Proc. 4th Europe Working Session on Learning*, Momtpellier 71-80

[GLF81] T. Garvey, J. Lowrance amd M. Fischler, (1981). An Inference Technique for Integrating Knowledge from Disparate Sources, *Proc. Seventh Int. Joint Conf. Artificial Intelligence*, 1, 319-325.

[GeN87] M. Genesereth and N. Nilson, (1987). *Logical Foundation of Artificial Intelligence*, Morgan Kaufmann.

[GoS88] R.M. Goodman, P. Smyth, (1988). Decision Trees design from A communication Theory Standpoint, *IEEE Trans. Infor. Theory*, Vol. 34, 979-994

[GrS87] B.J. Gragun and H.J. Studel, (1987). A Decision-Table Based Processor for Checking Completeness and Consistency in Rule-Based Expert-Systems, *Int. J. Man-Machine Studies* 26(5), 633-648

[Grz88] Grzymala-Busse, (1988). Knowledge Discovery Under Uncertainty- A Rough Set Approach, *J. Intell. Rob. Systems*, vol. 1, 3-16

[HCC92a] J. Han, Y.Cai, N. Cercone, (1992a). Knowledge Discovery in Databases: An Attribute-Oriented Approach, *Proceeding of the 18th VLDB Conference*, Vancouver , B.C., Canada, 335-350.

[HCC92b] J. Han, Y.Cai, N. Cercone, (1992). Data_Driven Discovery of Quantiative Rules in Relational Databases, *IEEE Trans. Knowledge and Data Engineering*, 5(2).

[Hau86] D. Haussler, (1986). Quantifying the Inductive Bias in Concept Learning, *Proceedings of 1986 AAAI Conference*, Philadelphia, PA, 485-489.

[Hau87a] D. Haussler, (1987). Bias, Version Spaces and Valient's Learning Framework, *Proc. 4th Int. Workshop on Machine Learning Workshop*, Irvine, CA, 324-336.

[Hau87b] D. Haussler, (1987). Learning Conjuctive Concepts in Structural Domains, *Proceedings of 1987 AAAI Conference*, Seattle, Washington, 466-470.

[HaM77] F. Hayes-Roth and J. McDermott, (1977). Knowledge Acquisition from Structural Descriptions, *Proceedings of 5th International Joint Conference on Artificial Intelligence,* Cambridge, MA,356-362.

[HoM91] J. Hong, C. Mao, (1991). Incremental Discovery of Rules and Structure by Hierarchical and Parallel Clustering, *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds), 177-194.

[HCH93] X. Hu, N. Cercone, J. Han, (1993). Discovery of Konwledge Associated With Conceptual Hierarchies in Databases, *Proc. Third International Conference for Young Computer Scientists, Beijing, China.* 2.106-2.109

[Hux94] X. Hu, (1994). Object Aggregration and Cluster Identification: A Knowledge Discovery Approach, *Applied Math. Letter.* 7(4), 29-34.

[HCH94a] X. Hu, N. Cercone, J. Han, (1993). A Rough Set Approach for Knowledge Discovery in Databases, *Rough Sets, Fuzzy Sets and Knowledge Discovery,* Springer_Verlag Press, W. Ziarko(ed), 90-99

[HuC94a] X. Hu, N. Cercone, (1994). Learning in Relational Databases: A Rough Set Approach, *Computational Intelligence : An International Journal* , special issue on rough set and knowledge discovery, 11(2), 323-338

[HuS94] X. Hu, N. Shan, (1994). Multiple Knowledge Bases and Rough Set, *Proc. of the 7th Florida Research Symposium on AI,* 255-258

[HCS94] X. Hu, N. Cercone. N. Shan, (1994). A Rough Set Approach to Compute All Maximal Generalized Rules, *Proc. of the 6th International Conference on Computing and Information,* Peterborough, Ontario, Canada, May 26-28. 1078-1089.

[HSCZ94] X. Hu, N. Shan, N. Cercone, W. Ziarko, (1994). DBROUGH: A Rough Set Based Knowledge Discovery System, *Proc. of the 8th International Symposium on Methodologies for Intelligent System*, Lecture Notes in AI 869 (Methodologies for Intelligent Systems), Spring Verlag, 386-395

[HCH94b] X. Hu, N. Cercone, J. Han, (1994). A Concept-based Knowledge Discovery Approach in Databases, *Proc. of the 10th Canadian Artificial Intelligence Conference*, 47-62, Banff, Alberta, Canada

[HCX94] X. Hu, N. Cercone, J. Xie. (1994). Learning Data Trend Regularities From Databases in A Dynamic Environment, *Proc. of the AAAI Knowledge Discovery in Databases Workshop,* 323-334

[HuC94d] X. Hu, N. Cercone, (1994). Discovery of Decision Rules from Databases: A Rough Set Approach, *Proc. of the Third Internatinal Conference on Information and Knowledge Management,* Gaithersburg, Maryland, Nov. 1994, 392-400

[HuC95a] X. Hu, N. Cercone, (1995). Rough Sets Similarity-Based Learning From Databases, *accepted in the 1st International Conference on Knowledge Discovery and Data Mining*, Montreal, Canada, Aug. 21-23, 1995

[HuC95b] X. Hu, N. Cercone, (1995). Knowledge Discovery in Databases: A Rough Set Approach, *submitted*

[Kon89] I. Kononenko, (1989). ID3, Sequential Bayers, Naive Bayes and Bayesian Neural Networks, *Europe Workshop on Learning.* 91-98.

[Kon91] I. Kononenko, (1991). An Experiment in Machine learning of Redundant Knowledge, *Proc. Intern Conf. MELECON*, Ljubljana 1146-1149

[KoK93] Igor Kononko, Matevz Kovacie, (1993). Learning as Optimization: Stochastic Generation of Multiple Knowledge, *Proceeding of the 9th International Workshop on Machine learning (ML93),* Aberden, Scotland, 259-262

[KMK91] K.A. Kaufman, R.S. Michalski and L. Kerschberg, (1991). Mining for Knowledge in Databases: Goals and General Descriptions of the INLEN System, *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds), 449-462.

[Lan77] P.W. Langley, (1977). Rediscovery Phisics with BACON 3, *Proceeding of the 5th IJCAI Conference*, Cambridge, MA, 505-507

[Len77] D.B. Lenat, (1977). On Automated Scientific Theory Formation: a Aase Study Using the AM program. *Machine Intelligence 9*, J. E. hayes, D. Michie and L. I. Mikulich (eds), Haalsted Press, 251-256.

[Lub89] D.J. Lubinsky, (1989). Discovery from Database: A Review of AI and Statistical Techniques, *Proceedings of IJCA-89 Worshop on Knowledge Discovery in Databases*, Detroit, Michigan, 204-218.

[MaK87] M.V. Manago and Y. Kodratoff, (1987). Noise and Knowledge Acquision, *Proceedings of the 10th IJCAI Conference* , Milan, Italy, 348-354.

[MAG80] A. Marchand, L. Van, R. Galen, (1980). The Assessment of Laboratory Test in the Diagnosis of Acute Appendicitis, *American Journal of Clinical Pathology*, 80(3), pp369-374

[MCP93] C.J. Matheus, P.K. Chan, and G. Piatetsky-Shapiro, (1993). Systems for Knowledge Discovery in Databases, *IEEE transaction on Knowledge and data Engineering*, Vol 5(6) 903-913

[McD82] J. Mcdermott, (1982). A Rule-based Configurer of Computer Systems, *Artificial Intelligence*, Jan. 1982

[MiC80] R.S. Michalski and R.L. Chilansky, (1980). Learning by Being Told and Learning from Examples: An Experienmental Comparision of the Two Methods of Knowledge Acquisition in the Context of Developing an Expert System for Soybean Disease Diagnosis, *International Journal of Policy Analysis and Information System* , Vol. 4, 125-161.

[Mic83] R.S. Michalski, (1983). A Theory and Methodology of Inductive Learning, *Machine Learning: An Artificial Intelligence Approach,* vol. 1, Morgan Kaufmann, 83-134.

[MiS83] R, Michalski, and R. Stepp, (1983). Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy, *IEEE Transaction on Pattern Analysis and Machine Intelligence, 5(4)*, 396-409.

[MMHL86] R. S. Michalski, L. Mozetic, J. Hong and N. Lavrac, (1986). The Multipurpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains, *Proceedings of 1986 AAAI Conference,* Philadelphia, PA, 1041-1045.

[Mic87] R.S. Michalski, (1987). How to Learn Imprecise Concepts: A Method for Employing a Two-tiered Knowledge Representation in Learning, *Proceedings of the 4th International Workshop on Machine Learning*, Irvine, CA, 50-57.

[Min89] J. Mingers, (1989). An Empirical Comparision of Selection Measures for Decision-Tree Induction, *Machine Learning 3:*, 319-342

[Mit77] T. M. Mitchell, (1977). Version Space: A Candidate Elimination Approach to Rule Learning, *Proceedings of the 5th IJCAI Conference*, Cambridge, MA, 305-310.

[Mit79] T.M. Mitchell, (1979). An Analysis of Generalization as a Search Problem, *Proceedings of the 6th IJCAI Conference*, Tokyo, Japan, 577-582.

[NgB92] O. K. Ngwenyama, N. Bryson, (1992). A Formal Method For Analyzing and Integrating the Rule-Sets of Multiple Experts, *Information Systems,* Vol. 17. No.1 1-16

[Nib87] T. Niblett, (1987). Constructing Decision Tress in Noisy Domains, *Proceeding of the 2nd Europe Woking Session on Learning*, 67-78.

[Out90] J.K. Ousterhout, (1990). TCL: An Embedded Comamnd Language, *Prod. 1990 Winter USENIX Conference*, Washington D.C., 133-146

[Paw82] Zdzislaw Pawlak, (1982). Rough Sets, *International Journal of Information and Computer Science* 11(5), 341-356

[Paw85] Zdzislaw Pawlak, (1985). Rough Sets and Fuzzy Sets, *Fuzzy Sets and Systems,* 17, 99-102

[PWZ88] Z. Pawlak, S.K.M Wong and W. Ziarko, (1988). Rough Set, Probabilistic versus Deterministic Approach, *Internat. J. Man-Machine Stud.*, Vol. 29, 81-97

[Paw91] Z. Pawlak, (1991). *Rough Sets: Theoretical Aspects of Reasoning About Data,* Kluwer Academic Publishers.

[Paw92] Zdzislaw Pawlak, (1992). Anathomy of Conflicts, *ICS Research Report 11/92*, Wawsaw University of Technology, Nowowiejska 15/19, 00-665, Warsaw, Poland

[Pia89] Piatetsky-Shapiro, (1989). Discovery of Strong Rules in Databases, *Proceedings of IJCAI-89 Workshop on Knowledge Discovery in Databases,* Detroit, Michigan, USA, 264-274.

[Qui83] J. R. Quinlan, (1983). Learning Efficient Classification Procedures and Their Appliccation to Chess End-Games, *Machine Learning: An Artificial Intelligence Approach*, Vol. 1, Morgan Kaufmann, 463-482.

[Qui86] J.R. Quilian, (1986). The Effect of Noise on Concept Learning, *Machine Learning: An Artificial Intelligence Approach*, Vol. 2, Morgan Kaufmann, 149-166.

[Qui87] J. R. Quinlan, (1987). Generating Production Rules from Decision Trees, *Proceedings of The 10 IJCAI*, pp304-307

[Rei84] R. Reiter, (1984). Towards a Logical Reconstruction of Relational Database Theory, *On Conceptual Modeling,* Spring-Verlag, M. Brodie, J. Mylopoulos and J. Schmids (Eds). 191-233.

[Ren86] L. Rendell, (1986). A General Framework for Induction and a Study of Selective Induction, *Machine Learning*, Vol. 1, 1986

[Rus88] S. J. Russell, (1988). Tree-Structure Bias, *Proceedings of 1988 AAAI Conference*, Minneapolis, Minnesota, 211-213.

[ScF86] J.C. Schlimmer, D. Fisher, (1986). A Case Study of Incremental Concept Induction, *Proc. of the Fifth National Conference on Machine Learning*, 496-501

[ShH94] Ning Shan. X. Hu, (1994). A Decision Matrix Approach to Construct Multiple Knowledge Bases, *Proc. of the 8th International Conf. on Industrial & Engineering Application of AI & Expert System.* Melbourne, Australia, June 1995. 517-524 (nominated for the best paper award)

[SHZC94] N. Shan, X. Hu, W. Ziarko, N. Cercone, (1994). A Generalized Rough Set Model, *Proc. of the Third Pacific Rim International Confernce on AI,* Beijing, China, pp437-443

[Sch91] J.C. Schlimmer, (1991). Learning Determinations and Checking Databases, *Knowledge Discovery in Database Workshop 1991.*

[Sha48] C.E. Shannon, (1948). A Mathematical Theory of Communication, *Bell System Tech. Journal*, 4(2) 379-423

[ShW64] C.E. Shannon, W. Weaver, (1964). The Mathematical Theory of Communication, Urbana, Illinois: University of Illinois Press

[She91] W.M. Shen, (1991). Discivering Regularities from Knowledge Bases , *Knowledge Discovery in Database Workshop.*

[SSU91] A. Silberschatz, M.Stonebraker and J.D.Ullman, (1991). Database Systems: Achievements and Opportunities, *Comm. ACM, 34(10)*, 94-109.

[SkR91] A. Skowron, C. Rauszer, (1991). The Discernibility Matrices and Functions in Information Systems. *ICS Research Report 1/91*, Wawsaw University of Technology, Nowowiejska 15/19, 00-665, Warsaw, Poland

[Slo92] Slowinski, R (ed.) (1992). *Intelligent Decision Support: Handbook of Applications and Advances of Rough Sets Theory.*

[SoS63] R.R. Sokal and R.H. Sneath, (1963). Principles of Numericcal Taxonomy, *W.H. Freeman*

[Sim84] R. Smith, (1984). On the Development of Commercial Expert Systems, *Artificial Intelligence Magazine*, Fall 1984

[SmG92] P. Smyth and R.M. Goodman, (1992). An Information Approach to Rule Induction from Databases, *IEEE Trans. on Knowledge and Data Engineering,* Vol. 4, 301-316

[Ste87] R.E. Stepp, (1987). Concepts in Conceptual Clustering, *Proceedings of the 10th IJACI Conference*, Milan, Italy, 211-213.

[SuF86] D. Subramanian and J. Feigenbaum, (1986). Factorization in Experiment Generalization, *Proc. 1986 AAAI Conf.*, Philadelphia, PA, 512-522.

[Tor93a] L. Torgo, (1993). Controlled Redundancy in Incremental Rule Learning, *Proc. of European Conf. on Machine Learning*, pp185-195

[Tor93b] L. Torgo, (1993). Rule Combination in Inductive Learning, *Proc. of European Conf. on Machine Learning*, pp384-389

[Utg88] P. Utgodd, (1988). ID5: An Incremental ID3, *Prof. of the Fifth Inter. Conf. on Machine Learning*, 107-120

[Ver75] S.A. Vere, (1975). Induction of Concepts in the Predicate Calculus, *Proceeding of the 4th International Joint Conference on Artificial Intelligence*, Los Altos, 281-287.

[WaE87] L. Watanabe and R. Elio, (1987). Guiding Constructive Induction for Incremental Learning from Examples. *Proceedings of the 10th IJCAI Conference*, Milan, Italy, 293-296.

[WeK89] S.M. Weiss and I. Kapouleas, (1989). An Empirical Comparision of Pattern Recognition Neural Nets, and Machine Learning Classification Methods, *Proc. of the 11th International Joint Conf. on AI*, pp781-787

[Win75] P. Winston, (1975). Learning Structure Descriptions from Examples, *The Psychology of Computer Vision*, Winston, P. (eds), McGraw-Hill, 157-209.

[WiH84] P. Winston and B.K.Horn, (1984). *LISP*, Reading,Mass.: Addison_Wesley.

[WoC87] B. Woolf, P. A. Cunningham, (1987). Multiple Knowledge Sources in Intelligent Teaching Systems. *IEEE Expert* 41-54

[WoC88] A. K. C. Wong and K.C.C. Chan, (1988). Learning from Examples in the Presence of Uncertainty , *Proceedings of International Computer Science Conference' 88*, Hong Kong, December, 369-376.

[WZY86] S.K.M Wong, Wi. Ziarko, R.L. Ye, [1986]. Comparision of Rough Set and statistical Methods in Inductive Learning, *Inter. J. Man-Machine Studies*, 24, 53-72

[Zia91] Wojciech Ziarko, (1991). The Discovery, Analysis, and Representation of Data Dependancies in Databases, in *Knowledge Discovery in Databases* G. Piatetsky-Shapiro and W. J. Frawlwy,(eds) Menlo Park, CA: AAAI/MIT, 213-228

[ZiS93] Wojciech Ziarko, Ning Shan, (1993). A Rough Set-Based Method for Computing All Minimal Deterministic Rules on Attribute-Value Systems, *Technical Report CS-93-02* Dept. of Computer Science, University of Regina, Canada

[Zia93a] Wojciech Ziarko, (1993). Variable Precision Rough Set Model, *Journal of Computer & System Science*, Vol. 46, No. 1, 39-59

[Zia93b] Wojciech Ziarko (1993). *Analysis of Uncertain Information in The Framework of Variable Precision Rough Sets*, Foundations of Computing and Decision Sciences, Vol. 18. No. 3-4, pp. 381-396.

[Zyt87] J. M. Zytkow, (1987). Combining Many Searches in the FAHRENHEIT Discovery System, *Proceedings of the 4th International Workshop on Machine learning*, Irvine, CA, 281-287.

[ZyB91] J. M. Zytkow and J. Baker, (1991). Interactive Mining of Regularities in Databases, *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds), 31-54.