CONCEPTUAL CLUSTERING AND CONCEPT HIERARCHIES IN KNOWLEDGE DISCOVERY

by

Xiaohua Hu

B.Sc., Wuhan University, China, 1985

M.Sc., Institute of Computing Technology, Academia Sinica, 1988

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
December 1992

ISBN 0-612-01128-3

APPROVAL

Name: Xiaohua Hu

Degree: Master of Science

Title of thesis: Conceptual Clustering and Concept Hierarchies in Knowledge Discovery

Examining Committee: Dr. Ramesh Krishnamurti
Chair

Dr. Nick Cercone, Senior Supervisor

Dr. Jiawei Han, Supervisor

Dr. Paul McFetridge, External Examiner

Date Approved: December 14, 1992

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

Conceptual Clustering and Concept Hierarchies in Knowledge Discovery

Author:

(signature)

Xiaohua HU

(name)

Dec 16, 1992

(date)

Abstract

Knowledge discovery is the nontrivial extraction of implicit, previously unknown, and potentially useful information from data. Knowledge discovery from a database is a form of machine learning where the discovered knowledge is represented in a high-level language. The growth in the size and number of existing databases far exceeds human abilities to analyse the data, which creates both a need and an opportunity for extracting knowledge from databases. In this thesis, I propose two algorithms for knowledge discovery in database systems. One algorithm finds knowledge rules associated with concepts in the different levels of the conceptual hierarchy; the algorithm is developed based on earlier attribute-oriented conceptual ascension techniques. The other algorithm combines a conceptual clustering technique and machine learning. It can find three kinds of rules, characteristic rules, inheritance rules, and domain knowledge, even in the absence of a conceptual hierarchy. Our methods are simple and efficient.

Acknowledgements

I would like to express my deepest gratitude to my senior supervisor Dr. Nick Cercone, supervisor Dr. Jiawei Han for their generosity, guidance encouragement, academic help and financial support throughout this research. I am also grateful to Dr. Paul McFetridge for his careful reading of the thesis and his valuable comments. My discussion with fellow student Yue Huang were very helpful in designing the program.

Contents

Chapter 1

Introduction

Learning is one of the most important characteristics of human and machine intelligence. Machine learning is a fundamental area in Artificial Intelligence which has achieved significant progress in the last two decades. Many learning systems have been constructed for scientific, business, industrial and medical appilcations; it is important to develop learning machanisms for knowledge discovery in large databases, especially relational databases.

An important machine learning paradigm, *learning from examples*, that is, learning by generalizing specific facts or observations [5,7], has been adopted in many existing induction learning algorithms. Current systems for *learning from examples* take training examples from various sources, such as, data extracted from experiments [1,24], examples given by teachers and experts [32], facts recognized by people [34] and rules accumulated from past experience[28], etc.. However, not many systems directly extract knowledge from data stored in relational databases.

Knowledge discovery is the nontrivial extraction on implicit, previously unknown, and potentially useful information from data [11].

The growth in the size and number of existing databases far exceeds human abilities to analyze this data, which creates both a need and an opportunity for extracting knowledge from databases. Recently, data mining has been ranked as one of the most promising research topics for the 1990s by both database and machine learning researchers [43].

From our point view, one of the major reasons that the learning systems do not integrate well with relational database systems is because of the inefficiency of current learning algorithms when applying to large databases. Most existing algorithms for *learning from examples* apply a tuple-oriented approach which 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 [7,28]. 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 previous studies [2,13,14], an attribute-oriented induction method has been developed for discovery in relational databases. The method integrates a machine learning paradigm, especially *learning from examples* techniques, with database operations and extracts generated data from actual data in databases. Two types of knowledge rules, characteristic rules and classification rules, can be learned. A key to the approach is attribute-oriented database operations which substantially reduce

the computational complexity of the database learning processes.

In this thesis, I further develop the results from previous studies [2,13,14] in two ways. The previous method is developed further to find knowledge rules associated with different levels of the concepts in the conceptual hierarchy. Furthermore, 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. This method combines our conceptual clustering technique with machine learning. It can find three kinds of rules even in the absence of a conceptual hierarchy. Our methods are simple but efficient.

This thesis is organized into seven chapters. A brief survey of the methods developed for learning from examples and knowledge discovery in large databases is presented in Chapter 2. Attribute_Oriented Induction in relational databases is addressed in Chapter 3. An algorithm for discovering knowledge rules associated with concepts of different levels in the conceptual hierarchy table and statistical rules with different concepts is explained in Chapter 4. Then we propose a new algorithm, knowledge discovery by conceptual clustering, in Chapter 5. In Chapter 6, the variations of our method, the comparision with other discovery systems are discussed. Finally, in Chapter 7, we conclude our research and propose some interesting topics for future research.

Chapter 2

Learning From 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. We review the basic components and the generalization 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 characterized 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 [12].

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 [28]. The training examples represent low-level, specific information. The learning task is to generalize 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) [12]. 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 [28]:

(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 generalization 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 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 *discrimination 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 [7,12]. A *statistical rule* is a rule associated with statistical information which assesses the representativeness of the rule.

Most learning algorithms are designed for learning admissible rules [7,28]. A few algorithms, such as INDUCE 1.2 [6] and SPROUTER [18], are designed for learning characteristic rules. DBLEARN [2,13,14,22] can discover all of the three kinds of rules.

2.1.3 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 [7]. 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 [33,34], the approach adopted in [18,48], the ID3 techniques [37] and the Bacon learning system [20].

In the model-driven methods, an priori model is used to constrain the search. These methods search a set of possible generalization in an attempt to find a few 'best' hypotheses that satisfy certain requirements. Typical examples of systems which adopt this startegy are AM [25], DENDRAL and Meta-DENDRAL [1], and the approach used in the INDUCE system [6].

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 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 [7]. 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 statistic measures of how well a proposed hypothesis accounts for the data.

2.2 Some Learning From Examples Models

Since the 1960's, many algorithms and experimental systems on *learning from examples* have been developed [33], which demonstrate aspects of machine learning in science, industry and business applications [17,39]. 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 [32]. 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 generalize the S set, and the negative examples force the program to specify 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 recomputation.

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 to become overly generalized. Similarly, a false example causes the concept set to become overly specialized. Eventually, noisy training examples can lead to a situation in which there are no concepts that are consisitent 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 represenatation language, the data-driven algorithm described above never generalizes. 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 arbitratrly many examples and arbitrarily many attributes in the hypothesis. 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 size of set S and set G can grow exponentially in the number of examples [16].

To improve computational efficiency, Haussler proposed a one-sided algorithm which is in constrast 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 [16,17].

2.2.2 AQ11 and AQ15 Systems

Michalski and his colleagues have developed a series of AQ learning systems. The AQ11 system [29] 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 descriptor* 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.

The discriminant rules developed by AQ11 correspond to the set of most general concepts consistent with the training examples. In many situations, it is also good to develop the most specific concepts of the class, thus permitting a very explicit handling of the unobserved portion of the space.

After developing the AQ11 system, Michalski et. al proposed another inductive learning system AQ15 in 1986 [31]. Ths 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.

2.3 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 [7]. Unfortunately, a relational database does not explicitly store negative data, 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.

Definition 2.3.1 *A generalized relation is a relation obtained by substituting the specific concept(s) by the general concept(s) in some attribute(s).*

2.3.2 Types of Rules

There are three types of rules, characteristic rules, classification rules and statistical rules which can be easily learned from relational databases.

Definition 2.3.2 *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 summarized as a characteristic rule.

Definition 2.3.3 *A classification rule is an assertion which descriminates the concepts of one class from other classes.*

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

Definition 2.3.4 *A statistical rule is a rule associated with statistical information which assesses the representativeness of the rule.*

Characteristic rules, classification rules and statistical rules are useful in many applications. A characteristic rule provides generalized concepts about a property which can help people recognize 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 statistical rules give the summary information about the data in the databases

The data relavant to the learning task can usually be classified ino 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 2.3.5 *A target class is a class in which the data are tuples in the database consistent with the learning concepts.*

Definition 2.3.6 *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.

In learning a characteristic rule, relevant data are collected into one class, the target class, for generalization. In learning a discirimination 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.

Since learning of these two rules represents two different learning tasks, different sets of examples are required for the learning processes. The characteristic rules only concern the characteristics of the data. Therefore, positive examples alone are enough to furnish the learning task. However, for learning classification rules, the negative examples must be incorporated into the learning process to derive the concepts which have the discrimination property.

2.3.3 Background Knowledge

Concept hierarchies represent 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 [2,33]. Using a concept hierarchy, the rules learned can be represented in terms of generalized concepts and stated in a simple and explicit form, whcih 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 domain experts. But if the concept hierarchies are not available, in some case , it is possible to construct them based on the data in the databases. I will address this problem in Chapter 5.

2.3.4 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.

Exceptional data often occurs in a large relation. It is important to consider exceptional cases when learning in databases. Statistical information helps *learning-from-examples* to handle exceptional and/or noisy data [13,14,22]. A special attribute, *vote*, can be added to each generalized relation to register the number of tuples in the original relation which are generalized to the current tuple in the generalized relation. The attribut *vote* carries database statistics and supports the pruning of scattered data and the generalization of the concepts which take a majority of votes. The final generalized rule will be the rule which represents the characteristic of a *majority* number of facts in the database (called an *approximate rule*) or indicate *quantiative* measurement of each conjunct or disjunct in the rule (called a *quantitative rule*).

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 uaually accompanied by substancial 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 [2,13,14,22].

2.4.1 INLEN System

The INLEN system was developed by Kaufman et. al in 1989 [23]. The system combines the 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 2.1 illustrates the general design of the system.

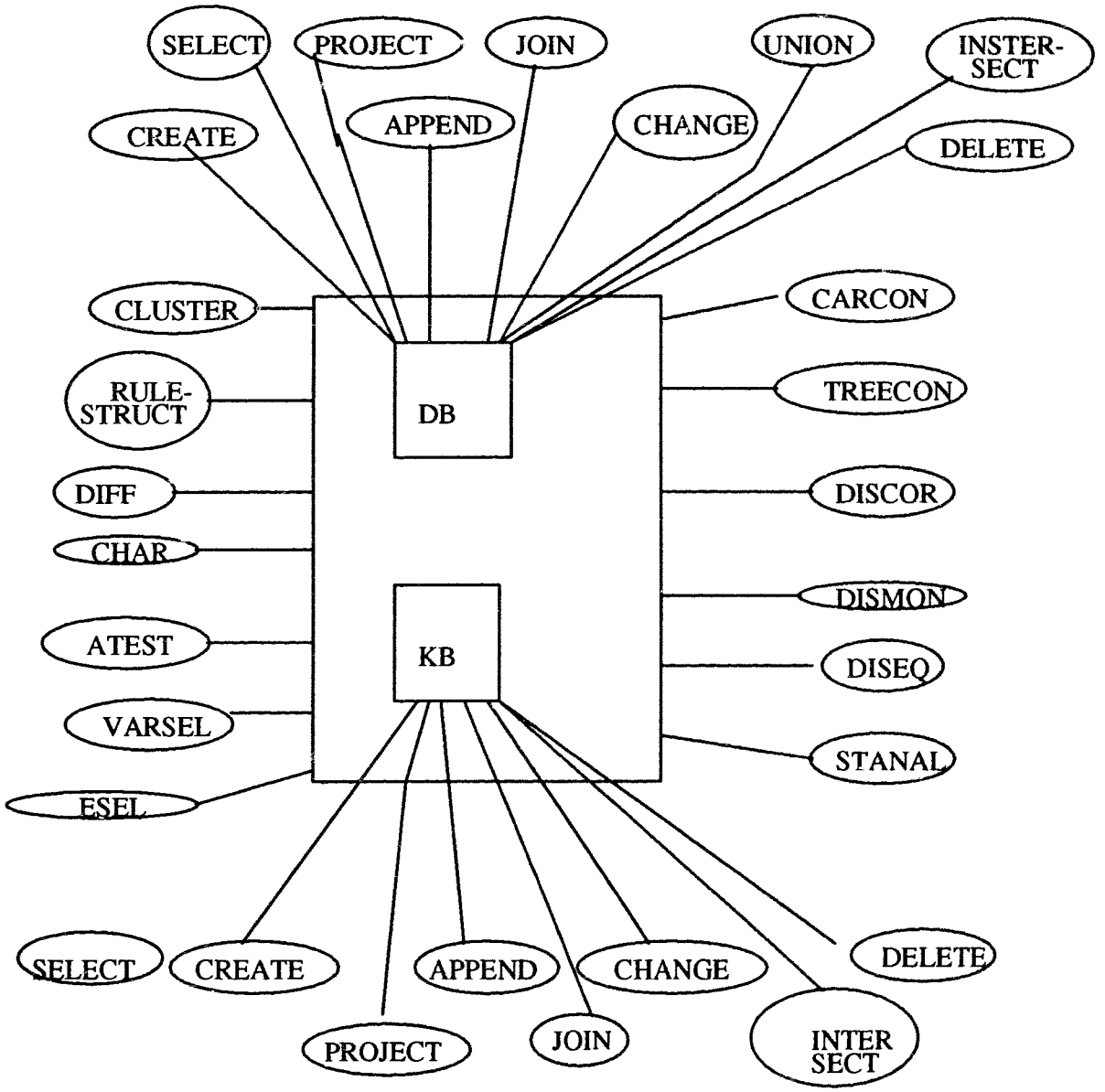

Figure 2.1: The organization of the INLEN System

The general design of INLEN is shown in Figure 2.1. The INLEN system consists of a relational database 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 metaknowledge 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 [29]. The operator DIFF determines the discrimination rules, and can be executed in the AQ program [29]. The operator CHAR discovers the characteristic rules, which is also implemented in an AQ program [29]. 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 [8]. ABACUS-2 is related to programs such as BACON [22] and FAHRENHEIT [47]. Most of these machine learning programs are invoked by EGOS and represent 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 An Algorithm for Discovering Strong Rules in Databases

Another interesting study on learning from relational databases was performed by Piatetsky-Shapiro [35]. He developed an algorithm to discover strong rules in relational databases. Somewhat different from an exact rule, which is a rule that is always correct, a strong rule is one that is always or almost always correct. This algorithm can find interesting rules of the form $(A = a) -> (B = b)$ from relational databases, that is, if the value of attribute A is a, then the value of attribute B is b.

This algorithm only requires one access to each database tuple. It is thus optimal to within a constant factor, since at least one access is needed to each tuple to check whether this tuple disproved any of the previously inferred rules.

The idea is to hash each tuple according to the value of A. When a tuple is hashed to an empty cell, the cell is initialized. Each cell contains the value of A, the *Count* of tuples hashed to that cell and a current cell *Tuple*. When a tuple is hashed to an occupied cell, it is compared with the cell for $(A = a)$ and it contains all the information necessary for deriving rules implied by $(A = a)$, such as, the number of tuples whose value of attribute A are a and the difference among those tuples which are hashed to the same cell.

A significant speed-up is achieved by using a test for early rejection of rules in an attribute. For a nominal attribute, if the value in the newly hashed tuple is different from the value stored in the cell *Tuple*, this attribute can be removed from further consideration. A taxonomic or an internal attribute is rejected when the intermediate result covers more than a user specified threshold value which is the maximum allowed sample coverage.

Piatesky-Shapiro has derived formulas for predicting rule accuracy on the entire database after rules are discovered in a sample. These formulas measure the significance of the correction between two attributes based on some statistical techniques.

This algorithm has been implemented in LISP and tested in relational databases. While most machine learning algorithms suffer from computational inefficiency, this algorithm can discover many strong rules from databases quickly, and can therefore be applied to relatively large databases. However, this algorithm may generate a large set of rules. For example, the author conducted an experiment on 500 tuples, each

having 12 attributes, and the learning algorithm produced 150 rules [35]. This system cannot perform incremental learning when the database is updated. The learning process must be restarted after the new data are insterted into a database because the criteria which determines whether a tuple should be rejected or saved have been changed.

2.4.3 Thought/KD1

Thought/KD1 consists of two components, one is a conceptual clustering system called Leobs [19] and the other is a multipurpose system called GS that uses learning from examples. Leobs is an extension of the well-known conceptual clustering CLUSTER/2. GS is a new multipurpose system of learning from examples which generates a description in DNF of a class of (positive) examples compared with the union of the rest of the classes of (negative) examples. Thought/KD1 first performs conceptual clustering using Leobs to partition the set of given examples into a certain number of subsets and then abstracts descriptions of the generated subsets. Then Thought/KD1 explores some implication relations between descriptions according to the relationships of the corresponding clusters. For rule formation, there are four algorithms of knowledge discovery in Thought/KD1: hierarchical knowledge discovery (HKD), parallel knowledge discovery (PKD), characteristic knowledge discovery (CKD), and inheritance knowledge discovery (IKD). HKD is based on hierarchical clustering and its corresponding abstraction. PKD is based on parallel clustering and its corresponding abstraction. CKD is based on classification and a characteristic description for each cluster. The HKD and CKD subsystems are used to discover domain

knowledge, but the PKD subsystem is used to discover not only the domain knowledge but also knowledge with uncertainty, noise, or exceptions. For HKD, the result is a hierarchy of clustering and corresponding descriptions, each of which comprises a family of clusters of its father clustering. For PKD, k clusterings are independently obtained. The second step is rule formation. That is, for hierarchical discovery, new knowledge is discovered by finding all possible implications between the descriptions in a clustering and those in its father clustering. The same holds for parallel discovery except for implications are found between cluster descriptions and clustering corresponding to large k's. For IKD, the rules are discovered by searching the path from the root of the hierarchy to the current cluster. For CKD, it just examines at the characteristic description for each cluster and finds the equivalent form of different attributes.

Chapter 3

A-O Induction in RDB

In this chapter, we explain and summarize the attribute-oriented method presented in [2,13,14].

3.1 The Basic Attribute_Oriented Induction Algorithm

Efficient induction techniques in relational databases are challenged by the large size of relational databases. Most existing algorithms for learning from examples conduct exhaustive searches of the given concept space, which makes the algorithms infeasibly slow for large database application [2]. Furthermore, although relational databases provide many facilities which have been well implemented, most machine learning algorithms do not take advantage of these facilities. Those learning systems suffer from computational inefficiency when they are used for learning from relational databases.

To make the learning mechanism applicable in relational databases, the learning

algorithm should be able to utilize the database implementation techniques and compute efficiently. The attribute-oriented induction approach can efficiently learn the characteristic rules and classification rules from relational databases [2,13,14]. The approach integrates database operations with the learning process and provides a simple and efficient way of learning from large databases. In contrast to the tuples-oriented approach, the attribute-oriented approach performs generalization attribute by attribute. The training data are examined one attribute at a time. After each attribute has been generated, the sub-concepts are combined to form the entire tree. The approach is demonstrated by two algorithms, the LCHR (for Learning Characteristic Rules) algorithm and the LCLR (for Learning Classification Rules) algorithm.

The general idea of basic attribute-oriented induction is that generalization is performed attribute by attribute using attribute removal and concept tree ascension. As a result, different tuples may be generalized to identical ones, and the final generalized relation may consist of only a small number of distinct tuples, which can be transformed into a simple logical rule. This basic approach can be illustrated by the follwoing algorithm:

Algorithm 3.1: Basic attribute-oriented induction in relational databases.

Input (i) a relational database, (ii) the learning task,(iii)the (optional) prefered concept hierarchies, and (iv) the (optional) preferred form to express learning results (e.g., generalization threshold).

Output

A characteristic rule learned from the database

Method.

1. Collection of the task-relevant data
2. Basic attribute-oriented induction
3. Simplification of the generalized relation, and
4. Transformation of the final relation into a logical rule

Notice that the basic attribute-oriented induction step is performed as follows:

begin { *basic attribute-oriented induction* }

for each attribute A_i ($1 \leq i \leq n$, where n = # of attributes)
in the generalized relation GR **do**
while # of distinct values A_i > threshold **do** {
if no higher level concept in the concept
hierarchy table for A_i
then remove A_i
else substitute for the values of the A_i's by
its corresponding minimal generalized concepts;
merge identical ones, }
while # of tuples in GR > threshold **do** {
selectively generalize attributes;
merge identical tuples }

end

In Step 1, the relevant set of data in the database is collected for induction. The then-part in the first while-loop of Step 2 incorporates attribute removal, and the

else-part utilizes concept tree ascension. The condition for the first while-loop is based on threshold control on each attribute and that for the second one on threshold control on the generalized relation. Attribute removal is used in Step 2 to ensure that the generalization is performed on the minimal decomposable components. Each generalization statement in both while-loops applies the least-commitment principle based on those stategies. Finally, Step 3 and Step 4 apply logic transformations based on the correspondence between relational tuples and logical formulas. Thus the obtained rule should be the desired result which summarizes the characteristics of the target class.

The basic attribute-oriented induction algorithm extracts a characteristic rule from an initial relation. Since the generalized rule covers all of the positive examples in the database, it forms the necessary conditions of the learning concept, that is, the rule is in the form of $learning\_class(x) -> condition(x)$ where $condition(x)$ is a formula containing x. However, since data in other classes are not taken into consideration in the learning process, there could be data in classes which can also meet the specified condition. Thus, $condition(x)$ is necessary but may not be sufficient for x to be in the learning class.

3.2 Learning Other Knowledge Rules by Attribute-Oriented Induction

The attribute-oriented induction method can also be applied to learning other knowledge rules, such as discrimination rules, data regularities, etc.

3.2.1 Learning Discrimination Rules

Since a discrimination rule distinguishes the concepts of the target class from those of contrasting classes, the generalized condition in the target class that overlaps the condition in contrasting classes should be detected and removed from the description of discrimination rules. Therefore, a discrimination rule can be extracted by generalizing the data in both the target class and the conceptual class synchronously and by excluding the properties that overlap in both classes in the final generalized rule.

To implement this notion, the basic attribute-oriented algorithm can be modified corresponding for discovery of discrimination rules. Since different classes may share tuples, the tuples shared by different classes are called *overlapping tuples*. In order to get an effective discrimination rule, care must be taken to handle the overlapping tuples. Usually these tuples should be marked and be excluded from the final discrimination rule, since the overlapping tuples represent the same assertions in both the target and the contrasting class, the concept described by the overlapping tuples cannot be used to distinguish the target class from the contrasting class. By detecting and marking overlapping tuples which have a discriminating property in the rule, which ensures the correctness of the learned discrimination rule.

3.2.2 Learning Data Evolution Regularity

Data evolution regularity reflects the trend of changes in database over time. Discovery of regularuties in an evolving database is important for many applications. To simplify our discussion, we assume that the database schema remains stable in data evolution. A database instance, DB_t, is the database state, i.e., all of the data in the

database, at time t.

Data evolution regularities can be classified into characteristic rules and discrimination rules. The former rules summarize characteristics of the changed data; while the latter distinguish general characteristics of the relevant data in the current database from those in a previous database.

In general, data evolution regularities can be extracted by collecting the learning task-relevant data (usually, the evolving portion) in different database instances and performing attribute-oriented induction on the corresponding task-revelant data set.

3.3 Implementation of the Database Learning Algorithms

To test and experiment on the algorithms, an experimental database learning system, DBLEARN, has been constructed and some interesting experiments have been conducted in the learning system.

DBLEARN is implemented in C and runs under Unix on a Sun workstation. It implements both the 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 knowlegde discovery in databases. The test result of applying DBLEARN to a relatively real large database: the NSERC Grant Information System [22] shows that DBLEARN is very efficient. The architecture of DBLEARN is presented in Figure 3.1:

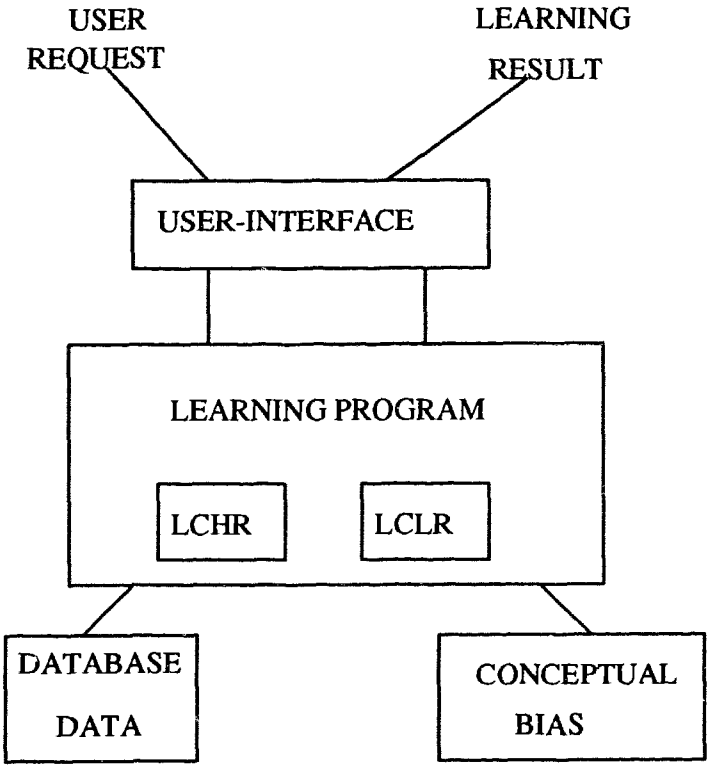

Figure 3.1. The architecture of DBLEARN

Chapter 4

D-K with Hierarchies

In chapter 3, we discussed the attribute-oriented method for discovering knowledge in relational databases. The method integrates a machine learning paradigm, especially *learning from example* techniques, with database operations and extracts generalized data from actual data in the databases. A key to the approach is attribute-oriented concept tree ascension for generalization which applies the well-developed set-oriented database operations and substantially reduces the computational complexity of the database learning process.

In this chapter, 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 containing noisy data and exceptional cases using database statistics.

4.1 An Algorithm for Discovering Three Kinds of Rules

In this section, a new algorithm is presented which is based on the attributed-oriented concept ascension techniques proposed [2,13 14]. In [2,13,14], the key to the approach is an attribute-oriented concept tree ascension technique for generalization which was implemented using well-developed set-oriented database operations, substantially reducing the computational complexity of the database learning task. The general idea of basic attribute-oriented induction is one in which generalization is performed attribute by attribute using attribute removal and concept tree ascension. As a result, different tuples may be generalized to identical ones, and the final generalized relation may consist of only a small number of distinct tuples, which can be transformed into a simple logical rule.

However, there are some drawbacks when using this method. The most obvious one is that the threshold has a great influence on the concept-tree ascension. If the threshold is too large, the algorithm stops at a very low level in the concept hierarchy and the discovered rules are too specific. Alternatively, if the threshold is too small, then the algorithm may not find a suitable concept for the generalized table, and the discovered rules are too general to be useful. Another limitation is that all the discovered rules are only related to some concepts at some level of the concept hierarchy, namely once the threshold is set, then the level of the concept is determined by the threshold value. Because of these limitations, we revise the previous method. Before we explain the details, we first give some definitions.

Definition 4.1.1 *An attribute in a relatively large relation is desirable for consideration for generalization if the number of distinct values it contains does not exceed a user-specified desirability threshold (usually 6 or less).*

Definition 4.1.2 *An attribute is generalizable if there are a large number of distinct values in the relation but there exists a concept hierarchy for the attribute (i.e., there are higher level concepts which subsume these attribute values). Otherwise, it is nongeneralizable.*

Attribute-oriented induction is performed in 3 steps. First, a set of data relevant to the learning task is collected by a database query. Secondly, the collected data is then generalized by (1) removal of nongeneralizable attributes; and (2) 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*, associated to register how many original tuples are generalized to this resultant tuple. The generalized relation obtained at this stage is called the *prime relation.* Thirdly, simplify the generalized relation, and transform the final relation into a logical rule. The core of the attribute-oriented induction is concept -tree ascension on generalizable attributes, which relies heavily on the concept hierarchy information available in the database. A stored concept hierarchy should be appropriately modified based on the statistics of relevant data sets and user preference in order to extract interesting rules.

A *prime relation* R_p for a set of data R stored in the relational table is an intermediate relation generalized from the relation R by removing nongeneralizable 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 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 4.1. Extraction of the prime relation from a set of data R

Input: (i)A set of data R, 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 nongeneralizable **then** remove A_i;
   if A_i is not desirable but generalizable **then** generalize A_i to desirable level;

   /* Generalization is implemented as follows. First, collect the distinct values in the relation and compute the lowest desirable level L on which the number of distinct values will be no more than T_i by synchronously ascending the concept

hierarchy from these values. Then generalize the attribute to this level L by substituting 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. $R_p := R_t$

Theorem 4.1.3 *Algorithm 4.1 correctly extracts the prime relation R_p from a data relation R.*

Proof 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 nongeneralizable, its 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 makes the tuple cover 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 desirable level, the generalized relation is the prime relation. □

For example, given the animal world depicted in Table 4.1 and the concept hierarchy for the attribute 'Animal' depicted in Table 4.2:

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim
1	tiger	yes	pointed	forward	no	claw	meat	yes	no	yes
2	cheetah	yes	pointed	forward	no	claw	meat	yes	no	yes
3	giraffe	yes	blunt	side	no	hoof	grass	yes	no	no
4	zebra	yes	blunt	side	no	hoof	grass	yes	no	no
5	ostrich	no	no	side	yes	claw	grain	no	yes	no
6	penguin	no	no	side	yes	web	fish	no	no	no
7	albotross	no	no	side	yes	claw	grain	no	yes	yes
8	eagle	no	no	forward	yes	claw	meat	no	yes	no
9	viper	no	pointed	forward	no	no	meat	no	no	no

Table 4.1:Animal World

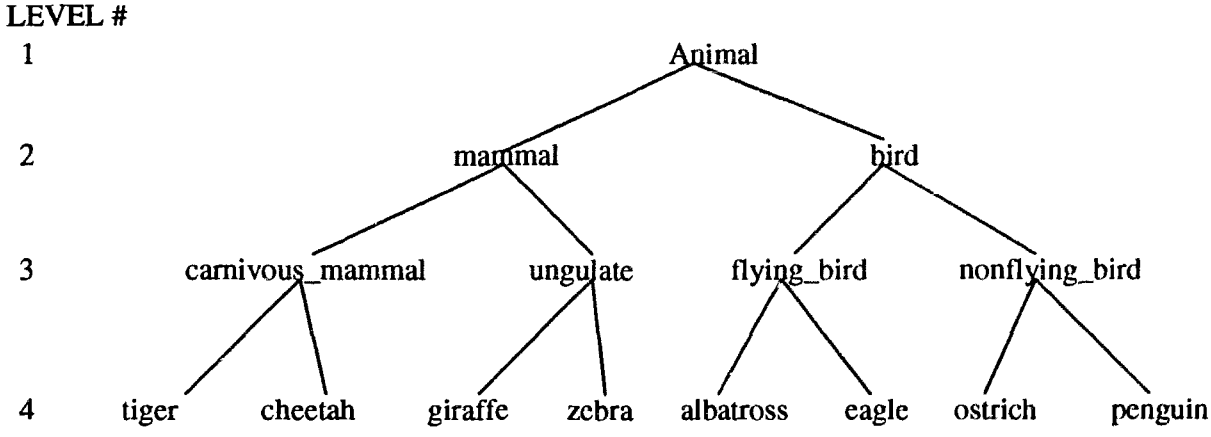

Table 4. 2: Conceptual Hierarchy of the Animal World

in the initial relation, for the attribute 'Animal', there are 9 distinct values, which

is greater than the threshold value for desirable level (suppose the desirability threshold is 6), the concept-tree ascension technique is applied, applying Algorithm 4.1, it is generalized to the desirable level (level 3) { carnivous_mammal, ungulate, flying_bird, nonflying_bird } in Table 4.2, result in a prime relation as shown in Table 4.3:

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	vote
1	cmammal	yes	pointed	forward	no	claw	meat	yes	no	yes	2
2	ungulate	yes	blunt	side	no	hoof	grass	yes	no	yes	2
3	nonfly	no	no	side	yes	claw	grain	no	no	no	1
4	nonfly	no	no	side	yes	web	fish	no	no	yes	1
5	flyingb	no	no	side	yes	claw	grain	no	yes	no	1
6	flyingb	no	no	forward	yes	claw	meat	no	yes	no	1
7	viper	no	pointed	forward	no	no	meat	no	no	no	1

Table 4. 3: Prime relation

The derivation and storage of prime relations for frequently inquired data sets may facilitate the extraction of different kinds of generalized rules from the prime relations. Further generalization can be performed on prime relations to derive characteristic or classification rules [2,13,21] if there are still many tuples in the prime relation. Moreover feature tables can be extracted from the prime relations, and relationships between generalized attribute values can be extracted directly from the feature tables as generalized rules. Based upon different interests, a generalized relation can be directly mapped into different feature tables. In general, we have the following algorithm for the extraction of a feature table from a generalized relation.

Algorithm 4.2. Extraction of the feature table T_A 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_I, (ii) j other attributes $B_1, ..., B_J$, (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 consisits of $m + 1$ rows and $l + 1$ columns, where l is the total number of distinct values in all the attributes. Each slot 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_i in R' **do**
$T_A[r.A, r.B_i] := T_A[r.A, r.B_i] + r.vote;$
$T_A[r.A, vote] := T_A[r.A, vote] + r.vote;$ }

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**
$T_A[p, s] := T_A[p, s] + T_A[t, s];$

Theorem 4.1.4 *Algorithm 4.2 correctly registers the number of occurrences for each general feature in the generalized relation R'.*

Proof 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. Thus we have the theorem. □

In our example, in order to get 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 4.4

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	vote
1	mammal	yes	pointed	forward	no	claw	meat	yes	no	yes	2
2	mammal	yes	blunt	side	no	hoof	grass	yes	no	yes	2
3	bird	no	no	side	yes	claw	grain	no	no	no	1
4	bird	no	no	side	yes	web	fish	no	no	yes	1
5	bird	no	no	side	yes	claw	grain	no	yes	no	1
6	bird	no	no	forward	yes	claw	meat	no	yes	no	1
7	other	no	pointed	forward	no	no	meat	no	no	no	1

Table 4. 4: the generalized relation

Then the feature table is extracted from the generalized relation by using algorithm 2 based on the attribute 'Animal' and the result is shown in Table 4.5. (since we are interested in learning for Animal)

animal Type	Hair		Teeth			...	Feather		..	Milk		vote
	yes	no	pointed	blunt	no	...	yes	no	...	yes	no	
mammal	4	0	2	2	0	...	0	4	...	4	0	4
bird	0	4	0	0	4	...	4	0	...	0	4	4
others	0	1	1	0	0	...	0	1	...	1	0	1
total	4	5	3	2	4	...	4	5	...	5	4	9

Table 4.5 : feature table for the attribute Animal

Different feature tables can be extracted from the prime relation based on the interest of different attributes. The extracted feature table is useful at derivation of the relationships between the classification attribute and other attributes at a high level. For example, the generalized rule *All the animal with hair are mammal* can be extracted from Table 4.5 based on the fact the class *mammal* takes all the with *Hair* count.

Next we present two algorithms for discovering different kinds of rules from database system.

Algorithm 4.3: An attribute-oriented algorithm for discovering different kinds of knowledge rules associated with the concepts for different levels in the concept hierarchy.

Input (i) a set of task-relevant data stored in a relational table (ii) a concept hierarchy

table

Output

A set of classification rules, domain knowledge, and characteristic rules.

Method.

1. using Algorithm 4.1, extract the prime relation from the original relational table, record number of votes.

2. save the prime relation .

3. (further generalize the attributes) based on the concept hierarchy, starting from the desirable level, each time climb one level up until it reach the next highest concept* in the concept hierarchy

4. using Algorithm 4.2, extract a feature table from the generalized relation based on some attribute A: if there are I distinct values for attribute A, then Algorithm 4.2 classifies the data into I classes.

5. assume there are total I classes, namely there are I distinct values for attribute A and J attributes for the data in the feature table. we use K to denote the number of distinct values for the attributes, for different attribute, K is different. According to feature table: for the k^{th} value (k=1,..., K) of the j^{th} attribute (j=1,..,J) in the i^{th} class (i=1,..,I), we associate two probability values: $b_{i,j,k}$ and $c_{i,j,k}$. We use $a_{i,j,k}$ denotes the number of tuples with the k^{th} values for the j^{th} attribute in the i^{th} class.

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

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

$b_{i,j,k}$ 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 particular class.

6. **iteration step**

discover the characteristic rules, classification rules and domain knowledge based on the probability for each distinct value of every attribute in each class in the feature table:

if both $b_{i,j,k} = c_{i,j,k} = 1$, **then** infer a rule:
j^{th} Attribute_name=i^{th} attribute_value $\Longleftrightarrow$ Class=i^{th} Class_name
check the values for the next attribute
(the $(j+1)^{th}$ attribute for the same class i)

if $b_{i,j,k} = 1$ and $c_{i,j,k} < 1$, **then** infer a rule:
j^{th} Attribute_name=i^{th} attribute_value $\longrightarrow$ Class=i^{th} Class_name
check the next value for the same attribute
(the $(k+1)^{th}$ value in the j^{th} attribute)

if $b_{i,j,k} < 1$ and $c_{i,j,k} = 1$, **then**
include j^{th} Attribute_name= i^{th} attribute_value as a component
for the corresponding classification rule forthe i^{th} class
check the next attribute
(the $(j+1)^{th}$ attribute in the same class i)
else

if $b_{i,j,k} \neq 0$ and $c_{i,j,k} \neq 0$ and $b_{i,j,k} * c_{i,j,k} \leq r_{frequency}$

then ignore this value,

check the next value for the same attribute

(the $(k+1)^{th}$ value for the j^{th} attribute)

else include the value as one of the characteristic

values for the attributes

check the next value for the same attribute

(the $(k+1)^{th}$ value for the j^{th} attribute)

iterate step 6 until all the classes are finished

/* since data in a database may be distributed along the full spectrum of the possible values, without using possible quantitative information, it is impossible to obtain a meaningful rule for such kind of data. However, using the quantitative information, various kinds of techniques can be developed to extract meaningful rules. Oue method treats the data which occur rarely in the entire database as exceptional or noise data and filters them using the $r_{frequency}$. $r_{frequency}$ is a small percentage number which is used to filter out those data in the entire database with a very low frequency ratio. */

7. simplify the learned rules: if the distinct data value set of an attribute does cover the entire set of values for the attribute, then remove this attribute and its asociated 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, then the 'not' operator is introduced to the rules to simplify it.

8. (discover the relationship between different attributes based on the feature table) for each class C_i, for any two attributes j_1,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 $a_{i,j_1,k_1} = a_{i,j_2,k_2} = vote$ then infer a rule:
   the j_1^{th} attribute_name=the k_1^{th} value $\Longleftrightarrow$
   the j_2^{th} attribute_name=the k_2^{th} value

   *next highest concept is the concept one level below the most generalized concept 'any'.

4.2 An Algorithm for Discovering Inheritance Rules

Algorithm 4.4: An 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 4.1 (ii) the concept hierarchy tables

Output

A set of inheritance rules

Method.

1. attach one class attribute to the prime table (we call it E-attribute, E means extra)

2. Each time descend one level from the next highest generalized concept according to the concept table until reaching the desirable level of the concept table.

   (a) fill the E-attribute with the generalized concept and the corresponding attribute with the concept one level down of the E-attribute value

   (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 generalized concept from the prime relation

   (d) find the inheritance rules: within the same generalized concept, check those attributes which have different values for different lower level concepts within the same generalized concept.

4.3 Test Example

In this section, we use a data set from [50] to demonstrate Algorithm 4.3 and Algorithm 4.4, step by step.

Given the animal world depicted in Table 4.1 and the concept hierarchy for the attribute 'Animal' depicted in Table 4.2:

First step: applying Algorithm 4.1 to Table 4.1, resulting in the prime relation of Table 4.3. then further generalize Table 4.3 to the generalized relation as shown in Table 4.4.

Second step: extracting the feature table based on the attribute 'Animal' depicted in Table 4.5.

Third step: in the feature table, there are three classes for animal category, mammal, bird and other. For Class=mammal, Hair=yes, $a_{1,1,1,} = 4$, $b_{1,1,1} = c_{1,1,1} = 1$, Class=mammal appears four times, and the total tuples for Class=mammal is four, Hair=yes only appears four times in the whole table, so we can infer a rule as follows:

(Hair=yes) $\longleftrightarrow$ (Class=mammal)

similarly we can get :

(Milk=yes) $\longleftrightarrow$ (Class=mammal)

(Class=mammal) $\longrightarrow$ (Feet=claw or hoof) $\bigwedge$ (Eats=meat or grass)

for Class=bird:

(Feather=yes) $\longleftrightarrow$ (Class=bird)

(Class=bird) $\longrightarrow$ (Feet=claw or web) $\bigwedge$ (Eats=grain or fish or meat)

The fourth step is to simplify the above rules, count the number of values appearing as the characteristic values for the attribute and compare 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. For example, the attribute 'Eats' has four distinct values: meat, grass, grain, and fish. In the dicovered rule:

(Class=bird) $\longrightarrow$ (Feet=claw or web) $\bigwedge$ (Eats=grain or fish or meat),

the Eats takes grain, fish and meat. So we can use not(Eats=grass) instead of (Eats=grain or fish or meat) as a component for the classification rule and the discovered rule can be simplified as:

(Class=bird) $\longrightarrow$ not(Feet=hoof) $\bigwedge$ not (Eats=grass)

similarly, the rule:

(Class=mammal) $\longrightarrow$ (Feet=claw or hoof) $\bigwedge$ (Eats=meat or grass)

can be simplified as

(Class=mammal) $\longrightarrow$ not(Feet=web) $\bigwedge$ (Eats=meat or grass)

The last step is to analyze the data between different attributes and find the relationship between them: for example, for Hair=yes, Feather=no,

(Hair=yes) $\Longleftrightarrow$ (Feather=No)

(Hair=yes) $\Longleftrightarrow$ (Milk=yes)

:

:

(Feathers=yes) $\Longleftrightarrow$ (Milk=No)

Then we can continue the process by using Algorithm 4. The prime relation table is illustrated in Table 4.3:

Attach the E_attribute to the table, use the next higher-level concept in the concept hierarchy for substitution, resulting in the temporary relation in Table 4.6:

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	E
1	cmammal	yes	pointed	forward	no	claw	meat	yes	no	yes	mammal
2	ungulate	yes	blunt	side	no	hoof	grass	yes	no	yes	mammal
3	nonflyb	no	no	side	yes	claw	grain	no	no	no	bird
4	nonflyb	no	no	side	yes	web	fish	no	no	yes	bird
5	flyingb	no	no	side	yes	claw	grain	no	yes	no	bird
6	flyingb	no	no	forward	yes	claw	meat	no	yes	no	bird
7	viper	no	pointed	forward	no	no	meat	no	no	no	other

Table 4. 6: temporary relation after substitution

note: cmammal=carnivorous mammal, nonflyb=non-flying bird, flyingb=flying bird

From Table 4.7, we can see that for mammals, Hair, Feather, Milk, Fly, and Swim do not distinguish mammals; Teeth, Eat do distinguish mammals, we can thus generalize the rules:

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	E
1	cmammal	yes	pointed	forward	no	claw	meat	yes	no	yes	mammal
2	ungulate	yes	blunt	side	no	hoof	grass	yes	no	yes	mammal

Table 4. 7: temporary relation for mammal

(Class=mammal) $\wedge$ (Teeth=pointed) $\longrightarrow$ (Animal=carnivorous_mammal)

(Class=mammal) $\wedge$ (Teeth=blunt) $\longrightarrow$ (Animal=ungulate)

:

:

(Class=mammal) $\wedge$ (Eats=meat) $\longrightarrow$ (Animal=cmammal)

(Class=mammal) $\wedge$ (Eats=grass) $\longrightarrow$ (Animal=ungulate)

#	Animal	Hair	Teeth	Eyes	Feather	Feet	Eat	Milk	Fly	Swim	E
3	nonflyb	no	no	side	yes	claw	grain	no	no	no	bird
4	nonflyb	no	no	side	yes	web	fish	no	no	yes	bird
5	flyingb	no	no	side	yes	claw	grain	no	yes	no	bird
6	flyingb	no	no	forward	yes	claw	meat	no	yes	no	bird

Table 4. 8: temporary relation for bird

Similarly for bird, based on Table 4.8, we can derive the following rules:

(Class=bird) $\wedge$ (Flies=yes) $\longrightarrow$ (Animal=flying_bird)

(Class=bird) $\wedge$ (Flies=no) $\longrightarrow$ (Animal=nonflyingb)

then continue the process, descending one level of the concept hierarchy

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	E
1	tiger	yes	pointed	forward	no	claw	meat	yes	no	yes	cmammal
2	cheetah	yes	pointed	forward	no	claw	meat	yes	no	yes	cmammal

Table 4. 9: temporary relation for cmammal

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	E
1	giraffe	yes	blunt	side	no	hoof	grass	yes	no	no	ungulate
2	zebra	yes	blunt	side	no	hoof	grass	yes	no	no	ungulate

Table 4.10:temporary relation for ungulate

Nothing interest can be found based on Table 4.9, Table 4.10. Because the information stored in the database is not enough to distinguish the animal: tiger and cheetah, giraffe and zebra.

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	E
5	ostrich	no	no	side	yes	claw	grain	no	no	no	nonflyb
6	penguin	no	no	side	yes	web	fish	no	no	yes	nonflyb

Table 4. 11: temporary relation for non-flying-bird

(Class=nonflying_bird) $\wedge$ (Feet=claw) $\longrightarrow$ (Animal=ostrich)

(Class=nonflying_bird) $\wedge$ (Eat=grain) $\longrightarrow$ (Animal=ostrich)

:

:

(Class=nonflying_bird) $\wedge$ (Feet=web) $\longrightarrow$ (Animal=penguin)

(Class=nonflying_bird) $\wedge$ (Swim=yes) $\longrightarrow$ (Animal=penguin)

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim	E
7	albatross	no	no	side	yes	claw	grain	no	yes	no	flyingb
8	eagle	no	no	forward	yes	claw	meat	no	yes	no	flyingb

Table 4.12 : temporary relation for flying-bird

(Class=flying_bird) $\wedge$ (Eye=side) $\longrightarrow$ (Animal=albatross)

(Class=flying_bird) $\wedge$ (Eats=grain) $\longrightarrow$ (Animal=aibatross)

(Class=flying_bird) $\wedge$(Eats=forward) $\longrightarrow$ (Animal=eagle)

(Class=flying_bird) $\wedge$ (Eats=meat) $\longrightarrow$ (Animal=eagle)

Chapter 5

K-Discovery by Clustering

5.1 Introduction

In the previous chapter, 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 database operations and extracts generalized data from actual data in the databases. A key to the approach is attribute-oriented concept tree ascension for generalization which applies the well-developed set-oriented database operations and substantially reduces the computational complexity of the databases learning process.

Since it is often necessary to incorporate higher level concepts in the learning process [33], candidate rules are restricted to formula with 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. The specification of conceptual bias is a

necessary and natural process for learning. Such a concept tree is specified using an IS-A hierarchy and stored in a relational table, the conceptual hierarchy table. In our previous method it is assumed that the concept hierarchy table is provided by the user or data analyst explicitly.

Since databases potentially store a large amount of data, it is important to develop efficient methods to explore regularities from them. Although data in a relational database are usually well-formatted and modeled by semantic and data models [2], 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 human tends 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.

In some applications, the conceptual knowledge is not available, So it is very useful if we can find some regularity from the database in the absence of a concept hierarchy table.

In this chapter, based on our previous research, 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.

5.2 Approaches to Conceptual Clustering

Conceptual clustering is a process which groups objects with common properties into clusters and extracts the characteristic of each cluster over a set of data objects. It is originally motivated and defined by Michalski and Stepp [30] as an extension of processes of numerical taxonomy. Currently there are two views regarding conceptual clustering. One view represents an extension to techniques of numerical taxonomy, the other view is a form of *learning by observations* or *conceptual formation* as distinct from methods of *learning from examples* or *concept identification.* Clustering algorithms which have been framed as extensions to numerical taxonomy techniques include CLUSTER/2 [30] and COBWEB [9]. The clustering algorithm can be viewed as an extension of *learning by observation*, which includes HUATAO [3] and Thought/KD1 [19]. Numerical taxonomy techniques are mainly used to form classification schema over data sets based on some numerical measure of similarity and they do not produce any conceptual description of the clusters. The problem of interpretation is simply left to the data analyst. Conceptual clustering as an extension of *learning by observation* not only considers the distance between objects as in numerical taxonomy, but also their relationships to other objects, and most importantly, their relationship to some predetermined concepts. The price for using such a 'concept-dependent' similarity measure results in significantly greater computational complexity, so this method is not feasible to knowledge discovery in database system since databases usually store a huge amount of data. Furthermore, these two techniques do not find any relationships between different clusters. Both the conceptual clustering and learning from examples methods are concerned with formulating some description that summarizes a set of data. In learning from examples, a tutor specifies which objects

should be assigned to which classes and the learner must characterize each class. In conceptual clustering the learner has the two-fold task of creating object classes as well as characterizing these classes. Among the many existing clustering algorithms such as CLUSTER/2 and COBWEB, none of them have been applied specifically to database applications and they tend to be computationally expensive.

Based on the relationships among the clusters and cluster descriptions, three different types of intercluster structures are commonly distinguished in the literature.

Optimization techniques of numerical taxonomy form a 'flat' (ie, unstructured) set of mutually exclusive clusters (ie. a partition over the input object set). Optimization techniques make an explicit search for a globally optimal K-partition of an object set, where K is a user supplied parameter. This search for globally optimal partitions make optimization techniques computationally expensive, thus constraining their use to small data sets and/or small values of K.

Hierarchical techniques form classification trees over object sets, where leaves of a tree are individual objects, and internal nodes represent object clusters. A 'flat' clustering of mutually-exclusive clusters may be obtained from the classification tree by severing the tree at some level. Hierarchical techniques are further divided into *divisive* and *agglomerative* techniques, which construct the classification tree top-down and bottom-up, respectively. Hierarchical techniques depends on 'good' clusterings arising from a serial of 'local' decisions. In the case of divisive techniques, a node in a partially constructed tree is divided independent of other (non-ancestrial) nodes of the tree. The use of 'local' decision-making in hierarchical methods make them computationally less expensive than optimization techniques with an associated probable reduction in the quality of constructed clusterings.

Clumping techniques return clusterings where constitutent clusters possible overlap. The possibility of cluster overlap stems from independently treating some number of clusters as possible hosts for an object which must be incorporated into a clustering.

A natural approach would combine the advantages of these techniques. We propose a new method for knowledge discovery that can find knowledge from databases by first clustering data using a numerical taxonomy, then extract a characteristic feature for the cluster, and finally treat each cluster as a positive example as in *learning from examples* and use existing machine learning methods to derive knowledge rules. Thus, there are three tasks which must be addressed by our algorithm:

(1) aggregating objects into different clusters;

(2) assigning conceptual descriptions to object classes; and

(3) learning from object classes.

In this chapter, we only consider problems (1) and (3) since problem (2) is identical to the well-studied task of learning from examples [2,4].

5.3 Knowledge Discovery by Conceptual 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 simply on different relationships between different classes.

5.3.1 Aggregating Objects into Different Clusters

For a numerical taxonomy, various measures of similarity have been proposed, most of them based on a Euclidean measure of distance between numerical attribute, consequently, the algorithm only works well on numerical data. More recently, database applications use non-numerical data. We propose a new measure; we use the number of common attribute values in two data set as a reasonable similarity measure between them. (the number of common attribute values of a data set with itself is defined as 0.)

Algorithm 5.1: a simple data clustering algorithm

Input. a set of data stored in the relational table

Output. a cluster hierarchy of the data set

Method.

1. (prelimemary step): generalize some attributes to a 'desirable level', for example, in a employer database, for the 'age' attribute, it is better to substitute the many different values into a few distinct values such as 'young', 'middle-aged', or 'old'. This may make the descriptions of the clusters concise and meaningful.

2. calculate the number of common attribute values between each pair of data.

3. based on the threshold value for the similarity, form a cluster for each data. (the threshold is changed in each iteration, it can be given by the user or determined by analyzing the distribution of the numbers of common attribute values)

4. delete the redundant clusters.

5. is there any new cluster produced? If not, terminate; otherwise

6. form the hierarchy based on the new and untouched* clusters

7. use the new cluster and the untouched clusters as data for the next iteration, go back to step 2

*Note: A cluster which is not an component of any newly formed clusters is called an untouched cluster.

5.3.2 Learning from object classes

We can discover three kinds of knowledge rules from object classes: hierarchical knowledge rules, the relationship between different attributes and inheritance knowledge rules. Given a set of data, after phase 1, the data is clustered into a hierarchy as illustrated in Figure 5.1

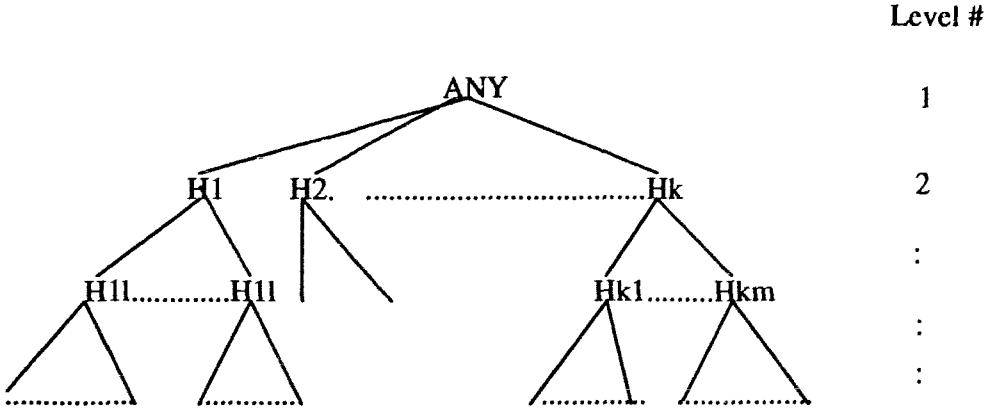

Figure 5. 1: Conceptual Hierarchy

where H's denote the clusters in the hierarchy, $H_{i,j}$ is a subset of H_i and the conceptual descriptions assigned to these classes are D_1,..., D_k, $D_{1,1}$, $D_{1,l}$,..,$D_{k,1}$,..,$D_{k,m}$,...,

and so on. The values of k, l,..,m depend on the actual data set.

For rule formation, there are three algorithms of knowledge discovery: Hierarchical Knowledge Discovery (HKD), Attribute Knowledge Discovery (AKD) and Inheritance Knowledge Discovery (IKD). For HKD, new rules are discovered by finding all possible implications between the descriptions of clusters in a cluster and those in its father clustering, namely $D_{i,j} \longrightarrow D_i$. For AKD, the algorithm just looks at 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 the users in terms of domain knowledge or labels are produced automatically by the system.

Clustering labels play an important role in knowledge discovery, the new rules discovered can be formed as

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

or

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

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 clustering.

5.4 An Example

In this section, we use a set of data from [50] to explain our method, step by step.

Given the animal world depicted in Table 5.1:

#	Animal	Hair	Teeth	Eyes	Feathers	Feet	Eat	Milk	Fly	Swim
1	tiger	yes	pointed	forward	no	claw	meat	yes	no	yes
2	cheetah	yes	pointed	forward	no	claw	meat	yes	no	yes
3	giraffe	yes	blunt	side	no	hoof	grass	yes	no	no
4	zebra	yes	blunt	side	no	hoof	grass	yes	no	no
5	ostrich	no	no	side	yes	claw	grain	no	yes	no
6	penguin	no	no	side	yes	web	fish	no	no	no
7	albotross	no	no	side	yes	claw	grain	no	yes	yes
8	eagle	no	no	forward	yes	claw	meat	no	yes	no

Table 5.1 Animal World

for example, 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 can not fly but can swim.

In Phase 1, the clustering algorithm is applied to classify the raw data. After the first iteration, the number of common attribute values between each pair of data is computed in Table 5.2:

	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	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 5. 2: number of common attribute values after 1st iteration

For example, the '9' in row 1, column 2 is computed by counting the number of common attribues between the data set in row 1 and row 2 of Table 5.1.

Suppose we choose 6 as the threshold value for similarity, the algorithm produces 8 clusters (1,2), (2,1), (3,4), (4,3), (5,6,7,8) ,(6,5), (7,5,8),(8,5,7), then 5 distinct clusters (1,2),(3,4),(5,6,7,8),(5,6),(5,7,8) are formed after deleting redundant ones and a hierarchy is formed as depicted in Figure 5.2:

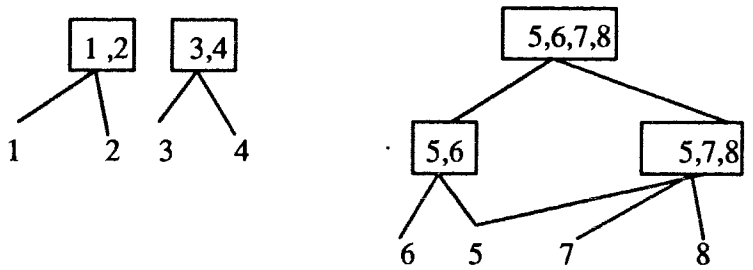

Figure 5. 2: Conceptual Hierarchy after 1st iteration

Next, the clustering algorithm is applied to (1,2), (3,4), (5,6,7,8), it calculate the similarity for the three clusters (1,2), (3,4), (5,6,7,8), the common attribute values are presented in Table 5.3 and the algorithm chooses 5 as the threshold value for this iteration, resulting in the hierarchy shown in Figure 5.3:

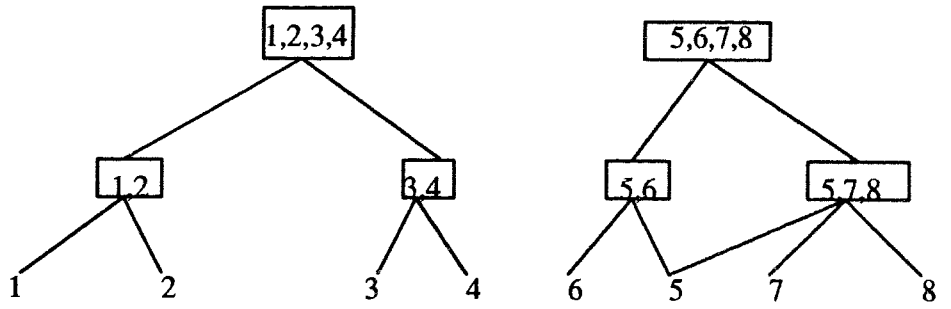

Figure 5. 3: Conceptual hierarchy after 2nd iteration

	(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

Table 5. 3: # of common attribute value after 2nd iteration

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

Table 5. 4: # of common attribute value after 3rd iteration

Finally, the clustering algorithm 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 Table 5.4 and the resultant conceptual hierarchy is illustrated in Figure 5.4. (characteristic descriptions of each cluster is the commom values for all the data in the cluster)

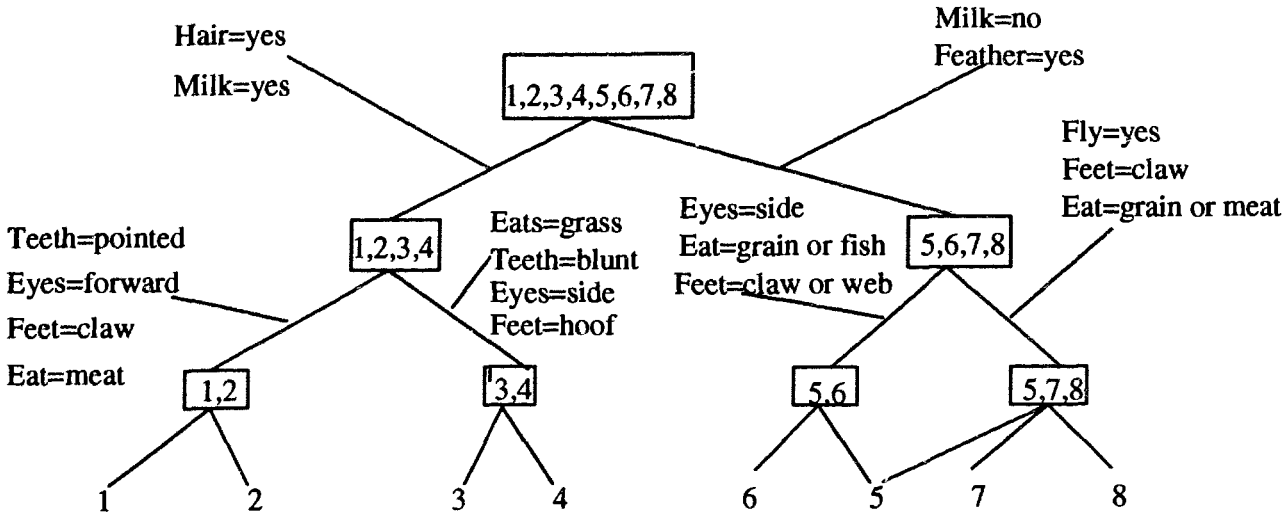

Figure 5. 4: Conceptual hierarchy after the 3rd iteration

In phase 3, the three Knowledge Discovery Algorithms HKD, AKD, and IKD as presented in section 3.2 are applied to the hierarchy depicted in Figure 5.4, respectively. Algorithm HKD results in Table 5.5, a rule set which can be interpreted as follows. Rule # 1 means *if a animal has hoof feet, then it gives milk.* The rule sets are generated as follows:

#	Knowledge Rules discovered by HKD
1	Feet=hoof-->Milk=yes
2	Teeth=pointed or blunt-->Milk=yes
3	Eat=grasss-->Milk=yes
4	Feet=hoof-->Hair=yes
5	Teeth=pointed or blunt-->Hair=yes
6	Eat=grass-->Hair=yes

Table 5. 5: Hierarchical Knowledge Rules

Algorithm AKD results in Table 5.6, a rule set which can be interpreted as follows.

#	Knowledge rules discovered b
1	Hair=yes<==>Milk=yes
2	Feather=yes<==>Milk=no

Table 5. 6: Equivalent logic

Rule # 1 means *if a animal has hair, then it gives milk, vice versa*

Algorithm IKD results in Table 5.7,a rule set which can be interpreted as follows.

Rule # 1 means *if a animal in Cluster Label(1,2,3,4,5,6,7,8) with hair or giving milk, then it belongs to cluster Label(1,2,3,4).*

#	Knowledge rules discovered by IKD
1	Label(1,2,3,4,5,6,7,8)and (hair=yes or Milk=yes)-->Label(1,2,3,4)
2	Label(1,2,3,4,5,6,7,8) and (Feather=yes or Milk=no)--> Label(5,6,7,8)
3	Label(1,2,3,4) and (Teeth=pointed or Eyes=forward or Feet=claw or Eats=meat)-->Label(1,2)
4	Label(1,2,3,4) and (Teeth=blunt or Eyes=side or Fee=hoof or Eats=grass)--> Label(3,4)

Table 5. 7: Inheritance Knowledge Rules

If we substitute the labels by the names given by an expert or users as shown in Table 5.8:

Labels given by system	Names given by expert or 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 mamal
Label(3,4)	ungulate
Label(5,6)	non-flying bird
Label(5,7,8)	meaningless cluster

Table 5. 8: Name list

we can obtain a set of meaningful rules as follows:

#	Substitute the labels by the names given by experts or users
1	(Thing=animal)and(Hair=yes or Milk=yes)->mammal
2	(Thing=animal)and(Feather=yes or Milk=no)->bird
3	(Animal=mammal)and(Teeth=pointed or Eyes=forward or Feet=claw or Eats=meat)->c_mammal
4	(Animal=mammal)and (Teeth=blunt or Eyes=side or Feet=hoof or Eats=grass)->ungulate

Table 5. 9: A set of meaningful rules after substitution
note: c_mammal=carnivorous mammal

Chapter 6

Discussion

In previous chapters, we presented two new methods for knowledge discovery in databases system. One new method finds knowledge associated with concepts at different levels in the conceptual hierarchy. The other method discovers knowlegde by conceptual clustering in the absence of a concept hierarchical table. In this chapter, we will discuss the two methods respectively.

6.1 Discovery of Knowledge Associated with a Concept Hierarchy

The algorithm is based on the previous research [2,13,14,20]. Thus our method inherits all the advantages of the attribute-oriented concept-tree ascension techniques. Moreover, our method can discover different kinds of rules associated with the concepts at different levels in the concept hirarchy

6.1.1 Search Space

Attribute-oriented induction provides a simple and efficient way to learn different kinds of knowledge rules in relational databases. As an emerging field, there have been only a few database-oriented knowledge discovery systems reported, most of which are based on previously developed learning algorithms. The major difference of our approach from the others is attribute-oriented induction vs. tuple-oriented induction. It is essential to compare these two approaches.

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 [7]

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 individual attributes, 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.

6.1.2 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 [2]. 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 attribute), 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 enviroment [2,13,14], our approach can use database facilities to improve the performance. Moreover, in contrast to many machine learning algorithms which can learn only qualitative rules, our approach can learn qualitative rules with quantative information, and statistical rules.

6.1.3 Conjunctive Rules, Disjunctive Rules and Incremental Learning

Many machine learning algorithms, such as Winston's algorithm for learning concepts about the blocks world [49] and the *candidate elimination* algorithm [7] are concerned with the discovery of a conjunctive rule when both positive examples and negative examples are presented. *Thoth* [47] and *SPROUTER* [18] both of which are designed

for finding the maximally-specific conjunctive generalizations of input positive examples, can only learn conjunctive rules [2]. The goal of the learning process performed by such algorithms is to induce a conjunctive rule which can be satisfied by all of the training examples. Since disjunctives allow a partially ordered rule space to become infinitely 'branchy', many algorithms do not permit disjunctives in the representation language. However in real world applications, there are many knowledge rules which should be expressed in a disjunctive norm form. Our algorithms can learn both conjunctive and disjunctive rules under the control of a specified threshold value. If the threshold value is set to 1, the learning result will be a conjunctive rule. Otherwise, if the threshold is a small integer greater than 1, the learning result will be a disjunctive rule consisting of a small number of conjuncts. When a new tuple is inserted into a database relation, rather than restarting the learning process from the beginning, it is preferable to amend and fortify what was learned from the previous data. Our algorithm can be extended to facilitate such *incremental learning* [2]. Let the generalized relation be stored in the database. When a new tuple is inserted into a database, the concepts of the new tuple are first generalized to the level of the concepts in the generalized relation. Then the generalized tuple can be naturally merged into the generalized relation.

6.1.4 Dealing with Different Kinds of Concept Hierarchies

In our examples, all of the concept hierarchies are reprsented 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 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.

Example 6.1 Handling an unbalanced concept tree

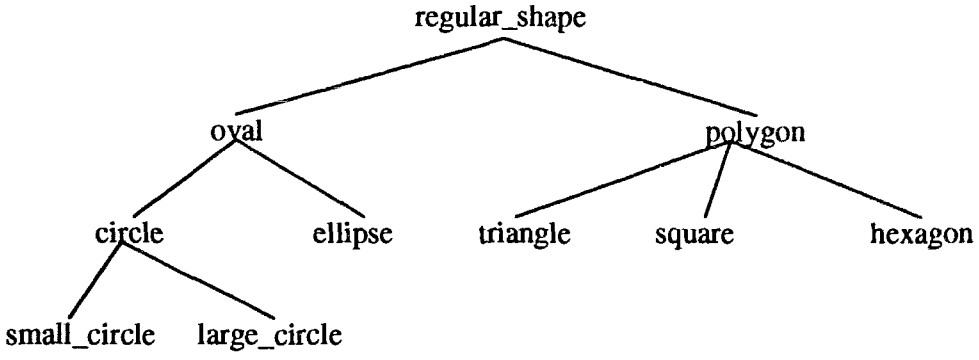

Figure 6.1. An unbalanced concept tree.

Figure 6.1 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 [31]. Our algorithms can 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.

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 have lattice. Figure 6.2 illustrates this case.

Example 6.2. Handling a concept with lattices.

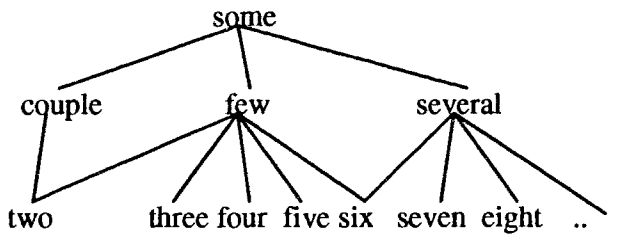

Figure 6.2: A concept hierarchy with lattices.

As illustrated in Figure 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 intermidiate 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.

6.2 Discovery of Knowledge by Conceptual Clustering

Most conceptual classification algorithms in the literature [21,30] are tuple-oriented algorithms. A tuple-oriented algorithm examines data in the database tuple by tuple

and performs generalization and classification based on the comparision 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 [21,30]. But our method use a new method to classify the data set based on the commom 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 choosen which is a similarity measure for classification.

Our method can discover three kinds of knowledge rules in the absence of a conceptual hierarchy. It first classifies the data into different clusters by using a attribute-oriented technique to cluster data, extracts the characteristic of each cluster, then discovers knowledge rules based on the relationship between different clusters.

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 a object has the same number of common attribute value to two clusters, then the

object is assigned to both clusters.

(3) All attributes are potentially significant.

Typically, objects to be clustered come from an experimenta; study of some phenomenon and are described by a specific set of attributes (variables) selected by the data analyst. The attributes selected by the data analyst are not always relevant to the clustering problem. In conventional approaches, the selection of relevant attributes is treated as a separate preliminary step. In the conjunctive conceptual clustering method, attribute selection is performed simultaneously with the formation of clusters. The method selects those attributes from the viewpoint of assumed criteria, allow it to 'simply' characterize the individual clusters in terms of available concepts. But classification can be based on any or all attributes simultaneously, not on just the most important one . This represents an advantage of our method over human classification and many existing conceptual clustering algorithms. In many applications, classes are distinguished not by one or even by several attributes, but by small differences in many attributes. Humans often have difficulty taking more than a few attributes into account and tend to focus on a few important attributes. Our method uses all attributes, permitting uniform consideration of all of the data.

(4) 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.

Chapter 7

Conclusion and Future Research

7.1 Conclusions

In this thesis, we have studied the methods of learning different kinds of rules such as characteristic rules, classification rules, hierarchy knowledge rules, inheritance knowledge rules and attribute rules. Our algorithms adopt the attribute-oriented induction approach, integrate database operations with the learning processes, and provide an efficient way of extracting knowledge from databases.

Our first method finds knowldge rules associated with concepts at different levels in the conceptual hierarchy. It adopts the attribute-oriented concept tree ascending technique which substitutes the lower-level concepts of the attribute in a tuple by its corresponding higher-level concepts and thus generalizes the relation. By eliminating the redundant tuples and applying a threshold value to control the generalization process, the final generalized relation consists of only a small number of tuples which can be transformed into a simple logic formula.

The other method we proposed is designed for discovery of knowledge in the absence of a conceptual hierarchy.

A comparision of our approach with many other algorithms for learning from examples shows that our algorithms have many distinct features, such as, the ability to use database facilities, learn disjunctive rules and statistical rules.

7.2 Future research

There are many interesting research issues related to learning from large databases

7.2.1 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

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 databses can only answer queries which involve the concepts in the database, but they cannot handle queries like 'What are the major characteristic

of mammal?' and 'How can we describe the major differences between mammal and bird?'. 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 in a natural way.

(3) Semantic query optimization using the learned rules.

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. Clearly, 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.

7.2.2 Construction of an Interactive Learning System

As shown 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.

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

(2) the entire learning process should be monitored and controlled by users. For

example, at some stage of the learning process, users may terminate th· generalization on some selected sttributes but continue the process on other attributes. In order to obtain multiple rules, users may influence the learning process using different threshold values.

Bibliography

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

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

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

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

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

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

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

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

[9] D. Fisher, Improving Inference Through Conceptual Clustering, *Proc. 1987 AAAI Conf.*, Seattle, Washington, July 1987, 231-239.

[10] D. Fisher, A Computational Account of Basic Level and Typicality Effects, *Proceedings of 1987 AAAI Conference*, Seattle, Washington, July 1987, 461-465.

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

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

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

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

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

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

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

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

[19] J. Hong, C. Mao, 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.

[20] X. Hu, N. Cercone, J. Han, Discovery of Konwledge Associated With Conceptual Hierarchies in Databases, submitted to Data and Knowledge Engineering.

[21] X. Hu, N. Cercone, J. Han, Object Aggregration and Cluster Identification: A Knowledge Discovery Approach, submitted to Applied Math. Letter.

[22] Y. Huang, J. Han, N. Cercone, G. Hall, Learning Statistical Rules from Relational Databases, to appear in AI: statistical, 1993

[23] K.A. Kaufman, R.S. Michalski and L. Kerschberg, 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.

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

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

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

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

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

[29] R.S. Michalski and R.L. Chilansky, 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, 1980, 125-161.

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

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

[32] R.S. Michalski, 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, 1987, 50-57.

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

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

[35] Piatetsky-Shapiro, Discovery of Strong Rules in Databases, *Proceedings of IJCAI-89 workshop on Knowledge Discovery in Databases,* Detroit, Michigan, USA, August 1989, 264-274.

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

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

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

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

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

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

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

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

[44] R.R. Sokal and R.H. Sneath Principles of Numericcal Taxonomy *W.H. Freeman 1963.*

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

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

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

[48] L. Watanabe and R. Elio, Guiding Constructive Induction for Incremental Learning from Examples. *Proceedings of the 10th IJCAI Conference*, Milan, Italy, August 1987, 293-296.

[49] P. Winston, Learning Structure Descriptions from Examples, *The Psychology of Computer Vision*, Winston, P. (eds), McGraw-Hill, 1975, 157-209.

[50] P. Winston and B.K.Horn *LISP*, Reading,Mass.: Addison_Wesley, 1984.

[51] A. K. C. Wong and K.C.C. Chan, Learning from Examples in the Presence of Uncertainty , *Proceedings of International Computer Science Conference' 88*, Hong Kong, December, 1988, 369-376.

[52] J. M. Zytkow, Combining Many Searches in the FAHRENHEIT Discovery System, *Proceedings of the 4th International Workshop on Machine learning*, Irvine, CA, 1987, 281-287.

[53] J. M. Zytkow and J. Baker, Interactive Mining of Regularities in Databases, *Knowledge Discovery in Database, AAAI/MIT Press*, G.Piatetsky-Shapiro and W.J. Frawley (eds), 31-54, 1991.