The Analysis of Relationships in Databases for Rule Derivation

Show-Jane Yen and **Arbee L.P. Chen***

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 300, R.O.C.
Email : alpchen@cs.nthu.edu.tw

Abstract

Owing to the rapid growth in the sizes of databases, potentially useful information may be embeded in a large amount of data. Knowledge discovery is the search for semantic relationships which exist in large databases. One of the main problems for knowledge discovery is that the number of possible relationships can be very large, thus searching for interesting relationships and reducing the search complexity are important. The relationships can be represented as rules which can be used in efficient query processing. We present a technique to analyze relationships among attribute values and to derive compact rule set. We also propose a mechanism and some heuristics to reduce the search complexity for the rule derivation process. An evaluation model is presented to evaluate the quality of the derived rules. Moreover, in real world, databases may contain uncertain data. We also propose a technique to analyze the relationships among uncertain data and derive probabilistic rules.

Keywords: knowledge discovery, semantic association degree, efficient rule derivation, learning algorithm, compact rules, uncertain data, probabilistic rules

1 Introduction

Because potentially useful information may be embeded in a large amount of data, it is difficult to discover by human beings. Hence, knowledge discovery in databases is becoming an increasingly important issue. The knowledge discovered can be used to answer cooperative

*To whom all correspondence should be sent.

queries [7, 16], handle null values [26] and facilitate semantic query optimization [9, 13, 14, 21, 22, 24, 28]. Knowledge discovery is a process of searching for interesting previously unknown relationships among attribute values. However, the number of possible relationships can be very large, thus searching for interesting relationships and reducing the search complexity are important. Moreover, in real world, databases may contain uncertain data. To search relationships in databases with uncertain data is thus necessary.

In this paper, we analyze relationships in databases to derive rules. These rules describe the dependencies between an attribute (called a *target attribute*) and other attributes (called *condition attributes*) in a relation. A rule consists of two parts, i.e., the antecedent and the consequent. The antecedent is a conjunction of $A_i = v_i$, where A_i is a condition attribute and v_i its associated value, while the consequent is $T = x$, where T is the target attribute and x its associated value. These derived rules can be used to provide high-level answers to user requests [12] and to improve query processing in databases. However, if many rules are derived or if antecendent of each rule involves many condition attributes, efficient use of these rules becomes difficult.

A *rule set* derived from a relation is a set of rules which characterize (or *cover*) all tuples in the relation, and each rule in the rule set has full confidence. A *compact rule set* is a rule set which causes the cardinality of the rule set and the number of attributes involved in the antecedent of each rule in the rule set to be diminished. A compact rule set can summarize and classify data concisely, and each rule in the compact rule set can make general description for the dependencies between attributes.

In a relation, if each attribute contains a large set of distinct values, rather complex rules may be derived and each rule may characterize just few tuples. We present a method to ascend specific values to higher-level concepts such that the number of distinct values can be diminished. By this way, the size of relations can also be reduced.

A mechanism to find relationships among data in different attributes and a technique to search important relationships in databases are also proposed. We first extract *semantic association relationships* [26, 27] from databases without uncertain data. A *learning algorithm*

was designed to identify important relationships to derive compact rule set.

For databases with uncertain data, rules which have full confidence cannot be derived. Hence, we propose a mechanism to analyze the semantic association relationships in databases with uncertain data and a learning algorithm to derive probabilistic rules with high confidence.

The rest of this paper is organized as follows. Section 2 describes the related work. Section 3 presents the basic concepts, the relationships in databases and the representation of uncertain data, and Section 4 a method to reduce relation sizes. Section 5 proposes a mechanism to analyze relationships and a learning algorithm to derive compact rules from databases without uncertain data. A mechanism to analyze relationships among uncertain data and a learning algorithm to derive probabilistic rules from databases with uncertain data are presented in Section 6. The analysis of computational complexity of our learning algorithm is discussed in Section 7. Finally, we conclude this paper and present directions for future research in Section 8.

2 Related Work

In this section, we discuss related works and compare other researchers' approaches with our techniques. The approach of Han, et al. [4, 5, 6, 10, 11] first manually identifies the relevant attributes for a target attribute and projects them out for rule derivation. This approach replaces lower-level concepts (or attribute values) of each attribute with higher-level concepts to reduce the number of distinct values in each attribute in the projected relation. For example, unique salary values are replaced by qualitative ranges of high, medium, or low. The number of tuples in the projected relation can be reduced, if some tuples have the same value in each attribute due to the above-mentioned replacements. From each tuple, a rule is derived. Since the cardinality of the reduced projected relation may still be large, relatively complex rules may remain. However, if the number of rules is required to be small, there may result in over-generalization and loss of valuable information. Also, the antecedent of each rule involves all condition attributes, which may be unnecessary.

Agrawal, et al. [2, 3] presented another approach to derive rules from a relation. Their approach derives rules for some combinations of values from different attributes, which appear in the same tuple in a relation with a frequency larger than a certain *threshold*. For example, if the values $I_1, I_2, ..., I_k$ of attributes $Y_1, Y_2, ..., Y_k$, respectively, $k \geq 2$, appear in the same tuple in the relation with a high frequency, then the rules are derived by using the values $I_1, I_2, ..., I_k$. The antecedent of each rule involves $k-1$ values from $I_1, I_2, ..., I_k$, and the consequent involves the remaining value. The number of rules derived depends on the setting of the threshold. If the threshold is small, many rules may be derived. Conversely, if the threshold is large, only a few rules are generated, which may result in loss of useful information about dependencies between attributes. It is difficult to find a suitable threshold to prevent problems of extreme rule sizes. Moreover, this approach requires many relation scans to derive rules, which is computationally inefficient.

Ziarko [29] analyzed data dependencies based on *rough set theory* [19]. A target attribute is identified and the remaining attributes in a relation are condition attributes. A measure was provided to analyze the dependency between every subset of condition attributes and a target attribute in a relation. A *minimal* subset is found from these subsets of which the dependencies between these subsets and the target attribute are functional dependency. This minimal subset and the target attribute are projected to derive rules. However, if there is no better mechanism to analyze the dependency between each subset of condition attributes and the target attribute, finding the minimal subset is time consuming. Moreover, it may be inefficient to separate data dependencies analysis and rule derivation.

We analyze relationships and identify important relationships in databases. Mechanisms are provided to extract the relationships such that our learning algorithm needs only one relation scan, which are much more efficient than other approaches [2, 3, 18, 20, 29] for which repeated scans of relation may be required. Our learning algorithm derives more compact rules than the rules derived by other approaches [2, 3, 4, 5, 6, 10, 11, 18, 20, 25]. A comparison of the degree of compactness of rules derived from various approaches appears in Section 5.4. Probabilistic rules are derived from databases with uncertain data using a

technique similar to that of deriving compact rules from databases without uncertain data. Other techniques have been proposed to discover knowledge from databases [1, 2, 3, 4, 6, 10, 17, 18, 20, 21, 25, 29]. However, these approaches all assume that the data in databases are certain.

3 Background Knowledge

In order to make data easier to characterize in terms of rules, one must diminish the number of distinct values in each attribute. The domain knowledge is provided to diminish the number of distinct values in an attribute. In this section, we describe the semantic association relationships for rule derivation. Moreover, since uncertain data exist in real world, the representations and definitions of uncertain data in databases are also given in this section.

3.1 Domain concept hierarchy

For each database attribute, if an attribute domain can be represented by higher-level concepts, then a *domain concept hierarchy* can be constructed for this attribute. The domain concept hierarchy is a specific-to-general structure that organizes varied levels of abstractions relevant to an attribute. The most specific values are domain elements of an attribute and the remaining values in the hierarchy are concepts specified by domain experts.

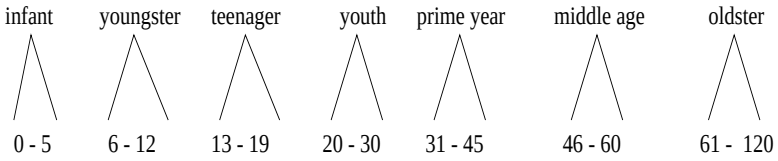

Figure 1: Domain concept hierarchy for "Age"

For a numerical attribute, according to the meaning of a numerical domain, we can divide its elements into groups, and represent each group by a concept. For example, suppose the attribute "Age" is a numerical domain in the range 0-120. The ages from 13 to 19 can be

represented by concept "teenager" (Figure 1). Some concepts can be further represented by higher-level concepts.

For a nonnumerical attribute, the domain concept hierarchy can be an 'IS-A' hierarchy, such as "a junior is an undergraduate student." The domain concept hierarchy can also be a taxonomy, such as "there are many countries in a continent and there are many cities in a country." For example, a domain concept hierarchy for the attribute "City" is shown in Figure 2.

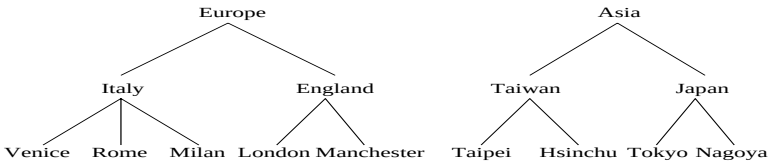

Figure 2: Domain concept hierarchy for "City "

3.2 Semantic association relationships

Semantic association relationships are associations between domain elements or concepts in different attributes. For example, suppose the relation EMPLOYEE contains attributes Degree and Salary and there exists a statistic "if degree is PHD, then salary is large," in which "PHD" is the domain element of attribute Degree and "large" is the concept specified in the domain concept hierarchy for the attribute Salary (Figure 3). Then, there is a semantic association from "Degree is PHD" to "Salary is large."

Semantic association degrees (SADs) are degrees of semantic associations between domain elements or concepts. For example, the SAD from "black eyes" to "Oriental" can be large, and that from "blue eyes" to "Oriental" close to zero. In our previous research [26], we suppose the SADs were specified by database designers. In this work, we provide a mechanism to efficiently obtain the SAD from a relation without uncertain data and that from a relation with uncertain data.

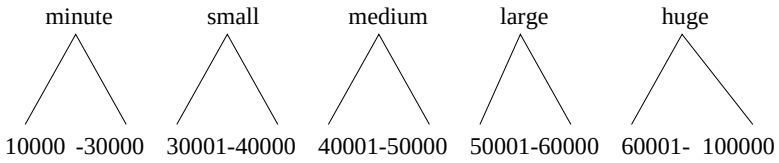

Figure 3: Domain concept hierarchy for "Salary"

For convenience, we discuss semantic association relationships in a relation. The relationships in separate relations can be found by joining these relations and applying the same approach with a single relation. Semantic association relationships are used to extract compact rules and probabilistic rules which will be discussed in the next two sections.

3.3 The representation of uncertain data

A *probabilistic partial value* [23] is a general representation of uncertain data. Uncertain data exist in databases due to their unavailability. In the following, we follow the definition of a probabilistic partial value given in [23], which is formally stated as follows:

Definition 1 [23]. A probabilistic partial value, denoted $v = [a_1^{p_1}, a_2^{p_2}, ..., a_n^{p_n}]$, associates with n possible values, $a_1, a_2, ..., a_n$, of the same domain D, where each a_i associates with a probability p_i such that $\sum_{i=1}^{n} p_i = 1$.

For a probabilistic partial value, $v = [a_1^{p_1}, a_2^{p_2}, ..., a_n^{p_n}]$, a function μ maps the probabilistic partial value to its corresponding finite set of possible values; that is, $\mu(v) = \{a_1, a_2, ..., a_n\}$ [23].

A definite value d can be regarded as a probabilistic partial value $[d^1]$. Besides, an applicable null value ϕ can be considered as a probabilistic partial value with $\mu(\phi) = D$, where D is the whole domain and the probabilities are uniformly distributed over the whole domain D.

Table 1 depicts a portion of a relation EMP which contains uncertain data. In relation EMP, the domain of attribute **specialty** contains PS(physics), BO(biology), CM(chemistry), MT(mathematics), CE(civil engineering), EE(electronic engineering), ES(English) and

GM(German), and the domain of attribute **position** contains LM(lab manager), JL(junior lecturer), SL(senior lecturer), ACP(associate professor), ASP(assistant professor) and FP(full professor). For the third tuple in relation EMP, the value in attribute **specialty** may be "EE" or "MT" with the probabilities $\frac{1}{5}$ and $\frac{4}{5}$, respectively. For the fifth tuple, the value in attribute **position** is an applicable null value ϕ which is a probabilistic partial value $[LM^{\frac{1}{6}}, JL^{\frac{1}{6}}, SL^{\frac{1}{6}}, ACP^{\frac{1}{2}}, ASP^{\frac{1}{6}}, FP^{\frac{1}{6}}]$.

name	**age**	**specialty**	**position**
Frank	$[35^{\frac{1}{2}}, 37^{\frac{1}{2}}]$	CE	LM
Tony	40	$[PS^{\frac{3}{4}}, BO^{\frac{1}{4}}]$	$[SL^{\frac{2}{3}}, ACP^{\frac{1}{3}}]$
Andy	$[28^{\frac{1}{3}}, 32^{\frac{2}{3}}]$	$[EE^{\frac{1}{5}}, MT^{\frac{4}{5}}]$	$[SL^{\frac{1}{2}}, ASP^{\frac{1}{2}}]$
Alex	65	$[ES^{\frac{1}{2}}, GM^{\frac{1}{2}}]$	FP
Jane	27	CM	ϕ
.	.	.	.
.	.	.	.

Table 1: Relation EMP

Definition 2 [23]. Two probabilistic partial values, say $u = [a_1^{p_1}, a_2^{p_2}, ..., a_n^{p_n}]$ and $v = [b_1^{q_1}, b_2^{q_2}, ..., b_n^{q_n}]$, are said to be equal, denoted $u = v$, if $a_i = b_i$ and $p_i = q_i$, for each $i = 1, ..., n$.

4 Reduction of Data Complexity

In a relation, if each attribute contains a large set of distinct values in the relation, the rules that are derived from the relation are not useful, because rather complex rules may be derived and each rule may characterize few tuples. Hence, the number of distinct values in each attribute needs to be limited.

Generalization is a process that ascends more specific values to higher-level concepts by climbing a domain concept hierarchy. Therefore, distinct values may correspond to the same concepts. In this way, the number of distinct values in an attribute can be diminished.

Hence, the preprocess of the knowledge discovery process is to perform generalization on each attribute in an *initial relation*. An initial relation is a relation formed by projecting

a target attribute and the condition attributes for the target attribute from a base relation. For example, suppose the base relation EMPLOYEE contains attributes Name, Age, Department, Salary and Position. If we seek to derive rules about attribute Position, attributes which are relevant to Position are selected. If we think that only attribute Name is irrelevant to Position, we can select attributes Age, Department and Salary from the base relation EMPLOYEE as the condition attributes and project attributes Age, Department, Salary and Position to form the initial relation.

Suppose IR is an initial relation, A_i is an attribute in IR, k_i is the number of distinct values in attribute A_i in IR, and n is the total number of tuples in IR. A *generalization threshold* is used to control the level of domain concept hierarchy ascension. The *generalization algorithm* (which is similar to the one used in Han, et al. [4, 5, 6, 10] to derive rules) is described as follows:

1. If the ratio of the number of distinct values in attribute A_i to the total number of tuples in IR, i.e. $\frac{k_i}{n}$, is greater than a generalization threshold, then execute step 2. Otherwise, the generalization algorithm on attribute A_i terminates

2. If there exist higher-level concepts in the domain concept hierarchy for the values in A_i, all values of A_i in IR are replaced according to their higher-level concepts by ascending the hierarchy one level. Otherwise, attribute A_i is removed from relation IR, because there is a large set of distinct values in A_i but attribute A_i cannot be generalized using higher-level concepts.

   For example, SSN (social security number) is an attribute in relation PERSON. However, since there is no domain concept hierarchy for attribute SSN, the values in attribute SSN cannot be replaced by higher-level concepts. Hence, attribute SSN is removed from relation PERSON after performing the generalization algorithm.

3. Since distinct values may be replaced by the same higher-level concepts, the number

of distinct values in A_i may be diminished and k_i may be modified.

The algorithm repeats Steps 1-3 until the ratio is less than or equal to the generalization threshold.

#	displace	fuelcap	mass	speed	cyl	cost
1	large	high	medium	medium	6	expensive
2	large	low	heavy	fast	6	expensive
3	medium	medium	light	fast	6	medium
4	small	low	light	slow	6	cheap
5	large	medium	medium	medium	6	expensive
6	large	medium	light	medium	6	expensive
7	small	low	light	medium	6	cheap
8	small	medium	light	slow	4	cheap
9	medium	low	medium	medium	6	medium
10	medium	high	medium	fast	6	expensive
11	medium	high	light	fast	6	expensive
12	small	high	heavy	medium	4	expensive
13	small	high	light	medium	4	cheap
14	medium	low	heavy	medium	4	expensive
15	medium	medium	medium	medium	4	medium
16	medium	high	medium	medium	4	medium
17	small	medium	medium	fast	4	medium
18	medium	medium	heavy	slow	4	expensive

Table 2: Relation GCAR

After performing generalization on each attribute in an initial relation, as distinct attribute values can be substituted by the same higher-level concepts, a set of tuples may be generalized to the same tuple. Redundant tuples are eliminated to generate a *generalized relation* (GR). Therefore, the size of an initial relation can be reduced. Suppose both relations GCAR and GEMP are generalized relations as shown in Table 2 and Table 3, respectively, where attribute # will be explained later.

#	age	specialty	position
1	$primeyear$	$engineering$	$staff$
2	$midage$	$science$	$[lecturer^{\frac{2}{3}}, professor^{\frac{1}{3}}]$
3	$[youth^{\frac{1}{3}}, primeyear^{\frac{2}{3}}]$	$[engineering^{\frac{1}{5}}, science^{\frac{4}{5}}]$	$[lecturer^{\frac{1}{2}}, professor^{\frac{1}{2}}]$
4	$oldster$	$literature$	$professor$
5	$youth$	$science$	ϕ
.	.	.	.
.	.	.	.

Table 3: Relation GEMP: a generalized relation of relation EMP

5 Rule Derivation in Databases without Uncertain Data

In this section, we discuss the semantic association relationships in databases without uncertain data. A mechanism for computing SAD is proposed. A learning algorithm for deriving compact rules follows. We give an example to illustrate the learning algorithm. Finally, an evaluation model is proposed to show that the set of rules derived according to our learning algorithm is more compact than that of other approaches [2, 4, 5, 6, 10, 11, 20, 25].

5.1 Information for computing SAD

We propose a mechanism by which the learning process needs to scan a GR only once. This scan serves to extract information from the GR for the learning algorithm. The other information for learning can be obtained by performing set operations on the extracted information. First of all, each tuple in the GR is assigned a unique *tuple number*. The attribute # in Table 2 and Table 3 show the numbers of the tuples.

Then, from the scan of the GR, the tuple numbers for the same attribute value in an attribute are recorded. For example, in the generalized relation GCAR (Table 2), the set of tuples with "medium" as a value in attribute **displace** includes {3, 9, 10, 11, 14, 15, 16, 18}. The tuple numbers for the same attribute values in **displace**, **fuelcap**, **mass**, **speed**, **cyl** and **cost** are shown in Table 4(a)−4(f), respectively.

displace	
large	{ 1,2,5,6 }
medium	{ 3,9,10,11,14,15,16,18 }
small	{ 4,7,8,12,13,17 }

(a)

fuelcap	
high	{ 1,10,11,12,13,16 }
medium	{ 3,5,6,8,15,17,18 }
low	{ 2,4,7,9,14 }

(b)

mass	
heavy	{ 2,12,14,18 }
medium	{ 1,5,9,10,15,16,17 }
light	{ 3,4,6,7,8,11,13 }

(c)

speed	
fast	{ 2,3,10,11,17 }
medium	{ 1,5,6,7,9,12,13,14,15,16 }
slow	{ 4,8,18 }

(d)

cyl	
6	{ 1,2,3,4,5,6,7,9,10,11 }
4	{ 8,12,13,14,15,16,17,18 }

(e)

cost	
expensive	{ 1,2,5,6,10,11,12,14,18 }
medium	{ 3,9,15,16,17 }
cheap	{ 4,7,8,13 }

(f)

Table 4: (a) The tuple numbers for the same attribute value in **displace**
(b) The tuple numbers for the same attribute value in **fuelcap**
(c) The tuple numbers for the same attribute value in **mass**
(d) The tuple numbers for the same attribute value in **speed**
(e) The tuple numbers for the same attribute value in **cyl**
(f) The tuple numbers for the same attribute value in **cost**

Suppose an event is an attribute-value pair; for example, event X=(**speed**, fast) denotes the value "fast" of attribute **speed**. A function TS maps one or two events into a set of tuples in which the event appears or the two events appear simultaneously in a GR, respectively. For example, in relation GCAR, if Y=(**displace**, medium) and Z=(**mass**, heavy) then $TS(Y)$={3, 9, 10, 11, 14, 15, 16, 18} and $TS(Y, Z)$={14, 18}.

Assume E_1 and E_2 are two events. $TS(E_1)$ and $TS(E_2)$ can be obtained according to the recorded information (e.g., Table 4). The set $TS(E_1, E_2)$ of tuples for the two events E_1 and E_2 appearing simultaneously can be obtained by performing a set intersection on $TS(E_1)$ and $TS(E_2)$:

$$TS(E_1, E_2) = TS(E_1) \bigcap TS(E_2) \qquad (1)$$

where the symbol "$\bigcap$" denotes set intersection.

For example, in relation GCAR, let E_1=(**displace**, medium) and E_2=(**mass**, heavy). $TS(E_1)$ and $TS(E_2)$ can be found according to Table 4(a) and 4(c), respectively. According to expression (1), the set of tuples for events E_1 and E_2 appearing simultaneously can be computed as follows:

$$\begin{aligned} & TS(E_1, E_2) \\ = \; & TS(E_1) \bigcap TS(E_2) \\ = \; & \{3, 9, 10, 11, 14, 15, 16, 18\} \bigcap \{2, 12, 14, 18\} \\ = \; & \{14, 18\} \end{aligned} \qquad (2)$$

5.2 A mechanism for computing SAD

In this subsection, we describe how to compute the semantic association degree (SAD) for a semantic association. A function f maps one or more events into the frequency of these events appearing in the same tuple in a GR. For example, in relation GCAR, if Y=(**displace**, medium) then $f(Y)$=8 as there are eight occurrences of value "medium" in attribute **displace** in GCAR.

Let N be the total number of tuples in the GR. A function P maps one or more events into the probability of these events appearing in the same tuple in the GR. We compute the probability $P(X)$ as:

$$P(X) = \frac{f(X)}{N} \qquad (3)$$

Let Z=(**mass**, heavy). Continuing the above example, $P(Y)$ and $P(Y, Z)$ are equal to $\frac{8}{18}$ and $\frac{2}{18}$ respectively, in which the total number of tuples in GCAR is 18.

Suppose E and H are two events. Let p be a SAD from E to H. That is, there is a degree p associated with "if E then H" as discussed in Section 2.2. The degree p is a conditional probability which can be represented as:

$$p(H|E) = \frac{P(E,H)}{P(E)} \qquad (4)$$

where event E is called a *condition event* and event H a *target event*.

By expression (3), $p(H|E)$ defined in the above can be computed as:

$$p(H|E) = \frac{f(E,H)}{f(E)} \qquad (5)$$

Suppose E_1, E_2, ..., and E_k are events. The frequency of events E_1, E_2, ..., and E_k appearing in the same tuple in a GR can be obtained as:

$$f(E_1, E_2, ..., E_k) = Card(\bigcap\{TS(E_1, E_2), TS(E_2, E_3), ..., TS(E_{k-1}, E_k)\}) \qquad (6)$$

where the symbol "$\bigcap$" denotes set intersection and $Card(S)$ denotes cardinality of set S.

Let H be a target event and E_1, E_2, ..., and E_k be condition events in a GR. The SAD from events E_1, E_2, ..., and E_k to event H can be obtained according to expression (5):

$$p(H|E_1, E_2, ..., E_k) = \frac{f(E_1, E_2, ..., E_k, H)}{f(E_1, E_2, ..., E_k)} \qquad (7)$$

For example, in relation GCAR, let events E_1 = (**displace**, medium), E_2 = (**mass**, medium) and E_3 = (**speed**, medium) be condition events and event H = (**cost**, medium) be a target event. Then, the frequency of events E_1, E_2 and E_3 appearing in the same tuple in relation GCAR can be computed according to expression (6):

$$f(E_1, E_2, E_3) = Card(\bigcap\{TS(E_1, E_2), TS(E_2, E_3)\}) \qquad (8)$$

From Table 4(a), 4(c), 4(d) and 4(f), $TS(E_1), TS(E_2), TS(E_3)$ and $TS(H)$ can be obtained, respectively. According to expression (1), TS(E_1, E_2) = {9, 10, 15, 16} and TS(E_2, E_3) = {1, 5, 9, 15, 16}.

$$
\begin{aligned}
f(E_1, E_2, E_3) &= Card(\bigcap\{\{9, 10, 15, 16\}, \{1, 5, 9, 15, 16\}\}) \\
&= Card(\{9, 15, 16\}) \\
&= 3 \qquad (9)
\end{aligned}
$$

Similarly, according to expression (1), TS(E_3, H) = {9, 15, 16}; the frequency of events E_1, E_2, E_3 and H appearing in the same tuple in relation GCAR is:

$$
\begin{aligned}
f(E_1, E_2, E_3, H) &= Card(\bigcap\{TS(E_1, E_2), TS(E_2, E_3), TS(E_3, H)\}) \\
&= Card(\bigcap\{\bigcap\{TS(E_1, E_2), TS(E_2, E_3)\}, TS(E_3, H)\}) \\
&= Card(\bigcap\{\{9, 15, 16\}, \{9, 15, 16\}\}) \\
&= Card(\{9, 15, 16\}) \\
&= 3 \qquad (10)
\end{aligned}
$$

Therefore, the SAD from events E_1, E_2 and E_3 to event H can be computed according to expression (7):

$$
p(H|E_1, E_2, E_3) = \frac{f(E_1, E_2, E_3, H)}{f(E_1, E_2, E_3)} = 1 \qquad (11)
$$

This result signifies that the SAD from combination (**displace**, medium), (**mass**, medium) and (**speed**, medium) of condition events to event (**cost**, medium) is 1. This semantic association is later regarded as a rule "IF **displace**=medium AND **mass**=medium AND **speed**=medium THEN **cost**=medium"; the set of tuples covered by this rule is {9, 15, 16}.

5.3 Learning algorithm

In the following, we describe a learning algorithm that makes use of semantic association relationships to derive compact rules about a target attribute from a GR.

Suppose there are k condition attributes CA_1, CA_2, ... and CA_k for a target attribute TA in a GR. Initially, in the first iteration of the learning algorithm, the SAD from each condition event to each target event is computed.

In the second iteration, the SAD from each combination of two condition events to each target event is computed. Subsequently, in the ith iteration, the SAD from each combination

of i condition events to each target event is computed, in which $1 \leq i \leq k$.

Let CE denote the combination (CA_i, ca_i), (CA_{i+1}, ca_{i+1}), ..., (CA_j, ca_j) of condition events and any combination which contains CE be denoted as CCE. We have the following Lemmas to be used in the learning algorithm.

Lemma 5.1 (Rule derivation) If the SAD from CE to event (TA, ta) is 1, a rule: "IF $CA_i = ca_i$ AND $CA_{i+1} = ca_{i+1}$ AND ... AND $CA_j = ca_j$ THEN $TA = ta$" is derived. Otherwise, no rule will be derived.

Rationale: Each rule in the compact rule set has full confidence. The SAD can be regarded as a confidence associated with the semantic association. As we seek to derive rules which have full confidence, only the semantic association of which degree is 1 becomes a rule in each iteration.

When a rule is derived, a set of tuples covered by this rule is recorded. This information can be obtained when the SAD for the rule is computed, as discussed in Section 5.2. Suppose $\Im$ is the set of all tuples in a GR, $\Im_R$ is a set of tuples covered by rule R and $\Im_u$ is a set of tuples which contain attribute value u. Assume that R_1, R_2, ... and R_{v-1} are previously derived rules.

Lemma 5.2 (Termination) When rule R_v is derived, if $\Im_{R_1} \cup \Im_{R_2} \cup ... \cup \Im_{R_v} = \Im$, the learning algorithm terminates.

Rationale: A compact rule set is a rule set which cover all tuples in the relation. When all the tuples in a GR are completely characterized by the derived rules, no rules need to be further derived.

Hence, if there are k condition attributes in GR, the learning algorithm requires to execute at most k iterations in the learning algorithm.

Lemma 5.3 (Removing rule) When rule R_v is derived, if $\Im_{R_i} \subset \Im_{R_v} (1 \leq i \leq v-1)$, rule R_i is discarded.

Rationale: Because all tuples covered by rule R_i are completely characterized by rule R_v, rule R_i is redundant which should be discarded. By this, the cardinality of the rule set can be diminished. Hence, this lemma retains the rule which covers more tuples.

Lemma 5.4 (Removing rule) If $\Im_{R_v} \subseteq \Im_{R_1} \cup \Im_{R_2} \cup ... \cup \Im_{R_{v-1}}$, rule R_v is discarded.

Rationale: Because all the tuples covered by rule R_v are completely characterized by the previously derived rules, rule R_v is redundant. Because the number of attributes involved in a previously derived rule is less than or equal to the number of attributes involved in the current derived rule, we prefer the previously derived rule. Hence, if the tuples covered by the current derived rule are covered by some previously derived rules, then the current derived rule is regarded as a redundant rule and is then discarded.

Lemma 5.5 (Reducing SAD computation) If the SAD from CE to event (TA, ta) is 0, the SADs from all CCEs to event (TA, ta) need not be computed in later iterations.

Rationale: In this case, CE is independent of target event (TA, ta). Hence, all CCEs are also independent of target event (TA, ta). The SAD from all $CCEs$ to target event (TA, ta) must be 0.

Lemma 5.6 (Reducing SAD computation) Continuing the case in Lemma 5.1, the SADs from all CCEs to any target event need not be computed in later iterations.

Rationale: If CE is combined with other events such that some rules are derived, the sets of tuples covered by these rules must be subsets of the set of tuples covered by the rule derived in Lemma 5.1. Moreover, if the SAD from CE to a target event is 1, then the SADs from CE to other target events must be 0. Hence, the SADs from all CCEs to any target event need not be computed in later iterations according to Lemmas 5.4 and 5.5.

Lemma 5.7 (Reducing SAD computation) When rule R_v is derived, if $\Im_{ta} \subseteq \Im_{R_1} \cup \Im_{R_2} \cup ... \cup \Im_{R_v}$, the SAD from any combination of condition events to target event (TA, ta) need not be computed later.

Rationale: When all tuples which contain a certain target attribute value in a GR are completely characterized by derived rules, no rules about this target attribute value need to be further derived.

Lemmas 5.5, 5.6 and 5.7 are heuristics to decrease the number of computations of SADs. The learning algorithm follows:

Algorithm LCR (Learning Compact Rules)

for $i = 1$ **to** k **do**

begin

for each combination $CA_i, ..., CA_j$ selected from CA_1, CA_2, ...,CA_k of size l **do**

DeriveRules($CA_i, ..., CA_j, TA$)

if all tuples in GR are characterized by derived rules

then terminate (Lemma 5.2)

else

if the set of tuples covered by derived rules of which

the consequents are "$TA = ta$" contains all tuples

in which event (TA, ta) appears in GR

then remove event (TA, ta) from attribute TA (Lemma 5.7)

end

DeriveRules($CA_i, ..., CA_j, TA$)

for each combination ca_i, ..., ca_j, denoted φ **do**

begin

for each value ta in attribute TA **do**

begin

if a combination which is contained in φ, value ta,

or the semantic association from a combination which is

contained in φ to value ta has not been removed

then

begin

compute SAD from φ to ta and

if SAD is 1

then

begin

derive a rule "IF $CA_i = ca_i$ AND ... AND $CA_j = ca_j$ THEN $TA = ta$"

(Lemma 5.1), record the set of tuples covered by this rule,

remove φ (Lemma 5.6), and
if the set of tuples covered by the previously derived rule is
a proper subset of the set of tuples covered by this rule
then discard the previously derived rule (Lemma 5.3)
if the set of tuples covered by this rule is a subset of the union
of the sets of tuples covered by previously derived rules
then discard this rule (Lemma 5.4)
end
else
if SAD is 0
then remove the semantic association from φ to value ta (Lemma 5.5)
end
end
end
end

A simple example is given below to demonstrate the LCR algorithm.

Example: Consider the relation GCAR shown in Table 2. We want to analyze the relationships between target attribute **cost** and other attributes in GCAR and derive compact rules about attribute **cost** from GCAR.

In the first iteration of the LCR algorithm, the SAD from each condition event to each target event in GCAR is obtained. Tables 5(a) and 5(b) show SADs from each value in condition attributes **displace** and **mass** to each value in target attribute **cost**, respectively. The SADs from event (**displace**, large) and from event (**mass**, heavy) to event (**cost**, expensive) are both 1. Hence, the rules "IF **displace**=large THEN **cost**=expensive" and "IF **mass**=heavy THEN **cost**=expensive" are derived and recorded as rules 1 and 2, respectively. The set of tuples covered by rule 1 is {1, 2, 5, 6} and the set of tuples covered by rule 2 is {2, 12, 14, 18}. Condition events (**displace**, large) and (**mass**, heavy) are then removed, because the two events need not be combined with other condition events in later iterations

according to Lemma 5.6.

	cost		
displace	expensive	medium	cheap
large	1	0	0
medium	$\frac{1}{2}$	$\frac{1}{2}$	0
small	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{2}{3}$

(a)

	cost		
mass	expensive	medium	cheap
heavy	1	0	0
medium	$\frac{3}{7}$	$\frac{4}{7}$	0
light	$\frac{2}{7}$	$\frac{1}{7}$	$\frac{4}{7}$

(b)

		cost		
displace	**mass**	expensive	medium	cheap
medium	medium	$\frac{1}{4}$	$\frac{3}{4}$	×
medium	light	$\frac{1}{2}$	$\frac{1}{2}$	×
small	medium	0	1	×
small	light	0	0	1

(c)

Table 5: Part of SADs

Table 5(c) shows the SAD from each combination of values in **displace** and **mass** to each value in **cost**. Because events (**displace**, large) and (**mass**, heavy) have been removed, there are four combinations of values in **displace** and **mass**.

Similarly, the rules "IF **displace**=small AND **mass**=medium THEN **cost**=medium" and "IF **displace**=small AND **mass**=light THEN **cost**=cheap" are derived and recorded as rules 3 and 4. The set of tuples covered by rule 3 is {17} and the set of tuples covered by rule 4 are {4, 7, 8, 13}. From Table 4(f), the set of tuples in which event (**cost**,cheap) appears is also {4, 7, 8, 13}. The target event (**cost**,cheap) is then removed according to Lemma 5.7, because all tuples which contain attribute value (**cost**,cheap) in GR are characterized completely by rule 4. Also, SADs from any combination of condition events to target event (**cost**, cheap) need not be computed later.

In the first iteration, the SADs from event (**displace**, medium) and from event (**mass**, medium) to event (**cost**, cheap) are both 0, respectively. Hence, the SAD from each combination which contains event (**displace**, medium) or event (**mass**, medium) to event (**cost**, cheap) need not be computed in later iterations according to Lemma 5.5. The entry of SADs from any combination of condition events which contains event (**displace**, medium) or event (**mass**, medium) to event (**cost**, cheap) is marked "$\times$", as shown in Table 5(c).

Continuing the LCR algorithm, the rule "IF **fuelcap**=medium AND **speed**=fast THEN **cost**=medium" is derived; the set of tuples covered by this rule is {3, 17}. The set of tuples covered by rule 3 is a proper subset of the set of tuples covered by this rule. Hence, rule 3 is discarded according to Lemma 5.3.

After completing algorithm LCR on GCAR , the derived rule set are shown in Figure 4. Notice that the numbers which follow a rule is the numbers of the tuples covered by the rule. This coverage measurement could help explain the significancy of each rules.

Rule 1: IF **displace**=large THEN **cost**=expensive {1, 2, 5, 6}
Rule 2: IF **mass**=heavy THEN **cost**=expensive {2, 12, 14, 18}
Rule 3: IF **fuelcap**=high AND
cyl=6 THEN **cost**=expensive {1, 10, 11}
Rule 4: IF **fuelcap**=medium AND
speed=fast THEN **cost**=medium {3, 17}
Rule 5: IF **displace**=small AND
mass=light THEN **cost**=cheap {4, 7, 8, 13}
Rule 6: IF **displace**=medium AND **mass**=medium
AND **speed**=medium THEN **cost**=medium {9, 15, 16}

Figure 4: A set of rules derived according to the LCR algorithm

5.4 Evaluation Model

An *evaluation model* is constructed in the following manner to evaluate how compact a rule set about a target attribute is. There are three observations: 1. the smaller the cardinality of the rule set is, the more compact is the rule set; 2. the more tuples covered by each rule in the rule set, the more compact is the rule set; 3. the smaller the number of condition attributes in the antecedent of a rule in the rule set is, the more compact is the rule set.

Suppose the cardinality of the rule set about a certain target attribute is r, the number of the tuples covered by rule i in the rule set is t_i, n_i is the number of tuples in which the target attribute value involved in the consequent of rule i appears, and the number of condition attributes in the antecedent of rule i is C_i, $1 \leq i \leq r$. The evaluation model is:

$$E = \frac{1}{r}\sum_{i=1}^{r} \frac{t_i}{n_i} \times \frac{1}{C_i} \qquad (12)$$

According to this evaluation model, we see that the larger the value E is, the more compact is the rule set; The value E for the rule set about target attribute **cost** in GCAR according to the LCR algorithm is 0.33.

The compactness degree E for a rule set derived from a relationsabout a certain target attribute may be 1. In this case, for each rule in the rule set, there is only one condition attribute in the antecedent of the rule, and all tuples containing the target attribute value involved in the consequent of the rule are covered completely by the rule. That is, the number of the distinct values of the target attribute in the relation is equal to the cardinality of the rule set. However, for a relation, the largest E value for all possible rule sets about a certain target attribute may be less than 1.

Hence, the upper (lower) bound of the value E for the rule sets derived from a relation about a certain target attribute is the largest (smallest) E value among all possible E values. The rule set with the largest (smallest) E value is the most (least) compact one among all possible rule sets.

Here, we illustrate two other approaches to show that the rule set derived according to the LCR algorithm is more compact than those derived by these two approaches.

Quinlan [20] proposed a learning method ID3 for classification. *Decision trees* are built for assigning tuples to a single class based on values of a certain target attribute. ID3 contructs a decision tree based on the information theory to choose a condition attribute for each node. From a node, distinct values of the attribute on the node divide tuples into different branches. If the tuples in a branch are all in the same class, these tuples need not

be divided again and the leaf node is the class name, i.e., the common target attribute value of these tuples. Otherwise, ID3 will continuously choose a condition attribute to divide these tuples. After a decision tree is constructed, each path from the root node to a leaf node is traced. For a path, a rule is derived. The conjunction of the condition attribute values on the branches of the path is involved in the antecedent of the rule, and the target attribute value in the leaf node is involved in the consequent of the rule. However, once an attribute has been chosen for a node, the antecedent of each rule which passes this node must involve the attribute. For example, in relation GCAR, attribute **displace** is chosen as the root node. Hence, the antecedent of each rule must involve attribute **displace** (see Figure 5). However, the antecedent of the rule in the compact rule set may not involve attribute **displace**.

The LCR algorithm extracts rules which involve minimal condition attributes in their antecedent and cover maximal tuples, such that the number of attributes involved in the antecedent of each rule and the cardinality of the rule set can be diminished. Hence, the rule set derived by LCR algorithm is more compact than the rule set derived by ID3 method.

Rule 1: IF **displace**=large THEN **cost**=expensive {1, 2, 5, 6}
Rule 2: IF **displace**=medium AND
mass=heavy THEN **cost**=expensive {14, 18}
Rule 3: IF **displace**=small AND
mass=heavy THEN **cost**=expensive {12}
Rule 4: IF **displace**=small AND
mass=medium THEN **cost**=medium {17}
Rule 5: IF **displace**=small AND
mass=light THEN **cost**=cheap {4, 7, 8, 13}
Rule 6: IF **displace**=medium AND **mass**=medium
AND **speed**=fast THEN **cost**=expensive {10}
Rule 7: IF **displace**=medium AND **mass**=medium
AND **speed**=medium THEN **cost**=medium {9, 15, 16}
Rule 8: IF **displace**=medium AND **mass**=light
AND **fuelcap**=high THEN **cost**=expensive {11}
Rule 9: IF **displace**=medium AND **mass**=light
AND **fuelcap**=medium THEN **cost**=medium {3}

Figure 5: A set of rules derived according to ID3

Applying ID3 method to derive rules about attribute **cost** from relation GCAR, the

result is shown in Figure 5. According to the evaluation model in expression (12), the value E of the rule set in Figure 5 is 0.172, which is smaller than that of the rule set derived according to LCR algorithm.

We also apply the learning algorithm presented by Han, et al. [4, 5, 6, 10, 11] to derive rules about attribute **cost** from relation GCAR. The relevant attributes are **displace**, **mass**, **fuelcap** and **speed**. Suppose the relation GCAR is a *final generalized relation* after applying their learning algorithm. Each tuple in the relation GCAR becomes a rule, and there are 18 rules in the rule set. According to the evaluation model, the value E of this rule set is 0.033.

6 Rule Derivation in Databases with Uncertain Data

Semantic association relationships among certain data have been discussed in Section 4. In order to extract the relationships in databases with uncertain data, the information and mechanism for computing SAD, and the learning algorithm need to be revised. In this section, we present a mechanism to efficiently compute SADs among uncertain data. A learning algorithm is also presented to derive probabilistic rules.

6.1 Information for computing SAD

A function T maps a value v into a set of tuple numbers, which contain v, and the associated probabilities; that is, $T(v) = \{N_1^{p_1}, N_2^{p_2}, ..., N_i^{p_i}\}$, where $N_1, N_2, ..., N_i$ are numbers of the tuples which contain value v and $p_1, p_2, ..., p_i$ are their associated probabilities, respectively. For example, in relation GEMP (Table 3), $T(\text{"}science\text{"}) = \{2^1, 3^{\frac{4}{5}}, ...\}$. For each value v in a relation, $T(v)$ will be recorded.

Lemma 6.1 If k values from k different attributes appear in the same tuple t and their associated probabilities are $p_1, p_2, ...$ and p_k, respectively, the probability of the k values appearing in tuple t is $p_1 \times p_2 \times ... \times p_k$.

Relationale: For a tuple, values from different attributes can be regarded as mutually independent.

A function TW maps two values from two different attributes into a set of tuples in which

the two values appear simultaneously, and the associated probabilities.

For example, in relation GEMP, $TW(\text{"}science\text{"}, \text{"}professor\text{"}) = \{2\frac{1}{3}, 3\frac{2}{5}, ...\}$ according to Lemma 6.1. For each value pair x and y from two different attributes in a relation, $TW(x, y)$ will be recorded. The set $T(v)$ and $TW(x, y)$ are called *probabilistic tuple sets*.

6.2 A mechanism for computing SAD

Suppose N is the total number of the tuples in GR. A function P maps one or more values from different attributes into the probability of their appearing in the same tuples in a relation.

If the probabilities of k (≥ 1) values from k different attributes simultaneously appearing in m tuples $t_1, t_2, ...$ and t_m are $p^{t_1}, p^{t_2}, ...$ and p^{t_m}, respectively, the probability of the k values appearing in the same tuples in the relation is:

$$P(V) = \frac{1}{N} \sum_{i=1}^{m} p^{t_i} \qquad (13)$$

in which V stands for the k values.

A function TS maps a probabilistic tuple set into a set of tuples contained in the set. For example, in relation GEMP, $TS(T(\text{"}science\text{"})) = \{2, 3, ...\}$ and $TS(TW(\text{"}science\text{"}, \text{"}professor\text{"})) = \{2, 3, 5, ...\}$.

The set of tuples in which k values $v_1, v_2, ...$ and v_k from k different attributes appear simultaneously can be obtained:

$$TM(v_1, v_2, ..., v_k) = \bigcap \{TS(TW(v_1, v_2)), TS(TW(v_2, v_3)), ..., TS(TW(v_{k-1}, v_k))\} \qquad (14)$$

in which the symbol "$\bigcap$" denotes set intersection.

For example, the values "*youth*", "*science*" and "*professor*", which are values of **age**, **specialty** and **position**, respectively, appear simultaneously in a tuple in relation GEMP

is:

$$\begin{aligned} & TM(\text{"}youth\text{"},\text{"}science\text{"},\text{"}professor\text{"}) \\ = \; & \bigcap\{TS(TW(\text{"}youth\text{"},\text{"}science\text{"})), TS(TW(\text{"}science\text{"},\text{"}professor\text{"}))\} \\ = \; & \bigcap\{\{3,\ 5,\ ...\},\{2,\ 3,\ 5,\ ...\}\} \\ = \; & \{3,\ 5,\ ...\} \qquad\qquad (15) \end{aligned}$$

An operation σ_{t_i} retrieves the probability with which tuple t_i in a probabilistic tuple set is associated. For example, $\sigma_3(T(science)) = \frac{4}{5}$ and $\sigma_3(TW(science, professor)) = \frac{2}{5}$.

In order to obtain the probability of k values from k different attributes appearing in the same tuples in a relation, it needs to scan a relation and apply expression (13). However, there is a mechanism to obtain the probability without any relation scan. Suppose the tuples in which values $v_1, v_2, ...$ and v_k appear simultaneously are $t_1, t_2, ...$ and t_m, respectively, which can be obtained according to expression (14).

The probability of values $v_1, v_2, ...$ and v_k appearing in the same tuples in the relation can be computed by expression (13); in which $V = \{v_1, v_2, ..., v_k\}$, and

$$p^{t_i} = \frac{\prod_{j=1}^{k-1} \sigma_{t_i}(TW(v_j, v_{j+1}))}{\prod_{l=2}^{k-1} \sigma_{t_i}(T(v_l))} \qquad (16)$$

For example, the tuples in which values "$youth$," "$science$" and "$professor$" appear simultaneously in a tuple in relation GEMP are $\{3, 5, ...\}$, which is computed in expression (15).

According to expression (16), the probability of values "$youth$," "$science$" and "$professor$" appearing in the third tuple in relation GEMP is computed in expression (17), in which $t_1 = 3, k = 3, v_1 =$ "$youth$," $v_2 =$ "$science$" and $v_3 =$ "$professor$."

By scanning relation GEMP and applying Lemma 6.1, the probability of the three values appearing in the third tuple is $\frac{1}{3} \times \frac{4}{5} \times \frac{1}{2} = \frac{2}{15}$. Hence, the correctness of expression (16) can be verified.

$$
\begin{aligned}
p^{t_1} &= \frac{\prod_{j=1}^{2} \sigma_3(TW(v_j, v_{j+1}))}{\prod_{l=2}^{2} \sigma_3(T(v_l))} \\
&= \frac{\sigma_3(TW(\text{"}youth\text{"}, \text{"}science\text{"})) \times \sigma_3(TW(\text{"}science\text{"}, \text{"}professor\text{"}))}{\sigma_3(T(\text{"}science\text{"}))} \\
&= \frac{\sigma_3(\{3^{\frac{4}{15}}, 5^1\}) \times \sigma_3(\{2^{\frac{1}{3}}, 3^{\frac{2}{5}}, ...\})}{\sigma_3(\{2^1, 3^{\frac{4}{5}}, 5^1\})} \\
&= \frac{\frac{4}{15} \times \frac{2}{5}}{\frac{4}{5}} \\
&= \frac{2}{15} \qquad (17)
\end{aligned}
$$

Suppose E_1, E_2, ... , E_k and H are attribute-value pairs (events) and involve values e_1, e_2, ..., e_k and h, respectively. Let p be a SAD from $E_1, E_2, ...$ and E_k to H. Then, there is a semantic association "if E_1 and E_2 and ... and E_k then H" with degree p discussed in Section 3.1. Hence, p is a conditional probability that can be obtained as:

$$p(H|E_1, E_2, ..., E_k) = \frac{P(e_1, e_2, ..., e_k, h)}{P(e_1, e_2, ..., e_k)} \qquad (18)$$

where $P(e_1, e_2, ..., e_k, h)$ and $P(e_1, e_2, ..., e_k)$ can be obtained according to expressions (13) and (16).

6.3 Learning algorithm

The learning algorithm is used to derive probabilistic rules about a certain target attribute from databases with uncertain data.

Suppose there are k condition attributes CA_1, CA_2, ... and CA_k for a target attribute TA. Initially, in the first iteration of the learning algorithm, the SADs from each value in CA_i (for all i, $1 \leq i \leq k$) to each value in TA is computed.

In the second iteration, the SADs from each combination of values from two different

condition attributes to each value in TA is computed. Subsequently, in the ith iteration, the SADs from each combination of values from i different condition attributes to each target attribute value is computed.

Lemma 6.2 (Rule derivation) If the SAD d from values ca_i, ca_{i+1}, ... and ca_j in CA_i, CA_{i+1}, ... and CA_j, respectively, to value ta in TA is larger than a threshold (named *max_threshold*), a rule "if $CA_i = ca_i$ and $CA_{i+1} = ca_{i+1}$ and ... and $CA_j = ca_j$ then $TA = ta$ with confidence d" will be derived.

Relationale: A SAD can be regarded as a confidence associated with the semantic association according to expression (15). Before a semantic association becomes a rule, its associated probability is a degree. A SAD becomes a confidence when the semantic association has been retrieved as a rule.

Lemma 6.3 (Reducing SAD computation) If the SAD from values ca_i, ca_{i+1}, ... and ca_j in CA_i, CA_{i+1}, ... and CA_j, respectively, to value ta in TA is smaller than a threshold (named *min_threshold*), the SAD from any combination which contains ca_i, ca_{i+1}, ... and ca_j to value ta need not be computed in later iterations.

Relationale: In this case, since ca_i, ca_{i+1}, ... and ca_j can not determine ta, any combination which contains ca_i, ca_{i+1}, ... and ca_j also can not determine ta. Hence, this Lemma is a heuristic to decrease the number of computations of SADs such that the efficiency of the learning algorithm can be improved.

7 Computational Complexity of rule derivation algorithm

In this section, we analyze the computational complexity of the LCR algorithm. Suppose N is the total number of tuples in GR and there are k condition attributes in GR. It takes time $O(N)$ to record the tuple numbers for the same attribute value in an attribute, and for each combination of attribute values in the same tuple from two different attributes. After recording this information, our learning algorithms need not scan GR again. However, learning techniques proposed in Ziarko [20] and Agrawal, et al. [2] may need to scan a

relation repeatedly.

We analyze the computational complexity of a SAD. Assume that A_1, A_2, ... and A_i are condition events and that A_{i+1} is a target event. Suppose the average cardinality of $TS(A_j, A_{j+1})$ is m, for all j within 1 and i. In order to obtain a SAD from the combination A_1, A_2, ... and A_i to event A_{i+1}, the intersections of tuple sets $TS(A_1, A_2)$, ... and $TS(A_i, A_{i+1})$ need to be computed, as discussed in Section 3.3. The intersection of two tuple sets requires at most $2m$ comparisons to find the same elements in the two tuple sets and in total, $i-1$ intersections are needed to find the same elements in the i tuple sets. Hence, in the worst case, $2m(i-1)$ comparisons are required to obtain a SAD in iteration i. As i is less than or equal to k and numbers m and k are both small numbers, few comparisons are needed to obtain a SAD.

Suppose, the average number of distinct values in each attribute is h. For iteration i, there are $C_i^k = \frac{k!}{(k-i)!i!}$ combinations of i attributes selected from k condition attributes, and there are h^i combinations of values from i condition attributes. Hence, there are at most $h^{i+1} \times C_i^k$ SADs to compute in iteration i. As there are heuristics proposed, the number of computations of SADs can be decreased. Moreover, our learning algorithms need to scan a relation only once. After this relation scan, the efficiency of our learning algorithms is independent of the size of the relation. However, most learning algorithms suffer from inefficiency problems in a large database environment [2, 8, 15, 29].

8 Conclusion and Future Work

In order to make data more regular and easier to characterize in terms of rules, we presented a generalization algorithm that can generalize the data and diminish their sizes. Domain concept hierarchies are introduced to support the generalization algorithm. In the real-world, the data in databases may be uncertain. Applying probabilistic partial values is a general way to represent these uncertain data.

Knowing semantic association relationships are important in databases. A learning algorithm is presented to identify important relationships and to derive compact rules in

databases without uncertain data. An evaluation model was developed to evaluate how compact a rule set about a target attribute is. The example rule set derived according to the LCR algorithm is demonstrated to be more compact than the rule sets derived according to other approaches. Probabilistic rules can also be derived from databases with uncertain data according to a similar learning algorithm.

Mechanisms used to obtain the SADs are proposed such that our learning algorithms need to scan a relation only once. Finally, we evaluate the efficiency of our learning techniques.

We shall extend our evaluation model to evaluate the quality of probabilistic rules. Also, we shall consider resource discovery which extracts knowledge from multidatabase systems. Since there exist incompatibilities in distinct databases, we need to consider resolving conflicts in the rule sets derived from these databases.

References

[1] R. Agrawal, S. Ghosh, T. Imielinski, B. Iyer, and A. Swami. An Interval Classifier for Database Mining Applications. In *VLDB-92*, pages 560–573, Vancouver, British Columbia, 1992.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of ACM SIGMOD*, pages 207–216, 1993.

[3] R. Agrawal and R. Srikant. Fast Algorithm for Mining Association Rules. In *Proc. 20th Intl. Conf. on Very Large Data Bases*, pages 487–499, 1994.

[4] Y. Cai, N. Cercone, and J. Han. Attribute-Oriented Induction in Relational Databases. In *Knowledge Discovery in Databases. Menlo Park, CA: AAAI/MIT*, pages 213–228, 1990.

[5] Y. Cai, N. Cercone, and J. Han. Learning Characteristic Rules from Relational Databases. In *Computational Intelligence, II, F.Gardin and G. Mauri (Editors)*, pages 187–196, 1990.

[6] Y. Cai, N. Cercone, and J. Han. An Attribute-Oriented Approach for Learning Classification Rules from Relational Databases. In *Proc. 6th Int. Conf. on Data Engineering, Los Angeles*, pages 281–288, Feb 1990.

[7] W. Chu, R.C. Lee, and Q. Chen. Using Type Inference and Induced Rules to Provide Intensional Answers. In *Seventh International Conference on Data Engineering*, pages 396–403, 1991.

[8] T.G. Dietterich and R.S. Michalski. A comparative review of selected methods for learning from examples. In *Machine Learning: An Artifical Intelligence Approach, Vol.1*, pages 41–82, 1983.

[9] M. Hammer and S.B. Zdondik. Knowledge-based query processing. In *Proceedings of the 6th VLDB Conference*, pages 137–146, 1980.

[10] J. Han, Y. Cai, and N. Cercone. Knowledge Discovery in Databases: An Attribute-Oriented Approach. In *Proceedings of the 18th VLDB Conference*, pages 547–559, 1992.

[11] J. Han, Y. Cai, and N. Cercone. Data-Driven Discovery of Quantitative Rules in Relational Databases. In *IEEE Trans. Knowledge and Data Engineering*, pages 29–40, 1993.

[12] T. Imielinski. Intelligent Query Answering in Rule-Based Systems. *Journal of Logic Programming*, 4:229–257, 1987.

[13] M. Jarke. Semantic Query Optimization in Expert Systems and Database Systems. In *Proceedings of the First International Conference on Expert Database Systems*, pages 467–482, 1984.

[14] C. Malley and S. Zdonik. A Knowledge-Based Approach to Query Optimization. In *Proceedings of the First Expert Database System Conference*, pages 243–257, 1986.

[15] R.S. Michalski. A theory and Methodology of Inductive Learning. *Machine Learning: An Artifical Intelligence Approach*, 1:83–134, 1983.

[16] A. Motro. Using Integrity Contraints to Provide Intensional Answers to Relational Queries. In *Proc. 15th Intl. Conf. on Very Large Data Bases*, 1989.

[17] G. Oosthuizen. Lattice-Based Knowledge Discovery. In *Knowledge Discovery in Databases. Menlo Park, CA: AAAI/MIT*, pages 221–235, 1991.

[18] J.S. Park, M.S. Chen, and P.S. Yu. An Effective Hash-Based Algorithm for Mining Association Rules. *SIGMOD RECORD*, 24(2):175–186, 1995.

[19] Z. Pawlak. Rough Sets. *International Journal of Computer and Information Sciences*, 11(5):341–356, 1982.

[20] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[21] Michael Siegeland Edward Sciore and Sharon Salveter. A Method for Automatic Rule Derivation to Support Semantic Query Optimization. In *ACM Transactions on Database Systems*, pages 563–600, 1992.

[22] M. Siegel. Automatic Rule Derivation for Semantic Query Optimization. In *Second International Conference on Expert Database Systems*, pages 371–385, 1988.

[23] F.S.C. Tseng, A.L.P. Chen, and W.P.Yang. Answering Heterogeneous Database Queries with Degrees of Uncertainty. *Distributed and Parallel Databases*, 1:281–302, 1993.

[24] U.Chakravarthy, D. Fishman, and J. Minker. Semantic Query Optimization in Expert Systems and Database Systems. In *Proceedings of the First International Conference on Expert Database Systems*, pages 326–340, 1984.

[25] Patrick Henry Winston. Learning by Building Identification Trees. In *Artificial Intelligence*, pages 423–432, 1992.

[26] S.J. Yen and A.L.P. Chen. Neighborhood/Conceptual Query Answering with Imprecise/Incomplete Data. In *Proceedings of the 12th International Conference on Entity-Relationship Approach*, pages 151–162, 1993.

[27] S.J. Yen and A.L.P. Chen. An Efficient Algorithm for Deriving Compact Rules from Databases. In *the 4th International Symposium on Database Systems for Advanced Application*, 1995.

[28] C. Yu and W. Sun. Automatic Knowledge Acquisition and Maintenance for Semantic Query Optimization. In *IEEE Trans. Knowl. Data Eng.*, pages 362–375, 1989.

[29] W. Ziarko. The Discovery, Analysis, and Representation of Data Dependencies in Databases. In *Knowledge Discovery in Databases. Menlo Park, CA: AAAI/MIT*, pages 195–209, 1991.