Computational Intelligence, Volume , Number ,

QUANTIFICATION OF UNCERTAINTY IN CLASSIFICATION RULES DISCOVERED FROM DATABASES

Y. Xiang, S.K.M. Wong and N. Cercone
Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2

We apply rough set constructs to inductive learning from a database. A design guideline is suggested, which provides users the option to choose appropriate attributes, for the construction of data classification rules. Error probabilities for the resultant rule are derived. A classification rule can be further generalized using concept hierarchies. The condition for preventing overgeneralization is derived. Moreover, given a constraint, an algorithm for generating a rule with minimal error probability is proposed.

1. INTRODUCTION

The rapidly growing size and number of databases, and the realization that intelligently analyzed data is a valuable resource have generated increasing demands for knowledge discovery in databases (Frawley 1991).

In this paper, we assume data are represented by a relational database in which information about individual objects in a domain is represented by a set of tuples of attribute values. Adopting the view of 'learning by examples' from Artificial Intelligence (AI), we may regard a database as a set of training examples. The objective of one form of learning is to produce a classification rule in a disjunctive normal form (DNF) for a particular concept or class. A learned rule can be generated using the vocabulary of attributes. We shall call the set of attributes used in the database a *basis set* (Genesereth 1987), and call the learning process *induction using attributes*. For complex domains, the rule generated using the basis set may contain too many conjuncts. One way to make the rule more compact, as well as more general, is to partition attribute values for each attribute using a concept hierarchy. The rule is then represented in a higher level language than the original basis set. We shall call this further generalization *induction by hierarchy*.

Given a basis set A' of attributes, a user may be interested only in a subset which is then used to create the rule. Given this restriction, we can generate the rule using a minimum subset $A \subset A'$ of attributes. Such a minimal subset may not be unique. We will discuss a design guideline based on rough sets (Pawlak 1982) to provide users with different options for the minimal subset.

Once a rule is generated by a learning system, a user may wish to know how reliable is the rule. We will show how error probabilities can be estimated for each component of the rule and for the rule as a whole.

Induction by hierarchy produces generalized rules. However, this induction process may overgeneralize, as discussed in Section 7, and thus increase the classification error of the resultant rule. We derive the condition under which induction by hierarchy does not introduce additional classification error.

In complex domains, the number of conjuncts in the resultant rule may be

large. A user may want to limit the number of conjuncts involved. We discuss how to produce a rule which satisfies such a restriction and minimizes the classification error.

In complex domains, there are often exceptions to general principles. For example, most birds fly but 'penguin' as a bird does not. Each exception will form a conjunct in a DNF rule. Sometimes, the user is interested in only the general principles. An error bound can be used to prune those conjuncts which correspond to exceptions. We will show how to generate a rule whose classification error is below a given threshold such that it includes minimal exceptions.

Our approach is based on the rough set theory (Pawlak 1982) which provides a theoretical framework for knowledge discovery in databases.

2. TERMINOLOGY

2.1. Basic Notions of Rough Sets

Let U be the *universe* of discourse, and let R be an equivalence relation on U. The pair $Z = (U, R)$ is called an *approximation space*. If $x, y \in U$ and $(x, y) \in R$, x and y are said to be *indistinguishable* in Z. Each equivalence class of the relation R is called an *elementary set* in Z. A finite union of elementary sets in Z is called a *composed definable set* or simply *composed set* in Z.

Let X be a subset of U. The least composed set in Z containing X is called *upper approximation* of X in Z, denoted by $\overline{Apr}(X)$; the greatest composed set in Z contained in X is called the *lower approximation* of X in Z, denoted by $\underline{Apr}(X)$. The set $Bnd(X) = \overline{Apr}(X) - \underline{Apr}(X)$ is called the *boundary* of X in Z.

2.2. A Database as Learning Examples

Refer to Figure 1 for illustration of the following notations. Ignore the lower right portion of the figure for now. Let Y be a domain of objects. Let $C \subset Y$ be a class of objects. We define a function $c : Y \to \{0, 1\}$ by the following rule: for each $y \in Y$, $c(y) = 1$ if $y \in C$, and $c(y) = 0$ if $y \notin C$. Let $S \subset Y$ be a finite set of sample objects. Let $A' = \{A_1, \ldots, A_n\}$ be a set of attributes, and let $V' = \{V_1, \ldots, V_n\}$ be the domains of these attributes. Let T' be the Cartesian product: $T' = V_1 \times \ldots \times V_n$. Let $f : Y \to T'$ be an one-to-one function such that each object $y \in Y$ is assigned to a tuple $t' \in T'$ $(f(x) = f(y)) \Rightarrow (x = y)$. The function f assigns each object to a *unique* tuple.

Let $D' = f[S]$ and $E' = f[C]$ be the images of S and C under the function f, respectively. D' represents a database of tuples as learning examples. $E' \cap D'$ is the set of positive examples of the class C, and $D' - E'$ is the set of negative examples. Throughout this paper, we will call an object $y \in S$ a *sample*, and call its value $f(y) = t' \in T'$ an *example tuple*.

Example 1. A bank has invested in a large number of Canadian manufacturing companies in the past as shown in Table 1. Some investments produced good returns and some didn't. The manager would like to study the investment history, taking investments with good returns as positive examples and poor returns as

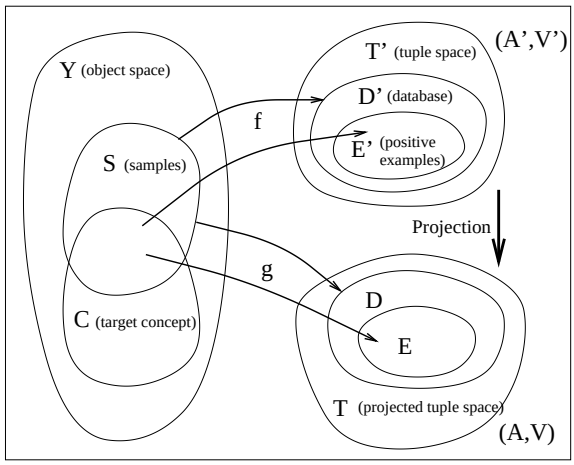

Figure 1. A model for knowledge discovery in databases

negative examples, so that prediction of good candidates for investment and poor ones can be improved in the future.

CompName	*Field*	*Location*	*Years*	*Rating*
MicroCom	*Computer*	*Vancouver*	*10>*	*Good*
CompGiant	*Computer*	*Burnaby*	*6-10*	*Good*
AutoContr	*Computer*	*Vancouver*	*10>*	*Good*
ToyHome	*Toy*	*Vancouver*	*10>*	*Good*
AllToys	*Toy*	*Windsor*	*2-5*	*Good*
Seasons	*Fashion*	*Burnaby*	*6-10*	*Good*
HomeWare	*Appliance*	*Windsor*	*2-5*	*Good*
FineDress	*Fashion*	*Vancouver*	*10>*	*Good*
NewWear	*Fashion*	*Windsor*	*2-5*	*Good*
Suit&Tie	*Fashion*	*Windsor*	*2-5*	*Poor*
HomeUse	*Appliance*	*Burnaby*	*6-10*	*Poor*
ToyManu	*Toy*	*Vancouver*	*10>*	*Poor*
NewToy	*Toy*	*Vancouver*	*10>*	*Poor*

Table 1. An example database on manufacturing companies invested in in the past.

3. SUPERFLUOUS ATTRIBUTES

Given positive examples $D' \cap E'$ and negative examples $D' - E'$, our task is to generate a classification rule for C such that, given the representation $t' \in T'$ of an object $y \in Y$, the membership of y in C can be determined with the minimal error (which will be referred to as the *classification error*).

We could describe the rule based on the entire set A' of attributes. However, for many practical reasons, we often use only a subset of attributes of A' to generate the classification rule. One reason is because we know some attributes have no dependence relation with the class to be described. For example, social insurance numbers do not help to distinguish a patient with tuberculosis from one without. Another reason may be because it is not appropriate to use some attributes in the classification task. For example, to exact a rule for selecting good employees, we may not want to include attributes sex and race in the rule. This restriction of attributes can be easily performed by using a *projection* operation in relational databases. The option of specifying a desired set of attributes should be given to the user of a learning system. The removal of a subset B of attributes represents a *conceptual bias* (Genesereth 1987) of the learning process.

We formalize the projection as follows: Let $B \subset A'$ be a proper subset of A'. A *projection* of the function f to the set of attributes $A = A' - B$ is defined as $g : U \rightarrow T$ where T is the Cartesian product $T = \times_i V_i$ and V_i is the domain of $A_i \in A$. We use $t = g(u)$ to denote the projected tuple obtained under g. Note that, different objects in Y can be mapped into the same tuple by g.

Example 2. Let the tuples in Example 1 be projected to

$$A = \{Field, Location, Years\}.$$

Let the set D of projected tuples be the universe. Let the equivalence relation R be $R(A)$. Then $(D, R(A))$ is the approximation space. The tuple $(Computer, Vancouver, 10 >)$ represents an elementary set.

Let E be the set of tuples with good returns. Then

$$\begin{aligned} \underline{Apr}(E) = \quad & \{(Computer, Van, 10 >), (Computer, Bur, 6-10), \\ & (Toy, Win, 2-5), (Fashion, Van, 10 >), \\ & (Fashion, Bur, 6-10), (Appliance, Win, 2-5)\} \\ Bnd(E) = \quad & \{(Toy, Van, 10 >), (Fashion, Win, 2-5)\} \end{aligned}$$

After the attributes have been restricted to a subset in the manner described above, it is often possible to further reduce the subset without increasing the classification error of the resultant rule. This involves the notion of *reduct* (Pawlak 1992, Ziarko 1991) in the rough set theory.

Let g be the projection of f, $D = g[S]$ be the set of learning examples described by the set of attributes $A = A' - B$, and $E = g[C] \cap D$ be the set of positive training examples described by A. Following the notation in Section 2.1, let D be the universe, and let the equivalence relation $R(A)$ be defined as follows: $(r, t) \in R(A)$ iff for every $\Lambda \in A$, $r_\Lambda = t_\Lambda$ where r_Λ is the value of the attribute Λ in the tuple r. An attribute $\Lambda \in A$ is *superfluous* in A if $R(A) = R(A - \{\Lambda\})$;

otherwise Λ is *indispensable* in A. If all attributes of A are indispensable in A, then A is *orthogonal*. A proper subset $W \subset A$ is a *reduct* of A iff W is orthogonal and $R(W) = R(A)$.

Since a reduct does not change the equivalence relation R, given a set $X \subseteq D$, none of $\overline{Apr}(X)$, $\underline{Apr}(X)$, or $Bnd(X)$ will change. This implies that the accuracy of classification relative to X does not change if we use a reduct as the basis set. The advantage of using a reduct rather than the original set A of attributes is that we have a more concise classification rule.

	Computer	Toy	Fashion	Appliance
Van.	MicroCom+ AutoContr+	ToyHome+ ToyManu- NewToy-	FineDress+	
Bur.	CompGiant+		Seasons+	HomeUse-
Win.		AllToys+	NewWear+ Suit&Tie-	HomeWare+

TABLE 2. Elementary sets with reduct $A_1 = \{Field, Location\}$.

Example 3. In the investment example, given a set of attributes $A = \{Field, Location, Years\}$ *in which a user is interested, there are two reducts of* A: $A_1 = \{Field, Location\}$ *and* $A_2 = \{Field, Years\}$. *Table 2 shows* $(D, R(A_1))$ *where each cell corresponds to an elementary set. Since tuples in an elementary set have identical attribute values, each of them is labeled instead using its company name. The rating is indicated by* $+$ *or* $-$.

Since given a set of examples D and the set A of attributes, there may exist more than one reduct, it would be useful for a learning system to provide several reducts to the user, and to proceed with the subsequent learning task using the reduct selected by the user. The user may select the reduct with minimum cardinality among the given alternatives, or the one which makes the most sense. We propose a model for the user interaction with a learning system as shown in Figure 2. The system interacts with a user in the order from left to right.

To compute a reduct, we remove an arbitrarily chosen attribute, say A_1, from the basis set A. We then check if all the elementary sets are unchanged. If so, we proceed with A_2, otherwise, we put A_1 back and proceed with A_2. We go through all the attributes in A in this fashion. The attributes left at the end constitute a reduct.

Although a single reduct can be computed relatively easily, the general problem of finding all reducts is NP-hard (Wong 1985, Ziarko 1991). In practice, if the cardinality of A is large, some compromise has to be made such that only a subset of all reducts is supplied.

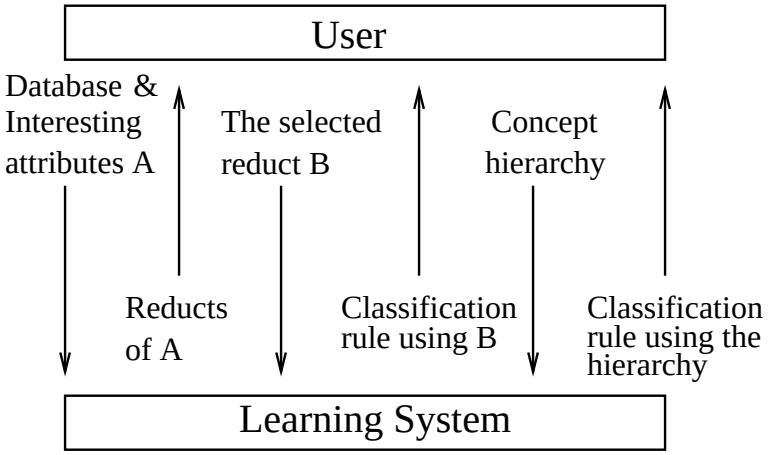

FIGURE 2. A model of learning systems

4. ERROR PROBABILITY ESTIMATION IN INDUCTION USING A REDUCT

Given the set D of examples and the set $E \subset D$ of positive examples, D can be partitioned into $Neg(E) = D - \overline{Apr}(E)$ which is a set of negative examples, $Pos(E) = \underline{Apr}(E)$ which is a set of positive examples, and $Bnd(E)$ which is a set of mixed positive and negative examples. In the following discussion, we will omit the variable E for brevity, and simply write Neg, Pos and Bnd.

Since all tuples in an elementary set are indistinguishable, we shall use $s[t]$ to denote the elementary set of a tuple t.

Throughout this paper, we assume that the set of examples D is truly a representative of the class C in the universe.

Assumption 1. Let Y be a domain and let D be a set of examples. Let the number of positive examples in an elementary set $s[t]$ be $n_+[t] \geq 0$ and the number of negative examples be $n_-[t] \geq 0$. The examples in D satisfy the following properties:

1. **Completeness** For every object $x \in Y$, there is a sample $y \in Y$ such that $g(x) = g(y)$.
2. **Proportion** For every elementary set $s[t]$ in D and every object $y \in Y$,
$$p(c(y) = 1 | g(y) = t) = n_+[t]/(n_+[t] + n_-[t]).$$
3. **Miniworld** For every elementary set $s[t]$ in D and every object $y \in Y$,
$$p(g(y) = t) = (n_+[t] + n_-[t])/Card(D).$$

Example 4. The existence of a company (PCMaker, Computer, Vancouver, 10>) is consistent with the Completeness *assumption. But the existence of a company (PCMaker, Computer, Windsor, 2-5) would be inconsistent with the assumption.*

The existence of nonequal number of good-return fashion companies and poor return fashion companies in Windsor would be inconsistent with the Proportion *assumption.*

If the number of computer manufacturing companies in Vancouver with 10 years or longer history is more than $2/13$ *of the total number of Canadian manufacturing companies, then the database would be inconsistent with the* Miniworld *assumption.*

The above representative assumption on examples is simply one way to formalize the assumption on samples in any inductive learning. The assumption allows us to associate the knowledge discovered with some estimation of error probability, to be presented in the rest of this section. The assumption cannot be verified given the limited data at hand, and may be invalidated when new data become available. We acknowledge the limited validity of the assumption. In the case where it is indeed invalidated by the new data, we again make the assumption on the updated set of examples, and update our previously discovered knowledge as well as our estimation of error probabilities.

In the following, we construct a classification rule for the class C expressed in terms of a disjunctive normal form:

$$\forall y \in Y((class(y) = 1) \Longleftrightarrow (t_1 \vee \ldots \vee t_n)),$$

where each $t_i \in D$ is a tuple (conjunct) corresponding to an elementary set, and $class(y) = 1$ means that the object y is classified by the rule as a member of the class C.

Proposition 1. Let t be a tuple in an elementary set $s[t] \subset Pos$. If $g(y) = t$ for $y \in Y$, then $c(y) = 1$.

Proof:

Suppose $g(y) = t$, and $c(y) = 0$ which means $y \notin C$. Since $g(y) = t$, $s[t]$ must be either a subset of Neg or Bnd. This contradicts the assumption $s[t] \subset Pos$. □

Based on Proposition 1, for each t such that $s[t] \subset Pos$, we include t as a conjunct in the classification rule. We label the conjunct with an error probability:

$$p(c(y) \neq 1 | g(y) = t) = 0$$

which means that if the new tuple $g(y)$ matches t, we can conclude $c(y) = 1$ with certainty due to the completeness assumption.

For each elementary set $s[t] \subset Bnd$, we include t as a conjunct in the classification rule if $n_+[t] \geq n_-[t]$. We label it with an error probability:

$$p(c(y) \neq 1 | g(y) = t) = n_-[t]/(n_+[t] + n_-[t]) \quad (1)$$

which means that if the new tuple $g(y)$ matches t, we can conclude $c(y) = 1$ with probability $1 - p(c(y) \neq 1 | g(y) = t)$. This is justified by the following Proposition.

Proposition 2. Let t be a tuple such that $s[t] \subset Bnd$ and $n_+[t] \geq n_-[t])$. If $g(y) = t$ for $y \in Y$, then

$$p(c(y) = 1 | g(y) = t) = n_+[t]/(n_+[t] + n_-[t]) \geq p(c(y) = 0 | g(y) = t)$$

Proof:

By the proportion assumption and the given condition, $p(c(y) = 1|g(y) = t) = n_+[t]/(n_+[t] + n_-[t]) \geq 0.5$. Therefore, $p(c(y) = 0|g(y) = t) = 1 - p(c(y) = 1|g(y) = t) = n_-[t]/(n_+[t] + n_-[t]) \leq 0.5$. □

The above proposition states that, to minimize the chance of error, if $n_+[t] \geq n_-[t]$, we should conclude $class(y) = 1$; otherwise conclude $class(y) = 0$ (by not firing the rule).

The error probability of the classification rule as a whole is determined by the following Theorem based on Assumption 1, and Proposition 1 and 2.

Theorem 1. The probability of false-positive error of the classification rule

$$\forall y \in Y((class(y) = 1) \Longleftrightarrow (\bigvee t))$$

for a given y is

$$p(class(y) = 1 \wedge c(y) = 0) = (\sum_{t \in Bnd(E) \wedge n_+[t] \geq n_-[t]} n_-[t])/Card(D).$$

The probability of false-negative error of the classification rule for a given y is

$$p(class(y) = 0 \wedge c(y) = 1) = (\sum_{t \in Bnd(E) \wedge n_+[t] < n_-[t]} n_+[t])/Card(D).$$

The error probability of the entire classification rule for a given y is

$$\begin{aligned} & p(class(y) \neq c(y)) \\ = \; & p(class(y) = 1 \wedge c(y) = 0) + p(class(y) = 0 \wedge c(y) = 1) \\ = \; & \sum_{t \in Bnd(E)} min(n_+[t], n_-[t])/Card(D)). \end{aligned}$$

Proof:

We prove the result for the false-positive error probability. The proof for the false-negative result is similar. The rest of the proof is trivial.

The false positive error happens when an object y $(c(y) = 0)$ has its tuple representation $g(y) = t$ $(t \in Bnd)$, and the classification rule contains t (because $n_+[t] \geq n_-[t]$). Given such a t, Proposition 2 and Equation 1 dictates that

$$p(c(y) = 0 | g(y) = t) = n_-[t]/(n_+[t] + n_-[t]).$$

The false positive error of the entire rule is the following weighted sum over every such t:

$$\sum_{t \in Bnd \wedge n_+[t] \geq n_-[t]} p(c(y) = 0 | g(y) = t) p(g(y) = t)$$

By the miniworld assumption,

$$p(g(y) = t) = (n_+[t] + n_-[t])/Card(D)$$

and the result for the false positive error probability follows. □

The discussion to this point leads naturally to the procedure for constructing a classification rule as realized in Algorithm 1.

Algorithm 1.

Input: A set $Z \subset D$ of distinct tuples, where D is the set of all examples.
Output: A classification rule in DNF.
(Notations follow that used in the text)

BEGIN
  Initialize $List$ to empty list
  FOR each $t \in Z$ such that $s[t] \subset Pos$ DO
    Label t with $p(c(y) \neq 1 | g(y) = t) = 0$
    Add t with its label to $List$
  END FOR
  FOR each $t \in Z$ such that $s[t] \subset Bnd$ and $n_+[t] \geq n_-[t]$ DO
    Label t with $p(c(y) \neq 1 | g(y) = t) = n_-[t]/(n_+[t] + n_-[t])$
    Add t with its label to $List$
  END FOR
  Construct the classification rule
    $\forall y \in Y((class(y) = 1) \Leftrightarrow (g(y) = t_1 \vee \ldots \vee g(y) = t_n))$
    where $t_i, \ldots, t_n$ are all the tuples in $List$
  Label the rule with $p(class(y) \neq c(y))$ as determined by Theorem 1
END

Example 5. Applying Algorithm 1 to the bank database in Example 1, we obtain the following rule: A Canadian manufacturing company y should be considered as a good candidate for investment if

$$\begin{aligned}
g(y) &= (Computer, Vancouver)[0] \text{ or} \\
g(y) &= (Computer, Burnaby)[0] \text{ or} \\
g(y) &= (Toy, Windsor)[0] \text{ or} \\
g(y) &= (Fashion, Vancouver)[0] \text{ or} \\
g(y) &= (Fashion, Burnaby)[0] \text{ or} \\
g(y) &= (Appliance, Windsor)[0] \text{ or} \\
g(y) &= (Fashion, Windsor)[0.5].
\end{aligned}$$

The rule as a whole is subject to a false prediction with a chance of $1/13 + 1/13 = 15\%$.

The error probability used to label the individual conjunct in a rule can be used for *posterior* decision-making. For example, the bank should be quite confident with an investment if the candidate is a computer manufacturer located in Vancouver. However, in evaluating an investment to a Windsor Fashion company, the bank knows that the chance of getting a good return is as likely as that of getting a poor one.

The overall error probability labeling the entire rule can be used for *prior* decision-making. Suppose another learning algorithm produced a rule with overall error probability of 0.19. If a decision must be made regarding which rule should be used for an automatic investment system, then the previous rule (with an error probability 0.15) would be preferred.

The overall error probability may also be used to limit the number of conjuncts contained in a rule. We discuss this issue in the next section.

5. RESTRICTING THE NUMBER OF CONJUNCTS

In order to increase the efficiency of a classification rule (less space to store and less time to apply), the user may impose a restriction on the number of conjuncts, subject to the minimization of error probability. To meet such a requirement, we can rank the conjuncts in a rule by their contribution to the overall error probability.

The removal of a conjunct from a rule increases only the false-positive error but not the false-negative error. In particular, the removal of a conjunct t_i will add n_{+i} to and subtract n_{-i} from the numerator of the error probability, where n_{+i} and n_{-i} are the number of positive examples and negative examples in $s[t_i]$, respectively. That is, the net increase to the numerator is $n_{+i} - n_{-i}$. Therefore, to find a conjunct whose removal causes minimal increase of error probability, we need only to select the one with minimal $n_{+i} - n_{-i}$. These observations lead to the following Algorithm and Proposition.

Algorithm 2 (SortConjuncts)

Input: A list I *of* m *conjuncts* $t_1, \ldots, t_m$*, the number of positive examples in each elementary set* $n_{+1}, \ldots, n_{+m}$*, and the number of negative examples in each elementary set* $n_{-1}, \ldots, n_{-m}$.

Output: The list O *of conjuncts such that the removal of a conjunct from the end of the list causes minimal increase of error probability.*

BEGIN
    Initialize O *to an empty list*
    WHILE $I \neq \phi$ *DO*
        Find t *in* I *with maximal* $n_+ - n_-$
        Remove t *from* I *and place it at the end of* O
    END WHILE
END

Proposition 3. Given a list of n conjuncts in a classification rule, and $m < n$ as an additional restriction on the number of conjuncts, Algorithm 2 sorts n conjuncts such that retaining the first m conjuncts in the list minimizes the increase of error probability.

Example 6. Table 3 shows a sorted list of the conjuncts/elementary sets for the rule in Example 5.

Elementary Set	$n_+ - n_-$
(Computer, Vancouver)	*(2)*
(Computer, Burnaby)	*(1)*
(Toy, Windsor)	*(1)*
(Fashion, Vancouver)	*(1)*
(Fashion, Burnaby)	*(1)*
(Appliance, Windsor)	*(1)*
(Fashion, Windsor)	*(0)*

TABLE 3. A sorted list of elementary sets

6. REMOVAL OF EXCEPTIONS

In many applications, the general principle is associated with some exceptions. For example, a bird is one that flies, but penguin is an exception to the 'flying' principle. Each exception will necessarily produce a conjunct in the classification rule. This example would produce a rule with two degenerated conjuncts: x is a bird iff (1) x flies or (2) x is a penguin. Sometimes we would like to remove such exceptions from the rule. Removing exceptions entails relaxing the error probability. The task presented to the learning system is then as follows: Given a threshold for error probability, find the minimal subset of conjuncts that satisfies the threshold.

The sorted list of conjuncts by Algorithm 2 can be used for this task. We may remove as many conjuncts as necessary subject to the threshold. At each step we remove the conjunct whose removal causes minimal increase of the overall error probability. The last conjunct in the sorted list is such a conjunct.

Algorithm 3 (RemoveExceptions)

Input: A list I of m conjuncts $t_1, \ldots, t_m$ sorted by Algorithm 2, and an error probability threshold P.

Output: The list O which contains the minimal set of conjuncts such that the error probability of the rule constructed from O is less than or equal to P.

BEGIN

Initialize p to the error probability of the rule obtained from I

If $p > P$, print an error message and exit
Initialize Last to null
Initialize O to I
WHILE $p \leq P$ DO
Remove Last from O
Assign the last conjunct in O to Last
Compute the error probability p of the rule
obtained from $O - \{Last\}$
END WHILE
END

7. INDUCTION BY CONCEPT HIERARCHY

One of the characteristics of knowledge discovery in databases is that the discovered knowledge is represented in a high-level language [3]. This aspect of knowledge discovery is different from learning in neural networks.

We consider here an externally provided generalization hierarchy [6] in which different levels of generalization are organized into a tree called a *concept tree* (Cai 1991).

Definition 1. Let Λ be an attribute with domain Δ. Let Γ be a rooted balanced [1] directed tree. Γ is a **concept tree** for the attribute Λ if the following conditions hold:

1. The leaves of Γ are labeled by the elements in Δ.
2. The set of leaves are partitioned and leaves in each partition are connected to a common parent node labeled by the partition.
3. The parent nodes of leaves are further partitioned and nodes in each partition are connected to a common parent node labeled by the partition. This process continues until all nodes at a level form a single partition. Their common parent, the root, is labeled 'any'.

In a concept tree, each node is identified with a unique label. Thus we will use terms *node* and *label* interchangeably. Figure 3 gives an example of a concept tree for an attribute *Location*.

In induction by concept hierarchy, we consider the issue of substituting the attribute values of some tuples by a more general concept: one of their ancestor labels in the concept tree. This will extend the domain of each attribute to include all concepts in the corresponding concept tree. Note that the values in the *extended domain* will no longer be exclusive any more. We call a tuple resulting from such a substitution a *generalized tuple*.

Definition 2. Let r and t be two tuples. Let Λ be an attribute. Tuples r and t are **incompatible** at only Λ if all corresponding attribute values of r and t are identical except the values for Λ.

[1] Notice that we consider here balanced trees whose leaves have identical height.

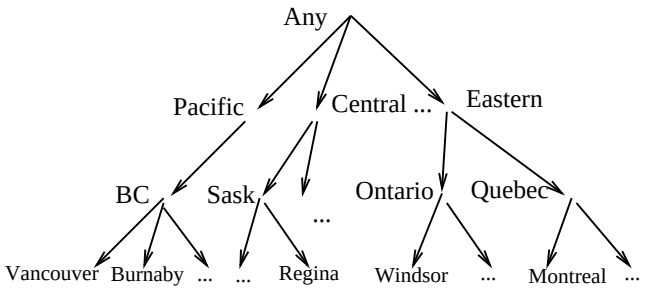

FIGURE 3. A concept tree for an attribute *Location*

For example, for a tuple schema *(Field, Location, Years)* representing a manufacturing company's field, location and years in business, the tuples *(Computer, Vancouver, >10)* and *(Toy, Vancouver, >10)* are incompatible at only *field*. We can extend Definition 2 to involve more than two tuples and to involve more than one attribute:

Definition 3. Let L be a set of tuples. Let $\Omega \subset A$ be a proper subset of attributes. Tuples in L are **incompatible** at Ω if all the corresponding attribute values of the tuples are identical except the values for attributes in Ω.

Suppose r and t are incompatible at only Λ, and their corresponding values are r_Λ and t_Λ. If we substitute r_Λ and t_Λ by one of their common ancestors in the concept tree, then we have replaced the two tuples by a new tuple u. In general, suppose r and t are incompatible at Ω. If we substitute the values for each attribute in Ω in r and t by a common ancestor in the corresponding concept tree, then we have replaced the two elementary sets $s[r]$ and $s[t]$ by a new elementary set $s[u]$.

Definition 4. Let r and t be two tuples that are incompatible at Ω. For each attribute $\Lambda \in \Omega$, let Γ be the concept tree of Λ and let α be a common ancestor of the attribute values r_Λ and t_Λ in Γ. The substitution of r_Λ and t_Λ with α for every $\Lambda \in \Omega$, is a **generalization** of r and t relative to Ω.

Since the generalization implies the substitution of the two tuples r and t by a new tuple, we shall call such a tuple substitution a *generalization* as well without confusion. The above definition of generalization can be extended to the generalization of a set L of tuples relative to Ω:

Definition 5. Let L be a set of tuples that are incompatible at Ω. For each attribute $\Lambda \in \Omega$, let Γ be the concept tree of Λ, let t_Λ be the value for Λ of a tuple $t \in L$, and let α be a common ancestor of t_Λ in Γ for every $t \in L$. The substitution of t_Λ with α in every $t \in L$ is a **generalization** of tuples in L relative to Ω.

In the following, we consider the effect of a generalization of a pair of tuples r and t on the overall error probability of the resultant rule. According to Theorem 1, the contribution of tuples r and t to the error probability of the original rule is

$$Contri_{r,t} = (min(n_+[t], n_-[t]) + min(n_+[r], n_-[r]))/Card(D).$$

The contribution of the new tuple u is

$$Contri_u = min(n_+[t] + n_+[r], n_-[t] + n_-[r])/Card(D).$$

The relationship between the two elementary sets $s[r]$ and $s[t]$, and the three sets Pos, Neg and Bnd can reflect one of the following six cases:

1. $s[r] \subset Neg$ and $s[t] \subset Neg$
2. $s[r] \subset Pos$ and $s[t] \subset Pos$
3. $s[r] \subset Bnd$ and $s[t] \subset Bnd$
4. $s[r] \subset Pos$ and $s[t] \subset Neg$
5. $s[r] \subset Pos$ and $s[t] \subset Bnd$
6. $s[r] \subset Neg$ and $s[t] \subset Bnd$

In cases 1 and 2, $Contri_{r,t} = Contri_u = 0$. No additional error is introduced to the new rule.

In case 3, if either $n_+[x] \geq n_-[x]$ for both $x = r$ and $x = t$, or $n_+[x] < n_-[x]$ for both $x = r$ and $x = t$, or $n_+[r] = n_-[r]$ and $n_+[t] < n_-[t]$, then $Contri_{r,t} = Contri_u$ and no additional error is introduced. However, if $n_+[r] > n_-[r]$ and $n_+[t] < n_-[t]$, then

$$Contri_u = min(n_+[t] + n_+[r], n_-[t] + n_-[r])/Card(D)$$

$$> \ Contri_{r,t} = (n_+[t] + n_-[r])/Card(D).$$

In cases 4, 5 and 6, the new tuple u causes reconfiguration of sets Pos, Neg and Bnd into Pos', Neg' and Bnd', and the elementary set $s[u]$ is a subset of Bnd'. In case 4,

$$Contri_u = min(n_+[r], n_-[t])/Card(D) \ > \ Contri_{r,t} = 0.$$

In cases 5, $Contri_u = min(n_+[t] + n_+[r], n_-[t])/Card(D)$

$$\geq \ Contri_{r,t} = min(n_+[t], n_-[t])/Card(D).$$

The equal sign holds iff $n_+[t] \geq n_-[t]$.

In case 6, $Contri_u = min(n_+[t], n_-[t] + n_-[r])/Card(D)$

$$\geq \ Contri_{r,t} = min(n_+[t], n_-[t])/Card(D).$$

The equal sign holds iff $n_+[t] \leq n_-[t]$.

We summarize the above analysis by the following definition 6 and Theorem 2:

Definition 6. Let $\Omega \subset A$ be a proper subset of attributes. Let L be a set of tuples incompatible at Ω. Let u be the tuple obtained by a generalization of tuples in L relative to Ω. The substitution of u for the tuples in L is a **proper generalization** if one of the followings holds:

- $(\bigcup_{t \in L} s[t]) \subset Neg$

- $(\bigcup_{t\in L} s[t]) \subset Pos$
- $(\bigcup_{t\in L} s[t]) \subset Bnd$, and there exist no two tuples $r,\ t \in L$ such that for one of $r,\ t,\ n_+ > n_-$, and for the other, $n_+ < n_-$.
- $(\bigcup_{t\in L} s[t]) \subset Pos \cup Bnd$, and $n_+[t] \geq n_-[t]$ for each $s[t] \subset Bnd$.
- $(\bigcup_{t\in L} s[t]) \subset Neg \cup Bnd$, and $n_+[t] \leq n_-[t]$ for each $s[t] \subset Bnd$.

Otherwise, the substitution is a **strict overgeneralization**.

Theorem 2. Let $\Omega \subset A$ be a proper subset of attributes. Let L be a set of tuples incompatible at Ω. Let u be the tuple obtained by a generalization of tuples in L relative to Ω.

The generalization does not increase the error probability of the resultant new rule iff it is proper. Otherwise, the amount of increase in the error probability is

$$(min(\sum_{t\in L} n_+[t], \sum_{t\in L} n_-[t]))/Card(D) - (\sum_{t\in L} min(n_+[t], n_-[t]))/Card(D) > 0.$$

Example 7. For the rule constructed in Example 5, the substitution of (Computer, Vancouver) *and* (Computer, Burnaby) *with* (Computer, BC) *is a proper generalization, so is the substitution with* (Computer, Any), *since both elementary sets are elements of Pos.*

However, the substitution of (Toy, Vancouver), *which is an element of Bnd, and* (Toy, Windsor), *which is an element of Pos, with* (Toy, Any) *is a strict overgeneralization. It causes an increase in the error probability of the resultant rule by* $2/13 - 1/13 = 0.077$.

Theorem 2 provides a guideline in the design of knowledge discovery algorithms.

8. REMARKS

In this paper, we apply rough set theory and probability concepts to inductive learning from databases. Under the assumption of a set of representative examples, we derive the error probabilities for components of a classification rule as well as for the rule as a whole. We derive the result for induction using attributes, and show the condition under which further induction can be performed without increasing the error probabilities.

Our work is closely related to other work in the area of knowledge discovery in databases: Ziarko (1991) discussed the application of reducts in inductive learning in databases. Cai, Cercone and Han (1991) and Han, Cai, and Cercone (1993) developed an attribute-oriented approach for inductive learning in databases using concept trees, and implemented in an experimental database learning system, DBLEARN. Our work extends their research and provides a theoretical framework to quantify the uncertainty in the knowledge discovered from databases. Pawlak, Wong and Ziarko (1988) discussed similar decision rules. However, they did not provide the error probabilities explicitly.

This work also relates to learning by examples in machine learning (Kodratoff 1990, Michalski 1986, Michalski 1983). We compare here with two well-known

methods. The decision tree approach (ID3) (Quinlan 1983) generates a decision procedure instead of a rule. It was extended later (C4) (Quinlan 1987) to handle noisy data by associating each leaf with an error probability similar to the error probability attached to each conjunct in our work. No error probability for the entire decision tree is estimated even though it could be added with an analysis similar to ours. It does not address the issue involved in induction by hierarchy. AQ11 (Michalski 1980) generates DNF rules in an incremental fashion from a preselected set of examples. It does not handle noisy data directly, and does not consider induction by hierarchy either (Forsyth 1986).

ACKNOWLEDGEMENTS

The authors are members of the Institute for Robotics and Intelligent Systems (IRIS) and wish to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, the Natural Sciences and Engineering Research Council, and the participation of PRECARN Associates Inc. We are grateful to the Canadian Cable Labs Fund for their financial assistance.

REFERENCES

CAI, Y., CERCONE, N., and HAN, J. 1991. Learning in relational databases: an attribute-oriented approach. *Computational Intelligence*, 7:119–132.

FORSYTH, R. and RADA, R. 1986. *Machine Learning: applications in expert systems and information retrieval.* Ellis Horwood.

FRAWLEY, W.J., PIATETSKY-SHAPIRO, G., and MATHEUS, C.J. 1991. Knowledge discovery in databases: An overview. In G. Piatetsky-Sapiro and W.J. Frawley, editors, *Knowledge Discovery in Databases*, AAAI/MIT, 1–27.

GENESERETH, M.R., and NILSSON, N.J. 1987. *Logical Foundations of Artificial Intelligence.* Morgan Kaufmann.

HAN, J., CAI, Y., and CERCONE, N. 1993. Data-driven discovery of quantitative rules in relational databases. *IEEE Trans. on Knowledge and Data Engineering*, 5(1):29–40.

HAUSSLER, D. 1987. Bias, version spaces and valiant's learning framework. In *Proc. 4th International Workshop on Machine Learning*, Irvine, CA, 324–336.

KODRATOFF, Y., and MICHALSKI, R.S., (editors) 1990. *Machine Learning: An Artificial Intelligence Approach*, volume 3. Morgan Kaufmann.

MICHALSKI, R.S., and CHILAUSKY, R.J. 1980. Learning by being told and learning from examples. *Journal of Policy Analysis & Information Systems*, 4.

MICHALSKI, R.S., CARBONELL, J.G., and MITCHELL, T.M., (editors) 1983. *Machine Learning: An Artificial Intelligence Approach*, volume 1. Tioga, 1983.

MICHALSKI, R.S., CARBONELL, J.G., and MITCHELL, T.M., (editors) 1986. *Machine Learning: An Artificial Intelligence Approach*, volume 2. Morgan Kaufmann.

PAWLAK, Z. 1982. Rough sets. *International Journal of Computer and Information Sciences*, 5:341–356.

PAWLAK, Z. 1992. Anatomy of conflicts. ICS Research Report 11/92, Warsaw University of Technology, Warsaw.

Pawlak, Z., Wong, S.K.M., and Ziarko, W. 1988. Rough sets: probabilistic versus deterministic approach. *International Journal of Man-Machine Studies*, 29:81–95.

Quinlan, J.R. 1983. Learning efficient classification procedures and their application to chess end games. In *Machine Learning: An Artificial Intelligence Approach, Vol.1*, Morgan Kaufmann, 463–482.

Quinlan, J.R. 1987. Decision trees as probabilistic classifiers. In *Proc. 4th International Workshop on Machine Learning*, Irvine, CA, 31–37.

Wong, S.K.M. and Ziarko, W. 1985. On optimal decision rules in decision tables. *Bulletin of Polish Academy of Sciences*, (11-12):693–696.

Ziarko, W. 1991. The discovery, analysis, and representation of data dependencies in database. In G. Piatetsky-Sapiro and W.J. Frawley, editors, *Knowledge Discovery in Databases*, AAAI/MIT, 195–209.