Quantifying Uncertainty of Knowledge Discovered From Databases

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

Abstract

This paper focuses on the application of 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 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 [2].

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' in AI, we may regard a database as a set of training examples. The objective 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 selected attributes a *basis set* [3], and call the learning process *induction using attributes*. The rule from induction using attributes can be further generalized using a concept hierarchy for each individual attribute. The rule is then represented in a higher level language and is more compact. We shall call this generalization process *induction by hierarchy*.

Given a basis set A' of attributes, a user my 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 minimal subset may not be unique. We will discuss a design guideline based on rough sets [6] to provide users with difference options.

Once a rule is generated by a learning system, a user may wish to know how reliable the rule is. 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 and thus increase the classification error of the resultant rule. In this paper, we derive a condition for induction by hierarchy, which guarantees that no additional error is introduced.

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 such 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 corresponds 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 [6] which provides a sound 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. Equivalence classes of the relation R is called *elementary sets* 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

Let Y be a domain of objects. Let $C \subset Y$ be a class of objects. We define a function $c : Y \rightarrow \{0, 1\}$ 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 $f : Y \rightarrow T'$ 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. This is the case in databases.

Let $D' = f[S]$ and $E' = f[C]$ be the images of S and C under 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.*

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 minimal 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 of attributes represents a *conceptual bias* [3] of the learning process.

We formalize the projection in the following: 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 $T = V_{j_1} \times \ldots \times V_{j_m}$ and V_{j_i} is the domain of $A_{j_i} \in A$. We use $t = g(u)$ to denote the projected tuple obtained under g. Note that, unlike f, g is not one-to-one. Multiple objects in Y can be mapped into the same tuple by g.

After the attributes have been restricted to a subset in the above manner, it is often possible to further reduce the subset without increasing the classification error of the resultant rule. This involves the notion of *reduct* [8, 11] 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 notion 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 subset $W \subseteq A$ 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 $\underline{Apr}(X)$, $\overline{Apr}(X)$, 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.

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 all the 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, or the one which makes the most sense to him.

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.

However, although a single reduct can be computed relatively easily, the general problem of finding all reducts is NP-hard [10, 11].

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 of interests.

Assumption 4.1 *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,*

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

3. **Miniworld** *For every elementary set $s[t]$ in D,*

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

where $p(g(y) = t)$ is the probability that a randomly chosen y from Y is represented by t.

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

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

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 4.1 *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 $s[t] \subset Pos$. $\Box$

Based on Proposition 4.1, 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.

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:

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

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 4.2 *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 Assumption 4.1 and the given condition, $p(c(y) = 1|g(y) = t) = n_+[t]/(n_+[t] + n_-[t]) \geq 0.5$. Therefore, $p(c(y) = 0|g(y) = t) = 1 - p(c(y) = 1|g(y) = t) = n_-[t]/(n_+[t] + n_-[t]) \leq 0.5$. $\Box$

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 Proposition.

Proposition 4.3 *The probability of false-positive error of the classification rule is*

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

The probability of false-negative error of the classification rule is

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

The error probability of the classification rule as a whole is

$$p(class(y) \neq c(y)) = p(class(y) = 1 \wedge c(y) = 0) + p(class(y) = 0 \wedge c(y) = 1).$$

We summarize the above discussion by the following Algorithm for the construction of a classification rule.

Algorithm 4.1 (Construct)

```
    Input: A set $Z \subset D$ of distinct tuples, where $D$ is the set of all examples.
    Output: A classification rule in DNF.

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
                $\forall y \in Y((class(y) = 1) \Leftrightarrow (t_1 \vee \ldots \vee t_n))$ 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 Proposition 4.3.
END
```

The error probability used to label the individual conjunct in a rule can be used for *posterior* decision-making. For example, after a patient's symptom matches a conjunct, the error probability labelling the conjunct tells the doctor the chance of misdiagnosis.

The overall error probability labelling the entire rule can be used for *prior* decision-making. Suppose two learning systems have learned the same concept from different sets of examples possibly using different attribute descriptions. If we must commit to one of the systems for future classifications, the overall error probabilities of the two systems help us to make a choice. For example, if we are to select one of two family doctors, we would prefer the one with lower overall rate of misdiagnosis.

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

5 Restriction on 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_{-j} from the numerator of the error probability, where n_{+i} and n_{-j} 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 5.1 (SortConjuncts)

Input: A list I of 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 5.1 *Given a list of n conjuncts in a classification rule, and $m < n$ as an additional restriction on the number of conjuncts, Algorithm 5.1 sorts n conjuncts such that retaining the first m conjuncts in the list minimizes the increase of error probability.*

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. The above example would produce a rule with two conjuncts: (x is a bird) $\Longleftrightarrow$ (x flies or x is a penguin). Sometimes we would like to remove such exceptions from the rule. This entails relaxing the error probability. The task presented to the learning system is then the following: given a threshold for error probability, find the minimal subset of conjuncts that satisfies the threshold.

The sorted list of conjuncts by Algorithm 5.1 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 6.1 (RemoveExceptions)

Input: A list I of conjuncts $t_1, \ldots, t_m$ sorted by Algorithm 5.1, 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 P.

```
BEGIN
    Initialize $p$ to the error probability of the rule from $I$
    If $p \leq P$, print an error message and exit
    Initialize Last to null
    WHILE $p \geq P$ DO
        Remove Last from $I$
        Compute the error probability $p$ of the rule from $I - \{Last\}$
    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 [2]. This aspect of knowledge discovery is different from learning in neural networks.

We consider here an externally provided generalization hierarchy [5] in which different levels of generalization are organized into a tree called a *concept tree* [1].

Definition 7.1 *Let Λ be an attribute with domain Δ. Let Γ be a rooted balanced* [1] *directed tree. Γ is a* **concept tree** *for an attribute Λ if the following conditions hold:*

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

In a concept tree, each node is identified with a unique label. Thus we will use terms *node* and *label* interchangeably. Figure 1 gives an example of a concept tree for attribute Birth_Place.

In induction by concept hierarchy, we consider the issue of substituting the attribute values of some tuples by a more general concept: their ancestor label 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 7.2 *Let Λ be an attribute with its concept tree Γ. Let a and b be two labels in Γ. Labels a and b are* **compatible** *if either (1) $a = b$, or (2) one of them is a descendent of the other. Otherwise, a and b are said to be* **incompatible**.

[1]We consider here balanced trees whose leaves have identical height.

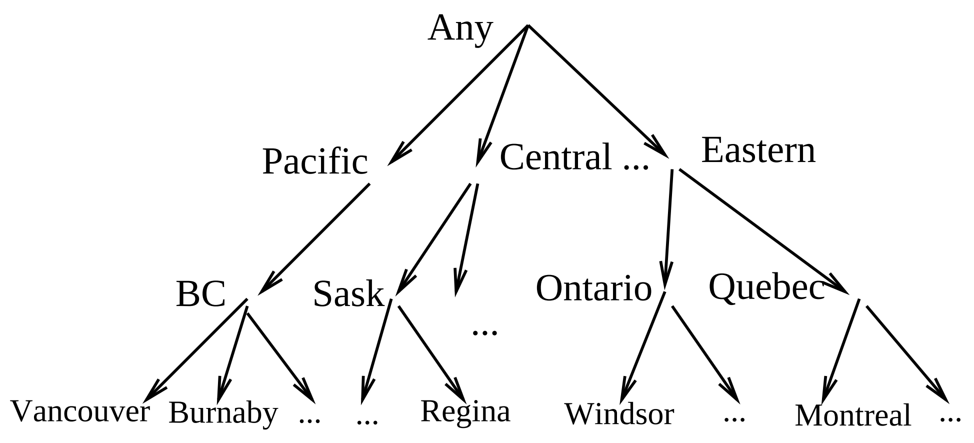

Figure 1: A concept tree for attribute BirthPlace

For example, in Figure 1, *Vancouver* and *BC* are compatible, but *Vancouver* and *Victoria* are incompatible, so are *Vancouver* and *Ontario*.

Definition 7.3 *Let r and t be two tuples. Let Λ be an attribute. Tuples r and t are* **compatible** *if all the values of corresponding attributes of r and t are compatible; r and t are* **incompatible** *at only Λ if all the values of corresponding attributes of r and t are compatible except the values for Λ.*

For example, for a tuple schema *(BirthPlace, Position)*, tuples *(Vancouver, assit-prof)* and *(BC, assit-prof)* are compatible; *(Vancouver, assit-prof)* and *(Victoria, assit-prof)* are incompatible at only *BirthPlace*.

Definition 7.4 *A set L of tuples in a database is a* **full sibling** *with respect to an attribute Λ and its corresponding concept tree Γ, if for every pair of tuples $r, t \in L$, (1) r and t are incompatible at only Λ, (2) r_Λ and t_Λ have a common parent node w in Γ, and (3) the children of w are exhausted in L. The node w is called the* **parent** *of the full sibling with respect to Λ.*

For example, for the tuple schema *(BirthPlace, Position)*, tuples *(Vancouver, assit-prof)* and *(Victoria, assit-prof)* are siblings, but *(Vancouver, assit-prof)* and *(Ontario, assit-prof)* are not siblings. Tuples *(Vancouver, assit-prof)*,$\ldots$, *(Victoria, assit-prof)* form a full sibling, but, if any one tuple is missing, the rest are no longer a full sibling.

Definition 7.5 *Let Λ be an attribute with its concept tree Γ. Let w be a label for a non-leaf node in Γ. Let L be a set of tuples. The substitution of w for the values of Λ in each $r \in L$ is a* **proper induction** *if*

1. *L is a full sibling with respect to Λ,*
2. *w is the parent of the full sibling with respect to Λ,*
3. *$L \subseteq Pos$;*

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

For example, if the positive examples for *Professor* include the following full sibling with respect to *BirthPlace*, $\{$*(Vancouver, assit-prof)*,..., *(Victoria, assit-prof)*$\}$, then the substitution of *Vancouver*, ..., *Victoria* by *BC* is a proper induction. As another example, suppose we have two tuples for classifying a preferred electronic appliance: 'Type=TV and Size=big' and 'Type=portable calculator and Size=small'. The concept tree for Size has only three nodes: a root 'any' and two leaves 'big' and 'small'. Substitution of 'big' and 'small' by 'any' for the attribute 'Size' causes a strict overgeneralization. Now 'Type=TV and Size=small' is classified as a preferred appliance which is not intended by the original example tuples.

Overgeneralization may or may not cause classification error. In the above example, if there is a small TV in our object domain, in which case, $L \cap Neg \neq \phi$, the overgeneralization will cause classification error. On the other hand, if there is no small TV in our object domain, in which case, $L \cap Neg = \phi$, the overgeneralization does not cause any error. This shows that, if we restrict to the object domain from which the samples are drawn, overgeneralization is not equivalent to the increase of error probability. However, overgeneralization *is* equivalent to the increase of error probability with an extended object domain, e.g., a domain including a small TV. A detailed discussion of this issue is beyond the scope of this paper.

Proposition 7.1 *A proper induction does not change the error probability of the resultant classification rule.*

Proof:

We refer the three conditions in Definition 7.5 as conditions 1, 2 and 3. Suppose the three conditions hold. After substitution, a new tuple z will be generated to replace the corresponding full sibling L in the classification rule. Because of conditions 1 and 2, every tuple $t \in Pos \cap L$ is compatible with z and will fire the conjunct correctly. Because of conditions 1, 2, and 3, no tuple $t \in Neg \cup Bnd$ is compatible with z, and t cannot incorrectly fire the conjunct. Since neither new false-positive nor false-negative error are introduced by the proper induction, the error probability remains the same. □

Recall that Algorithm 4.1 labels each conjunct of the rule as well as the entire rule with error probabilities. Proposition 7.1 shows that the error probability of the entire rule remains the same. By Definition 7.5, the only conjuncts that are changed during a proper induction are from *Pos*, and their error probabilities are zero. The conjuncts generalized from them have the identical zero error probability. Each conjunct from *Bnd* should not be changed and their error probabilities also remain the same.

8 Remarks

In this paper, we apply the rough set theory and probability concepts to inductive learning from databases. Under the assumption of representative set of 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 are closely related to several other work in the area of knowledge discovery in databases: Ziarko [11] discussed the application of reducts in inductive learning in databases. Cai, Cercone and Han [1] and Han, Cai, and Cercone [4] developed an attribute-oriented approach for inductive learning in databases using concept trees, and implemented in an experiment database learning system, DBLEARN. Our work extends theirs and attempts to provide a theoretical basis for knowledge discovery in databases. Pawlak, Wong and Ziarko [7] discussed similar decision rules in Section 4. However, they did not provide the error probabilities explicitly.

There are cases where the representative assumption on examples is not practical. In these cases, our estimation provides a lower bound for the error probability. We plan to implement and test our results in the next version of DBLEARN.

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 Canadian Cable Labs Fund for their financial assistance.

References

[1] Y. Cai, N. Cercone, and J. Han, "Learning in relational databases: an attribute-oriented approach", *Computational Intelligence*, 7, 119-132, 1991.

[2] W.J. Frawley, G. Piatetsky-Shapiro, and C.J. Matheus, "Knowledge discovery in databases: An overview", in *Knowledge Discovery in Databases*, eds., G. Piatetsky-Sapiro and W.J. Frawley, AAAI/MIT, 1-27, 1991.

[3] M.R. Genesereth and N.J. Nilsson, *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, 1987.

[4] J. Han, Y. Cai, and N. Cercone, "Data-driven discovery of quantitative rules in relational databases", *IEEE Trans. on Knowledge and Data Engineering*, 5 (1): 29-40, 1993.

[5] D. Haussler, "Bias, version spaces and Valiant's learning framework", *Proc. 4th International Workshop on Machine Learning*, Irvine, CA, 324-336, 1987.

[6] Z. Pawlak, "Rough sets", International Journal of Computer and Information Sciences, 11 (5): 341-356, 1982.

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

[8] Z. Pawlak, "Anatomy of conflicts", *ICS Research Report 11/92*, Warsaw University of Technology, Warsaw, May, 1992.

[9] J.R. Quinlan, "Learning efficient classification procedures and their application to chess end games", in *Machine Learning: An Artificial Intelligence Approach*, Vol.1, Morgan Kaufmann, 463-482, 1983.

[10] S.K.M. Wong and W. Ziarko, "On optimal decision rules in decision tables", *Bulletin of Polish Academy of Sciences*, 33 (11-12): 693-696, 1985.

[11] , "The discovery, analysis, and representation of data dependencies in database", in *Knowledge Discovery in Databases*, eds., G. Piatetsky-Sapiro and W.J. Frawley, AAAI/MIT, 195-209, 1991.