Using Rough Sets as Tools for Knowledge Discovery

Ning Shan, Wojciech Ziarko, Howard J. Hamilton and Nick Cercone

Department of Computer Science, University of Regina
Regina, Saskatchewan, Canada S4S 0A2
E-Mail: {ning,ziarko,hamilton,nick}@cs.uregina.ca

Abstract

An attribute-oriented rough set method for knowledge discovery in databases is described. The method is based on information generalization, which examines the data at various levels of abstraction, followed by the discovery, analysis and simplification of significant data relationships. First, an attribute-oriented concept tree ascension technique is applied to generalize the information; this step substantially reduces the overall computational cost. Then rough set techniques are applied to the generalized information system to derive rules. The rules represent data dependencies occurring in the database. We focus on discovering hidden patterns in the database rather than statistical summaries.

Introduction

Knowledge discovery in databases (KDD), in general, is "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data" (Piatetsky-Shapiro et al. 1991). The central question in knowledge discovery research is how to turn information, expressed in terms of stored data, into knowledge expressed in terms of generalized statements about characteristics of the data. Some machine learning techniques are appropriate for analyzing databases (Fayyad et al. 1993; Gemello and Mana 1989). Knowledge discovery methods based on the principles of machine learning must provide computational efficiency to deal with very large databases and robustness to cope with superfluous or "noisy" data.

The theory of rough sets (Pawlak 1991) offers a new approach to reasoning from data. This methodology, which is complementary to statistical inference, provides new insights into properties of data and has been successfully applied in knowledge acquisition, forecasting and predictive modeling, expert systems, knowledge discovery in databases (Slowinski 1992; Ziarko 1994; Ziarko and Shan 1995).

We present an approach to knowledge discovery from large relational databases. Our approach can discover different kinds of knowledge from relational databases, including *empirical keys*, *minimal relations*, and *discrimination rules*. An empirical key is a minimal subset of attributes of a relation which uniquely identifies every tuple in the relation. In contrast to the notion of a key known in database theory, the empirical key is derived from data rather than from known data dependencies and may not preserve the tuple identification property in all instances of the relation. The set of tuples of the relation projected on an empirical key is a minimized representation of the relationship encoded in the original relation. This set of tuples is called a minimal relation. A set of discrimination rules is a description of a class that distinguishes the target class from a fixed number of other classes. The approach integrates the concepts of *information generalization*, *information reduction*, and *rule generation* which significantly reduce the computational complexity of analyzing large databases. An *attribute-oriented concept tree* ascension technique is used to generalize an information system (which is either a relational database or a relation extracted by a query from a database). The generalization process is guided by domain experts. After the generalization process, a rough set method is applied to the generalized information system. The *reduction technique* introduced in rough sets is applied to generate a minimalized information system called a *reduct* which contains the minimal subset of the generalized attributes. By analyzing relationships between attributes, irrelevant attributes are removed without losing any essential information. For the generation of discrimination rules, a set of *maximally general rules* can be derived directly from the reduct. The maximally general rules embody the most general patterns within the database. The rules can be used to interpret and better understand the database.

Make_Model	Fuel	Disp	Weight	Cyl	Power	Turbo	Comp	Tran	Mileage
Ford Escort	EFI	Medium	876	6	High	yes	high	auto	medium
Dodge Shadow	EFI	Medium	1100	6	High	no	medium	manu	medium
Ford Festiva	EFI	Medium	1589	6	High	no	high	manu	medium
Chevrolrt Corvette	EFI	Medium	987	6	High	no	medium	manu	medium
Dodge Stealth	EFI	Medium	1096	6	High	no	high	manu	medium
Ford Probe	EFI	Medium	867	6	High	no	medium	manu	medium
Ford Mustang	EFI	Medium	1197	6	High	no	high	manu	medium
Dodge Daytona	EFI	Medium	798	6	High	yes	high	manu	high
Chrysler Le Baron	EFI	Medium	1056	4	Medium	no	medium	manu	medium
Dodge Sprite	EFI	Medium	1557	6	High	no	medium	manu	low
Honda Civic	2-BBL	Small	786	4	Low	no	high	manu	high
Ford Escort	2-BBL	Small	1098	4	Low	no	high	manu	medium
Ford Tempo	2-BBL	Small	1187	4	Medium	no	high	auto	medium
Toyota Corolla	EFI	Small	1023	4	Low	no	high	manu	high
Mazda 323	EFI	Medium	698	4	Medium	no	medium	manu	high
Dodge Daytona	EFI	Medium	1123	4	Medium	no	medium	manu	medium
Honda Prelude	EFI	Small	1094	4	High	yes	high	manu	high
Toyota Paseo	2-BBL	Small	1023	4	Low	no	medium	manu	high
Chevrolet Corsica	EFI	Medium	980	4	High	yes	medium	manu	medium
Chevrolet Beretta	EFI	Medium	1600	6	High	no	medium	auto	low
Chevrolet Cavalier	EFI	Medium	1002	6	High	no	medium	auto	medium
Chrysler Le Baron	EFI	Medium	1098	4	High	no	medium	auto	medium
Mazda 626	EFI	Small	1039	4	Medium	no	high	manu	high
Chervolet Corsica	EFI	Small	980	4	Medium	no	high	manu	high
Chevrolet Lumina	EFI	Small	1000	4	Medium	no	high	manu	high

Table 1: A collection of car information.

Information Generalization

Formally, an *information system* S is a quadruple $< U, A, V, f >$, where $U = \{x_1, x_2, ..., x_n\}$ is a nonempty finite set of objects, A is a finite set of attributes, $V = \bigcup_{p \in A} V_p$ is a nonempty finite set of values of attributes where V_p is the *domain* of attribute p (the set of values of attribute p), and $f : U \times A \rightarrow V$ is an *information function* which assigns particular values from the domains of attributes in A to objects such that $f(x_i, p) \in V_p$ for all $x_i \in U$ and $p \in A$. In a relational database system, the function f assigns each object to a unique tuple of a database table to avoid duplication and to provide storage efficiency. We view a relational database table as an information system and use the terms *relational database* table and *information system* interchangeably. Table 1 shows an information system for a collection of Japanese and American cars.

A concept hierarchy for an attribute A_i is represented as an attribute value classification tree, called a *concept tree*. A higher-level value is a more general value corresponding to the set of values of lower-level values in the concept tree. Typically, concept hierarchies for a database are provided by knowledge engineers or domain experts, but some hierarchies can be constructed automatically (Michalski 1983). Figure 1 shows concept hierarchies for the "Make_Model" and "Weight" attributes of Table 1.

A large database usually contains many distinct attribute values and is too refined for a clear representation of knowledge. For the purpose of knowledge discovery it is often useful to replace the original attribute values of an information system with more general categories, such as value ranges and higher level concepts. Thus, patterns not visible within the original database may become visible.

{Honda_Civic, ..., Honda_Accord}	$\subset$ Honda
{Toyotal_Tercel, ..., Toyota_Camry}	$\subset$ Toyota
{Mazda_323, Mazda_626, ... Mazda_939}	$\subset$ Mazda
{Honda, Toyota, ..., Mazda}	$\subset$ Japan(Car)
{Ford_Escort, Ford_Probe, ..., Ford_Taurus}	$\subset$ Ford
{Chevrolet_Corvette, ..., Chervolet_Corsica}	$\subset$ Chevrolet
{Dodge_Stealth, ..., Dodge_Dynasty}	$\subset$ Dodge
{Ford, Dodge, ..., Chevrolet}	$\subset$ USA(Car)
{Japan(Car), ..., USA(Car)}	$\subset$ Any(Make_Model)
{0..800}	$\subset$ Light
{801..1200}	$\subset$ Medium
{1201..1600}	$\subset$ Heavy
{Light, Medium, Heavy}	$\subset$ Any(Weight)

Figure 1: Examples of concept hierarchies for Table 1.

In our opinion, the procedure of information generalization should be guided by domain experts, because deep understanding of the meaning of the higher level concepts is required. By identifying relevant attributes and guiding the generalization to a certain level in the concept hierarchy, the domain expert can obtain an appropriate generalized information system.

An attribute is *generalizable* if there exists a concept hierarchy for the attribute. Otherwise, it is *nongeneralizable*. If an attribute is generalizable, it can be generalized to a higher level by concept tree ascension, which corresponds to the *climbing generalization trees* generalization rule (Cai et al. 1991). If an attribute is nongeneralizable, it can be removed, which corresponds to the *dropping conditions* generalization rule (Michalski 1983). This removal can be viewed as a generalization to the most general concept ANY. The replacement of the value in a tuple by a higher level concept generalizes the tuple by making it cover more cases than the original one. The final generalized information system may consist of only a small number of distinct tuples. Table 2 shows the result of the

Make_Model	Fuel	Disp	Weight	Cyl	Power	Turbo	Comp	Tran	Mileage	CNo
USA	EFI	Medium	Medium	6	High	yes	high	auto	medium	1
USA	EFI	Medium	Medium	6	High	no	medium	manu	medium	3
USA	EFI	Medium	Heavy	6	High	no	high	manu	medium	1
USA	EFI	Medium	Medium	6	High	no	high	manu	medium	2
USA	EFI	Medium	Light	6	High	yes	high	manu	high	1
USA	EFI	Medium	Medium	4	Medium	no	medium	manu	medium	2
USA	EFI	Medium	Heavy	6	High	no	medium	manu	low	1
Japan	2-BBL	Small	Light	4	Low	no	high	manu	high	1
USA	2-BBL	Small	Medium	4	Low	no	high	manu	medium	1
USA	2-BBL	Small	Medium	4	Medium	no	high	auto	medium	1
Japan	EFI	Small	Medium	4	Low	no	high	manu	high	1
Japan	EFI	Medium	Light	4	Medium	no	medium	manu	high	1
Japan	EFI	Small	Medium	4	High	yes	high	manu	high	1
Japan	2-BBL	Small	Medium	4	Low	no	medium	manu	high	1
USA	EFI	Medium	Medium	4	High	yes	medium	manu	medium	1
USA	EFI	Medium	Heavy	6	High	no	medium	auto	low	1
USA	EFI	Medium	Medium	6	High	no	medium	auto	medium	1
USA	EFI	Medium	Medium	4	High	no	medium	auto	medium	1
Japan	EFI	Small	Medium	4	Medium	no	high	manu	high	1
USA	EFI	Small	Medium	4	Medium	no	high	manu	high	2

Table 2: A generalized car information system.

generalized car information system created by generalizing the attributes "Make_Model" and "Weight" to a higher level. The levels of the other attributes are already considered appropriate. The numeric column "CNo" is the number of tuples in the original database which support the generalized tuple. It provides a measure of strength or confidence associated with the tuple (Ziarko 1991). A *generalized information system* maintains the relationship among generalized data in different attributes for a frequently queried data set.

The following simple algorithm extracts a generalized information system from a relation R. More sophisticated algorithms are given in (Han 1994) and (Carter and Hamilton 1995).

Generalization Algorithm: Extracts a generalized information system from a relation.

Input: (i) A set of task-relevant data R, a relation of arity n with a set of attributes A_i ($1 \leq i \leq n$); (ii) a set H of concept hierarchies, where each $H_i \in H$ is a hierarchy on the generalized attribute A_i, if available; (iii) l_i is the specified level for attribute A_i;

Output: The generalized information system R'

$R' \leftarrow R$

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

  if A_i is not generalizable **then**

    Remove attribute A_i from R'

  else

    if level is not appropriate **then**

      Ascend tree H_i to the specified level l_i and make appropriate substitutions in R'

    endif

  endif

endfor

Remove duplicates from R'

Suppose there are n tuples in a database table with a attributes. The time for substituting the lower level concepts by the higher level concepts is an for all attributes and the time for deleting redundant tuples is $n \log n$. Thus, the total time is at most $(an + n \log n)$. Since a are much smaller than n in a large database, the time complexity of the attribute-oriented method is $O(n \log n)$.

Discovery of Empirical Keys, Minimal Relations, and Core Attributes

Suppose we wish to discover minimal relations characterizing Japanese cars using subset of attributes in the car information system. This discovery task can be represented as an SQL-like request (Cai et al. 1991) that includes the attribute "Make_Model=Japan". The result of the query is the relation shown in Table 3.

The tuples in a generalized information system represent a certain relationship between attributes of the objects (cars). Every relationship among attributes A corresponds to a classification of objects of an information system into disjoint equivalence classes where objects belonging to the same equivalence class have the same attribute values. The classification corresponding to the set of attributes A can be represented by an equivalence relation $R(A) \subseteq OBJ \times OBJ$. Two relationships, one between attributes Q and the other between attributes P, of an information system are equivalent if they produce the same classification of OBJ, that is if $R(P) = R(Q)$.

Based on the above observations, one can see that the same relationship may be described in a simpler way by using the technique of *attribute reducts* (Pawlak 1991), resulting in a simpler information representation and better understanding of the nature of the relationship. A *reduct* is a minimal sufficient subset of attributes $RED \subseteq A$ such that

(a) $R(RED) = R(A)$, i.e., RED produces the same classification of objects as the whole attribute collection A, and

(b) for any $a \in RED$, $R(RED - \{a\}) \neq R(A)$, that is, a reduct is a minimal subset with respect to the property (a).

By definition, each reduct represents an alternative and simplified way of expressing the same relationship between attributes. It is easy to see that

Fuel	Disp	Weight	Cyl	Power	Turbo	Comp	Tran	Mileage	CNo
2-BBL	Small	Light	4	Low	no	high	manu	high	1
EFI	Small	Medium	4	Low	no	high	manu	high	1
EFI	Medium	Light	4	Medium	no	medium	manu	high	1
EFI	Small	Medium	4	High	yes	high	manu	high	1
2-BBL	Small	Medium	4	Low	no	medium	manu	high	1
EFI	Small	Medium	4	Medium	no	high	manu	high	1

Table 3: A generalized car information system.

reduct has the same properties as key defined in relational database theory (with respect to a specific instance of a relation only), and therefore it can be called an empirical key in this context. For example, in the generalized information system given in Table 3 one can identify the following reducts (empirical keys): {Fuel, Weight, Power}, {Fuel, Power, Comp} and {Weight, Power, Comp}. The projections of Table 3 on above empirical keys result in three different minimal relations.

The *core attributes* are the ones which cannot be eliminated from A without affecting our ability to classify an object into a particular relationship category. They are the most significant attributes in the analyzed relationship. The core attribute set is equal to the intersection of all computed reducts (Pawlak 1991). For example, the core attribute set of the generalized information system given in Table 3 is "Power" which means that it is the fundamental factor in the relationship between the given attributes. In other words, the relationship class of an object cannot be determined without knowing "Power" attribute values.

Discovery of Discrimination Rules

In this section, a method for discovering discrimination rules using rough set theory is introduced. Suppose our objective is to discover a set of decision rules which identify the features of a car that determine the "Mileage". The discovery task extracts the relevant set of data shown in Table 2 from the car's database. In such an information table, attributes A can be partitioned into two (disjoint) subsets, the *condition* attributes C and *decision* attributes D. Let $R^*(C) = \{X_1, X_2, ..., X_n\}$ be the collection of *equivalence classes* of the relation $R(C)$, and let $R^*(D) = \{Y_1, Y_2, ..., Y_m\}$ be the collection of equivalence classes of the relation $R(D)$, where each element Y_i is a group of objects having the same values for all attributes in D and creates a concept on the universe U. The pair $AS = (U, R(C))$ is called an *approximation space* (Pawlak 1991). The *lower approximation* in the approximation space AS, denoted as $LOW(C, D)$, is defined as the union of those equivalence classes in $R^*(C)$ which are completely contained by one of the equivalence classes $Y_i \in R^*(D)$, that is $LOW(C, D) = \bigcup_{Y_i \in R^*(D)} \{X \in R^*(C) : X \subseteq Y_i\}$. The *degree of dependency* $K(C, D)$ in the relationship between the groups of attributes C and D can be defined as $K(C, D) = \frac{card(LOW(C,D))}{card(U)}$, where *card* yields set cardinality. The dependency between two sets of attributes C and D indicates the extent to which values of attributes in D depend on the values of attributes in C. By definition, $0 \leq K(C, D) \leq 1$ because $LOW(C, D) \subseteq U$. If $K(C, D)$ is equal to 1, the dependency, as represented in the information system table, is fully functional. $K(C, D)$ is equal to 0 when none of the values of attributes in D can be uniquely determined from the values of attributes in C.

Attribute Reduction

A *relative reduct* (Pawlak 1991) is a minimal sufficient subset of a set of attributes which preserves the degree of dependency with respect to another set and which has the same ability to discern concepts as when the full set of attributes is used.

Definition A subset B of a set of attributes C is a *relative reduct* of C with respect to a set of attributes D if: 1) $K(B, D) = K(C, D)$, and $K(B, D) \neq K(B - \{a\}, D)$, for any $a \in B$.

The first condition ensures that a relative reduct preserves the degree of dependency with respect to D, and the second condition ensures that a relative reduct is a minimal subset and that any further removal of condition attributes will change the degree of dependency. The following algorithm constructs a relative reduct for an information system.

Generalization Algorithm: Computes a relative reduct
Input: (i) A generalized information system S;
(ii) A set of attributes C over S;
(iii) The degree of dependency $K(C, D)$ in S;
Output: A relative reduct SM
Compute the significance value for each attribute $a \in C$
Sort the set of attributes C in ascending order
of the significance values
SM $\leftarrow C$
$N \leftarrow$ |SM|
for i = 0 **to** $N - 1$ **do**
Remove the i^{th} attribute a_i from the set SM
if $K(\text{SM}, D) \neq K(C, D)$ **then**
SM $\leftarrow$ SM $\cup\, a_i$
endif
endfor

First, the algorithm assigns a significance value based on an evaluation function to each attribute and sorts the attributes based on their significance values. To compute a reduct, the algorithm remove attributes one

Make_Model	Weight	Power	Comp	Tran	Mileage	CNo
USA	Medium	High	high	auto	medium	1
USA	Medium	High	medium	manu	medium	4
USA	Heavy	High	high	manu	medium	1
USA	Medium	High	high	manu	medium	2
USA	Light	High	high	manu	high	1
USA	Medium	Medium	medium	manu	medium	2
USA	Heavy	High	medium	manu	low	1
Japan	Light	Low	high	manu	high	1
USA	Medium	Low	high	manu	medium	1
USA	Medium	Medium	high	auto	medium	1
Japan	Medium	Low	high	manu	high	1
Japan	Light	Medium	medium	manu	high	1
Japan	Medium	High	high	manu	high	1
Japan	Medium	Low	medium	manu	high	1
USA	Heavy	High	medium	auto	low	1
USA	Medium	High	medium	auto	medium	2
Japan	Medium	Medium	high	manu	high	1
USA	Medium	Medium	high	manu	high	2

Table 4: A relative reduct of the generalized car information system.

by one from set SM. At each step, the degree of dependency is calculated based on the remaining attributes in the set SM. If the degree of dependency has been changed, then the attribute is restored to the set SM; otherwise, the attribute is permanently discarded. The attributes remaining in set SM at the end form a relative reduct. Table 4 shows a relative reduct with respect to the decision attribute "Mileage" for the generalized car information system given in Table 2.

For n objects (tuples) with a attributes, the time complexity of our algorithm in the worst case is $O((2a+1)n + a\log a)$, because computing the degree of dependency (by using a hashing technique) is $O(n)$, computing attribute significance values is $O(an)$, sorting the attributes based on the significance values is $O(a\log a)$, and creating the relative reduct is $O(an)$.

Simplification of Decision Rules

A *rule* is a combination of values of some condition attributes such that the set of all objects matching it is contained in the set of objects labeled with the same class, and such that there exists at least one such object. Traditionally, a rule r is denoted as an implication

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

where $a_{i1}, a_{i2}, ...,$ and a_{in} are the condition attributes and d is a decision attribute. The set of attribute-value pairs occurring on the left hand side of the rule r is referred to as the rule condition part, denoted $cond(r)$, and the right hand side is the decision part, $dec(r)$, so the rule can be expressed as $cond(r) \rightarrow dec(r)$. Including more condition attributes in $cond(r)$ makes the rule more specific. A reduced information system can be considered as a set of specific decision rules $RULE = \{r_1, r_2, ..., r_n\}$. Each rule in $RULE$ corresponds to exactly one equivalence class $X_i \in R^*(RED)$. Such decision rules can be generalized by dropping conditions. The process by which the maximum number of condition attribute values are removed without loosing essential information is called *Value Reduction* (Pawlak 1991) and the resulting rule is called *maximally general* or *minimal length*. Discovering the optimal rules is of particular importance in knowledge discovery in databases because they represent the maximally-general patterns that exist in databases. The *maximally general rules* minimize the number of rule conditions and are optimal because their conditions are non-redundant.

To obtain a set of maximally general rules $MRULE$, each rule $r \in RULE$ is considered for dropping conditions. The algorithm initializes $MRULE$ to empty and copies one rule $r_i \in RULE$ to rule r. A condition is dropped from rule r, and then rule r is checked for decision consistency with every rule $r_j \in RULE$. If rule r is inconsistent, then the dropped condition is restored. This step is repeated until every condition of the rule has been dropped once. The resulting rule r is a maximally generalized rule. Before rule r is added to $MRULE$, it is checked for rule redundancy. If rule r is logically included in any rule $r_a \in MRULE$, rule r is discarded. If any rules in $MRULE$ are logically included in rule r, these rules are removed from $MRULE$. After all rules in $RULE$ have been processed, $MRULE$ contains a set of maximally general rules.

Generalization Algorithm: Rule Generation
Input: A set of specific decision rules $RULE$
Output: A set of maximally general rules $MRULE$
$MRULE \leftarrow \emptyset$
$N \leftarrow |RULE|$
for $i = 0$ **to** $N-1$ **do**
  $r \leftarrow r_i$
  $M \leftarrow |r|$
  for $j = 0$ **to** $M-1$ **do**
    Remove the j^{th} condition attribute a_j in rule r
    if r inconsistent with any rule $r_n \in RULE$ **then**
      Restore the dropped condition a_j
    endif
  endfor
  Remove any rule $r' \in MRULE$ that is logically included in rule r
  if rule r is not logically included in a rule $r' \in MRULE$ **then**
    $MRULE \leftarrow r \bigcup MRULE$
  endif
endfor

Make_Model	Weight	Power	Comp	Tran	Mileage	CNo
-	Heavy	-	medium	-	low	2
USA	Medium	High	-	-	medium	9
USA	Medium	-	medium	-	medium	8
-	Medium	-	-	auto	medium	4
USA	-	Light	-	-	medium	1
-	Heavy	-	high	-	medium	1
-	-	Medium	high	medium	high	3
Japan	-	-	-	-	high	6
-	Light	-	-	-	high	3

Table 5: A set of maximally general rules.

Suppose there are n' tuples (decision rules) with a' attributes in the reduced information system. Finding a maximally general rule for one decision rule requires $O(a'n')$ time and finding maximally general rules for n' decision rules requires $O(a'n'^2)$ time. Eliminating redundant rules is $O(n'^2)$. Consequently, the time complexity of our algorithm is $O((a'+1)n'^2) = O(a'n'^2)$.

Table 5 shows the set of maximally general rules corresponding to the values in Table 4; in Table 5, "-" indicates "don't care". For example, the first entry in the table corresponds to:

$$(\text{Weight} = \text{Heavy}) \wedge (\text{Comp} = \text{medium}) \rightarrow (\text{Mileage} = \text{low})$$

The rules in Table 5 are more concise than the original data in Table 1, and they provide information at a more abstract level. Nonetheless, they are guaranteed to give decisions about "Mileage" consistent with the original data. The higher the value of "CNo", the more the rule is confirmed. Usually, we would not rely on a rule based on one or very few support tuples unless it is known that the contents of the information table exhausts *all* feasible combination values of attributes.

Summary and Conclusions

We presented an approach to knowledge discovery which provides an effective way to discover hidden patterns and to transform information in a database into simplified, easily understood form. During the information generalization stage, undesired attributes are eliminated, primitive data are generalized to higher level concepts according to concept hierarchies and the number of tuples in the generalized information system is decreased compared with the original relation. The rough sets technique, used at the data analysis and reduction stages, provides an effective tool to analyze the attribute dependencies and identify irrelevant attributes during the information reduction process. The rules computed from reduced information system are usually concise, expressive and strong because they are in the most generalized form and only use necessary attributes. The rules represent data dependencies occurring in the database.

References

Cai, Y.; Cercone, N.; and Han, J. 1991. Attribute-Oriented Induction in Relational Databases. In *Knowledge Discovery in Databases*, AAAI/MIT Press. pp. 213-228.

Carter, C.L., and Hamilton, H.J. 1995. A Fast, Online Generalization Algorithm for Knowledge Discovery. *Appl. Math. Lett. 8(2):5-11.*

Fayyad, U.M.; Weir, N.; and Djorgovski, S. 1993. SKICAT: A Machine Learning System for Automated Cataloging of Large Scale Sky Surveys. In Machine Learning: Proc. of the Tenth International Conference. Morgan Kaufmann. pp. 112-119.

Gemello, R., and Mana, F. 1989. An Integrated Characterization and Discrimination Scheme to Improve Learning Efficiency in Large Data Sets. In Proceedings of the 11th IJCAI-89. Vol. 1. pp. 719-724.

Han, J. 1994. Toward Efficient Induction Mechanism in Database System. *Theoretical Computer Science.* Oct., 1994.

Michalski, R.S. 1983. A Theory and Methodology of Inductive Learning. In Michalski, R.S.; Carbonell, J.G.; and Mitchell, T.M. (eds). 1983. *Machine Learning: An Artificial Intelligence Approach*, Vol. 1, pp. 83-134.

Pawlak, Z. 1991. *Rough Sets: Theoretical Aspects of Reasoning About Data.* Kluwer Academic.

Piatetsky-Shapiro, G., and Frawley. W.J. (eds). 1991. *Knowledge Discovery in Databases*, AAAI/MIT Press.

Slowinski, R. (ed). 1992. *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory*, Kluwer Academic.

Ziarko, W. 1991. The Discovery, Analysis, and Representation of Data Dependencies in Databases. In *Knowledge Discovery in Database*, AAAI/MIT Press. pp. 195-209.

Ziarko, W. (ed). 1994. *Rough Sets, Fuzzy Sets and Knowledge Discovery.* Springer-Verlag.

Ziarko, W., and Shan, N. 1995. Discovering Attribute Relationships, Dependencies and Rules by Using Rough Sets. In the Proceedings of the 28th Hawaii International Conference on Systems Sciences, Vol. 3, pp. 293-299.