Incremental Versus Non-incremental: Data and Algorithms Based on Ordering Relations

Xiuyi Jia , Lin Shang , Jiajun Chen , Xinyu Dai

State Key Laboratory for Novel Software Technology, Nanjing University,
Nanjing, 210093, China

Department of Computer Science and Technology, Nanjing University,
Nanjing, 210093, China

E-mail: jiaxy@nlp.nju.edu.cn, {shanglin,chenjj}@nju.edu.cn, dxy@nlp.nju.edu.cn

Accepted: 08-11-2009
Received: 09-09-2010

Abstract

Based on multi-dominance discernibility matrices, a non-incremental algorithm RIDDM and an incremental algorithm INRIDDM are proposed by means of Dominance-based Rough Set Approach. For the incremental algorithm, when a new object arrives, after updating one row or one column in the matrix, we could get the updated rule sets. Time complexity analysis and experimental results show that the incremental algorithm INRIDDM is superior to some other non-incremental algorithms when dealing with large data sets. This paper also explores the influence of data saturation and data concentration on rule induction algorithms. We come to conclude that the data saturation and data concentration are important for the performance analysis of one learning algorithm.

Keywords: Incremental rule induction, dominance-based rough set approach, data saturation, data concentration.

1. Introduction

It is typical to represent knowledge by rules [1] and rough set theory can be used to induce rules. The classical rough set [6] is induced by the equivalence relation and can be generalized by considering any binary relation [7]. The Dominance-based Rough Set Approach(DRSA) employs ordering relations [2].

In paper [8,9,10], Yao and his colleagues proposed that the form of rules falls into at least two types under the generalized rough set: type 1 rules defined on attribute level and attribute value level in type 2 rules. The two types of rules were defined as:

$$\begin{aligned} &type\ 1\ rules: && I_a(x) \geqslant_a I_a(y) \Rightarrow I_d(x) \geqslant_d I_d(y);\\ &type\ 2\ rules: && I_a(x) \geqslant_a v_a \Rightarrow I_d(x) \geqslant_d v_d. \end{aligned}$$

A type 1 rule states that “if two objects have the same ordering relation on attribute a, then they have the same ordering relation on attribute d” and a type 2 rule states that “if the value of an object is equal to or larger than v_a on attribute a, then the value of object is equal to or larger than v_d on attribute d”. While type 1 rules focus on a pair of objects, type 2 rules focus on a single object. Type 1 rules are useful for the study of relationships between objects and type 2 rules are useful for classification.

In [8,9], the information table with ordering relations was firstly transformed into a binary information table, then standard machine learning and data mining algorithms could be used to mine type 1 rules. Most rule induction algorithms generate type 2 rules. Greco and Slowinski proposed rule induction algorithms DOMLEM [2] and AllRules [3] in DRSA for inducing type 2 rules. Both of them need to check all attribute-value pairs in order to choose proper pairs as the condition parts of the generated rules. Their computational efficiency are low. In this paper, we give an efficient incremental rule induction algorithm INRIDDM for generating type 2 rules.

Generally, a set of attributes, which could be used to classify two objects with different decision class labels, can be computed from the discernibility matrix. For classical rough set theory, through a single discernibility matrix we could get the discernibility attribute set because it is enough to express the discernibility of any two different objects. Otherwise, as in DRSA, if we only consider the decision labels, we will lose some other discernibility information. For example, for an upward union of objects, there may exist two objects with different decision class labels while they are indiscernible for the same upward union. So, if we want to get the discernibility information from an information table with ordering relations, one discernibility matrix is insufficient. In this paper, we study the discernibility of the dominance discernibility matrix and propose a notion of multi-dominance discernibility matrices for DRSA, which considers both the discernibility of two different decision class labels and the discernibility of two different upward or downward unions. Based on multi-dominance discernibility matrices, a non-incremental rule algorithm RIDDM [4] and an incremental algorithm INRIDDM are proposed.

For the incremental algorithm, when a new object arrives, the updated rules can be induced by updating one row or one column in the matrix only. When evaluating the performance of an algorithm, many researchers usually take account of its time usage and space usage only, but ignore the characteristics of the given data. In this paper, we also explore the relationship between algorithms and data. For presented non-incremental and incremental rule induction algorithms, three measurements of data saturation, data concentration and data redundancy [5] are considered. For two given data sets with the same number of objects and attributes, a dealing algorithm may produce different performance results if these data sets have different data saturations and data concentrations. Experimental results show that the incremental algorithm INRIDDM is superior to some other non-incremental algorithms on dealing with large data sets.

The rest of this paper is organized as follows. Section 2 briefly introduces some preliminary knowledge on DRSA, multi-dominance discernibility matrices and measurements of data. Section 3 gives the details of the non-incremental and incremental rule induction algorithms. Section 4 presents the influence of data saturation and data concentration on rule induction algorithms through the experiments. Finally, Section 5 concludes.

2. Preliminary

2.1. Dominance-based Rough Set Approach

In this section, we present some basic definitions used in DRSA. More details of DRSA can be found in [2].

Assuming that learning examples are represented in information table $DT = (U, C \cup D)$, where U is a set of examples(objects), C is a set of condition attributes describing objects and D is a set of decision attributes, $C \cap D = \emptyset$. This kind of table can also be called decision table. Let $f(x,q)$ denotes the value of attribute $q \in C$ taken by object $x \in U$, V_q is a domain of q [2].

Let $\mathbf{Cl} = \{Cl_t, t \in T\}$, $T = \{1,\ldots,n\}$, be a set of classes such that each $x \in U$ belongs to one and only one $Cl_t \in \mathbf{Cl}$. We suppose that the classes are ordered, i.e., for all $r,s \in T$, $r > s$, such that the objects from Cl_r are preferred to the objects from Cl_s.

The sets to be approximated are *upward union* and *downward union* of classes, respectively: $Cl_t^{\geqslant} = \bigcup_{s \geqslant t} Cl_s, Cl_t^{\leqslant} = \bigcup_{s \leqslant t} Cl_s, t = 1,\ldots,n$. Usually we do not take $Cl_n^{\leqslant}$ and $Cl_1^{\geqslant}$ into consideration because both of their values are U, it is useless for decision maker.

The equality relation is substituted by a *dominance relation*. We say that x *dominates* y with respect to $P \subseteq C$, denoted by xD_Py, which means "x is at least as good as y with respect to P".

-a set of objects dominating x, called *P-dominating set*, $D_P^+(x) = \{y \in U : yD_Px\}$,

-a set of objects dominated by x, called *P-dominated set*, $D_P^-(x) = \{y \in U : xD_Py\}$.

Definition 1. Using $D_P^+(x)$ and $D_P^-(x)$, P-lower and P-upper approximation of $Cl_t^{\geqslant}$ and $Cl_t^{\leqslant}$ are defined as:

$$\begin{aligned}
\underline{P}(Cl_t^{\geqslant}) &= \{x \in U : D_P^+(x) \subseteq Cl_t^{\geqslant}\}, \\
\overline{P}(Cl_t^{\geqslant}) &= \bigcup_{x \in Cl_t^{\geqslant}} D_P^+(x), for\ t = 1, \ldots, n. \\
\underline{P}(Cl_t^{\leqslant}) &= \{x \in U : D_P^-(x) \subseteq Cl_t^{\leqslant}\}, \\
\overline{P}(Cl_t^{\leqslant}) &= \bigcup_{x \in Cl_t^{\leqslant}} D_P^-(x), for\ t = 1, \ldots, n.
\end{aligned}$$

2.2. *Multi-dominance discernibility matrices*

In DRSA, the decision class union is considered. The discernibility information between two objects with different decision class labels can be obtained from a single dominance discernibility matrix, but for two different decision class unions, it is not enough. For example, consider two different objects x and y with decision class labels Cl_1 and Cl_2 correspondingly, it is obvious that x and y are different through a dominance discernibility matrix, but they are indiscernible for $Cl_3^{\leqslant}$. To solve this problem, the notion of multi-dominance discernibility matrices corresponding to multi-decision class unions in DRSA is proposed . The matrices are defined as the following:

Definition 2. For the given information table DT, the dominance discernibility matrix of decision class union $Cl_t^{\geqslant}$ is $M = \{m_{ij}\}$, where $Cl_t \in \mathbf{Cl}$.

$$m_{ij} = \begin{cases} \{q \in C : f(x_i, q) > f(x_j, q)\}, & \\ \qquad x_i \in \underline{C}(Cl_t^{\geqslant}) \& x_j \in Cl_{t-1}^{\leqslant}; & \\ \emptyset, & otherwise. \end{cases}$$

The matrix of decision class union $Cl_t^{\leqslant}$ is denoted as $M^* = \{m_{ij}^*\}$, which is constructed as following:

$$m_{ij}^* = \begin{cases} \{q \in C : f(x_i, q) < f(x_j, q)\}, & \\ \qquad x_i \in \underline{C}(Cl_t^{\leqslant}) \& x_j \in Cl_{t+1}^{\geqslant}; & \\ \emptyset, & otherwise. \end{cases}$$

The matrices of $Cl_1^{\geqslant}$ and $Cl_n^{\leqslant}$ don't be considered in this paper, as their lower approximations are the domain of objects U, which gives nothing to decision makers. Different from common single discernibility matrix, $(2*n-2)$ matrices are constructed for all decision class unions. In this definition, the rows of matrices are labelled with the objects which belong to the lower approximation of the union of decision classes, and the columns are labelled with the negative set, which means those objects haven't been classified into the decision class union according to their decision class labels.

Example 1. Table 1 is an information table with six objects and four attributes, the set of the condition attributes is $C = \{C_1, C_2, C_3\}$, the decision attribute presented D with two values. Assume the larger values of all attributes are better for decision maker. We will use this table for the rest of the paper.

Table 1. An information table.

U	C_1	C_2	C_3	D
x_1	1	1	1	2
x_2	1	1	1	1
x_3	1	0	1	1
x_4	0	0	1	1
x_5	0	1	1	1
x_6	2	1	2	2

For Table 1, we have following partitions defined by the set C.

$$\underline{C}(Cl_2^{\geqslant}) = \{x_6\} \qquad Cl_1^{\leqslant} = \{x_2, x_3, x_4, x_5\}$$

$$\underline{C}(Cl_1^{\leqslant}) = \{x_3, x_4, x_5\} \qquad Cl_2^{\geqslant} = \{x_1, x_6\}$$

The following dominance discernibility matrices are constructed for $Cl_2^{\geqslant}$ and $Cl_1^{\leqslant}$ according to Definition 2, respectively:

$$M(Cl_2^{\geqslant}) =$$

$$\begin{array}{ccccc} & x_2 & x_3 & x_4 & x_5 \\ x_6 & [\{C_1, C_3\} & \{C_1, C_2, C_3\} & \{C_1, C_2, C_3\} & \{C_1, C_3\}] \end{array}$$

$$M^*(Cl_1^{\leqslant}) = \begin{array}{c} \\ x_3 \\ x_4 \\ x_5 \end{array} \begin{array}{c} \begin{array}{cc} x_1 & x_6 \end{array} \\ \left[\begin{array}{cc} \{C_2\} & \{C_1,C_2,C_3\} \\ \{C_1,C_2\} & \{C_1,C_2,C_3\} \\ \{C_1\} & \{C_1,C_3\} \end{array} \right] \end{array}$$

So each matrix presents the discernibility information to the corresponding decision class union but not the whole information table.

2.3. *Data saturation and data concentration*

To estimate the characteristics of a data set, three definitions of the data characteristics [5] are introduced. The data saturation of an information table is a measurement of the data items' capability relative to a set of attributes. So the data saturation can be considered as the largest number of data items that the attributes set can contain. The data saturation of an information table is defined as follows:

Definition 3. Data Saturation. In an information table $DT = (U, C \cup D)$, C is the set of the condition attributes, D is the set of the decision attributes. $P \subseteq (C \cup D)$, here $P = \{p_1, p_2, p_3, \ldots, p_n\}$, the data saturation of P is denoted as ξ_P, which can be calculated as

$$\xi_P = \prod_{i=1}^{n} N_{P_i},$$

where N_{P_i} is the number of the attribute values of p_i. If $P = \emptyset$, then $\xi_P = 1$. The definition of data saturation represents the largest number of items that the attributes set P contains. It can also be viewed as the largest boundary relation in the information table.

A concept of data concentration is presented to describe the percent of the equivalent in the data set. The definition of data concentration is given as following:

Definition 4. Data Concentration. In an information table $DT = (U, C \cup D)$, the data concentration of U is denoted as θ, which can be calculated as

$$\theta = \frac{|U/IND(C \cup D)|}{\xi_{C \cup D}},$$

where $IND(C \cup D)$ is an equivalence relation with respect to $C \cup D$, $U/IND(C \cup D)$ means the partition of U induced by $IND(C \cup D)$. Once the data set U has $\theta = 1$, U is saturated.

Another concept can be well described by the data concentration, which be called data redundancy. The following is the definition of data redundancy [5]:

Definition 5. Data Redundancy. In an information table $DT = (U, C \cup D)$, θ is the data concentration, ξ is the data saturation. The data redundancy of U is denoted as γ, which can be calculated as

$$\gamma = 1 - \frac{\theta * \xi}{|U|}.$$

The redundancy means the repetition of the equation classes.

3. Rule induction algorithms

After constructing the multi-dominance discernibility matrices, the decision rules in terms of "if ... then ..." can be induced. The following is the details of rule induction algorithms.

3.1. *Non-incremental rule induction algorithm* RIDDM

The multi-dominance discernibility matrices according to Definition 2 are constructed first, while rows present the objects from the lower approximations of the decision classes unions, the columns present the negative set of the union of decision classes. Next the major-disjunctive normal form of every row in the matrix is calculated, the result is a set of condition attributes which can be used to distinguish the objects with other negative objects. We call each conjunctive normal form in the major-disjunctive normal form *attributes candidate set*, and the set is not empty. If it were empty, the object presented by the row will not belong to the lower approximation union by the definition of the matrix. After computing the major-disjunctive normal form of every row of the matrix, more than one *attributes candidate set* is obtained. Then the most frequent *attributes candidate set* is chosen as the rule condition part to generate a decision rule with those rows(objects) containing the corresponding *attributes candidate set*. More than one rule may be generated for this *attributes candidate set* in this procedure, then they are combined into a rule set. After getting the rule set for the most frequent *attributes candidate set*, those rows containing the *attributes candidate set* are deleted from the matrix and a new different matrix left. For this new matrix, the same operations are applied to get other new rule sets until the matrix is empty. At last the set of all generated rule set is what we induced for the decision union of classes, which is a *complete* set that cover all objects in lower approximations of unions [3]. The general scheme of the algorithm is presented below.

```
Algorithm 1. RIDDM(Rule Induction based on Dominance Discernibility Matrix)
Input: (1)The lower approximation of decision union $Cl_t^{\geqslant}(Cl_t^{\leqslant})$: $U1 = \underline{C}(Cl_t^{\geqslant})(U1 = \underline{C}(Cl_t^{\leqslant}))$,
       (2)The negative set of decision union $Cl_t^{\geqslant}(Cl_t^{\leqslant})$: $U2 = Cl_{t-1}^{\leqslant}(U2 = Cl_{t+1}^{\geqslant})$.
Output: The dominance discernibility matrix $M$ and rule set of $Cl_t^{\geqslant}(Cl_t^{\leqslant})$: Rule.
BEGIN
Step 1. get the matrix $M$ with rows are $U1$ and columns are $U2$ by Definition 2;
Step 2. $Cond = \emptyset$;/*the set of the attributes candidate sets*/
Step 3. $Rule = \emptyset$;/*the set of rule sets*/
Step 4. FOR each row of $M$ DO
Step 4.1. compute the major-disjunctive normal form and get the set of attributes candidate sets of this row $C$;/* sum up the frequency of each attributes candidate set */
Step 4.2. FOR each attributes candidate set $c \in C$ DO
Step 4.2.1. IF $c \in Cond$ THEN
Step 4.2.1.1. sum of $c$ plus 1;
Step 4.2.2. ELSE
Step 4.2.2.1. $Cond = Cond \cup \{c\}$;
Step 4.2.3. END IF
Step 4.3. END FOR
Step 5. END FOR
Step 6. $Rule =$ generate_the_rules ($M$,$Cond$);
Step 7. RETURN $Rule$;
END BEGIN

FUNCTION generate_the_rules.
(Input: $M$ and $Cond$.
Output: $Rule$.)
BEGIN
Step 1. $Rule = \emptyset$;
Step 2. $M1 = M$;
Step 3. WHILE matrix $M1$ is not empty DO
Step 3.1. Sort($Cond$);/*sort all the attributes candidate sets in descending order*/
Step 3.2. $bestC =$ Select($Cond$);/*choose the most frequent attributes candidate set $bestC \in Cond$*/
Step 3.3. $R =$ GenerateRule($M1, bestC$);/*generate the rule set for those rows containing $bestC$ in $M1$, $bestC$ is the condition part of each rule*/
Step 3.4. $Rule = Rule \cup R$;
Step 3.5. delete those rows containing $bestC$ from $M1$;
Step 3.6. count all attributes candidate sets in new $M1$;
Step 4. END WHILE
Step 5. RETURN $Rule$;
END BEGIN
```

In the algorithm, a function named **generate_the_rules** is applied to generate the rule set. Some steps need to be explained in detail here. In Step 4.1 of Algorithm 1, some optimization laws are used to reduce the computation time when computing the major-disjunctive normal form, such as *absorption laws*, et.al. In Step 3.3 of FUNCTION **generate_the_rules**, if there is more than one rule generated, merge them if one of them could be covered by other rules, otherwise, reserve them. For a singleton decision class or other kind of decision class unions like $Cl_t \cup Cl_k$, the steps are similar.

For decision class union $Cl_1^{\leqslant}$ in Table 1, the major-disjunctive normal forms are $\{C_2\}, \{C_1\} \wedge \{C_2\}, \{C_1\}$. The two *attributes candidate sets* are: $\{C_2\}$ and $\{C_1\}$, both of them appear twice, we choose $\{C_2\}$ as the condition part of a rule first, the generated rules for objects x_3 and x_4 are: "$if\ \ f(x,C_2) \leqslant 0, then\ \ x \in Cl_1^{\leqslant}$", "$if\ \ f(x,C_2) \leqslant 0, then\ x \in Cl_1^{\leqslant}$". We merge the two rules into one because they are same. For the rows containing

$\{C_2\}$, x_3 and x_4 are deleted, the new matrix contains only one row, and only one *attributes candidate set* $\{C_1\}$. The new rule generated for x_5 is "$if\ f(x,C_1) \leqslant 0, then\ x \in Cl_1^{\leqslant}$", the matrix is empty after the row containing $\{C_1\}$ be deleted. Algorithm ends. The final rule sets are "$if\ f(x,C_2) \leqslant 0, then\ x \in Cl_1^{\leqslant}$" and "$if\ f(x,C_1) \leqslant 0, then\ x \in Cl_1^{\leqslant}$".

3.2. Incremental rule induction algorithm

An incremental rule induction algorithm based on RIDDM algorithm will be presented in this section. For simplification, we assume that the domain of decision attribute(s) D are $1,2,\dots,n$ unchanged when information table increases dynamically. For object x and object y, we say they are inconsistent when they have different dominance relations on condition attributes and decision attribute(s), i.e., x is dominating(dominated by) y on condition attributes and dominated by(dominating) y on decision attribute(s). Otherwise, they are consistent. For a new added object x, if $\forall y \in U1$, x and y are consistent, then x is consistent with $U1$. While if $\exists y \in U1$, x and y are inconsistent, then x is inconsistent with $U1$.

The key idea of the incremental procedure is that, for a new object x and the matrix M generated by algorithm RIDDM, we just get the updated matrix $M(x)$ of $(U1 \cup U2 \cup \{x\})$ instead of recomputing the whole matrix M. When a new object x arrives, x is positive or negative for one decision class union. According to the definitions of $U1$ and $U2$ in a dominance discernibility matrix, the updating methods of M are presented as following:

(1) If x is positive for the decision class union and there is no inconsistent relation between x with objects of $U2$, then x is classified to the lower approximation of the decision class union, $U1 = U1 \cup \{x\}$, add a new row in M according to the definition of matrix.

(2) If x is positive for the decision class union and there exists inconsistent relation between x with some objects of $U2$, then x is classified to the boundary of the decision class union, and M remains unchanged.

(3) If x is negative for the decision class union and there exists inconsistent relation between x with some objects of $U1$, then find the inconsistent object(s) y(maybe there exists more than one y) in $U1$, delete the corresponding row(s) from the matrix, $U1 = U1 - \{y\}$, and add the corresponding column(s) respectively, $U2 = U2 \cup \{x\}$.

(4) If x is negative for the decision class union and there is no inconsistent relation between x with objects of $U1$, then add corresponding column in M, $U2 = U2 \cup \{x\}$.

Then the description of the incremental rule induction algorithm is given as following:

Algorithm 2. INRIDDM(INcremental Rule Induction based on Dominance Discernibility Matrix)

```
Input: (1) dominance discernibility matrix $M$,
       (2) $U1$ and $U2$ of $M$,
       (3) the set of attributes candidate sets $Cond$ of $M$,
       (4) new added object $x$.
Output: new matrix $M(x)$ and the rule set $Rule$ of decision class union $Cl_t^{\geqslant}$ ($Cl_t^{\leqslant}$).
BEGIN
Step 1. $Rule = \emptyset$;
Step 2. IF $x$ is positive for decision class union THEN
Step 2.1. IF $x$ is consistent with $U2$ THEN
Step 2.1.1. add a row according to Definition 2 in $M$, get $M(x)$;
Step 2.1.2. update $Cond$;
Step 2.1.3. $U1 = U1 \cup \{x\}$;
Step 2.2. ELSE
Step 2.2.1 $M(x) = M$;
Step 2.3. END IF
Step 3. ELSE
Step 3.1. IF there exists inconsistent relation between $x$ with $U1$ THEN
Step 3.1.1. find the inconsistent object set $Y$ with $x$ in $U1$;
Step 3.1.2. FOR each $y \in Y$ DO
Step 3.1.2.1. delete the row corresponding to $y$ from $M$;
Step 3.1.2.2. $U1 = U1 - \{y\}$;
Step 3.1.3. END FOR
Step 3.1.4. add corresponding column $x$ in $M$, get $M(x)$;
Step 3.1.5. $U2 = U2 \cup \{x\}$;
Step 3.1.6. update $Cond$;
Step 3.2. ELSE
Step 3.2.1. add corresponding column $x$ in $M$, get $M(x)$;
Step 3.2.2. $U2 = U2 \cup \{x\}$;
Step 3.2.3. update $Cond$;
Step 3.3. END IF
Step 4. END IF
Step 5. $Rule$ = **generate_the_rules**$(M(x), Cond)$;
Step 6. RETURN $Rule$;
END BEGIN
```

In this algorithm, the rules covering more objects are prior to those covering less objects. This kind of greedy strategy may lead the solution into a local optimal result, i.e., not the rule set with least rules. The post-processing procedure can be applied into the result to get the global optimal rules, as in DOMLEM [2]. The post-processing procedure needs to judge if the rule is minimal, which means to scan the whole data table. The post-processing do not be considered in algorithms RIDDM and INRIDDM, because for most decision maker(s), both more choices and less computation cost are better.

The following is an example of adding different objects in Table 1, which shows the detail of the incremental procedure.

a. For Table 1, add a new object $x(2,2,2,2)$, then x is positive for $Cl_2^{\geqslant}$, and it satisfies the updating situation (1): $U1 = \{x_6, x\}$, $U2$ remains unchanged, and $M(x)$ of $Cl_2^{\geqslant}$ is updated to:

$$\begin{array}{c} \\ x_6 \\ x \end{array}\begin{array}{c}\begin{array}{cccc} x_2 & x_3 & x_4 & x_5 \end{array}\\ \begin{bmatrix} \{C_1,C_3\} & \{C_1,C_2,C_3\} & \{C_1,C_2,C_3\} & \{C_1,C_3\} \\ \{C_1,C_2,C_3\} & \{C_1,C_2,C_3\} & \{C_1,C_2,C_3\} & \{C_1,C_2,C_3\} \end{bmatrix}\end{array}$$

Meanwhile, x is negative for $Cl_1^{\leqslant}$, it satisfies the updating situation (4): $U1$ remains unchanged, $U2 = \{x_1, x_6, x\}$, $M(x)$ of $Cl_1^{\leqslant}$ is updated to:

$$\begin{array}{c} \\ x_3 \\ x_4 \\ x_5 \end{array}\begin{array}{c}\begin{array}{ccc} x_1 & x_6 & x \end{array}\\ \begin{bmatrix} \{C_2\} & \{C_1,C_2,C_3\} & \{C_1,C_2,C_3\} \\ \{C_1,C_2\} & \{C_1,C_2,C_3\} & \{C_1,C_2,C_3\} \\ \{C_1\} & \{C_1,C_3\} & \{C_1,C_2,C_3\} \end{bmatrix}\end{array}$$

b. For Table 1, add a new object $x(0,1,1,2)$, then x is positive for $Cl_2^{\geqslant}$, and it satisfies the updating situation (2): $M(x)$ of $Cl_2^{\geqslant}$ remains unchanged. Meanwhile, x is negative for $Cl_1^{\leqslant}$, it satisfies the updating situation (3): x is inconsistent with x_5 in $U1$, $U1 = \{x_3, x_4\}$, $U2 = \{x_1, x_6, x\}$, $M(x)$ of $Cl_1^{\leqslant}$ is updated to:

$$\begin{array}{c} \\ x_3 \\ x_4 \end{array}\begin{array}{c}\begin{array}{ccc} x_1 & x_6 & x \end{array}\\ \begin{bmatrix} \{C_2\} & \{C_1,C_2,C_3\} & \{C_2\} \\ \{C_1,C_2\} & \{C_1,C_2,C_3\} & \{C_2\} \end{bmatrix}\end{array}$$

From *a*. and *b*. we know that we just need to update the corresponding dominance discernibility matrix when a new object arrives, and generate the rule set from the updated matrix.

Let us discuss the computation complexity of the algorithm briefly. For space complexity, the algorithm requires multi-dominance discernibility matrices. It means $2d-2$ matrices(assuming there are d different decision class labels in the information table) need to be constructed, and the space of each matrix is $|U1| * |U2|$. For time complexity, let m denotes the number of attributes, in the worst case, it needs at most $m * (|U1| + |U2| + 1)$ operations in order to judge which set contains the new object. On the other hand, updating matrix and generating rules need just $m * max(|U1|, |U2|) + m * (|U1| + 1)$ operations at most. Thus at most $n * m * (2(|U1| + 1) + |U2| + max(|U1|, |U2|))$ operations are needed if n objects arrive incrementally. So, the complexity of the algorithm is polynomial.

4. Experiments and analysis

4.1. Design of experiments

The aim of the experiments is to compare the incremental algorithm with non-incremental algorithms. Beyond comparing the algorithms, we want to find how the distribution of training samples influence the computation and performance of algorithms. Here, the distribution of training samples are expressed by data saturation and data concentration.

With the experiments carried out in this work we want to answer the following questions:

-How does the computational time grow with the increasing number of objects?

-How does the data saturation and data concentration affect the running time? Are the distributions of training samples important for all algorithms?

The following experiments are conducted on a Pentium(R) D-2.8GHz CPU with 512MB main memory running Windows XP. All algorithms are implemented in C# and executed on the Visual Studio.NET 2005.

For simplification, the rule sets generated by all algorithms only contain the decision class unions like $Cl_t^{\geqslant}$ and $Cl_t^{\leqslant}$, not $Cl_t \cup Cl_k$. In these experiments, four algorithms are implemented, including DOMLEM [2], AllRules [3] and other two algorithms proposed in this paper: RIDDM and INRIDDM.

4.2. *Incremental vs non-incremental algorithms analysis on data sets*

Two kinds of data sets(artificial and real) are used to analyze the performance of incremental and non-incremental algorithms. The artificial data sets are randomly generated according to pre-setting parameters, those parameters include the number of objects and attributes. The artificial data sets are used mainly for experiments concerning the efficiency and the affection of algorithms at the time when the distributions of provided data sets are different.

Among the real data sets, *iris*, *pimaindians* and *mushroom* are obtained from the repository of Machine Learning databases at UCI[a], *homeprice* is from the data and story library of CMU[b] and *DOM_DATA* is from paper [2]. These data sets are adapted for multi-criteria classification problems by specifying the preference order on some regular attributes.

Table 2. Characteristics of real data sets used for experiments.

Data set	Number of objects	Number of attributes	decision classes
iris	150	4	3
pimaindians	768	8	2
mushroom	8124	22	2
homeprice	117	7	3
DOM_DATA	17	3	3

[a]http://archive.ics.uci.edu/ml/

[b]http://lib.stat.cmu.edu/DASL/allsubjects.html

Firstly, all algorithms are compared on the real data sets, the details are presented in Table 2. For the data sets *iris*, *pimaindians*, *homeprice* and *DOM_DATA*, the numbers of objects are not large, so we only run non-incremental algorithms on these data sets. The results including the number of generated rules and the computation time are presented in the following two tables.

Table 3. The number of generated rules by non-incremental algorithms on real data sets.

Data sets	DOMLEM	AllRules	RIDDM
iris	8	13	10
DOM_DATA	8	13	9
homeprice	18	21	23
pimaindians	157	7	–

Table 4. The running time(in seconds) of non-incremental algorithms worked in real data sets.

Data sets	DOMLEM	AllRules	RIDDM
iris	0.546	5.312	0.328
DOM_DATA	0.015	0.046	0.001
homeprice	0.937	25.015	0.203
pimaindians	733.78	1784.59	–

Comparing the number of generated rules, RIDDM is not less than DOMLEM, the reason is that RIDDM does not consider post-processing, which we have discussed above. The rules generated by AllRules are more than those by DOMLEM on all data sets except *pimaindians*, because the rules generated by AllRules are based on *basis rule* [3]. For data set *pimaindians*, the objects covered by the 7 "*basis rules*" are much less than those covered by 157 rules generated by DOMLEM. The symbol "–" means that the algorithm has exceeded the accepted resources. There are only 8 attributes and 2 decision class unions in data set *pimaindians*, but there are too many values for each attribute and the number of objects is large, it means the matrices constructed by RIDDM may exceed the available memory resources. This is the limitation of RIDDM. Except the data set *pinaindians*, we can see that the time cost of RIDDM is the lowest on other data sets from Table 4.

Now we consider the performance of incremental algorithm on the large data sets. Table 5 is the running time of all algorithms worked in data set *mushroom*. In order to deal with data incrementally, first 500 objects of *mushroom* are taken as training samples of the first non-incremental process part, and the incremental algorithm is run with 500, 1000 and 1500 additional objects added each time. For those non-incremental algorithms, take all given objects as the input of them.

Table 5. The running time(in seconds) of all algorithms worked in *mushroom*.

	Num	of	objects	
Algorithms	500	1000	1500	2000
DOMLEM	27.70	191.01	570.43	1859.35
AllRules	–	–	–	–
RIDDM	4.56	20.53	44.34	79.03
INRIDDM	0	2.70	3.35	7.03

From the Table 5 we can see that the incremental algorithm INRIDDM is much faster than other non-incremental algorithms. In order to conduct a further analysis on INRIDDM, the algorithm INRIDDM is run on a group of artificial data sets, the size of these sets are $10000, 20000, \ldots, 100000$, with 7 condition attributes and 1 decision attribute. First 100 objects are taken to construct the matrices in the non-incremental process part. The time of computing the 100 objects is too short to be considered in the experiment.

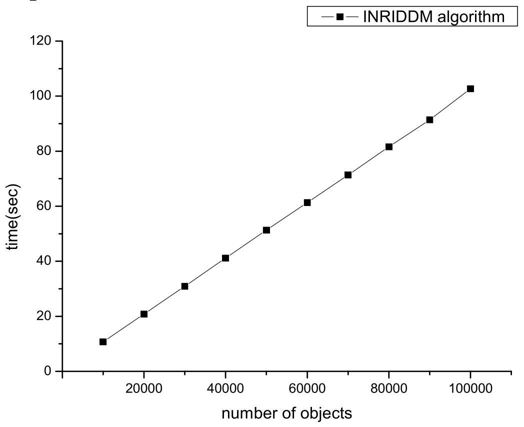

Fig. 1. The running time(in seconds) of INRIDDM on large data sets.

The experiment results detailed in Fig 1 show that the computation complexity of INRIDDM is polynomial. Comparing with DOMLEM, INRIDDM with $n * m * (2(|U1| + 1) + |U2| + max(|U1|, |U2|))$ operations needed is better, while DOMLEM needs at most $n * m * (n + 1) * (m + 1)/4$ operations [2]. The advantage of the incremental algorithm is that INRIDDM just needs to update the matrices when new objects arrive, this is faster than reconstructing new matrices. So, from the above analysis and Table 5 we can come to a conclusion that while dealing with large data set, even small data set, incremental algorithm INRIDDM is the best choice.

4.3. *Incremental algorithm on data saturation and data concentration*

In this section, we try to investigate the relationship between the data saturation, data concentration and the running time of incremental algorithm INRIDDM.

We have known that the time complexity of INRIDDM is $n * m * (2(|U1| + 1) + |U2| + max(|U1|, |U2|))$, it depends on the object number n, the attribute number m and two object sets: $U1$ and $U2$. Generally most algorithms were only concerned with n and m when they conducted a complexity analysis, but the data distribution information was neglected. Here, the data distribution information is denoted by data saturation, data concentration and data redundancy.

For the algorithm INRIDDM, the major operations are to update the existed matrices according to the judgement of whether those new arrived objects belong to $U1$ or $U2$. For saturated data set, the $U1$ and $U2$ of the given data set are remain unchanged when a new object arrives, because the object which has the same value with the new object can be easily found in $U1$ or $U2$. But for non-saturated data set, $U1$ and $U2$ may changed because the new object needs to be inserted into $U1$ or $U2$, or inconsistent objects need to be deleted from $U1$ or $U2$. So, for saturated and non-saturated data set with the same object number and attribute number, the running time of INRIDDM maybe different, and dealing with saturated data is faster than non-saturated data. In order to verify the conclusion, another experiment is conducted, the used data sets are detailed in Table 6.

Table 6. Characteristics of artificial data sets used for experiments, 2 groups of data sets

		saturated data		non-saturated data	
$\|C \cup D\|$	$\|U\|$	$N_{P_1} * N_{P_2} * \cdots * N_{P_i}$	θ	$N_{P_1} * N_{P_2} * \cdots * N_{P_i}$	θ
6	10000	5*7*6*5*2*3	1.0	8*7*6*5*3*3	0.53
6	20000	8*7*6*5*2*3	1.0	8*7*6*5*7*3	0.85
6	30000	8*7*6*5*3*3	1.0	8*7*6*6*6*3	0.83
6	40000	8*7*6*5*4*3	1.0	8*7*7*7*7*3	0.69
6	50000	8*7*6*5*5*3	1.0	8*8*8*8*8*3	0.51

In Table 6, $|C \cup D|$ denotes the number of attributes, $|U|$ denotes the number of objects, N_{P_i} denotes the number of values at i-th attribute, θ denotes the data concentration of data set. The data saturation of the sample with 10000 objects in the saturated data set is $\xi = 5*7*6*5*2*3 = 6300$, respectively, the sample in non-saturated data sets is $\xi = 8*7*6*5*3*3 = 15120$. The experiment result is detailed in Fig 2.

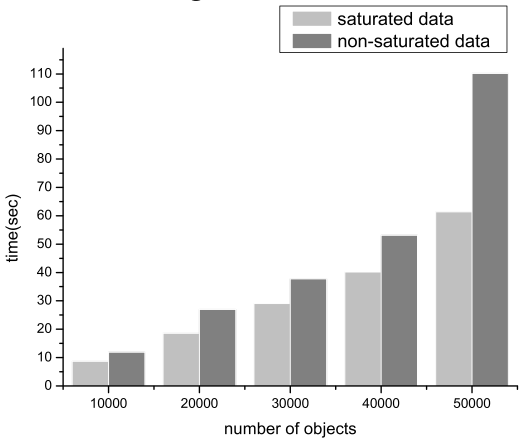

Fig. 2. The computation time(in seconds) of INRIDDM on saturated data sets and non-saturated data sets.

Now we concern the data concentration, the redundancy of data set can be well described by the data concentration [5]. The redundancy means the repetition of decision classes, it is also concerned with $U1$ and $U2$. According to the above analysis on data saturation for INRIDDM, the data redundancy, i.e., another form of data concentration, should also affect the performance of INRIDDM. The followed Table 7 is the detail of other artificial data sets, which are used to test whether the data redundancy affects the algorithm or not. The γ denotes the redundancy of the data set.

Table 7. Characteristics of artificial data sets with different redundancy

$\|U\|$	$\|C \cup D\|$	ξ	γ	θ
50000	6	4100	0.28	0.1
50000	6	4100	0.36	0.2
50000	6	4100	0.44	0.3
50000	6	4100	0.52	0.4
50000	6	4100	0.60	0.5
50000	6	4100	0.68	0.6
50000	6	4100	0.76	0.7
50000	6	4100	0.84	0.8
50000	6	4100	0.92	0.9

Fig 3 shows that INRIDDM needs less time to update matrices when the data redundancy is larger, because the data redundancy is related to $U1$ and $U2$, which are two factors of the time consuming of the incremental algorithm.

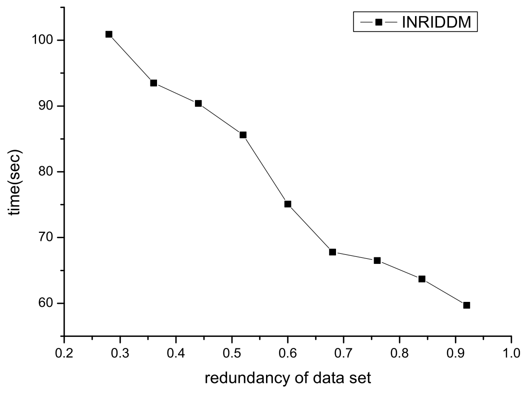

Fig. 3. The computation time(in seconds) of INRIDDM on data sets with different redundancy.

From above analysis and experiments we can answer the second question that the data saturation and data concentration are important for the incremental algorithm INRIDDM in performance, the reason is that they are relevant to $U1$ and $U2$. For those non-incremental algorithms, they don't consider $U1$ and $U2$, so there is not much connection between the data distribution and the algorithm performance.

5. Conclusion

In this paper the rule induction algorithms are studied under the frame of Dominance-based Rough Set Approach. The notion of constructing multi-dominance discernibility matrices to deal with the information table with ordering relations is proposed, which can highlight the differences between two decision classes. And then an incremental algorithm INRIDDM and a non-incremental algorithm RIDDM based on multi-dominance discernibility matrices are introduced. For incremental dealing, when a new object arrives, the updated rule sets can be obtained after updating one row or one column in the matrix. The analysis on INRIDDM and the results of comparing experiments with other non-incremental algorithms show the effectiveness of the INRIDDM. For large data sets or small data sets, the incremental algorithm INRIDDM is a better choice even if it is a non-incremental problem.

Besides the performance analysis of INRIDDM and experiments on it, the relationship between the data distribution and the running time of the incremental algorithm is also explored in this paper. For saturated data, the cost of INRIDDM is less than non-saturated data. If there exists redundancy in data set, it will reduce the running time of INRIDDM. So, we come to the conclusion that the algorithm INRIDDM prefers saturated data with redundancy.

In this paper we only consider generating the classification rules that cover the objects of lower approximations of the decision class unions, the following work will be on studying the rules which covering the different objects and on inducing the type 1 rules in the future.

Acknowledgments

As an extension of the paper published in RSKT2009 Proceedings, the authors thank Prof. Yiyu Yao and the anonymous reviewers for the constructive comments and suggestions. This work is supported by the Natural Science Foundation of Jiangsu Province under Grant No. 2009233.

References

1. Z. L. Chen and G. Q. Chen, “Building an associate classifier based on fuzzy association rules,” *International Journal of Computational Intelligence Systems*, **1-3**, 262–272 (2008).
2. S. Greco, B. Matarazzo, R. Slowinski and J. Stefanowinski, “An algorithm for induction of decision rules consistent with the dominance principle,” *Proc. Intl. Conf. On Rough Sets and Current Trends in Computing*, 304–313 (2000).
3. S. Greco, R. Slowinski, J. Stefanowski and M. Zurawski, “Incremental versus non-incremental rule induction for multicriteria classification,” *Transaction on Rough Sets* **II**, 33–53 (2004).
4. X. Y. Jia, L. Shang and J. J. Chen, “Rule induction algrithom based on dominance-based on rough set approach(in Chinese),” *Journal of Jiangnan University(Natural Science Edition)*, **6(6)**, 686–689 (2007).
5. Y. Liu, C. F. Xu, T. Y. Lin and Y. H. Pan, “Incremental supervised rule mining from inconsistent redundancy data,” *Proc. Intl. Conf. On Data Mining WORKSHOP Foundations of Secmantic Oriented Data and Web Mining*, 46–48 (2005).
6. Z. Pawlak, “Rough Sets, Theoretical Aspects of Reasoning about Data,” Kluwer Academic Publishers, Dordrecht, Boston, London, 1991.
7. Y. Y. Yao and T. Y. Lin, “Generalization of rough sets using modal logic,” *Intelligent Automation and Soft Computing*, **2(2)**, 103–120 (1996).
8. Y. Y. Yao and Y. Sai, “Mining ordering rules using rough set theory,” *Bulletin of International Rough Set Society*, **5**, 99–106 (2001).
9. Y. Y. Yao, “Mining high order decision rules,” *Rough Set Theory and Granular Computing*, 125–135 (2003).
10. Y. Y. Yao, B. Zhou and Y. H. Chen, “Interpreting low and high order rules: a granular computing approach,” *Proc. Intl Conf. On Rough Sets and Emerging Intelligent System Paradigms*, **LNAI 4585**, 371–380 (2007).