Enhanced Rule Induction Using Incremental Approach for a Dynamic Information System

B. K. Tripathy, Kumaran K., M. Sumaithri, and T. Swathi

Abstract— **In the present day scenario, there are large volumes of data available in several fields, which we can make use of effectively, for decision making. This can be achieved by inducing rules through various rule induction approaches that are available. In this paper, we proposed a rule induction algorithm, ELEM, which is an enhanced version of one of the existing rule induction algorithms, LEM1 [3]. This is made effective by reducing the database scans required to generate the rules. Also, it provides an incremental approach which makes use of ELEM and deals with any kind of data changes in a dynamic information system. The incremental technique is a way to solve the issue of added-in data without re-implementing the original algorithm in a dynamic database. In this paper, an incremental rule-extraction algorithm is proposed to resolve therefore mentioned issues. Applying this algorithm, while a new object is added to an information system, it is unnecessary to re-compute rule sets from the very beginning. The proposed approach updates rule sets by partially modifying the original rule sets, which increases the efficiency. This is especially useful while extracting rules in a large database.**

Index Terms— **ELEM, Global cover, Incremental approach, Rule Induction**

I. Introduction

Now-a-days, we are inundated with volumes of data. Business concerns have been accumulating vast amounts of data in accounting, inventory and sales records. Also large amounts data are available on internet. For decades this data has been entered and stored on computers. However, if the training data is viewed as an information system, then the procedures and methods of data mining can be used to find the previously unrecognized relationships in the data that will convert the data to information.

Rough set theory is a new mathematical approach to imperfect knowledge developed by Pawlak (7). The main advantage of rough set theory in data analysis is that it does not need any preliminary or additional information about data. Thus it has gained importance in rule induction.

To obtain meaningful decision rules, we underwent the following stages. Firstly, the data is pre processed. And then the rule induction algorithm ELEM is applied to the pre-processed data. This global cover, also known as relative reduct, based rule induction algorithm generates decision rules, which can reveal profound knowledge and provide new insights. The current traditional approaches do not consider the added-in data and the classification quality of decision tables. This also resulted in numerous studies in incremental approaches (3), (4), and (6). However, the existing incremental approaches still cannot deal with the problems of a large database. Moreover, for dealing with the new added-in data set, these approaches often re-implement the reduction algorithm and rule extraction which results in more computational time and wastage of memory space.

Therefore, to solve this dynamic database problem, an incremental rule extraction algorithm (1) is proposed based on the ELEM. Applying this algorithm, while a new object is added to an information system, it is unnecessary to re-compute rule sets from beginning, instead, we can make use of an incremental approach for the same.

Manuscript received October 9, 2010; revised July 10, 2011.

B. K. Tripathy and Kumaran K., VIT University, Vellore, India, email: tripathybk@vit.ac.in

M. Sumaithri, LAPG ,SISO, Bangalore, India, email: sumaithri.m@samsung.com

T. Swathi, Retail Divison, TCS India, email: thallamswathi89@gmail.com

II. Literature Review

A. Basic Rough Sets

Let U be a universe of discourse, which cannot be empty and R be an equivalence relation or indiscernibility relation [4], [8], [10] over U. By U/R we denote the family of all equivalence class of R, referred to as categories or concepts of R and the equivalence class of an element x $\in$ U is denoted by $[x]_R$. By a knowledge base, we understand a relational system k = (U, R), when U is as above and R is a family of equivalence relation or indiscernibility relation over U and k is called an approximation space. Elementary sets in k are the equivalence classes of R and any definable set in k is a finite union of elementary sets in k.

Therefore for any given approximation space defined on some universe U and having a n equivalence relation R imposed on it, U is partitioned into equivalence classes called elementary sets which may be used to define other sets in k; Given that X $\in$ U, X can be defined in terms of definable sets in k by the following

Lower approximation of X in A is the set

$$\neq \phi \mathbf{X} = \cup\{\mathbf{Y} \in \mathbf{U} \mid \mathbf{R}: \mathbf{Y} \subseteq \mathbf{X}\}$$

Upper approximation of X in A is the set

$$\overline{R}\,\mathbf{X} = \cup\{\mathbf{Y} \in \mathbf{U} \mid \mathbf{R}: \mathbf{Y} \cap \mathbf{X} \neq \phi\}$$

Another way to describe the set approximations is as follows. Given the lower and upper approximations $\underline{R}$X and $\overline{R}$X, of X a subset of U, the R-positive region of X is $POS_R(X)$ and is given by $POS_R(X) = \underline{R}X$, the R-negative region of X is $NEG_R(X)$ and is given by $NEG_R(X) = U - \overline{R}X$, and the boundary or the R-borderline region of X is $BN_R(X)$ and is given by $BN_R(X) = R\,X - \underline{R}\,X$. The elements of $\underline{R}\,X$ are those of U which can certainly be classified as elements of X and the elements of $\overline{R}\,X$ are those elements of U, which can possibly be classified as elements of X, employing knowledge of R. We say that X is rough with respect to R if and only if $\underline{R}\,X \neq \overline{R}\,X$, equivalently $BN_R(X) \neq \phi$. X is said to be R-definable if and only if, $\underline{R}\,X = \overline{R}\,X$ or $BN_R(X) \neq \phi$.

The tuple $\{\underline{R}\,X, \overline{R}\,X\}$ composed of the lower and upper approximations of X is called a rough set, associated with X with respect to R.

B. Rule Induction

Rule induction [2], [3], and [5] is one of the most important techniques of machine learning. Regularities hidden in data are frequently expressed in terms of rules; rule induction is one of the fundamental tools of data mining. Rules are generally in the following form

If (attribute$_1$, value$_1$) and (attribute$_2$, value$_2$) and (attribute$_n$, value$_n$) then (decision, value)

Data from which rules are induced are usually presented in a form similar to a table in which cases (or examples) are labels (or names) for rows and variables are labeled as attributes and a decision. *Attributes* are independent variables and the *decision* is a dependent variable. The set of all cases labeled by the same decision value is called a *concept*. For example, for the table 3.1, the set {1, 2, 4, 5} is a concept of all cases affected by flu (for each case from this set, the corresponding value of Flu is yes).

There are several studies of incremental approach in rough set theory [1, 4 and 5]. However, these previous incremental approaches cannot deal with the problems of a large database. Moreover, for dealing with the new added-in data set, these approaches often re-implement the reduction algorithm and rule-extraction to generate reduces and decision rules. The following table (Table 3.1) shows a simple example of the same. Here, Temperature, Headache, Weakness, Nausea are called *Attributes*, and the decision is Flu. The set of all cases labeled by the same decision value is called a concept. For Table3.1, case set {1, 2, 4, 5} is a concept of all cases affected by flu (for each case from this set the corresponding value of Flu is yes).

III. Solution Approach

A. Data extraction and Attribute Reduction

In this step, we have a database to store the values. Since the database cannot be accessible to everyone. We used the xml query processing. This converts the table into an xml file which is accessible to everyone. The xml file is then converted into a text file using a Windows 32 application. The major problem we face in rule induction is the null values, or missing values or unknown. To avoid this, we can create a web page and get the data from the user, where we apply validation control so that it prevents entering null values and missing values into the database. Thus pre-processing is done and missing values are eliminated. The necessary attributes are then selected as the condition attributes which determine in making a decision.

B. ELEM

This module includes two components

1) Generation of global covering:

To select the best global cover of the existing ones, we can make use of condition indispensible attributes accordingly. Some points for Global covering in incremental approach

- We can generate a list of possible covering with a flag field.
- Starting with the current global cover, as we proceed, if we find a sub cover which is not a global covering we can set the flag.

TABLE I: Decision Table

Case	Attributes				Decision
	Temperature	Headache	Weakness	Nausea	Flu
1	Very_high	yes	yes	no	yes
2	high	yes	no	yes	Yes
3	normal	no	no	no	No
4	normal	yes	yes	yes	Yes
5	high	no	yes	no	Yes
6	high	no	no	no	No
7	normal	no	yes	no	No

- Next time, when we came across sub covers of another global cover, just checking the flag we can discard it.
- This procedure shall reduce the search space.

2) Implementation of ELEM algorithm

- ELEM algorithm firstly calculates the minimal set of attributes that must be present in generating the rule set which is the global covering.
- ELEM then computes the necessary attribute value-pairs and the unnecessary ones are removed and converted into a rule set.

After the implementation of the algorithms, we found ELEM is better equipped in handling the incremental methodology and so we chose ELEM and proceeded for the incremental approach.

C. Incremental Approach

A rough set rule induction algorithm generates decision rules, which can reveal profound knowledge and provide new insights. But these traditional approaches do not consider the added-in data and the classification quality of decision tables. This resulted in numerous studies in incremental rough set theory. However, the existing incremental approaches still cannot deal with the problems of a large database. Also, for dealing with the new added-in data set, these approaches often re-implement the reduction algorithm and rule extraction which results in more computational time and wastage of memory space.

Therefore, to solve this dynamic database problem, a new incremental approach is proposed as follows

The decision table is consists of condition attributes and decision attributes. ELEM algorithm is applied to the decision table to generate rules on the existing records. These rules are stored in a file. When a new data set is added to the decision table, in general, ELEM algorithm is applied on the whole table again to generate the new rules. This seems to be ineffective when many records are added. To resolve this problem, a new approach using ELEM, Incremental Approach, is proposed which is shown above.

Each step in the incremental approach diverges to different cases. We considered all types of cases that can occur when a new data is added. All the cases are explained in the following. We now move on to the algorithm steps of incremental approach.

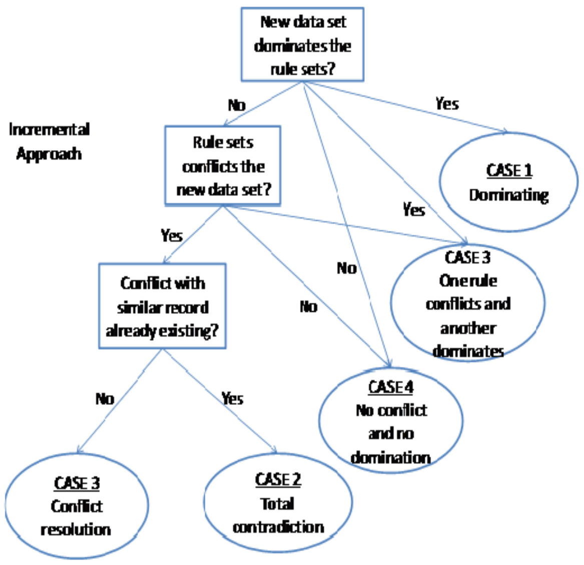
Incremental Approach
New data set dominates the rule sets?
No
Yes
Rule sets conflicts the new data set?
CASE 1 Dominating
Yes
Yes
No
No
CASE 3 One rule conflicts and another dominates
Conflict with similar record already existing?
No
Yes
CASE 4 No conflict and no domination
CASE 3 Conflict resolution
CASE 2 Total contradiction

Fig. 1. Incremental approach

Case 1: New tuple is dominated by the existing rules

If the new data set added is dominated by the existing rules then there is no change in the existing rules and the new tuple is added to the decision table.

Case 2: Total contradiction

Here, the conflict occurs with the existing rule, where there is no change in any of the condition attributes of the rule and the change is only in the decision attribute value. This can be termed as total contradiction. The conflicting rule has to be deleted from the set of rules.

The conflicting rule is placed in a new category named as inconsistent rules. The remaining rules will be called as consistent rules. ELEM algorithm is applied to the new data set and the new rule is generated. This rule has to be placed in the inconsistent rules. In due course time, we use support value of the rules in inconsistent category to reduce their number. This will optimize the number of rules which helps user to consider minimum number of rules.

Case 3: Conflict Resolution

The new tuple conflicts with the rule due to change in the condition attributes of the rule. Now, consider the tuples that are being covered by the conflicting rule. ELEM algorithm is applied combined on these tuples and the new data set. The rules are updated with the new rules generated.

Case 4: No conflict and no domination

The new tuple might have a new attribute value where there is no conflict and no domination. In this case ELEM is applied on the new data set and the rules are generated. Rules are updated with the new rules that are generated.

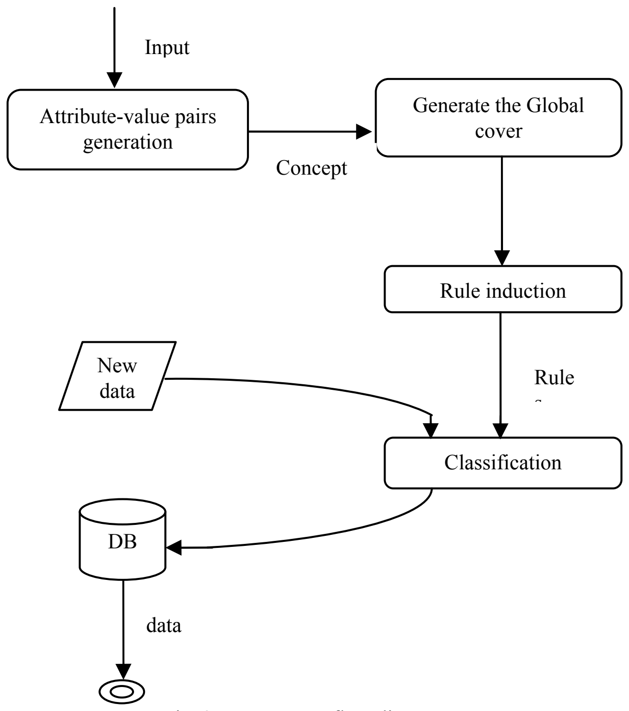

Fig. 2. ELEM Data flow diagram

IV. Proposed Algorithms

A. ELEM

The following gives the algorithm for the ELEM approach.

Notations

A set of all attributes

$\{d\}$ decision attribute

$\{d\}^*$ partition of $\{d\}$

$\{G\}$ global cover where $\{G\}=\{g_1,g_2, \ldots, g_n\}$, $g_1,g_2, \ldots g_n \in A$

g_a attribute name, a= 1, 2, … p

v_{ab} value of the attribute g_a, b= 1, 2, …q

(g_a, v_{ab}) denotes a attribute-value pair

R set of rules generated

Input the decision table with c, condition attributes and d, decision attributes

output rule sets

```
For each tuple in decision table
$R' := \emptyset$ , $G' := G$
While ($k > 1$)
$G' := G' - g_k$
if ( $\cap (g_a , v_{ab})) \le \{d\}^*$ ) $\forall g_a \in G'$ of the tuple then
$G' := G'$
Else
$G' := G' + \{g_k\}$
$K := k-1$
END if
If ($k = 1$)
$R = G'$
Else
$R' = R' + \{g_k\}$
END if
END while
$R = R + \{R'\}$
END for
```

1) Test Cases for ELEM

Notations:

A- Set of all attributes

B- A non-empty subset of A

U- Set of all cases

IND (B) – an equivalence relation on U.

Equivalence classes of IND(B) – elementary sets of B

Consider a decision table (Table I),

A decision $\{d\}$ depends on B, if and only if, $B^* \leq \{d\}^*$

Let B = {Temperature, Headache}

A decision $\{d\}$ depends on B are $B^* = \{\{1\}, \{2\}, \{3, 7\}, \{4\}, \{5, 6\}\}$

A Global covering of {d} is a subset B of A such that $\{d\}$ depends on B and B is minimal in A.

The following is the procedure for finding the global cover.

$\{\text{Temperature, Headache, Weakness, Nausea}\}^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}\}$

$\{\text{Flu}\}^* = \{\{1, 2, 4, 5\}, \{3, 6, 7\}\}$

$\{T, H, W, N\}^* \leq \{F\}^*$

Dropping Temperature,

$\{H, W, N\}^* = \{\{1\}, \{2\}, \{3, 6\}, \{4\}, \{5, 7\}\} \leq F^* \;|$

Dropping headache,

$\{T, W, N\}^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}\} \leq F^*$

So, $\{T, W, N\}$ is accepted.

Dropping weakness,

$\{T, N\}^* = \{\{1\}, \{2\}, \{3, 7\}, \{4\}, \{5, 6\}\} \leq F^* \;|$

Dropping Nausea,

$\{T, W\}^* = \{\{1\}, \{2, 6\}, \{3\}, \{4, 7\}, \{5\}\} \leq F^* \;|$

Therefore, Total covering is $\{T, W, N\}$

Now, consider each case in the table,

Case 1:

(Temperature, very-high) & (Weakness, yes) & (Nausea, no) → (Flu, yes)

Now, Drop (Temperature, very-high)

{1, 4, 5, 7} ∩ {1, 3, 5, 6, 7} = {1} covers cases {1, 5} and {7} i.e, two different concepts.

So, (Temperature, very-high) cannot be dropped.

Drop (Weakness, yes),

{1} ∩ {1, 3, 5, 6, 7} = {1} covers only case 1.

So, (Weakness, yes) can be dropped.

Therefore, (Temperature, very-high) & (Nausea, no) → (Flu, yes)

Drop (Nausea, no),

{1} covers only case 1.

So, (Nausea, no) can be dropped.

Finally the rule is, (Temperature, very-high) → (Flu, yes)

Case 2:

(Temperature, high) & (Weakness, no) & (Nausea, yes) → (Flu, yes)

Drop (Temperature, high),

{1, 4, 5, 7} ∩ {2, 4} = {4} covers case 2.

So, (Temperature, high) can be dropped.

Drop (Weakness, no),

{2, 4} covers {2, 4} cases from same concept

So, (Weakness, no) can be dropped,

Finally, (Nausea, yes) → (Flu, yes)

Case 3:

(Temperature, normal) & (Weakness, no) & (Nausea, no) → (Flu, no)

Drop (Temperature, normal),

{2, 3, 6} ∩ {1, 3, 5, 6, 7} = {3, 6} covers {3, 6} from same concept.

So, (Temperature, normal) can be dropped.

Drop (Weakness, no),

{1, 3, 5, 6, 7} covers {1, 5} and {3, 6, 7} with different concepts

So, (Weakness, no) cannot be dropped.

Drop (Nausea, no),

{2, 3, 6} covers {2} and {3, 6} with different concepts

So, (Nausea, no) cannot be dropped.

Finally, (Weakness, no) & (Nausea, no) → (Flu, no)

Case 4:

(Temperature, high) & (Weakness, yes) & (Nausea, no) → (Flu, yes)

Drop (Temperature, high),

{1, 4, 5, 7} ∩ {1, 3, 5, 6, 7} = {1, 5, 7} covers different concepts.

So, (Temperature, high) cannot be dropped.

Drop (Weakness, yes)

{2, 5, 6} ∩ {1, 3, 5, 6, 7} = {5, 6} covers different concepts.

So, (Weakness, yes) cannot be dropped.

Drop (Nausea, no)

{2, 5, 6} ∩ {1, 4, 5, 7} = {5} covers same concepts.

So, (Nausea, no) can be dropped.

Finally, (Temperature, high) & (Weakness, yes) → (Flu, yes)

Case 5:

(Temperature, normal) & (Weakness, yes) & (Nausea, no) → (Flu, no)

Drop (Temperature, normal),

{1, 4, 5, 7} ∩ {1, 3, 5, 6, 7} = {1, 5, 7} covers different concepts.

So, (Temperature, normal) cannot be dropped.

Drop (Weakness, yes),

{3, 4, 7} ∩ {1, 3, 5, 6, 7} = {3, 7} covers same concepts.

So, (Weakness, yes) can be dropped.

Drop (Nausea, no),

{3, 4, 7} covers different concepts

So, (Nausea, no) cannot be dropped.

Finally, (Temperature, normal) & (Nausea, no) → (Flu, no)

Therefore, the rule sets of ELEM algorithm are:

(Temperature, very-high) → (Flu, yes)

(Nausea, yes) → (Flu, yes)

(Weakness, no) & (Nausea, no) → (Flu, no)

(Temperature, high) & (Weakness, yes) → (Flu, yes)

(Temperature, normal) & (Nausea, no) → (Flu, no)

B. Incremental Approach

The following gives the algorithm for the incremental approach.

Notations

D it is the new data set added.

D_c	data set's condition values
D_d	data set's decision values
P	set of all rules (in general consistent)
P	any rule
ICR	Inconsistent rules
p'	newly added rules
DT	Decision Table
R	a tuple for DT (decision table)

Step 1: Check if new data set conflicts with any existing rules with no change in condition attributes. (i.e. a tuple already exists such that, the change is only in decision attributes) CASE 2 – Total Contradiction

```
for each p in P
if ( $D_c=p_c$ and $D_d \neq p_d$ ) then
for each R in DT
if ( $D_c=R_c$ ) then
goto Step 6
else  goto Step 2
END if
END for
END if
END for
```

Step 2: Check if the new data set conflicts with any existing rules with a change in condition attributes. CASE 3 – Conflict resolution

```
for each p in P
 if ( $D_c=p_c$ and $D_d \neq p_d$ ) then
for each R in DT
if ( $D_c \neq R_c$ ) then
goto Step 7
else  goto Step 3
END if
END for
END if
END for
```

Step 3: Check if the new data set conflicts with one rule and another rule dominates it. CASE 3 – Conflict resolution.

```
flag:=0
for each p in P
if ( $D_c=p_c$ and $D_d=p_d$ ) then
flag:=1
END if
END for
for each p in P
if ( $D_c=p_c$ and $D_d \neq p_d$ ) then
for each R in DT
if ( $D_c \neq R_c$ ) AND (flag=1) then
goto Step 7
else  goto Step 4
END if
END for
END if
END for
```

Step 4: Check if the data set is neither conflicting nor dominated by the existing rules. CASE – 4

```
count:=0
for each p in P
if ( $D_c \neq p_c$ ) then
count++
END if
END for
if ( count= |rules|) then
Goto Step 8
else goto Step 3
END if
```

Step 5: Check if the new data set is dominated by existing rules. CASE – 1

```
for each p in P
if ( $D_c=p_c$ and $D_d=p_d$ ) then
Add the tuple to decision table
END if
END for
```

Step 6: Total Contradiction

Step 6.1: Add p to Inconsistent rules

Step 6.2: Remove the conflicting rule from the set of rules

Step 6.3: Apply ELEM to D, and the new rules:= p'

Step 6.4: Update Inconsistent rules by adding p'

Step 6.5: Add the tuple to decision table

Step 7: Conflict Resolution

Step 7.1: Retrieve the tuples [{*R*}] covered by the conflicting rule p

Step 7.2: Apply ELEM to D + {R}, and the new rules:=p'

Step 7.3: Update the consistent rules by adding p'

Step 7.4: Add the tuple to decision table

Step 8: Neither a conflict nor domination, but the existing rules do not cover the new data set

Step 8.1: Apply ELEM to D, and the new rules:=p'

Step 8.2: Update the consistent rules by adding p'

Step 8.3: Add the tuple to decision table

In the similar way, when a tuple is deleted,

If there is a rule that covers only the deleted record, then it can be removed

else, only the tuple is removed from the table and the rule sets remain intact.

1) Test Cases for Incremental Approach

Consider the TABLE I and now the rule sets P are

1. (Temperature, very high) $\Rightarrow$ (Flu, yes) covers 1st tuple.
2. (Nausea, yes) $\Rightarrow$ (Flu, yes) covers 2nd and 4th tuple.
3. (Weakness, no) & (Nausea, no) $\Rightarrow$ (Flu, no) covers 3rd and 6th tuple.
4. (Temperature, high) & (weakness, yes) $\Rightarrow$ ((Flu, yes) covers 5th tuple.
5. (Temperature, normal) & (Nausea, no) $\Rightarrow$ (Flu, no) covers 7th tuple.

Example for case 1:

S No	Temperature	Headache	Weakness	Nausea	Flu
8	Normal	Yes	No	No	No

Step 5: Dominating existing rules

For each rule p in P

if ((D_c{(Temperature , normal) & (Nausea , no)} = p_c {(Temperature , normal) & (Nausea , no)}) &

(D_d {(flu , yes)} = p_d {(flu , yes)})) i.e, the new data set dominates the existing rules

then

Add 8th tuple to the decision table and no change in the

existing rules.
End if
End for

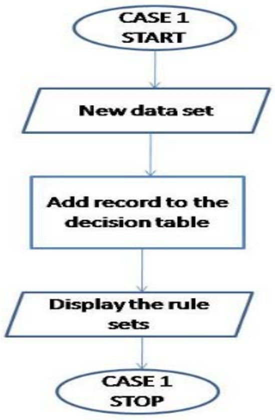

Fig. 3. Case 1

Example for case 2:

S No	Temperature	Headache	Weakness	Nausea	Flu
9	High	Yes	No	Yes	No

Step 1: Total contradiction

Check if new data set conflicts with any existing rules with no change in condition attributes. (i.e, a tuple already exists such that, the change is only in decision attributes)

for each p in P

if ((D_c{(Nausea , yes)} = p_c{(Nausea , yes)}) &

(D_d{(flu , yes)} ≠ p_d{(flu , no)})) i.e, the new data set is conflicted by the 3rd rule in the decision

Attribute, then

for each R in decision table(DT)

if(D_c{(Temperature , high) & (Headache , yes) & (Weakness , no) & (Nausea , no)} =

R_c{(Temperature , high) & (Headache , yes) & (Weakness , no) & (Nausea , no)})

i.e the new data set is totally contradicted by the 2nd tuple and 2nd rule , then

GOTO *step* 6.

Step 6: Total contradiction

Step 6.1 Add 2nd rule to the Inconsistent rules.

Step 6.2 Remove the 2nd rule from the set of rules.

Step 6.3 Apply ELEM to the new data set 9 and generate new rules p′.

Step 6.4 Update the inconsistent rules by adding p′.

Step 6.5 Add 9th tuple to the decision table.

Example for case 3:

S No	Temperature	Headache	Weakness	Nausea	Flu
10	high	yes	Yes	no	no

Step 2: Conflict resolution

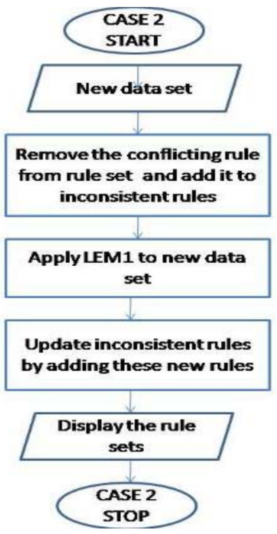
CASE 2 START
New data set
Remove the conflicting rule from rule set and add it to inconsistent rules
Apply LEM1 to new data set
Update inconsistent rules by adding these new rules
Display the rule sets
CASE 2 STOP

Fig. 4. Case 2

Check if the new data set conflicts with any existing rules with a change in condition attributes.

for each p in P

if((Dc{(Temperature,high)&(Weakness,yes)}=p_c{(Temperature , high) & (Weakness , yes)}) &

D_d{(flu , no)} ≠ p_d{(flu , yes)}) i.e., new data set is conflicted by 4th rule since it has same

Condition attributes and different decision attribute which covers 5th tuple, then

for each R in decision table(DT)

if(Dc{(Temperature , high) & (Headache , yes) & (Weakness , yes) & (Nausea , no)} ≠

R_c{(Temperature , high) & (Headache , no) & (Weakness , yes) & (Nausea , no)} i.e, the new data set

has different condition attributes to the 5th tuple , then

GOTO *step* 7.

Step 7: Conflict resolution

Step 7.1 Retrieve 5th tuple covered by the conflicting rule.

Step 7.2 Apply ELEM to new data set and the 5th tuple and generate new rules p′.

Step 7.3 Update the consistent rules by p′.

Step 7.4 Add the new tuple to the decision table.

Example for case 3:

S No	Temperature	Headache	Weakness	Nausea	Flu
11	Very High	No	No	No	No

Step 3: Conflict resolution

Check if the new data set conflicts with one rule and another rule dominates it.

Let us consider flag:=0

for each p in P

if((D_c{(Weakness , no) & (Nausea , no)} = p_c{(Weakness , no) & (Nausea , no)}) &

D_d{(flu , no)} = p_d{(flu , no)}) i.e the new data set is dominated by the 3rd rule , then

Flag:=1

End if

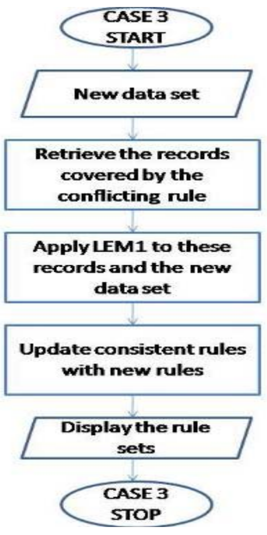
CASE 3 START
New data set
Retrieve the records covered by the conflicting rule
Apply LEM1 to these records and the new data set
Update consistent rules with new rules
Display the rule sets
CASE 3 STOP

Fig. 5. Case 3

if((D_c{(Temperature , very high)} = p_c{(Temperature , very high)}) & D_d{(flu , no) ≠ p_d{(flu , yes)})

i.e, the new data set is conflicted by the 1st rule which covers 1st tuple then ,

for each R in decision table(DT)

if(((D_c{(Temperature , very high) & (Headache , no) & (Weakness , no) & (Nausea , no)} ≠

R_c{(Temperature , very high) & (Headache , yes) & (Weakness , yes) & (Nausea , no)}) AND

(flag:=1)) i.e, the new data set is conflicted by the 1st rule with no similar tuple and flag =1 is

satisfied , then

GOTO *step* 7 which is already described above.

Same procedure has to be followed by retaining the dominated rule and the rules are updated.

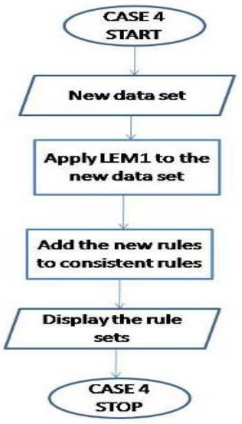
CASE 4 START
New data set
Apply LEM1 to the new data set
Add the new rules to consistent rules
Display the rule sets
CASE 4 STOP

Fig. 6. Case 4

Example for case 4:

S No	Temperature	Headache	Weakness	Nausea	Flu
12	High	No	moderate	no	Yes

Step 4: Check if the data set is neither conflicting nor dominated by the existing rules.

Let us consider count:=0

for each p in P

if((D_c{(Temperature , high) & (Headache , no) & (Weakness , moderate) & (Nausea , no)} ≠ p_c{∅})

i.e no rule dominates or conflicts the new data set , then increment count

count = 5 (checks for all the five rules)

End if

End for

if(count = |rules|) i.e, count(5)=|rules|(5) is satisfied then

GOTO *step* 8

Step 8:

Step 8.1 Apply LEM1 to new data set and generate new rules p′.

Step 8.2 Update the consistent rules by p′.

Step 8.3 Add the new tuple to the decision table.

END

Thus, all the cases are tested and verified with examples as shown above. This shows that the algorithm followed through all the paths as it was specified.

V. Conclusion

In this paper, we proposed an enhanced version of LEM1 i.e. ELEM, a rule induction algorithm which has several advantages over the original algorithm. First, it requires less number of database scans. Secondly, it facilitates in providing an incremental approach which updates the rule sets when there is a change in data in the information system. We also dealt with addition and deletion of tuples in the decision table. However, there is enough scope for improvement in the proposed algorithm. For example, an efficient technique to deal with the problem of generation of global covers when there is a change in data. Also, one may need to have knowledge about how often to check for changes in the Information System which plays a crucial role in implementing the incremental approach.

References

[1] Blaszczynski J, Slowinski R, “Incremental Induction of Decision Rules from dominance-based Rough approximations”, Electronic notes in Computer Science (2003)

[2] Grzymala-Busse J W, ″Rough Set Theory with Applications to Data Mining″, KES Conference Tutorials (2004)

[3] Grzymala-Busse J W, “Rule Induction”, Chapter 1, pp 01-19 in Intelligent Decision support – Handbook of Application and Advances of the Rough set Theory (Ed : Slowinski. R), Volume 11,(1992).

[4] Guo, S Wang, Z Y Wu, Z C & Yan, “A novel dynamic incremental rules extraction algorithm based on Rough set theory”, in the proceedings of the fourth International Conference on machine learning and cybernetics pp 18 – 21.

[5] Slowinski. R., ed. Intelligent Decision support – Handbook of Application and Advances of the Rough set Theory Volume 11 of System Theory, Knowledge Engineering and problem solving, kluwer Academic Publishers. Dordrecht, The Netherland(1992).

[6] YU-Neng Fan, Tzu-Liang(Bill), Ching-Chin Chern, Chun-Che Huang, “Rule induction based on an incremental Rough Sets”, Expert Systems with Applications pp 11439 – 11450 (2009)

[7] Zdzislaw Pawlak, Jerzy Grzymala-Busse, Roman Slowinski, and Wojciech Ziarko, ”Rough Sets” Communications of the ACM, Vol. 38, No. 11 (1995)

[8] Zdzislaw Pawlak and Andrzej Skowron,“Rudiments of Rough Sets” Institute of Mathematics Warsaw University Banacha 2, 02-097 Warsaw, Poland.