A Rule Extraction Based on the Rough Set Theory with Decremental Data

Horng-Fu Chuang
Dept. of Accounting Information
Da-Yeh University
Zhang-Hua, Taiwan, ROC
frank@mail.dyu.edu.tw

Chun-Che Huang
Dept. of Information Management
National Chi Nan University
Puli, Taiwan, ROC
cchuang@ncnu.edu.tw

Ying-Ling Hsieh
Dept. of Information Management
National Chi Nan University
Puli, Taiwan, ROC
s98213544@ncnu.edu.tw

Zhi-Xing Chen
Dept. of Information Management National Chi Nan University
Puli, Taiwan, ROC
s99213514@ncnu.edu.tw

***Abstract*—In a dynamic database (DB), data deletion operations are a frequent feature of database management activities. Unfortunately, most existing DM algorithms assume that the database is static and that updating a database requires re-computation of all the patterns to enable rule extraction. Of the DM techniques available, the rough set (RS) approach is a knowledge discovery tool that can be used to help identify logical patterns hidden in massive data. It is also useful for knowledge discovery, pattern recognition and decision analysis. However, traditional RS approaches cannot produce rules with preferential order and often lack focus. They generate too many rules and cannot guarantee that the decision table is credible. This study proposes a DAREA (Decremental Alternative Rule-Extraction Algorithm) to address the issue of data deleted from the database and to generate preference-based rules, according to a strength index (SI), specifically for the case wherein the desired reducts are not necessarily unique. The algorithm does not need to re-compute rule sets that can quickly generate and complete rules, from the very beginning. Experiments are presented to validate that the proposed approach is superior to the traditional RS approach.**

***Keywords*—Data mining, Dynamic databases, Rough set approach, Decremental Algorithm, Decremental data, Rule induction**

I. Introduction

In the past, researchers usually assumed databases were static, to simplify data-mining problems [1]. In real-world applications, these database change, over time, because of the data insertion, deletion and modification operations that are frequently used in database management activities [2]. If this behavior changes, over time, the continued use of the original system could lead to unacceptable results and produce unacceptable decisions, based on these results [3]. Business databases are dynamic in the sense that (1) the data in the database may be continually updated, over time, so the content and the size can change [4], (2) old data must be deleted from the database [2, 5] and (3) distributed databases are being updated with a new block of data, at regular time intervals [6].

Data deletion is one of the most frequently used operations in many business databases [4]. The data are deleted are referred to decremental data. When some database transactions are deleted, or modified, the content of the database is been updated and this database is referred to as an updated database. Mining the updated database is referred to as decremental mining [4]. It is obviously inefficient and time-consuming to repeatedly perform the data mining algorithm, in order to analyze the whole database, including the decremental data and the original data [5], especially for decremental rule extraction, which extracts a rule from the data source, by comparing the updated database with the data deleted. These database changes occur, over time, because of the deletion of old data operations that are frequently used in database management activities [2]. Unfortunately, most existing data-mining algorithms assume that the database is static and that a database update requires rediscovery of all of the patterns, by scanning the entire data, old and new, for the purposes of mining and rule extraction [4].

In the field of data mining, the rough set (RS) approach can deal with qualitative information, based on an individual object model [7]. Rough set theory was developed by Pawlak [8]. The theory has been extensively used in decision-making, particularly for sorting and classifying problems with multiple criteria [9]. However, up to date, few RS approaches have considered decremental rule extraction, when data is deleted from databases due to out of date [2, 5].

In addition, literature relating to knowledge discovery [10] reveals that using RS induct attributes often generates too many rules and has no focus. These rough set approaches cannot guarantee that the classification of a decision table is credible [11]. Therefore, Tseng et al. [12] proposed the AREA (Alternative Rule Extraction Algorithm) to solve the problem. The AREA (Alternative Rule Extraction Algorithm) discovers preference-based rules, in accordance with the strength index (SI) of the reducts, specifically for the case wherein the desired reducts are not necessarily unique, since several reducts can have the same SI value. Using the AREA, an alternative rule can be defined, which is the rule with identical preference to the original decision rule, but which may be more attractive to a decision-maker than the original one.

To address the issues associated with dynamic DB's, with respect to data deletion and extraction of creditable and alternative rules, this study proposes a DAREA (Decremental Alternative Rule Extraction Algorithm) for dynamic DB's, in which some data can be deleted. The algorism solves the problem of how to extract rules from the data source, for decremental rule extraction, by comparing the updated database and particular data deleted from the original database, rather than implement AREA once again. With the proposed approach, the results of rule extraction correctly reflect the current situation and there is need to re-run the algorithm for rule extraction, to analyze the whole database, after data is deleted.

This paper is organized as follows. Section II describes the proposed approach and provides illustrated examples. Section III details an experiment that compares the application's software to run the two algorithms (AREA and DAREA). Finally, Section IV offers conclusions and suggestions for future research.

II. Solution Approach

A. *Structure of the proposed solution*

The proposed algorithm is based on the reduct generation procedure of Pawlak [13] and the alternative rule extraction algorithm of Tseng *et al.* [12]. The proposed approach updates rule sets by partially modifying the original rule sets.

Since the objects in most databases are always changing (e.g., objects are often added, deleted, or updated), a practicable method for use in real applications must be able to cope with changes of objects [14]. With the decremental data, the final rules may be classified to five cases.

Case 1. No rule Generated; No rule Replaced; No rule Deleted: A deleted object does not result in any rule change.

Case 2. No rule Generated; No rule Replaced; Rule Deleted: A deleted object results in the original rule being deleted.

Case 3. No rule Generated; Rule Replaced; No rule Deleted: A deleted object results in an original rule being replaced by a new rule.

Case 4. Rule Generated; No rule Replaced; No rule Deleted: A deleted object results in a new rule.

Case 5: Hybrid case

Case 5 is a hybrid of cases 1, 2, 3 and 4. In practice, the hybrid case often occurs because a deleted object may result in a combination of the previous 4 cases.

Case 5-1: Rule Generated; Rule Replaced; No rule Deleted The case wherein a deleted object results in the generation of a new rule generation and the original rules are simultaneously replaced by new rules.

Case 5-2: Rule Generated; No rule Replaced; Rule Deleted The case wherein a deleted object results in the generation of a new rule and the original rules are also deleted.

Case 5-3: No rule Generated; Rule Replaced; Rule Deleted The case wherein a deleted object results in the original rules being replaced by new rules and the original rules are simultaneously deleted.

Case 5-4: Rule Generated; Rule Replaced; Rule Deleted The case wherein a deleted object results in generation of a new rule and the original rules are deleted and simultaneously replaced by new rules.

B. *Lemma*

Three lemma are used in the decremental algorithm:

Lemma 1: If $D(X_i)_{Aj}$ - num_{del} = Ø, then a new reduct is generated.

Lemma 2: The new reduct must be different from the previous reduct. ($Reduct_{new} \notin Reduct_{old}$)

Lemma 3: If Temp has not been changed, then only one rule will be reduced, or the rules are not affected. (If Temp=0, then old rule set $\subseteq$ new rule set.)

Notations: $D(X_i)_{Aj}$: the j-th attribute of the object, X_i, column in the difference set table. For example, $D(X_1)_{A2}$ is the column set of object 1's attribute, A_2, in the difference set table. *Temp:* he record that determines if the order of SI has changed and is different from the original;

C. *The algorithm*

The principal elements of the algorithm are presented as follows: Firstly, all parameters are set to null. Then, the deleted objects are selected and their object number is set to numdel. Secondly, if any numdel is in the sets of columns in D, numdel is deleted. Based on Lemma 1, if any column sets become empty, then the reduct generation procedure of Pawlak [13] is used to generate a new reduct of D(Xi)Aj, which is added to reduct set of table (R). Thirdly, the possible reducts affected by numdel in R are identified and deleted. The value of the new temp, T_{new}, is then compared with the value of original temp, T_{old}, to decide whether the rules should extracted, once more. Based on Lemma 3, if T_{new} is different from T_{old}, and then the rules are extracted, again, according to AREA.

Notations:

U: a finite set of objects; A: an attribute set; d: a "decision attribute set"; i: the object index; j: the attribute index; n: the reduct index; l: the value of new level; L: the number of original levels; q: the number of object data; r: the number of attributes; k: the number of reducts in the "reduct set of the table"; S: the case number, which is determined by the decisive attribute and U_i; T_{new}: the number of the sorting SI, according to case number, S, in the new reducts set, in the table; T_{old}: the number of the sorting SI, according to case number, S, in the original reducts set, in the table; *Temp:* the record that determines if the order of SI has changed and is different from the original; X_i: the object number, e.g., object $X_{i\in 1}$=object 1; a_{ij}: the j-th value set of the attribute for object X_i; num_{del}: the number of a deleted object, e.g., if the deleted object is object 1, then num_{del}=1; $I()$: the original information, in the table; $I(X_i)_{Aj}$: the j-th attribute of the object, X_i, column in I, e.g., $I(X_i)_{Aj}$ is X_i's A_j column set in the original information, in the table; $D()$: the difference set of each attribute, A_j, (the equivalent class of each objects), and attributes, d (the equivalent class of each objects, corresponding to a decision), in the table; $D(X_i)_{Aj}$: the j-th attribute of the object, X_i, column in the difference set table, e.g., $D(X_1)_{A2}$ is a column set of object 1's attribute A_2, in the difference set table; E: extension of table; R: reduct set in the table; F: final rules in the table; F_{old}: the set of rules (reducts) selected from the original rules; F_{new}: the set of rules (reducts) selected from the new rules; $F_{generate}$: the set of rules (reducts) selected by generating rules; $F_{replace}$: the set of rules (reducts) selected by replacing rules; F_{delete}: the set of rules (reducts) selected by deleting rules; OC: column of object cardinality, e.g., $I(num_{del})_{oc}$ is num_{del}'s object cardinality column, in the original information in the table; MO: column of merged object No's, e.g., R_{MO} is the merged object column in the reduct set in the table; SO: column of support object No's, e.g., F_{SO} is the support object column in the final decision rules, in the table;

Procedure:

```
Input: The number of a deleted object, $num_{del}$.
Output: The set of decision rules and alternative rules, $F_{new}$.
Step 0    Initialization
          (i). When an object is deleted, set the object number to $num_{del}$.
          (ii).Set $S = 1$, $l=1$, $F_{old}$ = original rules set, $F_{new} = \varnothing$,
          $F_{generate} = \varnothing$, $F_{replace} = \varnothing$, $F_{delete} = \varnothing$.
Step 1    Check if there is any $num_{del}$ in sets of columns in $D$
          and delet $num_{del}$.
          For $i = 1$ to $q$
              For $j = 1$ to $r$
                If $num_{del} \in D(X_i)_{Aj}$ && $D(X_i)_{Aj} - num_{del}$ ==
                    empty
                    Based on Lemma 1, go to step 1.1.
                Else go to step 2
                    End If
              Endfor
          Endfor
Step 1.1  Apply the reduct generation procedure of Pawlak
          [17], to generate a new reduct of $D(X_i)_{Aj}$.
Step 1.2  Check whether the new reduct exists in the $R$.
          For $i = 1$ to $k$
              If new reduct $\in R_i$
                  Based on Lemma 2, go to step
          1.2.1.
              Else go to step 1.2.2.
              End If
          Endfor
Step 1.2.1 Add the new reduct to $R$.
Step 1.2.2 Merge the new reduct with the identified original
          reduct, into $R$. The new reduct of object number
          joins $R_{MO}$ and the cardinality is also added to $R_{OC}$.
Step 2    Check whether the new reduct is better than the
          original.
          For $l = 1$ to $L$
              If a new reduct is not yet generated,
                  Go to step 2.1.
              Else go to step 3.
              End If
          Endfor
Step 2.l  Check whether the intersection of $A_j$ and $D(X_i)$ is
          empty.
          If it is empty
              Go to Step 1.1.
          Else go to step 2.
          End If
Step 3    Find any reduct that is possibly affected by $num_{del}$,
          in $R$:
          For $n = 1$ to $k$
              If $num_{del} \in R_{MO}(n)$
                  Go to step 3.1.
              End If
          Endfor
          Go to step 3.2.
Step 3.1  When the $num_{del}$ is deleted from $R_{MO}$, subtract
          $I(num_{del})_{OC}$ from $R_{OC.}$
Step 3.2  Re-compute the strength index, SI. Sort SI
          according to case number, $S$, and $T_{new}$ is stored, with
          the order, for each reduct.
Step 4    Check whether the new order has not changed, in $R$.
          $Temp=0$
          For $n = 1$ to $k$
              If $T_{new}(n)$ != $T_{old}(n)$
                  $Temp=1$
              End If
          Endfor
Step 5    Decide whether to re-extract the rules, according to
          the value of $Temp$.
          Based on Lemma 3, if $Temp = 1$, the rules must be
          re-extracted, so go To Step 5.1, else go To Step 6.
Step 5.1  Re-extract the rules, according to AREA
          Set $F_{old}$=the original rules，$F_{generate}$=the new rules，
          $F_{replace}$=the replaced rules，$F_{delete}$=the deleted rules.
Step 6    Print $F_{new}= \{F_{old} + F_{generate} + F_{replace} - F_{delete}\}$.
```

D. Time complexity for the proposed decremental algorithm

Using the proposed algorithm, the time complexities for the 5 cases are presented in Table I. The algorithm addresses the problems of deleted data deleted, without the need to re-process all of the data, from the very beginning. Using this approach, the method proposed in this paper not only solves decremental problems, but also decreases the time complexity. As already mentioned, when objects are deleted from the database, using a DAREA algorithm, the time complexity for the worst cases (Case 3 and Case 5) are $O(nm(N_{cor})+q(N_{nr})+r(N_r))$. When objects are deleted from the database with the original AREA algorithm, the time complexity for the worst case is $O(m^2k(N_{cor}) + qk(N_{nr}) + r(N_r))$, as presented in Table I.

TABLE I. THE COMPLEXITY FOR THE PROPOSED ALGORITHM, IN FIVE CASES

Case number	Description	Time complexity in the worst case
1	A deleted object does not result in any rule change.	$O(nm(N_{cor}))$
2	A deleted object results in the original rule being deleted	$O(nm(N_{cor})+r(N_r))$
3	A deleted object results in an original rule being replaced by a new rule.	$O(nm(N_{cor})+q(N_{nr})+r(N_r))$
4	A deleted object results in a new rule.	$O(nm(N_{cor})+q(N_{nr}))$
5	A hybrid case of Case 1, 2, 3 and 4.	$O(nm(N_{cor})+q(N_{nr})+r(N_r))$
Original AREA algorithm	Without the solution to decremental data.	$O(m^2k(N_{cor}) + qk(N_{nr}) + r(N_r))$

n: total number of original objects; m: total number of attributes; r: total number of rules; q: total number of reducts, after removal; k: total number of objects, after removal; N_{cor}: total number of original reducts from the object that generates the rule (or reduct); N_{nr}: total number of reducts, from the new data set; N_r: total number of rules, from the rule set;

Comparing these two complexities, it is obvious that the proposed algorithm is more efficient. Since the complexity of the decremental algorithm is $O(n^2)$, the worst case for the decremental algorithm must have less complexity than the original AREA algorithm, which is $O(n^3)$. Therefore, the proposed algorithm is much more efficient than that without a decremental AREA algorithm.

III. EXPERIMENTS

Firstly, the traditional AREA algorithm is used and the CPU computation time for deletion of data is measured. The unit of CPU computation time is milliseconds (ms). O*bject No.* represents the data variables. The number of objects varies from 1 to 1000. The attribute variable (*Ai)* represents the number of attribute variables. The number of attributes varies from 1 to 20. The *DB* variable represents the original database, and *db* variables represent the objects that are assumed to be deleted from *DB*. |*DB*| and |*db*| denote the size of *DB* and *db*, respectively.

In order to compare the efficiency, using AREA and DAREA, the number of objects in the database is 100, 200, 400, 500, 700 and 1000 and the ratios of data deleted from the database are *0.01*DB, 0.1*DB, 0.4*DB* and *0.7*DB*. The rules generated and the CPU computation time required for re-extracting the rules in Table II, after data deletion were measured. At first, the number of the objects and the number of the attributes were set and then random numbers were automatically generated by the system. Finally, the AREA rules were re-extracted.

A database, *A5.D100,* was generated, as *DB,* and the ratio of data deleted, *db,* was 1%. The AREA button was then pressed, to re-extract the rules and the execution screen showed the CPU computation time, after re-extracting the rule. The CPU computation time is presented in Table II. Data deletion was then performed, for *db=0.1*|*DB*|, so *db* = 10. The AREA button was pressed, to re-extract the rules. The CPU computation time is presented in Table II. Data deletion continued, in a similar fashion, until the data deleted was *db=0.7*|*DB*|, so *db* =70.

TABLE II. USING AREA TO RE-EXTRACT RULES (WITH ATTRIBUTES=5)

DB	1%|DB|	10%|DB|	40%|DB|	70%|DB|
100	286	246	171	134
200	721	595	361	185
400	2309	1815	959	400
500	3397	2685	1326	547
700	6462	5088	2501	895
1000	13257	10233	4635	1453

Secondly, the CPU execution time, for data deletion, was measured for the decremental algorithm, DAREA. O*bject No.* represents the data variables. The number of objects varies from 1 to 1000. The attribute variable (*Ai*) represents the number of attribute variables. The number of the attributes varies from 1 to 20. The *DB* variable represents the original database and the *db* variables represent the objects that are assumed to have been deleted from *DB*. |*DB*| and |*db*| denote the size of *DB* and *db*, respectively.

The number of the objects in the database is 100, 200, 400, 500, 700 and 1000, and the ratios of data deleted from the database are *0.01*DB, 0.1*DB, 0.4*DB* and *0.7*DB*. The rules generated and the CPU computation time required to re-extract rules, after data deletion, are recorded in Table III. Firstly, the number of the objects and the number of the attributes were set and then random numbers were automatically generated by the system. Finally, the AREA rules were re- extracted.

The same database, *A5.D100,* is used, as *DB,* and the ratio of data deletion, as *db*, was 1%. The DAREA button was then pressed, to re-extract the rules, and the execution screen shows the CPU computation time, for re-extracting the rule. The CPU computation time is presented in Table III.

Data deletion was then performed, for *db=0.1*|*DB*|, so *db* = 10. The DAREA button was pressed, to re-extract the rules, and the CPU computation time is presented in Table III. Data deleting continued in a similar fashion, until the data deleted was db=0.7|DB|, so *db* = =70

TABLE III. USING DAREA TO RE-EXTRACT RULES (WITH ATTRIBUTES=5)

DB	1%|DB|	10%|DB|	40%|DB|	70%|DB|
100	112	110	114	81
200	173	189	203	91
400	403	360	226	143
500	423	385	238	141
700	667	594	350	173
1000	1120	866	481	196

With reference to the computation times in Tables II and III, the line chart, below, shows a comparison of the CPU computation times for the AREA and DAREA. The values of |*DB*| are 100, 200, 400, 500, 700 and 1000 respectively.

Fig. 1 shows the ratio of computation times, for DAREA and AREA. The CPU execution time of AREA is divided by the CPU execution time of DAREA. The results show that DAREA performs much better than AREA, for several ratios of data deletion from the database, and that the efficiency of the computation increases, as the number of objects increases from 100 to 1000.

If |*db*| = *1%|DB|*, *10%|DB|*, *40%|DB|* and *70%|DB,|* respectively, then the figure represents the ratio of AREA to DAREA CPU computation times.

Fig. 1 shows that DAREA performs 5 to 9 times better than AREA, when the number of the objects is 700, and that DAREA performs 7 to 11 times better than AREA, when the number of the objects is 1000.

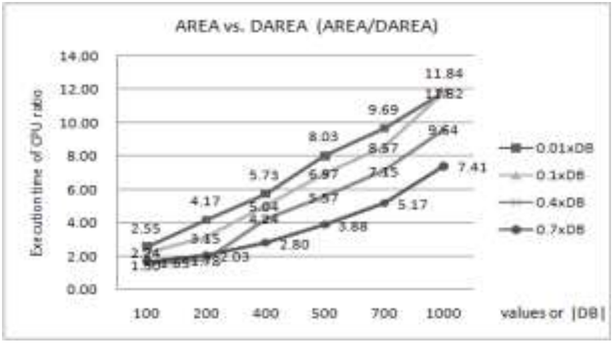

Figure 1. A comparison of the ratio of AREA to DAREA CPU execution times

IV. CONCLUSIONS

This study reviewed literature related to traditional rough set approaches and decremental techniques and presented the drawbacks of previous studies. An decremental alternative rule-extraction algorithm was proposed, based on the AREA of Tseng *et al.* [12], to address the aforementioned drawbacks, which require re-processing of the entre database, when objects are deleted. Illustrative example cases were then presented, to show how the approach searches for solutions. Finally, experiments proved the proposed approach to be superior to the traditional approach. The contributions of the paper are summarized, as follows:

- In a dynamic database, data deletion operations are frequently used, in database management activities. When an object is deleted from the dynamic DB, it is unnecessary to re-extract the rules, by re-computing with AREA, from the beginning; the proposed approach updates reduct sets, using a difference set, thereby reducing computation time.
- The proposed decremental Alternative rule-extraction algorithm is based on the AREA of Tseng *et al.* [12], so it o takes advantage of the Tseng method [11], to identify preference-based rules, according to the strength indices of reducts, and guarantees that the classification of a decision table is credible.
- The AREA can select more than one of the maximum SI's. Therefore, this study, addresses the problem of incomplete rules.

REFERENCES

[1] T.-P. Hong, C.-Y. Wang, and S.-S. Tseng, "An incremental mining algorithm for maintaining sequential patterns using pre-large sequences," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7051-7058, 2011.

[2] S. Zhang, J. Zhang, and C. Zhang, "EDUA: An efficient algorithm for dynamic database mining," *Information Sciences*, vol. 177, no. 13, pp. 2756-2767, 2007.

[3] F. Crespo, and R. Weber, "A methodology for dynamic data mining based on fuzzy clustering," *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 267-284, 2005.

[4] S. Zhang, J. Zhang, and Z. Jin, "A decremental algorithm of frequent itemset maintenance for mining updated databases," *Expert Syst. Appl.*, vol. 36, no. 8, pp. 10890-10895, 2009.

[5] F. A. Sohel, and C. M. Rahman, "Association Rule Mining in Dynamic Database using the Concept of Border Sets," *Asian Journal of Information Technology*, vol. 3, pp. 508-515, 2004.

[6] M. E. Otey, C. Wang, S. Parthasarathy *et al.*, "Mining Frequent Itemsets in Distributed and Dynamic Databases." pp. 617-620.

[7] A. Kusiak, "Feature transformation methods in data mining," *Electronics Packaging Manufacturing, IEEE Transactions on*, vol. 24, no. 3, pp. 214-221, 2001.

[8] Z. Pawlak, "Rough Sets," *International Journal of Information and Computer Sciences*, vol. 11, no. 5, pp. 341-356, 1982.

[9] Q. Yuhua, L. Jiye, and D. Chuangyin, "Incomplete Multigranulation Rough Set," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 40, no. 2, pp. 420-431, 2010.

[10] P. V. Gorsevski, and P. Jankowski, "Discerning landslide susceptibility using rough sets," *Computers, Environment and Urban Systems*, vol. 32, no. 1, pp. 53-65, 2008.

[11] T.-L. B. Tseng, "Quantitative Approaches for Information Modeling," University of Iowa, Iowa city, 1999.

[12] T.-L. B. Tseng, C.-C. Huang, and J. C. Ho, "Autonomous Decision Making in Customer Relationship Management: A Data Mining Approach."

[13] Z. Pawlak, "Rough Sets: Theoretical Aspects of Reasoning about Data," *Rough Sets: Theoretical Aspects of Reasoning about Data*, Boston.: Kluwer Academic Publishers, 1991.

[14] N. Zhong, J.-Z. Dong, S. Ohsuga *et al.*, "An incremental, probabilistic rough set approach to rule discovery." pp. 933-938.