A parallel algorithm for knowledge discovery system

Jakub Wróblewski

Institute of Mathematics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
e-mail: jakubw@jakubw.pl
http://www.jakubw.pl/about

Abstract.

A rough set based knowledge discovery system is presented. The system is based on the decomposition of large data tables into smaller ones in such a way that the approximation of global decision algorithm (related to the whole table) from local ones (related to these smaller tables) can be obtained. The set of these smaller tables can be considered separately and the rules can be calculated in parallel. Then, a set of locally generated rules can be collected for achieving the sufficient approximation of the global decision algorithm. The so called "templates" are presented as the efficient generators of the smaller tables.

On the other hand, a rule generator for one table can be implemented in parallel too. A system for efficient rule generating based on the notion of reduct is presented. A fast, parallel algorithm for short reduct finding is described. Moreover, since a genetic algorithm is used as a driving force of the reduct generator, it can be implemented in parallel in natural way.

A parallel system of reduct finding is described and discussed. Experiment results on large data tables are presented.

1 Introduction

Many problems connected with rule generation from data is NP-hard, so there is no easy method to solve them in deterministic and complete way in reasonable (non-exponential) time. One must use approximate methods such as fast heuristics or evolutionary algorithms. These methods are developed and successfully implemented in knowledge discovery and expert systems. Unfortunately, these methods are still too time-consuming, especially for large data tables. The method to manage this situation is to perform parallel or distributed computations.

Rough set expert systems base on the notion of a *reduct* ([7], [8]), a minimal subset of attributes which is sufficient to discern between objects with different decision values. A set of short reducts can be used to generate rules ([1]). A problem of short reducts generation is NP-hard, but an approximate algorithm (like the genetic one described in [9], [4] and implemented successfully - see [6]) can be used to obtain reducts in reasonable time. On the other hand, rules generated basing on reducts are often too specific and cannot classify new objects. Another types of reducts have been considered to improve efficiency on new objects (see [2]).

One of the methods is to calculate reducts basing on a single object. Let $A = (U, A \cup \{d\})$ be an *information system* (see [8]), where U - set of objects, A - set of attributes, d - decision.

Definition: A *local reduct* $R(o_i) \subseteq A$ (or a *reduct relative to decision and object* $o_i \in U$; o_i is called a *base object*) is a subset such that:

a) $\forall\ o_j \in U,\ d(o_i) \neq d(o_j) \Longrightarrow \exists\ a_k \in R$: $a_k(o_i) \neq a_k(o_j)$

b) R is minimal with respect to inclusion.

A rule generated by a local reduct is concerned with the base object and may not recognize any other object from U. To assure that a set of rules will recognize (at least) all objects from the training set, we have to generate a local reduct for every object. A fast approximation algorithm for local reducts generation is presented in [12]. Although very efficient for small and medium sizes of database, this algorithm is not suitable for large ones (with millions of records) because of computation time.

2 Decomposition problem

Our approach (see [4]) is based on the decomposition of large data tables into smaller ones in such a way that the approximation of global decision function (related to the whole table) from local ones (related to these smaller tables) can be obtained. The set of these smaller tables can be treated as a set of generators that used together with appropriate operators, like grouping, generalization, contraction, can be applied for achieving the sufficient approximation of the global decision function.

The simplest way to perform such a decomposition is to do it randomly. This method is relatively fast and gives satisfactory results (supposing the random sample is large enough). On the other hand, in some cases (e.g.

when the database is accessible only using SQL) random sampling may be hard.

In [3] the so called "templates" was proposed to use as the decomposition generators. Templates can be described as conjunctions of "attribute=value" expressions. We look for templates with high quality e.g. characterized by the number of objects supporting a template times the number of attribute=value pairs describing the template. This approach allow us to decompose a given universe into a family of relatively large subsets of objects sharing many common features. Hence these subsets can be treated as subdomains. One can expect to find strong regularities (rules) for these subdomains. In [3] a few algorithms for template generation are presented. Results show, that expert system with decomposition based on templates in some cases gives better classification algorithm than generated from the whole table.

3 Parallelization

An algorithm presented in [12] realizes the following objective: assuming the information system is consistent, find a family of subsets R_1, R_2,... R_k such that for any object o_i from U at least one R_j is a local reduct (we will say, that R_j covers o_i). We will look for possibly small family R_1,... R_k, i.e. we will prefer these subsets which cover possibly many objects. We assume, that these subsets reflect regularities in data and generate more general rules - it means better classification of new samples and less memory required to store rules.

1. Let σ be a random permutation of attributes.
2. Let $R = A$ and N_1,... N_n - a table of numbers of local reducts found for each object. Set $N_j = 0$.
3. Test whether R is a local reduct for any object (the method of determining for which objects R is superreduct is described in [12]). If so, increment N_i for these objects and store rules.
4. Let $R = R - a_i$, where a_i - the first attribute from R. Calculate a number M_i of these objects, for which R is a (super)reduct, and which are not covered by reducts found previously. Let $R = R + a_i$.
5. Continue step 4. with the next attribute from R. Finish after collecting numbers M_i for all attributes.

6. Find the maximal number among M_i; if there are more than one such a number - get the first one with respect to the permutation σ. Let a_j - an attribute associated with this maximum. Let $R = R - a_j$. Continue from 3. until R is empty.

7. If there is at least one uncovered object - let $R = A$, continue from 4.

In general, this algorithm can be parallelized in three ways:

- We can perform the check from step 4 of algorithm in parallel (for all i). Unfortunately, in this case we cannot use some optimization techniques described in [12].
- We can test several permutations (step 1) in parallel.
- We can implement this algorithm sequentially and run it in parallel for many subtables (generated randomly or using templates). This way of parallelization is the most natural and easy to implement. Moreover, in this case the information exchange between processes seems to be minimal.

4 Implementation and experimental results

A system with simple (random) table decomposition and local reduct-based rule generator was implemented as experimental tool. We have considered two ways of implementation: on Hitachi SR2201 parallel computer (8 processing units), and on a number of PCs connected with LAN. The first solution assures high communication speed between tasks, the second one is more scalable, cheaper and easier to use in practice. Moreover, since our algorithms use huge amount of integer computations, we cannot utilize pseudo-vector processing capabilities of SR2201. Experiments show, that in our kind of applications one processor of SR2201 acts much worse than fast PC (computation time of one program run: SR2201 - 56 sec, Pentium 166 - 45 sec, PentiumII 300 - 29 sec.). Since we do not need to exchange any information between processes except initial data and results, the communication speed is not very important.

We have implemented our system on a number of PCs connected with LAN. One PC acts as decomposition server: it has a direct access to data source and produces random samples (subtables). These samples are sent by LAN (in compressed form) to client machines where a rule generator (based on fast, sequential algorithm for local reducts finding) is implemented. The results are sent to another machine, where rule sets are combined.

We have used medium-size database (220 000 records) in our experiments. The direct analysis of such database with local reduct rule generator is possible, but very time-consuming: it takes more than 4 hour (Pentium 200) to obtain results. We have three profits from random decomposition: first, since rule generation has time complexity of $n \log(n)$, the sum of computation time on subtables is less than the time for large table; second, from technical reasons (virtual memory mechanisms, caching etc.) the less memory we occupy, the faster computations we perform; third, we can do it in parallel. Below we present results of computation time when analysis process is implemented in sequential way. The computation time (Pentium 200) includes database random decomposition, compressing and decompressing subtables, local reducts finding and rules generation, saving results on disk.

Sub. size (obj.)	Sub. size	No.sub.	Dec.time	Analysis time	Total
No decomp.					4h 10min
10 000	233 KB	22	11min 53s	35s	24min 43s
30 000	698 KB	7	4min 26s	3min 40s	31min 6s

Sub. size (obj) - size of one subtable in objects;
Sub. size - size of compressed subtable;
No.sub. - number of subtables the database was decomposed into;
Dec.time - random decomposition time and subtable compressing;
Analysis time - time of local reduct generation for one subtable;
Total - total time of computation in sequential way.

Results show, that decomposition and parallel implementation of rules generation system can speed up computations many times. A size of subtables the database is decomposed into, depends on the number of available machines. E.g. when we have only two machines, total time in case of 10000 objects in subtable is about 12 minutes and in case of 30000 objects it is about 16 minutes. On the other hand, with 8 machines we have about 12 minutes and 8 minutes respectively.

5 Conclusions and future work

A parallel system of rules generation from large databases was presented. Results show, that computation time is acceptable, even when hardware background is cheap and not very complicated (a few PCs connected with LAN).

Some topics, as usage of template-based decomposition and efficient methods for combining results, needs more research and experiments. A module for database access via SQL (in decomposition server) is to be implemented too.

5.1 Acknowledgment

This work was supported by Polish State Committee for Scientific Research grant #8T11C01011 and Research Program of European Union - ESPRIT-CRIT2 No. 20288.

References

[1] Bazan J., Skowron A., Synak P., 1994. *Dynamic reducts as a tool for extracting laws from decision tables*, Proc. of the Symp. on Methodologies for Intelligent Systems, Charlotte, NC, October 16-19, 1994, Lecture Notes in Artificial Intelligence 869, Springer-Verlag, Berlin 1994, 346-355, also in: ICS Research Report 43/94, Warsaw University of Technology.

[2] Bazan J., 1998. *A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Tables.* In: L. Polkowski, A. Skowron (eds.). Rough Sets in Knowledge Discovery. Physica Verlag, 1998.

[3] Nguyen S. H., Polkowski L., Skowron A., Synak P., Wróblewski J.,1996. *Searching for Approximate Description of Decision Classes*, Proc.of The Fourth International Workshop on Rough Sets, Fuzzy Sets and Machine Discovery, RSFD'96, November 6-8, 1996, Tokyo, Japan, pp:153-161.

[4] Nguyen S. H., Skowron A., Synak P., Wróblewski J., 1997. *Knowledge Discovery in Databases: Rough Set Approach.* Proc. of The Seventh International Fuzzy Systems Association World Congress, vol. II, pp. 204-209, IFSA97, Prague, Czech Republic.

[5] Nguyen H. S., Nguyen S. H., 1998. *Discretization Methods in Data Mining.* In: L. Polkowski, A. Skowron (eds.). Rough Sets in Knowledge Discovery. Physica Verlag, 1998.

[6] Øhrn A., Komorowski J., 1997. *Rosetta - A rough set toolkit for analysis of data.* Proc. of Third International Join Conference on Information Sciences (JCIS97), Durham, NC, USA, March 1 - 5, 3 (1997), pp. 403-407.

[7] Pawlak Z., 1991. *Rough sets: Theoretical aspects of reasoning about data.* Kluwer: Dordrecht 1991.

[8] Skowron A., Rauszer C., 1992. *The Discernibility Matrices and Functions in Information Systems.* In: R. Slowiński (ed.): Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory. Kluwer: Dordrecht 1992, pp: 331 - 362.

[9] Wróblewski J., 1995. *Finding minimal reducts using genetic algorithms.* Proc. of the Second Annual Join Conference on Information Sciences, pp.186-189, September 28-October 1, 1995, Wrightsville Beach, NC. Also in: ICS Research report 16/95, Warsaw University of Technology.

[10] Wróblewski J., 1996. *Theoretical Foundations of Order-Based Genetic Algorithms.* Fundamenta Informaticae, vol. 28 (3, 4), pp: 423-430. IOS Press, 1996.

[11] Wróblewski J., 1998. *Genetic algorithms in decomposition and classification problem.* In: L. Polkowski, A. Skowron (eds.). Rough Sets in Knowledge Discovery. Physica Verlag, 1998.

[12] Wróblewski J., 1998. *Covering with reducts - a fast algorithm for rule generation.* Proc. of the International Conference on Rough Sets and Current Trends in Computing, RSCTC'98, pp: 402 - 407. Springer-Verlag (Lecture Notes in AI 1424), 1998.