Adaptation of decomposition tree to extended data

Piotr Synak
Polish-Japanese Institute of Information Technology
Koszykowa 86, 02-008 Warsaw, Poland
email: synak@pjwstk.waw.pl

Abstract

We investigate the problem of large data tables decomposition when the amount of data is being increased in time. For this purpose we apply binary decomposition tree which should be built on the base of time dependencies in data. In the paper we try to deal with the problem of rebuilding an existing decomposition tree in case the data was extended. We prefer to adapt the tree to the new situation, instead of building it from the beginning, since the proces of tree generation is much time consuming.

Keywords: data mining, increase of data in time, decomposition of large databases, rough sets.

1 Introduction

One of the main problems in modern data analysis is size of databases and amount of information to be processed. Many techniques try to deal with this problem - one of them is data decomposition, which leads to partitioning of data into compact, homogeneous and manageable parts. There are many different attempts to decomposition, one of them is based on decision trees, where one try to find repeatedly optimal split condition that separates objects of different kind. Process of such decomposition tree generation is quite complex and needs to be repeated each time a database is updated with new records. From the other side many real problems are described by data which size grows in time. In such a case the decomposition may need to be computed several times what, again, is complex and time consuming.

In [14] we proposed a method of building decomposition tree stable in time. In this paper we investigate the problem of decomposition tree adaptation to extended data. We present a method for fast modification of existing tree, so it is also valid for new information, given by new cases.

2 Basic notions

An *information system* [10] is a pair $\mathbf{A} = (U, A)$, where U is a non-empty, finite set called the *universe* and A is a non-empty, finite set of *attributes*. Each $a \in A$ corresponds to function $a : U \to V_a$, where V_a is called the *value set* of a. Elements of U are called *situations*, *objects* or *rows*, interpreted as, e.g., cases, states, patients, observations.

A special case of information systems is a *decision table* $\mathbf{A} = (U, A \cup \{d\})$, where $d \notin A$ is a distinguished attribute called *decision*. The elements of A are called *conditional attributes (conditions)*.

Suppose, that the set V_d of values of the decision d is equal to $\{1, \ldots, r(d)\}$, for some positive integer $r(d)$ called *the range of* d, the decision d determines the partition $\{C_1, ..., C_{r(d)}\}$ of the universe U, where $C_k = \{u \in U : d(u) = k\}$ for $1 \leq k \leq r(d)$. The set C_k is called the *k-th decision class* of $\mathbf{A}$.

Any expression of form $a \in v$, where $a \in A \cup \{d\}, v \subseteq V_a$ we call a *descriptor*. A descriptor $a \in v$ is a *conditional descriptor* iff $a \in A$, and *decision descriptor* iff $a \in \{d\}$.

We say that an object $x \in U$ *matches* a descriptor $a \in v$ iff $a(x) \in v$.

A *decision rule* is any expression of form $p \Rightarrow q$ iff p is a Boolean combination of conditional descriptors and q is a decision descriptor.

A *binary decomposition tree* for decision table $\mathbf{A} = (U, A \cup \{d\})$ is a binary tree $T = (V, E)$, where V is a set of vertices (nodes), E is a set of edges, and:

1. each vertex has a corresponding sub-table $\mathbf{B} \subseteq \mathbf{A}$

2. *root* corresponds to $\mathbf{A}$
3. each vertex $v \in V$, such that v is not a leaf of T, has a corresponding conditional descriptor p_v (*split condition*)
4. for each vertex $v \in V$, such that v is not a leaf of T, if v corresponds to $\mathbf{B} = (U_{\mathbf{B}}, A \cup \{d\})$ then left child of v corresponds to $\mathbf{B}_l = (\{x \in U_{\mathbf{B}} : x \text{ matches } p_v\}, A \cup \{d\})$ and right child to $\mathbf{B}_r = (\{x \in U_{\mathbf{B}} : x \text{ matches } \neg p_v\}, A \cup \{d\})$

A *level* of a vertex v is the number of vertices in the path from root vertex to v.

3 Formulation of the problem

We consider the case, that amount of data can grow in time what can make our decomposition to be invalid or not close to the optimal one. Generation of decomposition, each time data are changed, can be not possible from practical point of view (see e.g. [8]) - one would prefer to generate decomposition valid also for some extension of data, even if it is not the optimal one. That addresses the first problem which was investigated in [14]:

Problem 1: *How to find time dependencies in a database and how to apply them to the process of decomposition?*

Second problem is related to low-cost update of existing decomposition after the amount of data was increased:

Problem 2: *How to modify the existing decomposition, so it expresses new facts included in new data, without rebuilding it from the very beginning?*

In this case we want our decomposition to adapt to the new situation without it's re-computation. Here, we deal between fast-and-good and slow-and-best solutions.

In many cases the process of slight modifications of the existing decomposition may not last too long. Sometimes, however, the data may be changed so much, that we really need to create new decomposition. Let us formulate the third problem in the following manner:

Problem 3: *When shall we stop updating our decomposition and build completely new one?*

This problem is very important, since we need to observe how data change in time and what are the symptoms of significant change of the situation described by these data. In many cases, if such a change occurs, the quality of rebuilt decomposition may be incomparably better.

4 Generation of decomposition tree

The process of decomposition tree generation is based on recursive search for optimal split. For each leaf we search for conditional descriptor which generates a pair of child nodes, optimal with respect to a quality function corresponding to a given problem. The process stops if all leaves are of satisfactory quality, e.g. size of a universe corresponding to given leaf exceeds given threshold, or the distribution of decision attribute in a given leaf is good enough.

Now, let us define the split quality function. We expect the split to separate different decision classes and give leaves of similar size, i.e. for a given vertex and decision table $\mathbf{A} = (U, A \cup \{d\})$ related to this vertex, we search for descriptor p such that the following quality function:

$$Q(\mathbf{A}, p) = l(\mathbf{A}, p) \cdot r(\mathbf{A}, p) - \sum_{i=1}^{r(d)} l_i(\mathbf{A}, p) \cdot r_i(\mathbf{A}, p) \quad (1)$$

takes maximal value, where

- $l(\mathbf{A}, p)$ is the number of objects of $\mathbf{A}$ satisfying condition p (i.e. number of objects of the left child of given vertex);
- $r(\mathbf{A}, p)$ is the number of objects of $\mathbf{A}$ satisfying condition $\neg p$ (i.e. number of objects of the right child);
- $l_i(\mathbf{A}, p)$ is the number of objects of $\mathbf{A}$ belonging to decision class i and satisfying condition p;
- $r_i(\mathbf{A}, p)$ is the number of objects of $\mathbf{A}$ belonging to decision class i and satisfying condition $\neg p$.

Above definition may be interpreted in the manner of indiscernibility relation [10], [11] as the number of pair of objects that are discerned by split p. In other words it is the number of conflicts that are resolved by p. To compare qualities of different vertices of the tree (also from different levels), we need to normalize the quality function. We can refer $Q(\mathbf{A}, p)$ to the number of all pairs of objects to be discerned [6] (number of all conflicts to be resolved):

$$Conflict(\mathbf{A}) = \frac{1}{2}\left(|U|^2 - \sum_{i=1}^{r(d)} |C_i^{\mathbf{A}}|^2\right) \quad (2)$$

where $C_i^{\mathbf{A}}$ is i-th decision class of $\mathbf{A}$. Thus, the normalized quality of split p of table $\mathbf{A}$ we define as:

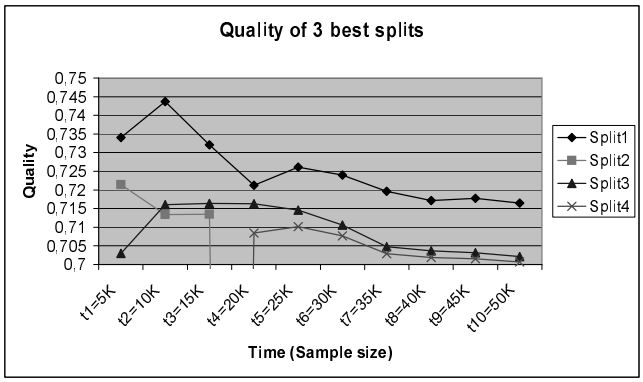

Figure 1: Three best split conditions found for $Dataset1$.

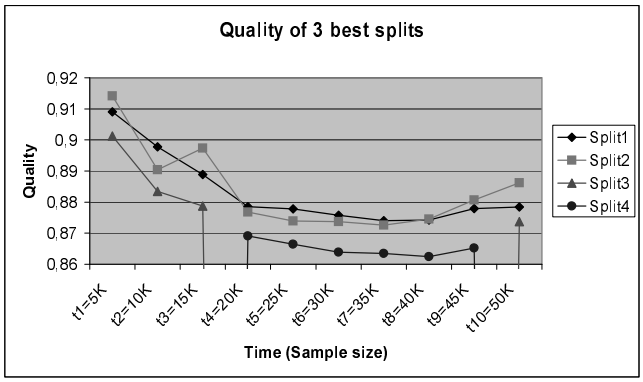

Figure 2: Three best split conditions found for $Dataset2$.

$$NQ(\mathbf{A}, p) = \frac{Q(\mathbf{A}, p)}{Conflict(\mathbf{A})} \tag{3}$$

Another definition of quality function can be found in [14].

While building the tree we have to be sure that found splits are not only optimal but also stable in a sense of new data. We propose, in the process of searching for splits, to choose those that are optimal according to both quality and dynamic behaviour on a growing (according to size) series of sub-tables.

Let $\mathbf{A} = (U, A \cup \{d\})$ be a decision table and let $\mathcal{F}_{\mathbf{A}} = \{\mathbf{A}_1, \mathbf{A}_2, ...\}$ be an ascending family of sub-tables of $\mathbf{A}$, such that $U_1 \subseteq U_2 \subseteq ... \subseteq U$. We propose, for each $\mathbf{A}_i \in \mathcal{F}_{\mathbf{A}}$ to search for k best splits $p^1_{\mathbf{A}_i}, p^2_{\mathbf{A}_i}, ..., p^k_{\mathbf{A}_i}$. Then, we can observe dynamic behaviour of splits, i.e. whether some splits are of high quality in several sub-tables and also how the quality of those splits changes itself in time. Each sub-table can be treated as a situation (observation) in a given time and we can consider periods of changes of data. By *period* $[t_i, t_j]$ we understand all sub-tables $\mathbf{A}_i, \mathbf{A}_{i+1}, \mathbf{A}_{i+2}, ..., \mathbf{A}_j$.

On Figure 1 and 2 we show results of experiments performed on two data sets. For each data set we created an ascending family of ten sub-tables with universe size from 5000 objects to 50000 objects. For each sub-table we were looking for $k = 3$ best splits. In this case $t_1 = 5K, t_2 = 10K, ..., t_{10} = 50K$.

On Figure 1 we can observe that $Split1$ is the dominant one for each sub-table and that $Split2, Split3$ have trends similar to the best one ($Split1$). In this case we deduce that $Split1$ is going to be the dominant one also in the future.

On Figure 2 we can notice different situation. In period $[t_1, t_3]$ there is an unstable situation - qualities of $Split1, Split2, Split3$ are changing rapidly. In period $[t_4, t_8]$ we observe stable situation - $Split1$ is dominant, however change of quality of $Split2$ comparing to $Split1$ is suggesting that $Split2$ is going to be the dominant one in the future. In time t_8 we see break even point, which means that situation described by data is changed. Then, according to our expectation $Split2$ is the dominant one in period $[t_9, t_{10}]$.

Please notice, that in case of both data sets we observe periods of perturbations and stability. Also in both cases there are some splits that are of high quality all the time, while some other splits appear and/or disappear. From above examples it follows that one time, in stable period, qualities of splits are correlated ($Dataset1$), and some other time the situation is being changed in time - dominant split is replaced with another one ($Dataset2$).

To discover stable periods we propose to observe standard deviation of quality changes. Firstly, let us define ΔNQ to measure value of change of normalized quality in a period $[t-1, t]$.

$$\Delta NQ(\mathbf{A}_t, p) = NQ(\mathbf{A}_t, p) - NQ(\mathbf{A}_{t-1}, p) \tag{4}$$

Secondly, we compute mean value of normalized quality change of k best splits.

$$E(\Delta NQ(\mathbf{A}_t, \cdot)) = \frac{1}{k} \sum_{i=1}^{k} \Delta NQ(\mathbf{A}_t, p^i_{\mathbf{A}_t}) \tag{5}$$

Finally, we compute standard deviation to measure magnitude of the deviations of quality change from the mean value.

$$D^2(\Delta NQ(\mathbf{A}_t,\cdot)) =$$

$$= \sqrt{\frac{1}{k}\sum_{i=1}^{k}\left(\Delta NQ(\mathbf{A}_t, p^i_{\mathbf{A}_t}) - E\left(\Delta NQ(\mathbf{A}_t,\cdot)\right)\right)^2} \quad (6)$$

We say, that period $[t_i, t_j]$ is *stable* iff

$$\forall_{t\in(t_i,t_j]} D^2(\Delta NQ(\mathbf{A}_t,\cdot)) < \tau \quad (7)$$

for a given threshold $\tau \in [0,1]$, otherwise it is *unstable*.

$Dataset1$ and in $Dataset2$ have the same stable period which is $[t_4, t_{10}]$. In both cases we used the same threshold $\tau = 0.003$.

Now, we can focus on the problem of quality analysis. Suppose we are building decomposition tree for $Dataset2$ in time t_8. We create sub-tables t_1, ...,t_7 and discover stable period $[t_4, t_8]$. We observe two candidates for split to be finally chosen: $Split1$ and $Split2$. Normally, we would choose $Split1$ because it was the dominant split during the whole stable period, but we observe that there is another split ($Split2$) with quality growing faster that in case of $Split1$. We can take our decision on the basis of prediction of quality for new data. If we predict quality of $Split2$ to be higher than of $Split1$, we would prefer to choose this split, even, if it is not the best split for current data.

For predicting the quality of a given split in case, that amount of data was increased, we propose to use the analogy of Taylor series for finite differences calculus. For a given split p we take the sequence $NQ(\mathbf{A}_1,p), NQ(\mathbf{A}_2,p), ..., NQ(\mathbf{A}_n,p)$ of qualities of p and treat this sequence as values of a quality function for p. Then, we compute:

$$NQ(\mathbf{A}_{n+1},p) \simeq \sum_{i=1}^{w}\frac{1}{i!}NQ^{(i)}(\mathbf{A}_n,p) \quad (8)$$

where w is the parameter saying how many sub-tables we used for our estimation.

5 Adaptation of decomposition tree to extended data

The method for stable decomposition tree generation presented in the previous section is time consuming, however, it assures that obtained tree is more general and universal. It means that we don't need to build a completely new tree when data is extended. The question is how can we know what nodes should be rebuilt, if any.

We propose to make a usage of information obtained in the process of tree generation. Having, in each node, k-best splits we can compute in a fast way which split is the optimal one for extended data. That determines nodes for which we have to recompute their sub-trees.

Suppose, we are given a decision table $\mathbf{A}$, binary decomposition tree $T = (V, E)$ for $\mathbf{A}$, decision table with new cases $\mathbf{N}$. We start searching T from root node using depth-first search. For a given node v we consider all k splits $p^1, p^2, ..., p^k$ that were generated in the process of tree generation. We compute their quality for extended data, i.e. for sub-table of $\mathbf{A}$, as well as of $\mathbf{N}$, corresponding to v. If optimal split was same (in some degree) as previous one we go to the child nodes, otherwise we mark v to be rebuilt and go to the parent node. After search is finished we rebuilt all nodes that are marked.

Note, that if root node was marked it means that the whole tree must be rebuilt. Similarly, if any of low level nodes (close to the root) are marked then we rebuilt almost everything. That gives us information about how the problem described by data changes in time.

6 Summary and future work

In the paper we presented the problem of building decomposition tree for data being increased in time. We proposed an algorithm for tree generation that results with decomposition possibly valid for extended data. Taking into account that for each node we can remember a number of best splits, we proposed how to adapt the tree to extended data. Moreover, if new data are significantly different from original the tree is automatically rebuilt in whole.

Acknowledgments: This work was supported by the grant of Polish National Committee for Scientific Research (KBN) No. $8T11C02417$ and $8T11C02519$.

References

[1] Bazan, J., Skowron, A. and Synak, P.: Discovery of Decision Rules from Experimental Data, *Proceedings of the Third International Workshop on Rough Sets and Soft Computing.* San Jose, California (1994) 526–533.

[2] Bazan, J., Skowron, A. and Synak, P.: Dynamic reducts as a tool for extracting laws from decision tables, *Proceedings of the Eighth International Symposium on Methodologies for Intelligent Systems (ISMIS'94), Lecture Notes in Artificial Intelligence 869.* Berlin: Springer-Verlag (1994) 346–355,.

[3] Kodratoff, Y., Michalski, M. (ed.): *Machine Learning vol. III.* Morgan Kaufmann, San Mateo, CA (1990).

[4] Mannila, H., Ronkainen, P.: Similarity of Event Sequences (revised version). In: *Proceedings of the Fourth International Workshop on Temporal Representation and Reasoning (TIME'97)* , Daytona Beach, Florida, USA, (1997) 136–139.

[5] Michalski, R.,S., Mozetic, I., Hong, J. and Lavrac, N.: The Multi-Purpose Incremental Learning System AQ15 and its Testing to Three Medical Domains, *Proceedings of AAAI-86.* Morgan Kaufmann, San Mateo, CA (1986) 1041–1045.

[6] Nguyen, H. S.: Discretization of Real Value Attributes: Boolean reasoning Approach. Ph.D. Thesis, Warsaw University, Warsaw, Poland (1997).

[7] Nguyen, H. S., Skowron, A.: Quantization of real value attributes, *Proceedings of Second Joint Annual Conf. on Information Sciences*, Wrightsville Beach, North Carolina, September 28 - October 1, USA (1995) 34–37.

[8] Nguyen S. Hoa, A. Skowron, P. Synak. Discovery of data pattern with applications to Decomposition and classification problems. In L. Polkowski, A. Skowron (eds.): Rough Sets in Knowledge Discovery **2**. Physica-Verlag, Heidelberg (1998) 55–97.

[9] Oliver, J., Hand, D.J.: On Pruning and Averaging Decision Trees, *Proceedings of the International Conference on Machine Learning ICML'95*, Kluwer (1995).

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

[11] Pawlak, Z., Skowron, A.: A rough set approach for decision rules generation, *ICS Research Report 23/93*, Warsaw University of Technology, *Proceedings of the IJCAI'93 Workshop W12: The Management of Uncertainty in AI*, France (1993).

[12] Quinlan, J., R.: Induction of Decision Trees, *Machine Learning 1.* Kluwer Academic, Boston, MA (1986) 81–106.

[13] Skowron, A.: Boolean reasoning for decision rules generation, *Proceedings of the 7-th International Symposium ISMIS'93*, Trondheim, Norway 1993, In Komorowski J. and Ras Z. (ed.), *Lecture Notes in Artificial Intelligence, vol. 689.* Springer-Verlag (1993) 295–305.

[14] Synak, P.: "Decomposition of large databases growing in time", Proc. of Eigth International Conference on Information Processing and Management of Uncertainty on Knowledge Based Systems (IPMU'2000), Madrid, Spain, (2000) 234–239.