Evaluation of Decision Table Decomposition Using Dynamic Programming Classifiers

Michal Mankowski[1], Tadeusz Luba[2], and Cezary Jankowski[2]

[1] Warsaw University of Technology, Institute of Radioelectronics, Warsaw, Poland
m.mankowski@stud.elka.pw.edu.pl
[2] Warsaw University of Technology, Institute of Telecommunications , Warsaw, Poland
luba@tele.elka.pw.edu.pl, c.jankowski@stud.elka.pw.edu.pl

Abstract. Decision table decomposition is a method that decomposes given decision table into an equivalent set of decision tables. Decomposition can enhance the quality of knowledge discovered from databases by simplifying the data mining task. The paper contains a description of decision table decomposition method and their evaluation for data classification. Additionally, a novel method of obtaining attributes sets for decomposition was introduced. Experimental results demonstrated that decomposition can reduce memory requirements preserving the accuracy of classification.

Key words: Data Mining, Decomposition, Clasifications, Dynamic Programming

1 Introduction

Increasing amount of data requires to develop new data analysis methods. A common approach in data mining is to make a prediction based on decision tables. Decomposition of a decision table to smaller subtables could be obtained by the divide and conquer strategy. This idea comes from logic synthesis and functional decomposition [1].

The fundamentals of decision system and logic synthesis are different, but there are many similarities between them. The decision system is usually described by a decision table, and combinational logic of a digital system by a truth table. Input variables of digital systems correspond to conditional attributes. Therefore, multiple terms from logic synthesis may be extended to data mining. Functional decomposition could be used to build a hierarchical decision system.

The functional decomposition was firstly used in logic synthesis of digital systems. In this situation, decomposition involves breaking a large logic functions, which are difficult to implement, into several smaller ones, which can be easy to implement. A similar problem in machine learning relies on disassembling the decision table to the subsystems in such a way that the original decision table can be recreated through a series of operations corresponding to the hierarchical decision making. But the most important is that we can induce noticeably simpler decision rules and trees for the resulting components to finally make the same decision as for the original decision table. [2–4]

For evaluation of decomposition, the decision trees and rules classifiers based on extensions of dynamic programming [5] were used. Decision tree and rules were sequentially optimized for different cost functions (for example, relative to number of misclassification and depth of decision trees). In the case of decision trees and rules this approach allowed to describe set of trees or rules by directed acyclic graph (DAG).

2 Basic Concepts

2.1 Preliminary Notions

Information system is a pair $\mathbb{A} = (U, A)$, where U is a non-empty, finite set of objects called the universe, A is a non-empty, finite set of attributes, i.e. each element $a \in A$ is a function from U into V_a, where V_a is the domain of a called value set of a. Then, the function ρ maps the product of U and A into the set of all values. The value of the attribute for a specific object is denoted by $\rho(u_t, a_i)$, with $u_t \in U$, $a_i \in A$

One or more distinguished attributes from set A of information system may indicate a decision from rest of attributes. Such information system is called *decision system*. Formally, *decision system* is information system denoted by $\mathbb{A} = (U, A \cup D)$, where $A \cap D = \emptyset$. Attributes in set A are referred to as conditional attributes while attributes in set D are referred to as decision attributes. However, in the case when function ρ maps $U \times (A \cup D)$ into the set of all attribute values such system is called *decision table*.

Let $\mathbb{A} = (U, A)$ be an information system. For each subset $B \subseteq A$ we define *B-indiscernibility relation* $IND_{\mathbb{A}}(B)$:

$$IND_{\mathbb{A}}(B) = \left\{(u_p, u_q) \in U^2 : \forall a_i \in B, \rho(u_p, a_i) = \rho(u_q, a_i)\right\} \tag{1}$$

The attribute values a_1, i.e. $\rho_{pi} = \rho(u_p, a_1)$ and $\rho_{qi} = \rho(u_q, a_1)$ are *compatible* ($\rho_{pi} \sim \rho_{qi}$) if, and only if, $\rho_{pi} = \rho_{qi}$ or $\rho_{pi} = *$ or $\rho_{qi} = *$, where "$*$" represents attributes value "do not care". In the other case ρ_{pi} and ρ_{qi} are *not compatible* ($\rho_{pi} \nsim \rho_{qi}$).

The consequence of this definition is *compatibility relation* $COM_{\mathbb{A}}(B)$ associated with every $B \subseteq A$:

$$COM_{\mathbb{A}}(B) = \left\{(u_p, u_q) \in U^2 : \forall a_i \in B, \rho(u_p, a_i) \sim \rho(u_q, a_i)\right\} \tag{2}$$

$COM_{\mathbb{A}}(B)$ classifies objects by grouping them into compatibility classes, i.e. $U/COM_{\mathbb{A}}(B)$, where $B \subseteq A$. Collection of subsets $U/COM(B)$ is called *r-partition* on U and denote as $\Pi_{\mathbb{A}}(B)$. R-partition on a set U may be viewed as a collection of non-disjoint subsets of U, where the set union is equal U. All symbols and operations of partition algebra [6] are applicable to r-partitions. The r-partition generated by a set B is the product of r-partitions generated by the attributes $a_i \in B$:

$$\Pi_{\mathbb{A}}(B) = \bigcap_i \Pi_{\mathbb{A}}(\{a_i\}) \tag{3}$$

If $B = \{a_{i1}, ..., a_{ik}\}$, the product can be expressed as: $\Pi(B) = \Pi(a_{i1}) \cdot ... \cdot \Pi(a_{ik})$. We will write often $\cdot$ instead of $\cap$.

2.2 Hierarchical Decomposition

To compress data and accelerate computations, hierarchical decomposition can be applied. The goal is to break down a decision table into two smaller subtables.

Let $\mathbb{F}$ be a functional dependency $D = \mathbb{F}(A)$ for a consistent decision system $\mathbb{A} = (U, A \cup D)$, where A is a set of conditional attributes and D is a set of decision attributes. Let B_1, B_2 be subsets of A such that $A = B_1 \cup B_2$ and $B_1 \cap B_2 = \emptyset$. A *simple hierarchical decomposition* relative to B_1, B_2 exists for $\mathbb{F}(A)$ if and only if:

$$\mathbb{F}(A) = H(B_1, G(B_2)) = H(B_1, \delta) \tag{4}$$

where G and H represent the following functional dependencies: $G(B_2) = \delta$ and $H(B_1, \delta) = D$, where δ is a intermediate attribute. The output of functions $\mathbb{F}(A)$ and H are exactly the same. In other words we try to find a function H depending on the variables of the set B_1 as well on the output δ of a function G depending on the set B_2.

Table 1. An example decision table $\mathbb{A} = (U, A \cup D)$

	a_0	a_1	a_2	a_3	a_4	a_5	d
0	0	0	0	0	0	0	1
1	0	0	1	1	0	0	1
2	1	2	2	0	1	1	2
3	0	1	1	0	0	1	2
4	0	1	0	2	0	1	3
5	1	2	2	3	2	0	2
6	1	2	2	2	0	1	1
7	0	0	1	1	0	1	3
8	0	1	0	3	2	0	4
9	2	2	2	3	2	0	4

Partition-based representation of data tables can be used to describe decomposition algorithms [6, 3]

Theorem 1. *([3]) Functional dependencies G and H represent hierarchical decomposition of function $\mathbb{F}(A) = H(B_1, G(B_2))$, if there exists a r-partition $\Pi_G \geq \Pi(B_2)$ such that:*

$$\Pi(B_1) \cdot \Pi_G \leq \Pi(D) \tag{5}$$

where all the r-partitions are over the set of objects, and the number of values of component G is equal to $L(\Pi_G)$ where $L(\Pi)$ denotes the number of blocks of r-partition Π.

In Theorem 1., r-partition Π_G represents component G, and the product of r-partitions $\Pi(B_1)$ and Π_G corresponds to H. The decision tables of the resulting components can be easily obtained from these r-partitions.

According to Theorem 1 the main problem is to find a partition Π_G. To solve that problem, a subset of original attributes B_2 and an m-block partition $\Pi(B_2) = \{K_1, K_2, ..., K_m\}$ generated by that subset is appropriate to consider. Two blocks K_i, K_j of partition $\Pi(B_2)$ are compatible if and only if the partition Π' obtained from $\Pi(B_2)$ by joining the blocks K_i and K_j into a single block K_{ij} (without changing the other ones) satisfies the equation (5), i.e. iff $\Pi' \cdot \Pi_G \leq \Pi(D)$. Otherwise, K_i, K_j are incompatible.

For decision table from Table 1 and sets of attributes $B_1 = \{a_0, a_5\}$, $B_2 = \{a_1, a_2, a_3, a_4\}$ the following set of incompatible pairs could be found:
$E = \{(K_1, K_8), (K_2, K_4), (K_2, K_8), (K_3, K_7)(K_4, K_5)\}$. The subset of n partition blocks, $\Pi(B_2) = \{K_{i_1}, K_{i_2}, ..., K_{i_n}\}$ where $K_{i_j} \in \Pi(B_2)$ is the compatible class of $\Pi(B_2)$ partition blocks iff all blocks of that subset are pairwise compatible. The compatibility class is referred to as maximal compatibility class (MCC) iff it does not belong to any other compatibility class of the partition concerned.

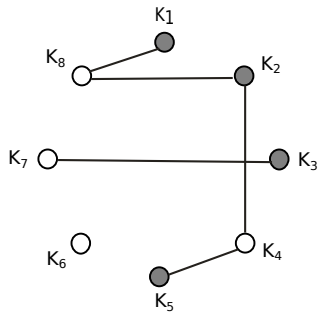

Fig. 1. Incompatibility graph and its coloring

The decomposition process may be interpreted in terms of an incompatibility graph (Fig. 1). The edges represent the incompatible pairs of partition $\Pi(B_2)$: (K_1, K_8), (K_2, K_4), (K_2, K_8), (K_3, K_7), (K_4, K_5). It is clearly visible that the proper coloring of the graph specifies the compatible classes: $\{K_1, K_2, K_3, K_5\}, \{K_4, K_6, K_7, K_8\}$ and, as a consequence, the partition

$$\Pi_G = \left\{\overline{0, 1, 2, 4, 5, 7, 9}; \overline{3, 6, 8}\right\}.$$

Another approach to building an incompatibility graph is to create a labeled partition matrix [7, 8] (Table 2). It should be noted that the columns represent all possible combinations of the attributes values in B_2. Each column thus denotes the behavior of decision table when the attributes in B_1 set are constant. Therefore each column can be treated as object from decision table. To build incompatibility graph it is necessary to apply the equation (2) to each pair of columns. When the compability relation is met then pair is compatible, otherwise it is incompatible.

Table 2. Partition matrix for the example set from Table 1 and the sets of attributes $B_1 = \{a_0, a_5\}$ and $B_2 = \{a_1, a_2, a_3, a_4\}$

		a_1	0	0	2	1	1	2	2	1
		a_2	0	1	2	1	0	2	2	0
		a_3	0	1	0	0	2	3	2	3
a_0	a_6	a_4	0	0	1	0	0	2	0	2
0	0		1	1	$*$	$*$	$*$	$*$	$*$	4
1	1		$*$	$*$	2	$*$	$*$	$*$	1	$*$
0	1		$*$	3	$*$	2	3	$*$	$*$	$*$
1	0		$*$	$*$	$*$	$*$	$*$	2	$*$	$*$
2	0		$*$	$*$	$*$	$*$	$*$	4	$*$	$*$
			K_1	K_1	K_3	K_4	K_5	K_6	K_7	K_8

2.3 Attributes Selection Criteria for Further Decomposition

Simple hierarchical decomposition requires to divide a set of conditional attributes A to two disjoint subsets B_1 and B_2. Proposed idea of obtaining sets is based on the attributes relationship called *attributes dependency* from Rough Set theory [9].

Let C and B be sets of attributes, then B depends entirely on a set of attributes C, denoted $C \Rightarrow B$, if all values of attributes from B are uniquely determined by the values of attributes from C. If B depends in degree k, $0 \leq k \leq 1$, on C, then:

$$k = \gamma(C, B) = \frac{|POS_C(B)|}{|U|} \tag{6}$$

where

$$POS_C(D) = \bigcup_{X \in U/B} C_*(X) \tag{7}$$

called a positive region of the partition U/B with respect to C, is the set of all elements of U that can be uniquely classified to blocks of the partition U/B, by means of C.

Proposed method allows us to measure dependency between all possible pairs of conditional attributes and decision attribute. *Related dependency* of one conditional attribute can be generated from a given information system: $\mathbb{A} = (U, A \cup \{d\})$, where $A = \{a_0, ..., a_k\}$ is a set of conditional attributes and d is a decision attribute:

$$r(x) = \frac{\sum_0^k \gamma(\{x, a_i\}, \{d\})}{|A|}, x \in A \tag{8}$$

The above function of related dependency is used for comparison of attributes. This function is being calculated for each attribute, then the results are being sorted by the value of function r. The most dependent attributes are put in set B_1, which corresponds to the final decision table H.

Example 1. For Table 1 first step of the algorithm is to built a matrix of attribute dependency between each pair of condition attributes and decision attribute. Then the mean of partial results is calculated, which is represented by related dependency $r(x)$ in Table 3. These results can be sorted by value and divided into two equinumerous sets. If the number of attributes is odd, then set $|B_1| = |B_2| + 1$. An example of sorting and assignment attributes is presented in Table 4. The calculation of related dependency $r(x)$ allows formulating an accurate method for assessing sets B_1 and B_2, i.e. $B_1 = \{a_1, a_3, a_5\}$ and $B_2 = \{a_0, a_2, a_4\}$. Therefore the decomposition is as follows: $\mathbb{F}(A) = H(\{a_1, a_3, a_5\}, G(\{a_0, a_2, a_4\}))$.

Table 3. Matrix of attributes dependency

	a_0	a_1	a_2	a_3	a_4	a_5
a_0	0	0.1	0.1	0.6	0.5	0.2
a_1	0.1	0	0.2	0.6	0.3	0.4
a_2	0.1	0.2	0	0.6	0.3	0.2
a_3	0.6	0.6	0.6	0	0.1	0.5
a_4	0.5	0.3	0.3	0.1	0	0.3
a_5	0.2	0.4	0.2	0.5	0.3	0
$r(x)$	0.250	0.267	0.233	0.400	0.250	0.267

Table 4. Attributes for partitions $G(B_2)$ and $H(B_1)$

a_2	0.233
a_0	0.250 B_2
a_4	0.250
a_5	0.267
a_1	0.267 B_1
a_3	0.400

3 Classification Schema

3.1 Hierarchical Decision Making

Due to the decomposition of decision tables, there is a need for hierarchical decision system to evaluate this method for the purpose of classification [6]. This method is based on disassembling the decision table into the subtables. The most important advantage is the possibility to induce a simpler classification model, for example shorter decision rules or smaller decision tree for the resulting components to finally make the same decision as for the original decision table. Following the process of decomposition, we propose to take decisions hierarchically. For the part of the attributes B_2 a prediction model to calculate intermediate decision was built. Then this intermediate decision was used simultaneously with the attributes from B_1 to build final classification model. Then, on the basis of both, i.e., these attributes and the intermediate decision δ, the final decision was taken (Fig. 2).

3.2 Dynamic Programing Classifiers

For decision prediction, the approach based on an extension of dynamic programming was used. These methods were developed in [5]. They allow sequential optimization of decision trees and rules relative to different cost functions, in particular between the

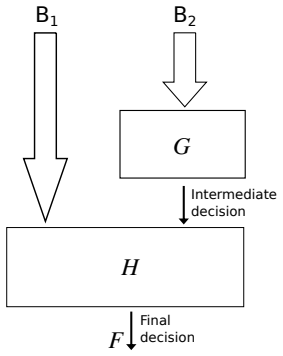

Fig. 2. Two-step implementation of the decision table

number of misclassifications and the depth of decision trees or the length of decision rules. Proposed algorithm constructs a directed acyclic graph (DAG), which represents structure of subtables of initial tables. For decision table $\mathbb{A}$ separable subtables of $\mathbb{A}$ described by systems of equalities of the kind $a_i = b$ are considered as subproblems, where a_i is an attribute and b is its value. Classification and optimization of decision trees and rules are discussed in details in [5, 10].

In the applied approach to optimization of decision trees directed acyclic graph (DAG) represents a set of CART-like decision trees [11]. Set of Pareto optimal points for bi-criteria optimization problem is constructed. Two types of decision tree pruning have been compared. First is the multi-pruning for which, using validation subtable (part of training subtable) for each Pareto optimal point, a decision tree with minimum number of misclassification is found. Second, as an improvement of multi-pruning, is to use only the best split for a small number of attributes in each nodes of DAG graph, instead of using the best split for all attributes. This pruning is called restricted multi-pruning.

The system of decision rules as a prediction model was also considered. As in case of decision trees we used dynamic programming algorithm to create and optimize decision rules [10].

4 Experiments

To evaluate the proposed decomposition algorithm and hierarchical decision making idea the $Dagger$ software system created in King Abdullah University of Science and Technology was used. Proposed algorithm has been tested on categorical datasets from UCI ML Repository [12]. A data sets were preprocessed. Duplicate rows were removed. There were some inconsistencies, i.e., there are instances with the same values of conditional attributes, but their decisions are different. The solution was to replace such set with a single row with most common decision. Results were obtained using the two-fold Cross-Validation evaluation repeated 100 times, each time using a different random selected testing subset. From training part, 70% of rows was used to generate decision trees and remaining part is preserved for validation.

Table 5. Compression results, where S is size of original decision table and S_D is a sum of tables after decomposition (H and G).

data set	rows	attributes	compression (S_D/S)
flags	196	27	0.801
house	281	17	0.395
kr-vs-kp	3198	37	0.209
breast cancer	268	10	0.754
cars	1730	7	0.261
spect-test	169	23	0.751
dermatology	366	35	0.352

The advantage of decomposition is due to the fact that two components (i.e. tables G and H) require less memory than the original decision table. Let us express the size of the table as $S = n \sum_i b_i$, where n is the number of objects, and $b_i = \lceil \log_2 |V_{a_i}| \rceil$ is the number of bits required to represent attribute a_i. Then, after the decomposition, we may compare the size of specific components with that of the original table (prior to decomposition). Results of compression are presented in Table 5.

Table 6. Comparison of accuracy error with and without decomposition.

data set	rows	attributes	decomp.	dp rules	dp tree MP	dp tree RMP
flags	196	27	no	0.402 ± 0.041	0.409 ± 0.047	0.399 ± 0.041
			yes	0.402 ± 0.041	0.404 ± 0.038	0.417 ± 0.045
house	281	17	no	0.092 ± 0.013	0.064 ± 0.008	0.067 ± 0.009
			yes	0.076 ± 0.045	0.080 ± 0.047	0.073 ± 0.031
kr-vs-kp	3198	37	no	0.019 ± 0.003	0.015 ± 0.004	0.062 ± 0.003
			yes	0.061 ± 0.015	0.064 ± 0.016	0.011 ± 0.015
breast cancer	268	10	no	0.287 ± 0.022	0.302 ± 0.023	0.304 ± 0.026
			yes	0.274 ± 0.024	0.295 ± 0.020	0.297 ± 0.029
cars	1730	7	no	0.070 ± 0.009	0.062 ± 0.010	0.068 ± 0.004
			yes	0.122 ± 0.013	0.159 ± 0.022	0.112 ± 0.018
spect-test	169	23	no	0.048 ± 0.025	0.048 ± 0.019	0.047 ± 0.000
			yes	0.056 ± 0.022	0.051 ± 0.016	0.049 ± 0.005
dermatology	366	35	no	0.229 ± 0.028	0.219 ± 0.028	0.221 ± 0.022
			yes	0.212 ± 0.030	0.226 ± 0.030	0.228 ± 0.027

Table 6. represents experimental results of classification using different dynamic programming algorithms. For each data set and method, accuracy error and standard deviation were calculated. As we can see the decomposition influences on accuracy. In cases of two data sets, i.e., kr-vs-kp and cars, accuracy error incomparably increased after decomposition. Note, that the compression for those date sets was very efficient.

However, for most of measurements, accuracy keeps a significant level or is slightly better. The biggest improvement occurring when dynamic programing rules were used.

5 Conclusion

Effective data aggregation algorithms have been sought after for a long time due to the increasing complexity of databases used in practice. Recently, some suggestions were put forward that decomposition algorithms, previously used mainly in logic synthesis of digital systems, may be applied for that purpose [13]. This approach is indeed very relevant as decision systems and logic circuits are very similar. Bearing this in mind, this paper demonstrates that a typical algorithm for the decomposition of binary data tables (representing Boolean functions) may be applied to the decomposition of data represented by multi-valued attributes used in decision systems.

The paper indicates the advantages and possibilities of decomposition algorithms for the purpose of classification. Results of experiments performed by proposed decomposition algorithm and $Dagger$ system has been presented. New attributes selection criteria describing partitions for decomposition has been introduced and used in the experiments. Proposed method is particularly efficient in data compression. It allows to build simple classification model and save memory, simultaneously keep the accuracy. To achieved better results in accuracy data set decomposition requires further research, particularity with attributes selection criteria. Also, there is a need to extend the decomposition to deal with continuous attributes and noise in data.

Acknowledgments. The authors would like to thank professor Mikhail Moshkov and his team for their support while writing this paper. This research has been supported by King Abdullah University of Science and Technology.

References

1. Perkowski, M.A., Luba, T., Grygiel, S., Burkey, P., Burns, M., Iliev, N., Kolsteren, M., Lisanke, R., Malvi, R., Wang, Z., et al.: Unified approach to functional decompositions of switching functions. PSU Electr. Engn. Dept. Report (1995)
2. Łuba, T., Lasocki, R., Rybnik, J.: An implementation of decomposition algorithm and its application in information systems analysis and logic synthesis. In Ziarko, W., ed.: Rough Sets, Fuzzy Sets and Knowledge Discovery. Workshops in Computing. Springer London (1994) 458–465
3. Luba, T., Lasocki, R.: On unknown attribute values in functional dependencies. In: Proceedings of the International Workshop on Rough Sets and Soft Computing,(San Jose, CA). (1994) 490–497
4. Rokach, L., Maimon, O.: Data mining using decomposition methods. In Maimon, O., Rokach, L., eds.: Data Mining and Knowledge Discovery Handbook. Springer US (2010) 981–998
5. Alkhalid, A., Amin, T., Chikalov, I., Hussain, S., Moshkov, M., Zielosko, B.: Optimization and analysis of decision trees and rules: dynamic programming approach. International Journal of General Systems **42**(6) (2013) 614–634

6. Borowik, G. In: Data Mining Approach for Decision and Classification Systems Using Logic Synthesis Algorithms. Volume 6. Springer International Publishing (2014) 3–23
7. Zupan, B., Prof, S., Bratko, D.I.: Machine learning based on function decomposition. Technical report, University of Ljubljana (1997)
8. Zupan, B., Bohanec, M.: Experimental evaluation of three partition selection criteria for decision table decomposition. Informatica (Slovenia) **22**(2) (1998)
9. Pawlak, Z.: Rough sets. International Journal of Computer & Information Sciences **11**(5) (1982) 341–356
10. Amin, T., Chikalov, I., Moshkov, M., Zielosko, B.: Dynamic programming approach to optimization of approximate decision rules. Information Sciences **221**(0) (2013) 403 – 418
11. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. CRC press (1984)
12. Lichman, M.: UCI machine learning repository (2013)
13. Maimon, O., Rokach, L.: Decomposition methodology for knowledge discovery and data mining. In Maimon, O., Rokach, L., eds.: Data Mining and Knowledge Discovery Handbook. Springer US (2005) 981–1003