New developments for Determinacy Analysis: diclique-based approach

GRETE LIND, REIN KUUSIK
Department of Informatics
Tallinn University of Technology
Raja 15, 12618 Tallinn
ESTONIA
grete@staff.ttu.ee, kuusik@cc.ttu.ee

Abstract: - Determinacy Analysis (DA) is a method that solves tasks of data mining (it enables to describe by the rules the subset of objects determined by the user). There are different treatments in DA: step by step and DAS-like in algoritmic view, one solution and multiple solutions as a result, additive and non-additive sets of rules in systematic view. Thereat the essence of the method itself does not change, only the rules change. There is a number of lacks in DA base algorithms: they are too labour-intensive (step by step approach) or they find only a limited set of rules (i.e. only one system of rules of many possible systems, in case of DAS). In this paper we show that DA can be reduced to the diclique finding task that is well-known from the graph theory, we present the prerequisites to take into account and explain how it influences the rules. The diclique-based DA enables to set up the DA tasks of new type: to find a system with minimal number of rules, to find a system with minimal number of shortest rules (for example). Reducing DA to a diclique finding task, the basis for the new generation of DA algorithms is founded.

Key-Words: - Determinacy Analysis, Data mining, Rules, Diclique, Diclique extracting task.

1 Introduction to Determinacy Analysis

Determinacy Analysis (DA) is a system of methods for the analysis of rules that was created at the end of 70s. Its approach combines mathematical statistics and logic. DA's methodology and the underlying mathematics are developed by Russian scientist Sergei Chesnokov [1], [2].

DA-technology provides an alternative way to perform factor analysis of qualitative and quantitative variables. It assists in obtaining regularities, explanations and prognostic rules.

DA has been used in sociology [3], linguistics [4], medicine [5], and other areas (for complete list of references see [6] or [7]).

The overview of determinacy analysis is based on [1], [2], [6], [7], [8].

1.1 Determination and its characteristics

The main idea behind DA is that a rule can be found based on the frequencies of joint occurrence or non-occurrence of events. Such rule is called a determinacy or determination, and the mathematical theory of such rules is called determinacy analysis [6].

If it is observable that an occurrence of X is always followed by an occurrence of Y, this means that there exists a rule "If X then Y", or $X \rightarrow Y$. Such correlation between X and Y is called determination (from X to Y). Here X is *determinative (determining)* and Y is *determinable*.

The determinative (X) consists of one or more factors. Factor is an attribute with its certain value. Each attribute can give as many different factors as many different values it has. Factors coming from the same attribute are not contained in the same X.

Each rule has two characteristics: accuracy and completeness[1].

Accuracy of determination $X \rightarrow Y$ shows to what extent X determines Y. It is defined as a proportion of occurrences of Y among the occurrences of X:

$A(X \rightarrow Y) = n(X\ Y) / n(X)$, where

$A(X \rightarrow Y)$ is accuracy of determination,

$n(X)$ is a number of objects having feature X and

$n(X\ Y)$ is a number of objects having both features X and Y.

[1] In the beginning (in [1], for example) "accuracy" (Russian "*точность*") was called "intensity" and "completeness" ("*полнота*") was called "capacity" ("*емкость*").

Completeness of determination X→Y shows which part of objects having Y can be explained by determination X→Y. It is a percentage of occurrences of X among the occurrences of Y:

C(X→Y) = n(X Y) / n(Y), where

C(X→Y) is completeness of determination,

n(Y) is a number of objects having feature Y and

n(X Y) is a number of objects having both features X and Y.

Both accuracy and completeness can have values from 0 to 1. Value 1 shows maximal accuracy or completeness, 0 means that rule is not accurate or complete at all. Value between 0 and 1 shows quasideterminism.

If all objects having feature X have also feature Y then the determination is (maximally) accurate. In case of accurate determination A(X→Y) = 1 (100%).

Majority of rules are not accurate. In case of inaccurate rule A(X→Y) < 1.

In order to make determination more (or less) accurate complementary factors are added into the first part of a rule. Adding factor Z into rule X→Y we get a rule XZ→Y.

Contribution of factor Z *to accuracy* of rule XZ→Y is measured by increase of accuracy ΔA(Z) caused by addition of factor Z into rule X→Y: ΔA(Z) = A(XZ→Y) – A(X→Y).

Contribution to accuracy falls into interval from -1 to 1.

If ΔA(Z)>0 then Z is a positive factor. Addition of positive factor makes rule more accurate, sometimes the resultant rule is (maximally) accurate.

If ΔA(Z)<0 then Z is a negative factor. Addition of negative factor decreases rule's accuracy, sometimes until zero.

If ΔA(Z)=0 then Z is a zero (or inessential) factor. Addition of zero factor does not change rule's accuracy.

If C(X→Y)=1 (100%) then the rule X→Y is (maximally) complete. It means that Y is always explained by X.

In case of incomplete rule C(X→Y)<1, it means that X does not explain all occurrences of Y.

Contribution of factor Z *to completeness* of rule XZ→Y is measured by increase of completeness ΔC(Z) by addition of factor Z into rule X→Y: ΔC(Z) = C(XZ→Y) – C(X→Y).

Contribution of whatever factor to completeness is negative or zero [8].

1.2 System of rules

System of rules is a set of rules in form $S_q = \{x_i \to y \mid i=1,2,...,q\}$, where q is the number of rules.

Every system is characterised by average accuracy, summarised completeness and summarised capacity (the number of objects covered by the rules).

Accuracy of system of rules is:

$$A(S_q) = A((\bigcup_{i=1}^{q} x_i) \to y) = \frac{n(y \bigcup_{i=1}^{q} x_i)}{n(\bigcup_{i=1}^{q} x_i)}. \quad (1)$$

Completeness of system of rules is:

$$C(S_q) = C((\bigcup_{i=1}^{q} x_i) \to y) = \frac{n(y \bigcup_{i=1}^{q} x_i)}{n(y)}. \quad (2)$$

Capacity of system of rules is:

$$n(S_q) = n(\bigcup_{i=1}^{q} x_i). \quad (3)$$

System of rules $S_q = \{x_i \to y \mid i=1,2,...,q\}$ is *additive* when x_i-s pairwise do not intersect (i.e. do not cover the same objects). Capacity of additive system is equal to the sum of capacities of rules it consists of:

$$n(S_q) = n(\bigcup_{i=1}^{q} x_i) = \sum_{i=1}^{q} n(x_i). \quad (4)$$

In case of additive system the previously given formulas can be simplified.

Completeness and capacity of additive system are just summed up completenesses and capacities of the rules:

$$C(S_q) = C((\bigcup_{i=1}^{q} x_i) \to y) = \frac{\sum_{i=1}^{q} n(yx_i)}{n(y)} = \quad (5)$$

$$= \sum_{i=1}^{q} C(x_i \to y);$$

$$n(S_q) = \sum_{i=1}^{q} n(x_i). \quad (6)$$

Accuracy of additive system is not additive (i.e. equal to the sum of rules' accuracies). It is found as a weighted average:

$$A(S_q) = A((\bigcup_{i=1}^{q} x_i) \to y) = \frac{\sum_{i=1}^{q} n(yx_i)}{\sum_{i=1}^{q} n(x_i)} = \\ = \sum_{i=1}^{q} \frac{n(x_i)}{\sum_{k=1}^{q} n(x_k)} A(x_i \to y) = \\ = \sum_{i=1}^{q} A((\bigcup_{k=1}^{q} x_k) \to x_i) A(x_i \to y). \quad (7)$$

System is called *complete* if its completeness is 1.

System is called *accurate* if its accuracy is 1. System is accurate when all of its rules are accurate.

The theory of DA covers also non-additive systems of (intersecting) rules, but this part of theory is not considered in this paper.

1.3 Main task of DA

The initial data table is given and some feature Y (as a certain class). The goal is to describe Y (possibly) completely by non-intersecting (possibly) accurate rules.

Substantially we search for the answer to the questions "Who are they (objects of class Y)?", "How can we describe them?", "What distinguishes them from the others?". This way, DA solves the data mining task.

The original application of DA (DAS) extracts the rules with equal length (= the number of factors in the rule), all of them contain the same attributes – this is a simple way to get an additive system of rules. In step by step approach (see below) the extracted rules can have different lengths while the system is additive.

Approach to DA described in [9] is a step by step approach where the order of factors (attributes) is essential, different orders may lead to the different results.

Attributes (factors) are added into the rules one by one. If some rule is accurate it will not be expanded by adding the next factor. At the same time the completenesses of found accurate rules are summed up. Reaching 100% the coverage is found.

The user decides about the order in which the attributes are included into the rules, from the beginning until the situation when all objects of the class are covered.

In the pseudocode of the algorithm the following notation is used:

`attrs` – set of attributes to include into the rules
`pot` – set of potential rules
`next_pot` – set of potential rules for the next iteration
`result` – set of found rules

Step by step algorithm for finding rules of DA:

```
determine Y, attrs
result ← ∅, total_C ← 0
pot ← {[]}
Find n(Y)
FOR EACH attribute in attrs DO
  next_pot ← ∅
  FOR EACH rule in pot DO
    FOR EACH value of attribute DO
      X = rule & attribute.value
      find n(X), n(X Y), A(X→Y), C(X→Y)
      IF A(X→Y)=1 THEN
        result ← result ∪ {X→Y}
        total_C ← total_C + C(X→Y)
        IF total_C=1 THEN goto END
      ELSEIF A(X→Y)>0 and A(X→Y)<1 THEN
        next_pot ← next_pot ∪ X
      ELSE
        //XY does not exist
      ENDIF
    NEXT value
  NEXT rule
  pot ← next_pot
NEXT attribute
END: All rules are found
```

In the following example step by step approach will be demonstrated.

1.4 Example

Data from [10] will be used in example (see Table 1). This data table contains 8 objects described by 4 attributes. The last attribute shows object's belonging to certain class (feature Y).

We will describe persons belonging to class "–". This class consists of three objects (n(Y)=3).

First we will use attributes in the (freely chosen) order: Hair, then Eyes and then Height.

Table 1. Quinlan's data

Height	Hair	Eyes	Class
tall	dark	blue	+
short	dark	blue	+
tall	blond	blue	-
tall	red	blue	-
tall	blond	brown	+
short	blond	blue	-
short	blond	brown	+
tall	dark	brown	+

Rules containing attribute Hair only are given in Table 2.

Table 2. Rules consisting of attribute Hair

Hair	n(X)	n(XY)	A	C	ΣC
dark	3	0	0	0	
red	**1**	**1**	**1**	**1/3**	1/3
blond	4	2	2/4	2/3	

Rule Hair.red→Class.– is accurate (A=1) and needs no additional factors. This rule is included into the result. It covers 1/3 of the objects of the class (C=1/3). The completenesses C of accurate rules will be summed up (ΣC), with hope to reach to 100% coverage (by accurate rules).

Rule with Hair.dark has zero accuracy (i.e. does not exist) in given class and will not be expanded.

Rule with Hair.blond has accuracy between 0 and 1, thus we expand it by adding the next attribute (Eyes) into it (see Table 3).

Table 3. Rules consisting of attributes Hair and Eyes

Hair	Eyes	n(X)	n(XY)	A	C	ΣC
blond	**blue**	**2**	**2**	**1**	**2/3**	1
blond	brown	2	0	0	0	

The first of found rules has accuracy 1 and will be included into the result. Its completeness is 2/3. Now the summed completeness is 100%, thus the class "–" is (fully) covered by the (accurate) rules.

At the same time we can see that the other branch (Hair.blond and Eyes.brown) has zero accuracy and there is no reason to expand it. As the found rules cover the class completely there is no reason to use the next attribute (Height), so the class is described without using it.

Class "–" is covered by two rules (rule system S1):

- Hair.red → Class.– (C = 33%)
- Hair.blond&Eyes.blue → Class.– (C = 67%)

This is one possible description for class "–".

Now we will describe the same class by the same attributes, adding them in the order they are given in the Table 1: Height, then Hair and then Eyes.

Rules consisting of attribute Height (only) are given in Table 4.

Table 4. Rules consisting of attribute Height

Height	n(X)	n(XY)	A	C	ΣC
short	3	1	1/3	1/3	
tall	5	2	2/5	2/3	

Neither of two (candidate) rules is accurate.

We add the next attribute (Hair) into both rules (see Table 5).

Table 5. Rules consisting of attributes Height and Hair

Height	Hair	n(X)	n(XY)	A	C	ΣC
short	dark	1	0	0	0	
short	red	0				
short	blond	2	1	1/2	1/3	
tall	dark	2	0	0	0	
tall	**red**	**1**	**1**	**1**	**1/3**	1/3
tall	blond	2	1	1/2	1/3	

Theoretically there can be 6 different value combinations of those two arguments. One of those 6 does not exist in the data (Height.short&Hair.red). Two of them do not exist in the given class (Height.short&Hair.dark; Height.tall& Hair.dark). Three others are proper for the class. One of them (Height.tall&Hair.red) has accuracy 1 and needs no additional factors. This rule covers 1/3 of the objects of the class. Two rules having accuracy between 0 and 1 have to be expanded again. Next we add attribute Eyes into those two rules (see Table 6).

Table 6. Rules consisting of attributes Height, Hair and Eyes

Height	Hair	Eyes	n(X)	n(XY)	A	C	ΣC
short	**blond**	**blue**	**1**	**1**	**1**	**1/3**	2/3
short	blond	brown	1	0	0	0	
tall	**blond**	**blue**	**1**	**1**	**1**	**1/3**	1
tall	blond	brown	1	0	0	0	

Two rules of four have accuracy 1 and both have completeness 1/3. We have found three (accurate) rules for the class "–" with overall completeness 1 (100%). Thus the class is completely described (S2):

- Height.tall&Hair.red → Class. – (C = 33%)
- Height.short&Hair.blond&Eyes.blue → Class. – (C=33%)

- Height.tall&Hair.blond&Eyes.blue → Class. – (C = 33%)

We have shown that the same class can be (completely) covered by different sets (systems) of (non-intersecting) rules depending on the order of inclusion of attributes into the rules.

There is no mathematical apparatus yet for deciding which rule set is better.

1.5 The essence of the problem of applying DA

Originally DA is realised in a software package "ДА-система" (DA-system or DAS) by "Контекст Медиа" [7]. This system uses a different approach compared to the step by step one (presented above). There the order of attributes has no effect. Only the set of attributes (used for finding the rules) is important – all attributes are included into all rules.

For example, using all three attributes the following rules are found (S3):

- Height.tall&Hair.red&Eyes.blue → Class. – (C = 33%)
- Height.short&Hair.blond&Eyes.blue → Class. – (C=33%)
- Height.tall&Hair.blond&Eyes.blue → Class. – (C = 33%)

In this result compared to the system S2 the only difference is in the first rule: it contains a zero factor (which does not change the rule's accuracy) Eyes.blue.

Using only two attributes – Hair and Eyes – DAS gives two rules (S4):

- Hair.red&Eyes.blue → Class.– (C = 33%)
- Hair.blond&Eyes.blue → Class.– (C = 67%)

Again, the first rule contains Eyes.blue as a zero factor.

Compared to the first set of rules S1 (found by step by step approach) a crystal-clear difference is seen. Let the reader decides which one to deem better in essence. Comparision of DAS and step by step approaches through diclique view will be done in 2.4.2.

By our opinion the inability of finding rules with different number of factors in rule makes DAS limited compared to the theory of DA by Chesnokov [8]. Chesnokov's goal is to find a system of so called normal rules – rules without zero factors [8]. Naturally these rules can have different number of factors (in the same system of rules). Our closer discussion about difficulties of getting normal rules is given in [11].

An important disadvantage is that DAS has no automated search strategies to find better solutions. It only helps manual search (by making calculations), but gives no advice for the subsequent search direction. The system does not say which attributes should be included into or excluded from analysis.

Also, the user manual does not include the methodology how to get needed result with DAS. Probably that knowledge is sold at special training courses.

Usually somebody is interested in the minimal number of rules. But we can say that DA treatment (beginning from 1 attribute and adding next ones step by step) described in sections 1.3 and 1.4 is not quite effective because of a lot of handwork. For example, if we have 10 dichotomous (2-valued) attributes, then we should look through 2^{10}-1 =1023 combinations of attributes, i.e. by 1, 2, 3, …, 10 attributes. It is not real to do by hand. The other base algorithm is needed for effective solution.

2 Pattern mining as a clique extracting task

Here we describe how to transform a data table into a graph, describe a pattern as a diclique and DA as a dicliques finding task.

2.1 Data table as a graph

Let a data table X(N, M) be given, i=1,...,N; j=1,...,M, Xij= 1, 2,..., Kj. For transforming we can create a bipartite graph, where nodes on the left side A of the graph are observations (objects), nodes on the right side B are variable values (factors). For example, let be given X(3,3), Kj=2, j=1,…,3 (see Table 7).

Table 7. Example X(3,3)

	V1	V2	V3
O1	1	2	1
O2	1	2	2
O3	2	2	1

We can present this data table as a graph (see Fig. 1).

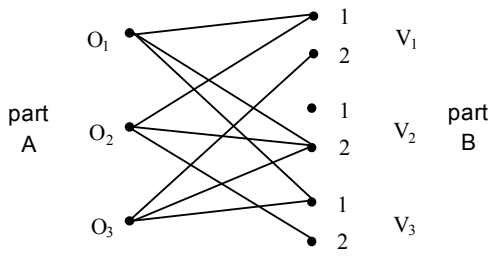

Fig. 1. Data table as a graph

Naturally we can present such a graph as a data table, with rows as nodes of the bipartite graph's part A and columns as nodes of the bipartite graph's part B. There is „1" in the table, if these nodes of the parts A and B are connected and „0" when not. For our graph we get a transformed data table shown in Table 8.

Table 8. Transformed data table

		Nodes of part B					
		V1=1	**V1=2**	**V2=1**	**V2=2**	**V3=1**	**V3=2**
Nodes of part A	*O1*	1	0	0	1	1	0
	O2	1	0	0	1	0	1
	O3	0	1	0	1	1	0

2.2 Pattern as a diclique

In general a pattern for the given variables (attributes) V1, V2, ..., Vm identifies a subset of all possible objects over these variables [12].

We can ask how to describe a pattern on a graph? It is a diclique. *Diclique* is a subgraph of the bipartite graph where all nodes of the part A are connected with all nodes of the part B [13].

For our example there are two dicliques with a frequency≥2 (see Fig. 2 and Table 9):

1) {(O1, O2); (V1=1, V2=2)},
2) {(O1, O3), (V2=2, V3=1)}.

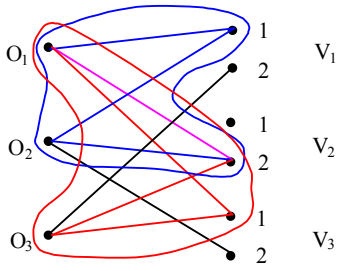

Fig. 2. Dicliques with a frequency ≥ 2

Table 9. Dicliques contained in the table

		Nodes of part B					
		V1=1	**V1=2**	**V2=1**	**V2=2**	**V3=1**	**V3=2**
Nodes of part A	*O1*	1	0	0	1	1	0
	O2	1	0	0	1	0	1
	O3	0	1	0	1	1	0

If the frequency ≥ 1, then we have 5 dicliques:

1) {(O1), (V1=1, V2=2, V3=1)},
2) {(O2), (V1=1, V2=2, V3=2)},
3) {(O3), (V1=2, V2=2, V3=1)},
4) {(O1, O2); (V1=1, V2=2)},
5) {(O1, O3), (V2=2, V3=1)}.

2.3 Determinacy Analysis as a Diclique extracting task

2.3.1 Any rule extracted by DA describes a pattern

Any rule R extracted by DA covers objects which belong to the class Y. It means that all objects covered by the rule R have the same attributes' values described by the rule R. It means that these objects belong to the same pattern described by the rule R. As we show in the section 2.2 every pattern can be presented as a diclique.

2.3.2 DA as a diclique extracting task

Theorem 1:

In DA the set (system) of rules is complete if every object of the class is covered by one rule (only).

Proof:

It is ensured the following way.

Looking at the certain attribute (determined by the user), we make sure if there exists a rule(s) (or not).

Having extracted a(n) (accurate) rule we do not permit to expand it with the next attribute (at the next iteration). Only value combinations that do not belong to any extracted rule can be expanded. It means that sets of objects corresponding to the rules cannot intersect. Thus every object can be covered by one rule only.

From here we can conclude that in case of DA number_of_rules ≤ number_of_objects.

Proceeding from the fact that pattern is describable as a diclique and based on the theorem 1, we can formulate Determinacy Analysis as a diclique extracting task:

to find a set of dicliques so that nodes of the bipartite graph's part A which belong to the class Y belong to some diclique so that every object from Y belongs only to one diclique.

2.3.3 DA as a diclique extracting task: an example

Here we present our example from section 1.4 as a diclique extracting task. Table 10 presents a transformed table of data given in Table 1 and Fig. 3 shows the corresponding graph. Here attribute Class is not shown on the right side B (among other attributes), it is used to group the nodes that correspond to the objects on the left side A.

Table 10. Transformed data table

	Height		Hair			Eyes		Class	
	tall	sh.	dark	bl.	red	blue	br.	"+"	"-"
O1	1	0	1	0	0	1	0	1	0
O2	0	1	1	0	0	1	0	1	0
O3	1	0	0	1	0	1	0	0	1
O4	1	0	0	0	1	1	0	0	1
O5	1	0	0	1	0	0	1	1	0
O6	0	1	0	1	0	1	0	0	1
O7	0	1	0	1	0	0	1	1	0
O8	1	0	1	0	0	0	1	1	0

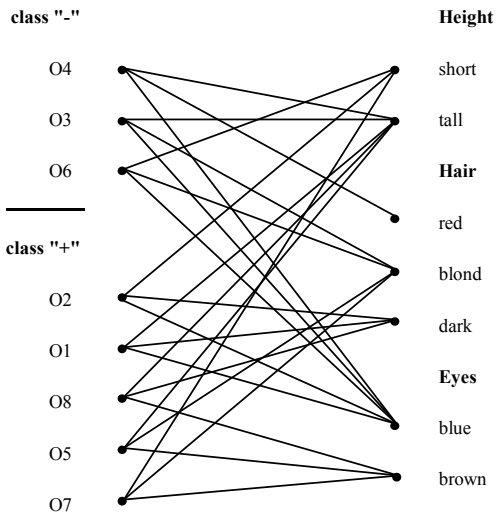

Fig. 3. Data from Table 10 as a graph

Table 11 and Fig. 4 present the result (consisting of 2 rules) S1 from section 1.4:

1) {(O3, O6), (Hair="blond", Eyes="blue")}
2) {(O4), (Hair="red")}

Table 11. Subgraphs corresponding to the rules in class "-"

	Height		Hair			Eyes		Class	
	tall	Sh.	dark	bl.	red	blue	br.	"+"	"-"
O1	1	0	1	0	0	1	0	1	0
O2	0	1	1	0	0	1	0	1	0
O3	1	0	0	1	0	1	0	0	1
O4	1	0	0	0	1	1	0	0	1
O5	1	0	0	1	0	0	1	1	0
O6	0	1	0	1	0	1	0	0	1
O7	0	1	0	1	0	0	1	1	0
O8	1	0	1	0	0	0	1	1	0

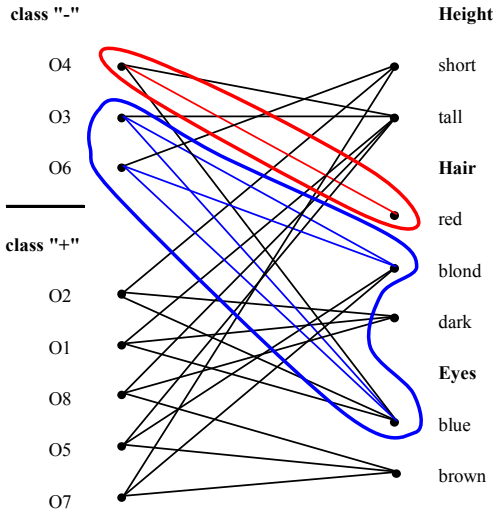

Fig. 4. Subgraphs corresponding to the rules in class "-"

Table 12 and Fig. 5 show the dicliques in class "–":

1) {(O3, O6), (Hair="blond", Eyes="blue")}
2) {(O4), (Hair="red", **Eyes="blue", Height="tall"**)}

Table 12. Dicliques in class "-"

	Height		Hair			Eyes		Class	
	tall	Sh.	dark	bl.	red	blue	br.	"+"	"-"
O1	1	0	1	0	0	1	0	1	0
O2	0	1	1	0	0	1	0	1	0
O3	1	0	0	1	0	1	0	0	1
O4	1	0	0	0	1	1	0	0	1
O5	1	0	0	1	0	0	1	1	0
O6	0	1	0	1	0	1	0	0	1
O7	0	1	0	1	0	0	1	1	0
O8	1	0	1	0	0	0	1	1	0

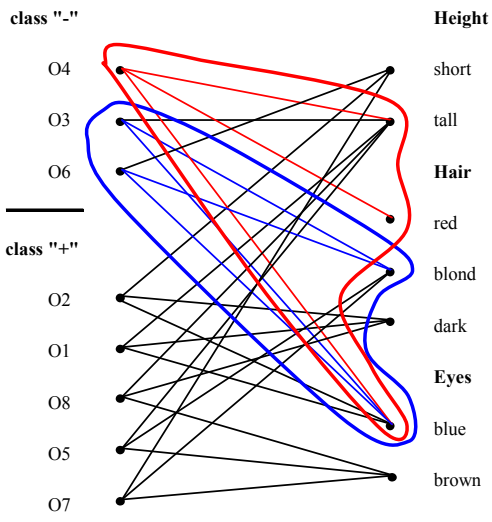

Fig. 5. Dicliques in class "-"

2.4 Some developments of DA based on diclique finding task

As we see the pattern determined by the second rule {(O4), (Hair="red")} (in Fig. 4) does not fulfill the condition of diclique, it is a subdiclique of the diclique {(O4), (Hair="red", Eyes="blue", Height="tall"} (see Fig. 5). It means that we can add to the rule such factors (Eyes="blue", Height="tall") which are *common* to all objects (i.e. nodes belonging to the diclique part A) covered by the rule (see Table 12). Added factors serve as only additional information for the researcher and this way do not change the essence of the rule. These factors do not change the rule's frequency and accuracy (zero factors in DA):

X = Hair.red	X' = Hair.red & **Eyes.blue & Height.tall**
n(X) = 1	n(X') = 1
n(XY) = 1	n(X'Y) = 1
A(X→Y) = 1	A(X'→Y) = 1

Adding this property (these factors) to the rule DA can be realised on the basis of diclique extracting algorithm. This way it would be a development of DA, that enables to automate the handwork and to extract the rules with different length at the same time.

We present next a task setup and an example describing possibilities of step by step and DAS treatments for DA.

2.4.1 New formulation of task

Now we present a new task for DA:

To find a minimal system of rules.

This task can be understood several ways:

1) Originating from the original DA, all the rules have equal length containing all given attributes. In this case the minimal system of rules should contain the minimal number of rules of equal length (i.e. number of factors).

2) Originating from the step by step approach, it is possible to find system of rules of different length.

In the first case, on the assumption of the essence of the DAS approach, there is exactly one solution. If each rule has to contain all (given) attributes, then we cannot take away any factor from any rule and consequently we cannot change such system of rules. Determining a different set of attributes (for description) we get a different system which is also the only possible solution for that set of attributes (in case of the same requirement – to contain all attributes).

In the second case the solution is a system of rules with minimal number of rules. At the same time one can be also interested in this, that the system with minimal number of rules had possibly short rules i.e. the double minimizing – over the systems of rules and additionally inside the system of rules – takes place.

2.4.2 An example for comparison of step by step approach and DAS: a diclique view

Next we present as an example the rules for Table 1 in cases of different orders of attributes (step by step approach) in diclique view and compare them with the result of DAS. We present the rules of class "+" this time and give also the corresponding binary tables and graphs for better understanding.

In case of step by step approach the result depends on the inclusion order of attributes. If the order is: Hair, then Eyes and then Height; then we get (S5):

- Hair.dark → Class.+ (C = 60%)
- Hair.blond&Eyes.brown → Class.+ (C = 40%)

See Table 13 and Fig. 6.

These rules contain only two first attributes, attribute Height is not needed for determining the class "+". On the graph both rules are dicliques.

Table 13. Rules for class "+" by step-by-step DA using attributes in the order: Hair, then Eyes, then Height

	Height		Hair			Eyes		Class	
	Tall	Sh.	dark	bl.	red	blue	br.	"+"	"-"
O1	1	0	1	0	0	1	0	1	0
O2	0	1	1	0	0	1	0	1	0
O3	1	0	0	1	0	1	0	0	1
O4	1	0	0	0	1	1	0	0	1
O5	1	0	0	1	0	0	1	1	0
O6	0	1	0	1	0	1	0	0	1
O7	0	1	0	1	0	0	1	1	0
O8	1	0	1	0	0	0	1	1	0

Table 14. Rules for class "+" by DAS using attributes Hair and Eyes

	Height		Hair			Eyes		Class	
	tall	sh.	dark	bl.	red	blue	br.	"+"	"-"
O1	1	0	1	0	0	1	0	1	0
O2	0	1	1	0	0	1	0	1	0
O3	1	0	0	1	0	1	0	0	1
O4	1	0	0	0	1	1	0	0	1
O5	1	0	0	1	0	0	1	1	0
O6	0	1	0	1	0	1	0	0	1
O7	0	1	0	1	0	0	1	1	0
O8	1	0	1	0	0	0	1	1	0

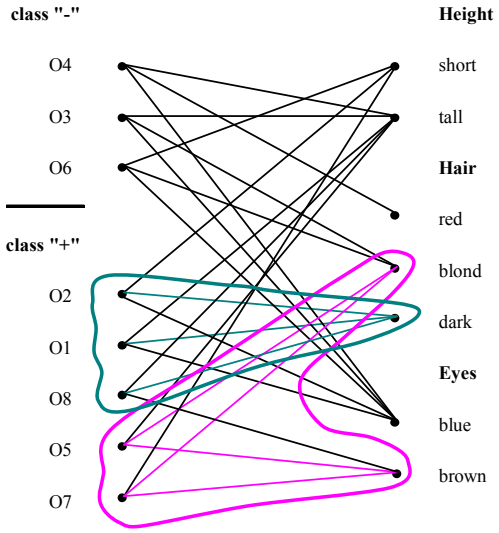

Fig. 6. Rules for class "+" by step-by-step DA using attributes in the order: Hair, then Eyes, then Height

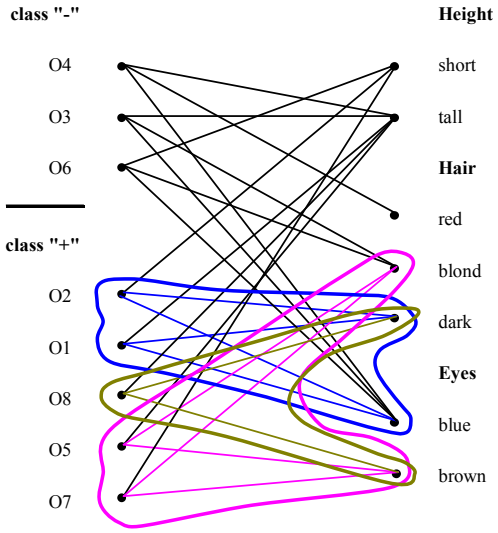

Fig. 7. Rules for class "+" by DAS using attributes Hair and Eyes

The result of DAS using these two attributes (Hair and Eyes, the order is not important) is (S6):

- Hair.dark&Eyes.blue → Class.+ (C = 40%)
- Hair.blond&Eyes.brown → Class.+ (C = 40%)
- Hair.dark&Eyes.brown → Class.+ (C = 20%)

See Table 14 and Fig. 7.

Compared to the previous set of rules, the second rule here is the same. The first rule of previous set is divided into two rules (the 1st and the 3rd) by adding the values of attribute Eyes.

In this set (S6) two first rules correspond to dicliques and the last one does not (- Height.tall is missing). The 3rd rule can be changed into a diclique without infriging the essence of DA by adding Height.tall as a zero factor.

Using another order (Height, then Hair and then Eyes), the step by step approach gives (S7):

- Height.tall&Hair.dark → Class.+ (C = 40%)
- Height.tall&Hair.blond&Eyes.brown → Class.+ (C = 20%)
- Height.short&Hair.dark → Class.+ (C = 20%)
- Height.short&Hair.blond&Eyes.brown → Class.+ (C = 20%)

See Table 15 and Fig. 8.

Here the 3rd rule (covering 1 object and describing it by 2 attributes) is not a diclique. It can be changed into a diclique without infriging the essence of DA by adding Eyes.blue as a zero factor.

Table 15. Rules for class "+" by step-by-step DA using attributes in the order: Height, then Hair, then Eyes

	Height		Hair			Eyes		Class	
	tall	sh.	dark	bl.	red	Blue	br.	"+"	"-"
O1	1	0	1	0	0	1	0	1	0
O2	0	1	1	0	0	1	0	1	0
O3	1	0	0	1	0	1	0	0	1
O4	1	0	0	0	1	1	0	0	1
O5	1	0	0	1	0	0	1	1	0
O6	0	1	0	0	0	1	0	0	1
O7	0	1	0	1	0	0	1	1	0
O8	1	0	1	0	0	0	1	1	0

Table 16. Rules for class "+" by DAS using attributes Height, Hair and Eyes

	Height		Hair			Eyes		Class	
	tall	sh.	dark	bl.	red	blue	br.	"+"	"-"
O1	1	0	1	0	0	1	0	1	0
O2	0	1	1	0	0	1	0	1	0
O3	1	0	0	1	0	1	0	0	1
O4	1	0	0	0	1	1	0	0	1
O5	1	0	0	1	0	0	1	1	0
O6	0	1	0	0	0	1	0	0	1
O7	0	1	0	1	0	0	1	1	0
O8	1	0	1	0	0	0	1	1	0

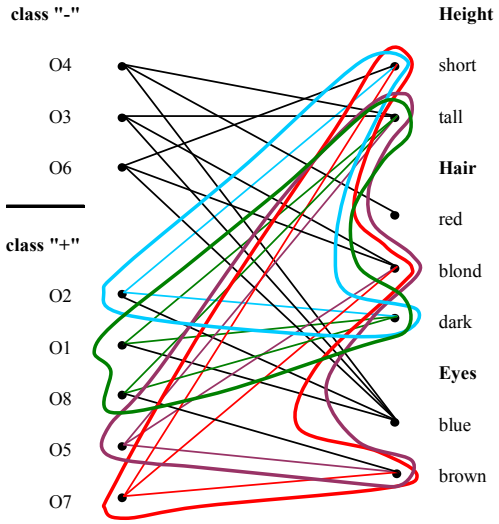

Fig. 8. Rules for class "+" by step-by-step DA using attributes in the order: Height, then Hair, then Eyes

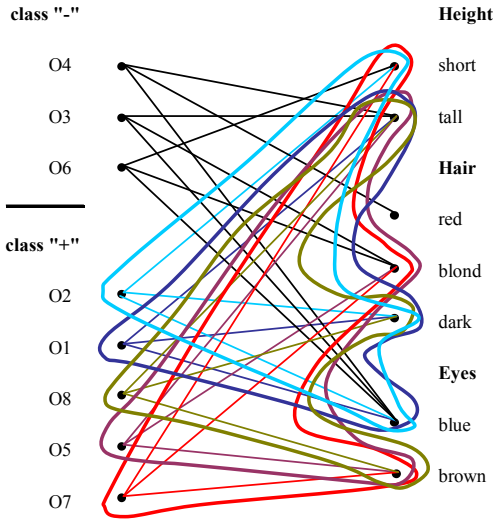

Fig. 9. Rules for class "+" by DAS using attributes Height, Hair and Eyes

Using all 3 attributes (Height, then Hair and then Eyes), DAS finds (S8):

- Height.tall&Hair.dark&Eyes.blue → Class.+ (C = 20%)
- Height.short&Hair.dark&Eyes.blue → Class.+ (C = 20%)
- Height.tall&Hair.blond&Eyes.brown → Class.+ (C = 20%)
- Height.short&Hair.blond&Eyes.brown → Class.+ (C = 20%)
- Height.tall&Hair.dark&Eyes.brown → Class.+ (C = 20%)

See Table 16 and Fig. 9.

Each rule here covers only one object using all possible attributes. All such rules are dicliques (if there are no contradictions in the data).

From the examples presented here we can see that

1) both step by step and DAS approach are reducible to the diclique finding task, only the first one has no contradiction with diclique extracting task formulated in 2.4.1
2) the examples brought out the strenghts and weaknesses of both approaches:
   - DAS-like approach [7] enables to find the rules that always contain all attributes (under consideration) and therefore have always equal length. This approach is good enough for avoiding algoritmic and complexity problems, but it limits the possibilities of DA significantly.
   - Step by step approach is sensitive to the order of passing through the attributes. For that reason the automation turns out to be essential to make for a possibility

to quickly find a system of shorter rules. The approach based on finding dicliques presented in this chapter enables it.

3 Conclusion

In this paper we have introduced DA which can determine several rule systems for objects with determined property (class). DA can answer to the question (for example) "Who are they (objects of class Y)? How we can describe them? What distinguishes them from the others?" This way, DA solves the data mining task. Thereat the specific difference is that in every system of rules each object of given class is described by one rule only. We can extract several rule systems, each of them covers given class 100%. There is no special mathematical apparatus to determine which rule system is the best, the substantial evaluations turn out to be decisive.

In the paper we showed also that DA is reducible to the graph-theoretical diclique extracting task. It enables to develop new DA treatments and elaborate algorithms, for example, to extract a rule system with a minimal number of rules.

Proceeding from different treatments of DA we can state that DAS does not enable the task setup with minimal number of rules, because it always finds only one system of rules, which consist of all attributes, i.e. in case of this approach only one solution exists. Step by step approach to DA enables to formulate and solve this task, but an effective algorithm for that is missing. Reducing this task to the diclique extracting task, the basis for the new generation of DA algorithms is founded. There exist several effective diclique extracting algorithms [14], [15], [16], [17], [18] applying them would enable to create the new effective approaches of DA that exclude manual work. These developments are the future works for the authors of the paper.

Acknowledgement

This work was supported in part by the Estonian Information Technology Foundation under Grant 08-03-00-25.

References:

[1] S. V. Chesnokov, *Determination-analysis of social-economic data in dialogical regime* (Preprint. Moscow, All-Union Institute for Systems Research, 1980, in Russian).

[2] S. V. Chesnokov, *Determinacy analysis of social-economic data* (Moscow, Nauka, 1982, in Russian).

[3] S. V. Chesnokov, Determinacy analysis of social-economic data, *Sociological Studies*, 1980, #3, pp. 179-189 (in Russian).

[4] P. A. Luelsdorff, S. V. Chesnokov, Determinacy Form as the Essence of Language, *Prague Linguistic Circle Papers*, 1996, v. 2, pp. 205-234.

[5] S. V. Chesnokov, Determinacy Analysis and the search for diagnostic criteria in medicine (the case of comprehensive ultrasonography), *Ultrasonic Diagnostics*, 1996, #4, pp. 42-47 (in Russian).

[6] DALSolution software and technology. Questions and Answers. http://www.dalsolution.com/faq.htm, retrieved 27.02.2007.

[7] ДА-система 4.0, версия 4.0 для Windows 95, Windows 98 и Windows NT. Вопросы-Ответы. ДА-система и технология анализа данных. © 1999 Фирма "Контекст".

[8] S. V. Chesnokov, Determinacy analysis of socio-economic data. Illustrative materials to lectures. Lecture 2: Rules. Lecture 3: Systems of rules. Lomonosov Moscow State University, Faculty of Economics, Moscow, 2002 (unpublished, in Russian).

[9] G. Lind, R. Kuusik. Some Ideas for Determinacy Analysis Realisation. *Proceedings of the 11th IASTED International Conference on Artificial Intelligence and Soft Computing*. Palma de Mallorca, Spain, August 29-31, 2007. ACTA Press, pp. 185-190.

[10] J. R. Quinlan, Learning efficient classification procedures and their application to chess and games, *J. G. Carbonell, R. S. Michalski, T. M. Mitchell (Eds.), Machine Learning. An Artificial Intelligence Approach*, Springer-Verlag, 1984, pp. 463-482.

[11] G. Lind, R. Kuusik. Some Problems in Determinacy Analysis Approaches Development. (*Proceedings of) the 4th International Conference on Data Mining (DMIN '08)*, Las Vegas, Nevada, USA, July 14-17, 2008, 7 pg (in press).

[12] D. Hand, H. Mannila, P. Smyth. *Principles of Data Mining*. MIT Press, 2001.

[13] R. M. Haralick. The Diclique Representation and Decomposition of Binary Relations. *JACM*, 21, 3, 1974, pp. 356-366.

[14] R. Kuusik. Extracting of all maximal cliques: monotonic system approach, *Proceedings of the Estonian Academy of Sciences. Engineering*, No. 1, 1995, pp. 113-138.

[15] D. Kumlander. Improving the maximum clique finding applications by using artificial intelligence principles. *WSEAS Transactions on Computers*, 8(5), August 2006, pp. 1726-1732.

[16] R. Kuusik, G. Lind, L. Võhandu. Frequent pattern mining as a clique extracting task. *The 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, July 18-21, 2004 - Orlando, Florida, USA, SCI 2004 Proceedings, Vol. IV, pp. 425-428.

[17] R. Kuusik, G. Lind. Algorithm MONSA for All Closed Sets Finding: basic concepts and new pruning techniques. *WSEAS Transactions on Information Science and Applications*, 5(5), May 2008, pp. 599-611.

[18] L. Võhandu, R. Kuusik, A. Torim, E. Aab, G. Lind. Some Monotone Systems Algorithms for Data Mining. *WSEAS Transactions on Information Science and Applications*, 4(3), April 2006, pp. 802-809.