New Solution for Extracting Inductive Learning Rules and their Post-Analysis

Rein Kuusik and Grete Lind
Department of Informatics
Tallinn University of Technology
Tallinn, Estonia
kuusik@cc.ttu.ee, grete@staff.ttu.ee

Abstract—**In this paper we present a new approach for machine learning task solution based on the new concept – the "Determinative set of rules" (DSR). We present a new inductive learning algorithm named MONSAMAX2 for finding DSR. MONSAMAX2 extracts it very effectively using some new pruning techniques. Compared to the former algorithm MONSIL it is much less labor-consuming. Also we present some ideas how to use DSR for post-analysis of rules.**

Keywords-machine learning; inductive learning algorithm; post-analysis of rules; determinative set of rules

I. INTRODUCTION

In the domain of inductive learning (IL) many different algorithms are used to solve different problems. There are several algorithms which try to solve the same task on different theoretical bases.

Some algorithms output rules as decision trees, some as set of rules, some of them find non-intersecting rules, some find overlapping rules, some algorithms find different systems of rules, some find a set of rules that meets certain requirements etc.

However, each method for finding a set of rules tries to prune the number of rules. This is expected, because the number of all possible rules in case of given sets of learning examples can be huge. Finding rules for bigger amounts of data is also very laborious. Because of this, we try to find such rules which have a stably good ability of recognition.

Thereby a number of different measures is used for evaluating the expediency of found rules. The main problem is not the existence of a rule for identifying some object but the correct identification of the object. Specifically as a result of using a rule set 3 situations can occur: 1) there is no rule for identifying the objects' belonging to a certain class, 2) the rule exists but identifies incorrectly, 3) the rule exists and it identifies correctly. The common approach uses the strategy that the first rule that identifies an object is used. But this does not guarantee certainty that the object is identified correctly. If several rules are used for identifying then the so-called rule conflict problem occurs. It occurs when two or more rules cover the same test example but predict different classes. The common strategy for preferring one of the conflicting rules is the best rule strategy [1]: for each rule a weight is calculated by some rule quality measure and the conflicting rule with the highest weight is chosen. Thereat the so-called preordering of rules [2] can be used for improving the result, as a consequence the result improves by 10-20%. It means that typically there are no actions for post-analysis of rules.

Also the approaches for the continuous development of the rule set have been created where the set of rules is complemented by adding new objects to the base of learning examples. Thereat the help of human experts can be used if the rule for identifying the object is missing [3].

It is clear that the pruning of the set of rules unavoidably leads to a loss of information and thereby to the possible increase of 1) the number of unidentified objects, 2) the number of misidentified objects. This paper offers one possible solution for lessening that.

The paper is structured as follows. The definitions of concepts used in inductive learning are given in Section II. Section III presents a proposed new approach for extracting rules. Conclusions are given in Section IV.

II. DEFINITIONS

We mainly follow the terms of the article [4].

The set of ***objects*** $X=\{x_1,\ldots,x_N\}$ can be described with ***attributes*** $t_1,\ldots, t_M$ so that every object x_i can be described as a tuple

$$x_i=(t_1(x_i),\ldots,t_M(x_i))=(x_{i1},\ldots,x_{iM}).$$

For each attribute t_j there exists a finite ***set of values*** A_j ($1\le j\le M$). So the ***attribute value*** x_{ij} of the object x_i belongs to the set A_j

$$x_{ij}=t_j(x_i)\in A_j.$$

Classes $C_1,\ldots,C_K$ are subsets of X such that

$$C_1\cup\ldots\cup C_K;\ \forall i\forall j,\ i\neq j : C_i\cap C_j=\varnothing.$$

The ***class value*** of the object $x\in X$ is c_j if $x_i\in Cj$. Let us denote the set of class values as

$$C=\{c_1,\ldots,c_K\}.$$

A ***learning example*** e_i is a tuple created from the object x_i and its class value

$$e_i=(x_i,c)=((t_1(x_i),\ldots,t_M(x_i)),c)=((x_{i1},\ldots x_{iM}),c).$$

Let us denote the ***set of examples*** E as

$$E=\{e_1,\ldots,e_n\}.$$

Let us denote the ***set of examples*** of class C_j as

$$E_j=\{e|e=(x,c_j),\ x\in C_j,\ C_j\subseteq X\}.$$

The ***positive example*** e_{j+} ***of the class*** C_j is an example which belongs to the set E_j, i.e.,

$$e_{j+}\in E_j\subseteq E.$$

The ***negative example*** e_{j-} ***of the class*** C_j is an example that does not belong to the set E_j, i.e.,

$$e_{j-}\notin E_j\subseteq E.$$

 ISBN: 978-1-61208-162-5

The ***majority class*** of set E is the class with the largest number of examples.

Function d which maps the class value c_j according to every element of the set X is called a ***concept***

$$d:X \rightarrow C;\ d(x_i)=c_j \Leftrightarrow x_i \in C_j.$$

In inductive learning the learning system should find, on the base of the learning examples, a concept description D which maps the class value for any object of the set X (not only for the objects of the example set) $D:X \rightarrow C$. Consequently the inductive learning system should (in an ideal case) find, on the base of learning examples, such a concept description D which maps the same class value as the concept d to every object

$$\forall x \in X,\ D(x)=d(x).$$

The ***(concept) description*** is the set of classification rules $D=\{r_1,\ldots,r_S\}$.

A ***classification rule*** (***decision rule***) is an implication where the condition part is a complex and the conclusion part is a class name:

$$r_j = \text{“}Com_j => c_k\text{”}$$

or

$$r_j = \text{“if } Com_j \text{ then } c_k\text{”}$$

or

$$r_j=(Com_j,c_k).$$

Complex Com_j is a tuple of ***selectors*** Sel_{jk} (k=1,…,M)

$$Com_j=(Sel_{j1},\ldots,Sel_{jM}).$$

Selector Sel_j is a subset of the set of values of the attribute t_j

$$Sel_j \subseteq A_j.$$

Description D ***maps a class value*** c_k ***for the object*** x_i if it contains a classification rule r_j which maps a class value c_k for the object x_i

$$\exists r_j \in D,\ r_j(x_i)=c_k \Rightarrow D(x_i)=c_k.$$

Rule $r_j=(Com_j,c_k)$ ***maps a class value*** c_k ***for the object*** x_i if its complex Com_j covers the object x_i

$$r_j=(Com_j,c_k),\ cover(Com_j,x_i) \Rightarrow r_j(x_i)=c_k.$$

Complex Com_j ***covers the object*** x_i if all its selectors Sel_{jk} cover this object

$$\forall k,\ 1 \le k \le M,\ cover(Sel_{jk},x_i) \Rightarrow cover(Com_j,x_i).$$

Selector Sel_{jk} ***covers the object*** x_i if the value of the attribute t_k of the object x_i is in the set Sel_{jk}

$$\forall j,\ 1 \le k \le M,\ x_{ik} \in Sel_{jk} \Rightarrow cover(Sel_{jk},x_i).$$

Description D is ***consistent*** on the set $X' \subseteq X$ if all its rules map the same class value for any object $x \in X'$

$$\forall r_i,r_j \in D,\ x \in X',\ X' \subseteq X,\ cover(Com_i,x),\ cover(Com_j,x) \Rightarrow r_i(x)=r_j(x).$$

Description D is ***complete*** on the set $X' \subseteq X$ if at least one rule for each object $x \in X'$ exists so that its complex covers this object

$$\forall x \in X',\ X' \subseteq X,\ \exists r_j \in D,\ cover(Com_j,x\}.$$

The inductive learning algorithms have to allow us to find descriptions that are at the same time both consistent and complete.

III. A NEW APPROACH

Next we present a new approach of IL which gives a new solution to previously named problems. At first we define a new concept “Determinative set of rules” (DSR), then describe an algorithm that can find it and describe how we can use this rule set for further analysis.

A. Basis of the New Approach

Let a data table X(N,M) be given and a set B of all possible rules for all classes and each rule in B is presented only once.

The Determinative set of rules (DSR) consists of all rules which are not contained in other rules of B.

B = {Ri}, i=1, 2, ..., K where K is a number of all possible rules. $Ri \neq Rj$, $i \neq j$.

$Ri \in DSR$ if there $/\exists\ Rt \in B, Ri \subset Rt, t \neq i$. $DSR \subseteq B$

It means that DSR does not contain the subrules of its rules. To get DSR from B we have to throw out all the subrules of the rules. We call this process „rule set compression“.

Example. Let B contain 4 rules:

r1: IF T1=1 & T2=1 THEN CLASS=1

r2: IF T1=1 & T3=2 THEN CLASS=2

r3: IF T2=1 THEN CLASS=1

r4: IF T3=2 THEN CLASS=2

As we see, the rule r1 is contained in r3 and r2 is contained in r4. According to the definition $DSR_B = \{r3, r4\}$.

The main features of DSR are:

1. there are no redundant attributes in rules,
2. the same object in X can be described by several rules.

B. Description of the Algorithm

Here we describe the algorithm realizing the new IL approach. The findable set of rules is DSR together with some redundant rules which are eliminated afterwards (rule set compression).

Algorithm MONSAMAX2 is given in Fig. 1.

This is a depth-first-search algorithm that makes subsequent extracts of objects containing certain factors (i.e., an attribute with a certain value). At each level first the rules (of that extract) are detected and then factors for making extracts of the next level are selected one by one.

The algorithm uses frequency tables for both all the objects in the current extract and each class of the current extract. We call them “3D frequency tables”. If a factor has equal frequencies for all objects and in any of the classes then this factor completes a rule. The rule includes also the factors chosen on the way to that extract.

The selection criteria for choosing the next factor are based on frequencies, the maximal frequency for all objects (of extract). If only one attribute (of the extract) has free (unused) value(s) (indicated by frequencies over zero) then it is not practical to make a next (further) extract because there would be no free factors to distinguish objects of different classes in that extract. If there are no free factors (i.e., no frequencies over zero) then obviously it is not possible to make a next extract. In both cases the algorithm backtracks to the previous level.

 ISBN: 978-1-61208-162-5

Algorithm MONSAMAX2
S0. t:=0; U_t:=∅
S1. Find frequencies in whole dataset and each class
  If t>0 then
    Bring zeroes down
    Backward comparison
S2. For each factor A such that its frequency in some class C is equal to its frequency in the whole set
    output rule {Ui}&A→C, i=0,…,t
    A←0
S3. If not enough free factors for making an extract then
    If t=0 then Goto End
    Else t:=t-1; Goto S3
S4. Choose a new (free) factor U_t
  U_t ←0; t:=t+1;
  extract subtable of objects containing U_t;
  Goto S1
End. Rules are found

Figure 1. Algorithm MONSAMAX2

Each factor that has been used for making an extract or completing a rule is set to zero in the corresponding frequency table. Zero in the frequency table means that this factor is eliminated from the analysis at this level.

Each frequency table (except for the initial level) inherits all zeroes of the previous level (we call it "bringing zeroes down").

Also, after making an extract, its (non-zero) frequencies are compared to the ones of the previous level. Equal frequencies at both levels mean that all objects containing that factor are contained in the extract of the current level and all possible rules containing them are found at current and subsequent levels. In order to prevent repetitious finding of such rules the frequencies of those factors are set to zero at the previous level. This technique is called "backward comparison". Using this pruning technique we can also determine the extractedness of all rules for some class, i.e., if for some class all frequencies are equal to zero at the initial level, it means that all rules for this class are found.

All these techniques can effectively decrease the number of extracts (nodes of the search tree) without losing the rules of DSR.

C. Example

In the following example data from [5] are used (Table I). In order to get a numerical representation the coding shown in Table II is used. Coded data are shown in Table III.

For given data frequencies are found across all data and across each class (see Table III). If frequencies of some factor are equal in the whole dataset and some class, we can complete the rule. In the given dataset/extract that factor determines the class. From the initial frequency tables (Table III) 3 rules are found this way:

R1: T2.1 → Class 1
R2: T3.2 → Class 1
R3: T2.2 → Class 2

TABLE I. EXAMPLE SET (FROM QUINLAN)

Object	Height	Hair	Eyes	Class
1	tall	dark	blue	–
2	short	dark	blue	–
3	tall	blond	blue	+
4	tall	red	blue	+
5	tall	blond	brown	–
6	short	blond	blue	+
7	short	blond	brown	–
8	tall	dark	brown	–

TABLE II. CODING OF VALUES

Attribute Value	Height T1	Hair T2	Eyes T3	Class
1	short	dark	blue	–
2	tall	red	brown	+
3		blond		

TABLE III. INITIAL DATA AND FREQUENCIES

Object	T1	T2	T3	Class
1	*2*	*1*	*1*	*1*
2	*1*	*1*	*1*	*1*
3	*2*	*3*	*1*	*2*
4	*2*	*2*	*1*	*2*
5	*2*	*3*	*2*	*1*
6	*1*	*3*	*1*	*2*
7	*1*	*3*	*2*	*1*
8	*2*	*1*	*2*	*1*

Value	T1	T2	T3	Class
1	3	**3**	5	
2	5	**1**	**3**	*all*
3		4		
1	2	**3**	2	
2	3	0	**3**	*1*
3		2		
1	1	0	3	
2	2	**1**	0	*2*
3		2		

The frequencies of those factors (T2.1, T2.2, T3.1) are set to zero in the current frequency table (see Table IV). Now the factor with the biggest frequency is selected for making an extract. We have two candidates: T1.2 and T3.1, both with frequency 5. As we do not have additional information we choose the first one. The chosen factor is T1.2 (with frequency 5). The extract by T1.2 and the corresponding frequencies are given in Table V.

The cells with grey backgrounds are prohibited factors that have zeroed frequencies in the previous level. This frequency table completes no rules. T3.1 with frequency 3 is chosen for making a subsequent extract (see Table VI).

In this frequency table the frequency of T2.3 in Class 2 is the same as in the previous level (see Table V), in the

previous level this frequency is set to zero (because everything connected to it will be done at a lower level).

From the current frequency table the next rule is found:

R4: T1.2&T3.1&T2.3 → Class 2

After completing a rule, the frequency of T2.3 is set to zero and the current frequency table contains no more usable frequencies.

Turning back to the previous level (Table V) it occurs that after zeroing the frequency of T3.1 (as a basis of the extract just made) there is only one usable factor (T2.3). It makes no sense to make an extract by it.

Therefore we turn back to the initial level. The frequencies are given in Table VII. The frequency of the last basis for the extract T1.2 is set to zero. The basis for the next extract is T3.1 with frequency 5. The extracted data and corresponding frequencies are given in Table VIII.

Backward comparison finds two factors with equal frequencies at the current and previous (see Table VII) levels: T1.1=1 in Class 2 and T2.3=2 in Class 2. Both frequencies are set to zero at the previous level. As we can see, the frequency table for Class 2 at the initial level is empty which means that all the rules for Class 2 will be extracted after traversing the extract by T3.1.

TABLE IV. FREQUENCIES AFTER EXTRACTING 3 RULES

Value	T1	T2	T3	Class
1	3	*0*	5	
2	**5**	*0*	*0*	*All*
3		4		
1	2	*0*	2	
2	**3**	*0*	*0*	*1*
3		2		
1	1	*0*	3	
2	**2**	*0*	*0*	*2*
3		2		

TABLE V. EXTRACT BY T1.2=5 AND CORRESPONDING FREQUENCIES

Object	T1	T2	T3	Class
1		*1*	*1*	*1*
3		*3*	*1*	*2*
4		*2*	*1*	*2*
5		*3*	*2*	*1*
8		*1*	*2*	*1*

Value	T1	T2	T3	Class
1		*0*	**3**	
2		*0*	*0*	*all*
3		2		
1		*0*	**1**	
2		*0*	*0*	*1*
3		1		
1		*0*	**2**	
2		*0*	*0*	*2*
3		1		

TABLE VI. EXTRACT BY T1.2&T3.1=3 AND CORRESPONDING FREQUENCIES

Object	T1	T2	T3	Class
1		*1*		*1*
3		*3*		*2*
4		*2*		*2*

Value	T1	T2	T3	Class
1		*0*		
2		*0*		*All*
3		**1**		
1		*0*		
2		*0*		*1*
3		0		
1		*0*		
2		*0*		*2*
3		**1**		

TABLE VII. FREQUENCIES AT THE INITIAL LEVEL

Value	T1	T2	T3	Class
1	3	*0*	**5**	
2	*0*	*0*	*0*	*All*
3		4		
1	2	*0*	**2**	
2	*0*	*0*	*0*	*1*
3		2		
1	**1**	*0*	**3**	
2	*0*	*0*	*0*	*2*
3		**2**		

TABLE VIII. EXTRACT BY T3.1=5 AND CORRESPONDING FREQUENCIES

Object	T1	T2	T3	Class
1	*2*	*1*		*1*
2	*1*	*1*		*1*
3	*2*	*3*		*2*
4	*2*	*2*		*2*
6	*1*	*3*		*2*

Value	T1	T2	T3	Class
1	2	*0*		
2	*0*	*0*		*all*
3		**2**		
1	1	*0*		
2	*0*	*0*		*1*
3		0		
1	**1**	*0*		
2	*0*	*0*		*2*
3		**2**		

From the current extract (Table VIII) we get a rule:

R5: T3.1&T2.3 → Class 2

After the frequency of T2.3 is set to zero (at the current level) only one non-zero frequency is left (for T1.1). The

possible extract by it cannot give any rules. Therefore algoritm backtracks to the initial level.

The current state of frequencies is given in Table IX.

The next extract is made by T2.3 (see Table X), there are no rules. There is only one frequency above zero in the frequency table, therefore we backtrack to the previous (initial) level.

In the frequency table of the initial level (see Table XI) there is now only one usable (non-zero) frequency. It cannot give a rule because if it could then it could be extracted from the initial table at the beginning. An extract is not made. The work is finished.

TABLE IX. FREQUENCIES AT THE INITIAL LEVEL

Value	T1	T2	T3	Class
1	3	*0*	*0*	
2	*0*	*0*	*0*	*all*
3		**4**		
1	2	*0*	*0*	
2	*0*	*0*	*0*	*1*
3		**2**		
1	*0*	*0*	*0*	
2	*0*	*0*	*0*	*2*
3		***0***		

TABLE X. EXTRACT BY T2.3=4 AND CORRESPONDING FREQUENCIES

Object	T1	T2	T3	Class
3	*2*		*1*	*2*
5	*2*		*2*	*1*
6	*1*		*1*	*2*
7	*1*		*2*	*1*

Value	T1	T2	T3	Class
1	2		*0*	
2	*0*		*0*	*all*
3				
1	1		*0*	
2	*0*		*0*	*1*
3				
1	*0*		*0*	
2	*0*		*0*	*2*
3				

TABLE XI. FREQUENCIES AT THE INITIAL LEVEL

Value	T1	T2	T3	Class
1	**3**	*0*	*0*	
2	*0*	*0*	*0*	*all*
3		*0*		
1	**2**	*0*	*0*	
2	*0*	*0*	*0*	*1*
3		*0*		
1	***0***	*0*	*0*	
2	*0*	*0*	*0*	*2*
3		*0*		

So, we extracted 5 rules: R1: T2.1 → Class 1, R2: T3.2 → Class 1, R3: T2.2 → Class 2, R4: T1.2&T3.1&T2.3 → Class 2, R5: T3.1&T2.3 → Class 2.

As we see the extracted rule set is not DSR because of the rule R4 which is a subrule of R5. After the compression of the extracted rule set we get a DSR: R1, R2, R3 and R5.

The number of extracted rules for MONSAMAX2 depends on the criteria of choosing the leader value for an extract in a situation when there are several candidates with equal frequencies. For example, if we would choose in the beginning of the algorithm T3.1 (with frequency 5) as a leader value instead of T1.2 (frequency=5), then we would extract only 4 rules (R1, R2, R3 and R5) and compression would not be needed (but we do not know this). MONSAMAX2 produces more additional information for effective rule set compression, but here is not enough space for presenting it.

D. Discussion

For the same purpose – to find a complete and consistent description – algorithms MONSIL [6] and DEILA [7] have been proposed. Each of the algorithms (MONSAMAX2, MONSIL and DEILA) work in a different way and usually give different descriptions. The common idea is the step following the main algorithm – compression of the found rule set in order to get a result as compact as possible .

Similarly to MONSAMAX2 the result of MONSIL (before compression) depends on the choice of leader value for making extract when there are several candidates with equal frequencies. For the same Quinlan's data [5] as here (see Table I), two different results of MONSIL are given (in [6]): the first one consisting of 8 rules and the second – 5 rules. In the latter, the redundant rule (T1.1&T2.3&T3.1 → Class 2) is not the same as in case of MONSAMAX2 (T1.2&T3.1&T2.3 → Class 2). The result which consists of 8 rules contains two more redundant rules (containing T1.1) in addition to these two.

We noticed that after compression the results of MONSIL and MONSAMAX2 are the same. This rule set is called DSR (determinative set of rules).

Algorithm DEILA finds more rules than MONSAMAX2 and MONSIL. From Quinlan's data it finds 15 rules. The result of DEILA may not contain all DSR rules. Some of them can be "replaced" by longer rules. This is due to DEILA's working principle – all found rules are dicliques.

The amount of extracted rules for MONSAMAX2 is smaller than for MONSIL because the first one extracts shorter rules first while the latter extracts longer rules first. The difference is in the number of (redundant) subrules – the rules that will be removed by compression. MONSAMAX2 finds fewer such rules (due to finding shorter rules first).

In order to determine the belonging of the extracted objects to the same class in the process of extracting rules, MONSIL must make an extract and usually the objects do not belong to the same class. It means that we have made a superfluous effort. MONSAMAX2 works so effectively because we have data to determine belonging of objects to the same class using 3D frequency tables, there is no need to make these extracts.

 ISBN: 978-1-61208-162-5

During the work of MONSAMAX2 we can also observe every class covering with rules: if 3D frequencies for some class are empty at the initial level it means that all rules for this class are extracted.

IV. CONCLUSION

This paper proposes a new approach and the corresponding algorithm MONSAMAX2 for finding a determinative set of overlapping rules. The algorithm is based on frequency tables and new pruning techniques which make it easy to detect a potential DSR rule.

MONSAMAX2 is more effective compared to the former algorithm MONSIL because it prevents the making of many unnecessary extracts due to using 3D frequency tables. Also it finds less redundant (i.e., non-DSR) rules because it finds shorter rules first while MONSIL starts from the longer ones.

On the basis of DSR we can form and solve next tasks, for example, to find

1. the shortest rules (by the number of attributes (selectors) in the rule),
2. the longest rules (by the number of attributes in the rule),
3. the rules with specific features (for example, all rules with r selectors),
4. the shortest rule system (i.e., the rule system with the smallest number of rules),
5. the rule system which consists of rules with minimal number of selectors,
6. all the rule systems we can form on the basis of DSR.

All these tasks are necessary for the post-analysis of the extracted rules. It means that several new possibilities are available for experimentation with several rule sets (subsets of DSR) and for describing them. We must not try to minimize the rule set during the work of a machine learning algorithm, we can find the best solution during the post-analysis of DSR.

Using DSR and the post-analysis of rules also gives the possibility to gather statistics about the use of rules in classification in order to analyze the rules' perspective and their power of classification. We can also see which rules classify more accurately and which do not on the basis of the information we have about classified (test-set and real) objects. On this basis we can reorder the rules in the rule set. DSR is a good basis for developing this approach.

Somebody might say that the finding of DSR is very laborious, especially in cases of large amounts of data. If so, the user can decide what is the purpose of the work. If the purpose is a quick one-time information gathering for a data set under analysis then the use of DSR-based IL approach may not be the best one. But if the purpose is to describe the data set and through that discover new knowledge and get an opportunity for post-analysis of the rule set then this approach is a good solution.

The post-analysis of rules will be the topic of the next paper.

REFERENCES

[1] L. Torgo, "Rule combination in inductive learning," in Machine Learning: ECML-93, ser. Lecture Notes in Computer Science, P. Brazdil, Ed. Springer Berlin / Heidelberg, 1993, vol. 667, pp. 384-389.

[2] T. Treier, "A new effective approach for solving the rules conflict problem", 2011 International Conference on Intelligent Computing and Control (ICOICC 2011), May 2011, in press.

[3] I. Birzniece, "From Inductive Learning towards Interactive Inductive Learning," in Scientific Journal of Riga Technical University, Computer Science. Applied Computer Systems, vol. 41, 2010, pp. 106-112.

[4] M. Gams and N. Lavrac, "Review of Five Empirical Learning Systems within a Proposed Schemata," in I. Bratko, N. Lavrac (Eds.), Progress in Machine Learning, Proceedings of EWSL 87: 2nd European Working Session on Learning, Bled, Yugoslavia, May 1987. Sigma Press, Wilmslow, 1987, pp. 46-66.

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

[6] P. Roosmann, L. Võhandu, R. Kuusik, T. Treier, and G. Lind, "Monotone Systems approach in Inductive Learning," in International Journal of Applied Mathematics and Informatics, Issue 2, Vol. 2, 2008, pp. 47-56.

[7] R. Kuusik, T. Treier, G. Lind, and P. Roosmann, "Machine Learning Task as a Diclique Extracting Task," 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery: FSKD'09, Tianjin, China, August 14-16, 2009; Los Alamitos, California: Conference Publishing Service, 2009, pp. 555-560.

 ISBN: 978-1-61208-162-5