A Minimization Approach to Propositional Inductive Learning

Dragan Gamberger

Rudjer Bošković Institute, 41000 Zagreb, Croatia

Abstract. An approach to the problem of propositional inductive learning of if-then-else rules, different from the commonly used ones, is presented in the paper. Main differences are: literal selection process that searches for the smallest set of literals so that the completely correct rule for all learning examples can be constructed and the nonnormal form of the generated rules built by the search for necessary and sufficient conditions of example classes. It is also presented how iterative application of the literal selection process can solve the problem of learning from noisy domains by appropriate exclusion of some learning examples. The results of application of the system that includes described algorithms on a few publicly available domains are discussed.

1 Introduction

The well known propositional inductive learning systems like ID3,AQ15,CN2, C4.5 have already been successfully applied on different learning domains [10, 8, 3, 12]. Although the research interests have moved towards learning first-order relations [11], some recent results have independently shown that the problem of propositional induction is neither completely solved yet. The main problem is, as shown in [2], that every of the known systems is well suited for some class of the problems while it can perform rather bad on the others. This is especially obvious for noisy domains when tree pruning or rule truncation must be used [1]. Also the results of application of m-estimates [4] as well as the application of CNF for generated rules [9] undoubtedly demonstrate that some modifications of the well known algorithms will be necessary in the future in order to make them more general. Besides, the development of LINUS shows that under some assumptions the propositional systems can be effectively used in building the first-order learning systems [7], what additionally stresses the importance of reliable and general propositional systems.

In this paper some novel ideas in building rule based propositional inductive learning systems, already applied on the ILLM (Inductive Learning by Logic Minimization) system will be presented [5]. Main differences to the known systems are as follows: literal selection process, the form of generated rules and noise handling procedure. These topics are described each in a separate section. At the end some comparative results are presented.

The term 'literal' is used in the paper in the sense as in [9]. It is equivalent to the term selector in AQ systems and logical test in ILLM. The form or complexity of literals is not restricted in any sense so that presented algorithms can be combined by some constructive induction methods or that new literals can be formed due to the background knowledge.

In the paper only the basic inductive learning problem with only two example classes, positive and negative, will be discussed. By known techniques the results can be applied on many-classes problems as well.

2 Literal Selection Process

All known propositional inductive learning systems select significant literals (attributes) and generate the rule (decision tree) at the same time. Some forms of information gain metric based on the total number of positive and negative examples before and after the selection is always used. Details can be found in [6, 3]. In ILLM system the procedures for literal selection and rule generation are separated. The aim of the first ILLM step is to select significant literals, and after it is completely done they are used in the rule generation process (second ILLM step described in the next section). Another difference is that the selection process does not use the information based metric. The significant literals are selected so that the smallest number of different literals should be used in the rule that will satisfy all learning examples.

This property has two main consequences. The first one is that in the basic form the suggested algorithm can work only on noise-free domains without contradictions. In Sec. 4 it will be shown how its iterative usage can enable its application on noisy domains as well. The second one is that total number of positive and negative examples that can be distinguished by some literal does not influence the selection process in the suggested algorithm. It means that, for example, presence or addition of example copies in the learning set can not change the subset of selected literals, what is not true for other systems. This characteristic enables elimination of example copies, and elimination of some other learning examples under conditions defined later in the section, before the actual beginning of the search process. This can simplify the learning task.

The two step inductive learning algorithm and so defined its first step introduces a completely new paradigm in inductive learning: the importance of the pairs of examples, each pair consisting of a positive and a negative learning example. Such pairs are called 1/0 pairs in the rest of the paper. Their importance stems from the fact that if we want to have a set of literals such that it is possible to form a rule that uses only literals from the set, then for any possible 1/0 pair a literal that covers it must be an element of the set [5]. A 1/0 pair is covered by a literal if it is true for the positive example and not true for the negative example in the pair. This directly implies the importance of 1/0 pairs built of the positive and negative learning examples that differ in a small number of attributes because such pairs are covered by a small number of literals and at least one of them must be an element of any minimal set of selected literals.

The main advantage of the algorithm is that it is concentrated on the search for the globally most relevant literals. The disadvantage is that it has space complexity $O(e \cdot l)$ and time complexity $O(e^2 \cdot l^K)$, where e is number of examples, l total number of possible literals and K number of selected literals. The algorithms time complexity can be reduced to $O(e^2 \cdot l \cdot K)$ if heuristic instead of exhaustive search is used. In this case space complexity is $O(e^2)$ and it can not be guaranteed that the real minimum is found. Practically, this algorithm characteristic limits its usage to the domains of up to 1000 learning examples. For greater domains a windowing procedure based on 1/0 pairs of small distance must be applied.

The search for the smallest subset of literals, either exhaustive or heuristic, can be realized in a few different ways. In the ILLM so called covering tables are introduced that, besides clear algorithm definition and fast manipulations, enable some reduction of the starting search space. These tables can be effectively used during the second algorithm step as well.

2.1 Definition and Reduction of Covering Tables

For learning domain two covering tables can be defined: one for positive and the other one for negative examples. The first one is called TO and it has as much rows as there are positive examples, while the other one is called TZ and it has as much rows as there are negative examples. Both tables have the same number of columns and it is equal to the number of possible literals that can be used. The elements of both tables are 1 and 0, including the cases when learning examples have some attributes of unknown value. An element of TO table has value 1 if the corresponding literal (by column) is true, or could be true by appropriate substitution of unknown values, for the corresponding example (by row). An element of TZ table has value 1 if the literal is *not true*, or could be *not true*, for the corresponding negative learning example. All other elements of TO and TZ tables have value 0. When a literal has value 1 for an example, either positive or negative, it is said that the example is covered by the literal.

After TO and TZ tables are formed for the domain, whole relevant information necessary for rule generation is contained in them. The algorithm does not need any further access to the learning examples.

The first ILLM step can be also defined as the minimal covering problem with object to find a minimal subset of literals that covers all *1/0 pairs*. The problem can be reduced if some columns and/or rows of TO and TZ tables are eliminated.

A column for a literal can be eliminated from both TO and TZ tables if there is another literal that covers the same, and potentially also some additional 1/0 pairs of examples. The condition is equivalent to the requirement that the column that can be eliminated has values 0 at least for all rows of both TO and TZ tables as some other column that will remain in the tables.

An example can be eliminated from either TO or TZ table if it is covered by same or potentially some additional literals as another example from the same class. The condition is equivalent to the requirement that the row that can be eliminated has values 1 in at least all columns as another row that would remain in the same table.

3 Form of Generated Rules

In most inductive learning systems that generate rules DNF is used. By experiments on 5 natural data sets, it has been recently shown that the application of CNF may lead in some cases to more concise rules [9]. In this section the concept of necessary and sufficient conditions which can result in the nonnormal form of rules that is generalization of both DNF and CNF will be presented. This algorithm is implemented in the second step of the ILLM system. It is supposed that the algorithm starts from the minimal set of literals generated by the first algorithm step although it can start from any set, including the set of all possible literals, that enables the rule that satisfies all learning examples to be constructed. The reason for the execution of the first step is, beside simplification of the rule generation process, selection of the globally significant literals.

A sufficient condition is a literal or conjunction of literals that covers some positive and all negative examples. A necessary condition is a literal or disjunction of literals that covers all positive and some negative examples. The rule generation process is iterative repetition of the following procedure:

For the given set of positive and negative learning examples (presented by TO and TZ tables respectively) find the minimal sufficient and the minimal necessary condition.

- If the minimal sufficient condition has less literals than the minimal necessary condition, or if the number of literals is equal but the sufficient condition covers greater number of examples than the necessary condition, then generate a part of the rule consisting of the minimal sufficient condition in the form of one literal or more literals connected by AND operation. Eliminate positive examples covered by the condition. If TO is empty stop rule generation process. Else add the sign of OR operation to the generated rule in order to connect the generated condition with the remaining part that will be generated in succeeding iterations.
- Else, if minimal necessary condition has less literals than minimal sufficient condition, or if the number of literals is equal but the necessary condition covers greater number of examples than the sufficient condition, then generate a part of the rule consisting of the minimal necessary condition. This condition has the form of one literal or more literals connected by OR operation and written in brackets. Eliminate negative examples covered by the necessary condition. If TZ table is empty stop the rule generation process. Else add the sign of AND operation to the generated rule in order to connect the generated condition with the remaining part that will be generated in succeeding iterations.

The main advantage of the approach is that it adapts the generated rule form to each domain. Beside DNF and CNF it can generate mixed forms that can be easily understood by users as well. The obtained rule form can be interpreted also as an extraction of common literals or groups of literals in order to simplify its presentation.

Example 1. A rule generated by literals lt1,lt2, .. , lt9 may look like:

$$\underbrace{\text{lt1}}_{\text{Suff.}} \vee \underbrace{\text{lt2} \cdot \text{lt3}}_{\text{Suff.}} \vee \underbrace{[\text{lt4} \vee \text{lt5}]}_{\text{Nec.}} \cdot [\underbrace{\text{lt6}}_{\text{Suff.}} \vee \underbrace{\text{lt7}}_{\text{Nec.}} \cdot \underbrace{\text{lt8}}_{\text{Nec.}} \cdot \underbrace{\text{lt9}}_{\text{Nec.}}]$$

4 Noise Handling Procedure

The algorithm described in Sec. 2 and 3 generates a rule that satisfies all learning examples. Because of its two step nature, truncation or similar procedures implemented on known inductive algorithms, can not be easily applied. However, this algorithm has rather precise measure of the complexity of the generated rules: the number of literals selected by its first step. This enables an iterative approach to the problem of noise in the learning domain.

The algorithm starts with the first step described in Sec.2. It selects the minimal set of literals that enables construction of the rule that satisfies all learning examples. Now, it is tested if by elimination of any of learning examples (or by elimination of a pair of learning examples in great domains) the minimal set of literals could be reduced. If yes, exclude the example (or the example pair) from the learning domain and repeat the procedure till the minimal set that can not be further easily reduced is obtained. That minimal set is input of the second step described in Sec.3. The formed rule will satisfy at least all nonexcluded examples.

The test that exclusion of an example (or an example pair) can reduce minimal literal set is a very complex task. But if the search space is reduced to the subsets of the already generated minimal set then it can be easily realized in the following way: for each literal from the minimal set find all 1/0 pairs of learning examples that are covered exclusively by the literal. If all this 1/0 pairs have an example (or an example pair) in common, by its elimination this literal will not be necessary any more.

The comparison of this noise handling procedure and the known truncation techniques is a problem of further research. It is obvious that the presented approach opens some novel possibilities like directed control of the class of excluded examples (positive or negative) what can be interesting for some, especially medical, domains.

5 Example

A small artificial learning domain is here included to show the presented concepts. In Table 1 the learning set is presented. There are 10 examples with 3 attributes of the quality type. Their names are c.bg (background color), c.frame (frame color), and c.object (object color). The classes are harmony 1 and 0. Attribute values are different colors.

For this learning set it is possible to generate 27 different literals of the form $(input = color)$, $(input \neq color)$, $(inputA = inputB)$, and $(inputA \neq inputB)$.

Table 1. Learning set used to learn the concept harmony

s	s	s	o
c.bg	c.frame	c.object	**harmony**
black	blue	yellow	1
red	yellow	yellow	1
blue	blue	white	1
red	white	white	1
yellow	blue	blue	1
white	yellow	yellow	0
black	red	red	0
red	white	blue	0
red	grey	white	0
white	grey	yellow	0

They are: $(c.bg = black)$, $(c.bg = red)$, $(c.bg = blue)$, $(c.bg = yellow)$, $(c.bg \neq white)$, $(c.bg \neq black)$, $(c.bg \neq red)$, $(c.frame = blue)$, $(c.frame = yellow)$, $(c.frame = white)$, $(c.frame \neq yellow)$, $(c.frame \neq red)$, $(c.frame \neq white)$, $(c.frame \neq grey)$, $(c.object = yellow)$, $(c.object = white)$, $(c.object = blue)$, $(c.object \neq yellow)$, $(c.object \neq red)$, $(c.object \neq blue)$, $(c.object \neq white)$, $(c.bg = c.frame)$, $(c.bg = c.object)$, $(c.frame = c.object)$, $(c.bg \neq c.frame)$, $(c.bg \neq c.object)$, $(c.frame \neq c.object)$. For these literals the generated TO and TZ tables are presented in Table 2.

Table 2. TO and TZ tables for 27 literals

TO

1	0	0	0	1	0	1	1	0	0	1	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1
0	1	0	0	1	1	0	0	1	0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	1	0
0	0	1	0	1	1	1	1	0	0	1	1	1	1	0	1	0	1	1	1	0	1	0	0	0	1	1
0	1	0	0	1	1	0	0	0	1	1	1	0	1	0	1	0	1	1	1	0	0	0	1	1	1	0
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	0

TZ

1	1	1	1	1	0	0	1	0	1	1	0	0	0	0	1	1	1	0	0	0	1	1	0	0	0	1
0	1	1	1	0	1	0	1	1	1	0	1	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1
1	0	1	1	0	0	1	1	1	0	0	0	1	0	1	1	0	0	0	1	0	1	1	1	0	0	0
1	0	1	1	0	0	1	1	1	1	0	0	0	1	1	0	1	0	0	0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	1	1	1	0	0	0	1	0	1	1	1	0	0	0	1	1	1	0	0	0

The columns in the tables correspond to the literals in the same order as they are cited. The rows of TO table correspond to ex1-ex5 and rows of TZ table to ex6-ex10. It can be noticed that in the tables there are some literals covered by other literals (e.g. column 1 is covered by column 8). After their elimination there remain only 11 literals that are interesting for rule generation.

They are: $(c.bg = red)$, $(c.frame = blue)$, $(c.frame = white)$, $(c.object = yellow)$, $(c.object = white)$, $(c.bg \neq white)$, $(c.frame \neq red)$, $(c.frame \neq white)$, $(c.frame \neq grey)$, $(c.object \neq blue)$, $(c.frame = c.object)$. The reduced TO and TZ tables are presented in Table 3.

Table 3. TO and TZ tables after column reduction

TO

ex1	0	1	0	1	0	1	1	1	1	1	0
ex2	1	0	0	1	0	1	1	1	1	1	1
ex3	0	1	0	0	1	1	1	1	1	1	0
ex4	1	0	1	0	1	1	1	0	1	1	1
ex5	0	1	0	0	0	1	1	1	1	0	1

TZ

ex6	1	1	1	0	1	1	0	0	0	0	0
ex7	1	1	1	1	1	0	1	0	0	0	0
ex8	0	1	0	1	1	0	0	1	0	1	1
ex9	0	1	1	1	0	0	0	0	1	0	1
ex10	1	1	1	0	1	1	0	0	1	0	1

It can be noted that although none of the rows of the original tables is covered by any other row of the same table, after reduction of the number of the literals (Table 3) ex6 covers ex10. After elimination of the row for ex10, the result is presented by the tables in Table 4. Any further reduction of the number of their rows and columns is not possible.

Table 4. TO and TZ tables after row reduction

TO

ex1	0	1	0	1	0	1	1	1	1	1	0
ex2	1	0	0	1	0	1	1	1	1	1	1
ex3	0	1	0	0	1	1	1	1	1	1	0
ex4	1	0	1	0	1	1	1	0	1	1	1
ex5	0	1	0	0	0	1	1	1	1	0	1

TZ

ex6	1	1	1	0	1	1	0	0	0	0	0
ex7	1	1	1	1	1	0	1	0	0	0	0
ex8	0	1	0	1	1	0	0	1	0	1	1
ex9	0	1	1	1	0	0	0	0	1	0	1

From 4 it can be seen that there are 3 necessary conditions (columns 6,7,9) and 1 sufficient condition (column 2) consisting of only one literal. The sufficient condition is selected because it covers 3 positive examples while each of necessary conditions covers only one negative example. After this selection and elimination of the mentioned 3 positive examples, beside previous 3 necessary conditions two other necessary conditions (columns 1,11) that cover two negative examples each, can be detected in the reduced TO table. By selection of these conditions all negative examples are eliminated. The generated rule has the form:

$$\textbf{harmony IF} \quad (c.frame = blue) \vee (c.bg = red) \cdot (c.frame = c.object).$$

6 Experimental Results

Apart from the results presented in [5] for the biodegradation domain, there are the concise results of application of the ILLM on three publicly available domains here.

For the 'Breast Cancer' database from the University of Wisconsin Hospitals obtained through UCI repository (699 learning examples), at first the ILLM excluded 24 examples and then selected following 5 literals:

A $(ClumpThickness > 5)$
B $(UniformityofCellSize > 2)$
C $(UniformityofCellShape > 2)$
D $(SingleEpithelialCellSize > 2)$
E $(BareNuclei > 3)$.

The generated rule

$$\textbf{malignant IF } \mathrm{CD} \vee \mathrm{AB} \vee \mathrm{E}[\mathrm{A} \vee \mathrm{B}[\mathrm{C} \vee \mathrm{D}]]$$

is not correct for 3 malignant and 18 benign learning cases. These results are very similar to those reported in [14] obtained by 3 other learning systems.

For the 'Mushrooms' database created at University of California at Irvine and obtained through the same repository (8124 learning examples), the ILLM used windowing based on 500 examples and after 4 iterations selected following 9 literals:

A $(cap - color = yellow)$
B $(bruises = yes)$
C $(bruises = no)$
D $(odor \neq none)$
E $(gill - spacing = close)$
F $(gill - size = narrow)$
G $(stalk - root = bulbous)$
H $(spore - print - color = green)$
I $(spore - print - color = white)$

The system does not suggest the exclusion of any example and generated the rule

$$\textbf{poisenous IF} \quad \mathrm{H} \vee \mathrm{CD} \vee \mathrm{AC} \vee [\mathrm{F} \vee \mathrm{D}][\mathrm{B} \vee \mathrm{E}][\mathrm{I} \vee \mathrm{BE}[\mathrm{F} \vee \mathrm{G}]]$$

that is correct for all learning examples. According to the knowledge of the author of the domain, it is the first such result for the domain [13].

For the 'Australian Credit Approval' database obtained from LIACC, University of Porto (690 learning examples), ILLM suggested exclusion of 32 examples and selected 15 literals. Due to its complexity the rule is not presented here. To illustrate the time complexity of the algorithm, let us mention that for that domain with only 14 attributes (descriptors) the selected starting set contained 1010 literals. The reason for such rather great number is that 6 attributes are of continuous type with lot of possible separation values. Even the minimization of covering tables before the first algorithm step did not reduce this number significantly. The minimization algorithm started from 690 learning examples and 969 literals. The result is that for one iteration of the first algorithm step (nonexhaustive search) the execution time was about 2 hours on a HP-Apollo workstation. The total time for the complete rule generation was about 60 hours. Such long execution time in this case is due to attribute characteristics as much as the consequence of very noisy domain and relatively weak correlation between attribute values and example classes. The results obtained by ILLM and other systems prove this statements. Nevertheless, from the rules generated by ILLM for such domains, one can expect relatively good prediction accuracy as well. For 10-fold cross validation on the 'Australian Credit Approval' database the ILLM system had prediction error rate of 13% while the reported error rate for all 22 different systems tested under Statlog project was greater than 13% (according to the results at LIACC, University of Porto).

7 Conclusions

The work shows that although the theory of propositional inductive learning is already well defined there is a need and a possibility for other approaches as well. Specially the generation of rules in a nonnormal form, that is a generalization of both the DNF and CNF, could be interesting for all rule generation systems.

The presented results obtained by the ILLM system for a few publicly available domains show rather great correlation in accuracy of generated rules and in the level of detected noise when compared to the results of commonly used systems. Another encouraging fact shown by the results is that iterative exclusion of 'suspicious' examples from noisy domains is a real alternative to the known algorithm of rule truncation. However, the unsolved problem of both suggested processes for literal selection and noise handling is their time complexity.

Acknowledgements

The 'Wisconsin Breast Cancer' and 'Mushrooms' databases were copied by anonymous ftp from ics.uci.edu from directory pub/machine-learning-databases. The 'Australian Credit Approval' database was copied by anonymous ftp from ftp.ncc.up.pt from directory pub/statlog/datasets.

References

1. I.Bratko, *Learning and Noise*. Proceedings of MLnet Summer School on Machine Learning and Knowledge Acquisition, Dourdan, France, 1994, pp.121-143.
2. P.Brazdil,J.Gama,B.Henery, *Characterizing the Applicability of Classification Algorithms Using Meta-level Learning*. Proceedings of MLnet Workshop on Industrial Applications of Machine Learning, Dourdan, France, 1994, pp.127-146.
3. P.Clark, T.Niblett, *The CN2 Induction Algorithm*. Machine Learning, Vol.3, 1989, pp.261-284.
4. S.Džeroski, B.Cestnik, I.Petrovski, *Using the m-estimate in Rule Induction*. Journal of Computing and Information Technology - CIT, Vol.1, 1993, pp.37-46.
5. D.Gamberger, S.Sekušak, A.Sabljić, *Modelling Biodegradation by an Example-Based Learning System*. Informatica, Vol.17, 1993, pp.157-166.
6. M.Holsheimer, A.Siebes,*Data Mining The Search for Knowledge in Databases*. Report CS-R9406, CWI, Amsterdam, 1994. (can be obtained by anonymous ftp from ftp.cwi.nl /pub/CWIreports/AA/CS-R9406.ps.Z)
7. N.Lavrač, S.Džeroski, *Weaking the Language Bias in LINUS*. J. Expt. Theor. Artif. Intell., Vol.6, 1994, pp.95-119.
8. R.S.Michalski, I.Mozetič, J.Hong, N.Lavrač, *The Multipurpose Incremental Learning System AQ15 and Its Testing Application to Three Medical Domains*. Proc. Fifth Nat'l Conf. Artificial Intelligence, Morgan Kaufmann, San Mateo, Calif., 1986, pp.1,041-1,045.
9. R.J.Mooney, *Encouraging Experimental Results on Learning CNF*. Technical report, University of Texas, October 1992.
10. J.R.Quinlan, *Induction of Decision Trees*. Machine Learning, Vol.1,1986, pp.81-106.
11. J.R.Quinlan, *Learning Logical Definitions from Relations*. Machine Learning, Vol.5, 1990, pp.239-266.
12. J.R.Quinlan, *C4.5:Programs for Machine Learning*, Morgan Kaufmann, 1992.
13. J.S.Schlimmer, *Concept Acquisition Through Representational Adjustment*, (Technical Report 87-19). Doctoral dissertation, Department of Information and Computer Science, University of California, Irvine, 1987.
14. W.H.Wolberg, O.L.Mangasarian, *Computer-Designed Expert Systems for Breast Cytology Diagnosis*. Analytical and Qauntitative Cytology and Histology, Vol.15, February 1993, pp.67-74.