Multivalue Learning in ILP

Orlando Muñoz Texzocotetla and René Mac Kinney Romero

Departamento de Ingeniería Eléctrica
Universidad Autónoma Metropolitana - Iztapalapa, P.O. Box 55-534 México D.F., México

Abstract. In this paper we present a method to make more expressive the hypotheses searched by ILP algorithms. This method allows to use more than one value in literals. The method we propose constructs new clauses using information obtained analyzing values in literals with multivalue learning. In order to discover such information, we draw from techniques used in inducing trees algorithms.

Keywords: Artificial Intelligence, Machine Learning, Unbiased Learning, ILP

1 Introduction

Learning can be viewed as a search for an hypothesis H which satisfies some quality criteria [6]. This search is carried out by a learner, which "can be described in terms of the structure of its search space, its search strategy and search heuristics" [4]. In Inductive Logic Programming (*ILP*), the search task can be described as follow: given sets of positive and negative examples (E^+ and E^-), and a background knowledge B, an ILP learner searches for an hypothesis H such that $B \wedge H \models E^+$.

Regarding the search space, in ILP is determined by the language of logic programs, which are formed with program clauses of the form $T \leftarrow Q$, where T is an atom $p(X_1, \ldots, X_n)$ and Q is a conjunction of literals $L_1, \ldots, L_m$. Also the search space's clauses form, is syntactically restricted by a bias language. This bias determines which clauses are searched from the vocabulary of predicates, function symbols and constants that are in the background knowledge [4].

When bias language is stronger (lack of expressiveness), the search space becomes smaller and more efficient, but it is likely that the final hypothesis can not represent an appropriate solution for the target problem. For instance, if the language restricts the use of the target atom into the clause's body then no hypothesis will represent an appropiate solution to any recursive problem.

Each ILP algorithm defines a language in order to construct theories with the highest expressiveness degree. However, current algorithms test only a single value in constructing literals, thus making the hypothesis search space to be constructed with lots of rules. If the background knowledge is large then it is more likely that the final theory contains many rules, hence making it more difficult to interpret.

In this paper we present a method to make more expressive the hypotheses constructed by ILP algorithms. This method allows to build literals that use more than one value. It constructs new clauses using information obtained analyzing values present in literals (values that are selected by the user) with multivalue learning.

To discover and extract that information, we implemented and adapted the algorithm of selection of split point used by two decision tree inducers: QUEST (Quick Unbiased Efficient Statistical Tree) [5] and CRUISE (Classification Rule with Unbiased Interaction Selection and Estimation) [3]. To choose the best attribute those algorithms use *analysis of variance (ANOVA)* or Levene's test for numeric attributes and *Pearson chi-square test* for categorical attributes[1]. In the case of CRUISE, this algorithm performs a *BOX-COX transformation* before QDA. To find the split point, we use *Quadratic Discriminant Analysis (QDA)*.

These new clauses (we call them multivalue clauses) are added to the background knowledge, thus allowing new literals to be used by the ILP algorithm search. This, we believe, allows ILP algorithms to construct hypotheses with fewer rules.

This article is organized as follows: in section 2 we describe, with an example, the problematic that we want to address; in section 3, we present the method to create new multivalue clauses and to make more literals available; in section 4 we show the experiments performed and the results obtained. Finally our conclusions and the future work are presented in section 5.

2 Univalue clauses

To describe our main goal, we present a pattern recognition problem *(Bongard)*[1]. This problem consists of finding a theory which describes the patterns related to the positive examples, these contains at least a triangle which points to some of the following directions: west (w), northwest (nw) and north (n). The target literal is: $\mathbf{bongard}\,(\mathbf{Example}) \leftarrow$. Hypotheses will be created from each value of following literals contained in the background knowledge:

- $triangle\,(Example, NumT)$. $circle\,(Example, NumC)$. $square\,(Example, NumS)$
- $direction\,(NumT, Direction)$ where $Direction \in \{w, nw, n, ne, e, sw, s, se\}$

In this case *Aleph* [11] (program that implements Stephen Muggleton's algorithm *Progol* [8]) returns the following theory, with three rules:

1. $bongard\,(A) : -triangle(A, B), direction(B, w).$
2. $bongard\,(A) : -triangle(A, B), direction(B, nw).$
3. $bongard\,(A) : -triangle(A, B), direction(B, n).$

[1] A categorical attribute takes values unordered, and a numerical attribute takes values on the real line.

To create this theory, ILP algorithms use clauses whose literals declare a single value to each literal. For instance, the second argument of the literal *direction*, *Direction*, is a categoric attribute with eight possible values. Each time that *direction* appears in some of the clauses which are contained in the search space, *Direction* presents only one of its values. This type of clauses, we'll call them *univalue*. Namely, if all the arguments of each literal within a clause's body declare only one value, then this is a *univalue clause*. If at least one argument presents more than one value, then we will call them *multivalue clauses*.

Thus if the number of values for attributes (categoric and/or numeric) increases significantly, then hypotheses may have lots of rules, therefore making these hypotheses more dificult to interpret. We can see that it would be very helpful to create multivalue clauses which would allow ILP algorithms to create smaller hypotheses. Therefore we can ask: is it possible that ILP algorithms can test more than one value (a set of values) at one time? How to create each set of values? and, Will multivalue clauses help to create hypotheses with fewer rules? The method we propose we believe answers the previous questions and it is detailed in the following section.

3 Multivalue clauses

The proposed method discovers information at the examples. This information is used to create new clauses. These clauses will be added to the background knowledge to allow the ILP algorithms to use them adding the necessary literals to the search space. We will use the most popular ILP algorithms FOIL [9] and [8] Progol. This method has the following steps:

1. **Creation of subsets of values.** In this step we make a binary split on the set of all values. For this we implemented the algorithm we presented in section 1 of selection of split point. Thus for each categoric value with a set $C = \{v_1, \ldots, v_m\}$, we will obtain two disjoint subsets C_1 y C_2 such that $C_1 \cap C_2 = \phi$. We also use this when we have a small number of discrete numeric values. For each numeric value we will obtain a split point d, which will divide the full set of values in two subsets. The first one will contain values less or equal than d ($C_1 = \{x \bullet x \leq d\}$), and the second one will contain values greater than d ($C_1 = \{x \bullet x > d\}$). $C_1 \cap C_2 = \phi$.
2. **Creation of multivalues clauses.** The subsets of values will be used to create multivalue clauses. Categoric attributes: each subset of values will be declared in the appropiate literal rather than a single value. Thus we will create two multivalue clauses. Numeric attributes: for these type of attributes we will create two clauses too. In the first one the split point will determine the values less or equal than d. In the second one the split point will determine the values greater than d.
3. **Modification of background knowledge.** In this step we add the multivalue clauses to the background knowledge.

4. **Usage.** The new information is used by the ILP algorithm. Making literals for the new clauses available to the ILP search.

3.1 Example

In order to explain the method proposed, we present a simple ILP example. We must find a theory on the number of sides that must have a figure that belongs to one of the following classes: quadrilateral or non_quadrilateral. The target literal is $\mathbf{class}\,(\mathbf{Fig}, \mathbf{Class}) \leftarrow$. The background knowledge declares the relation $\mathbf{side}\,(\mathbf{F}, \mathbf{S})$, which indicates the number of sides S of the figure F.

Aleph returns the following theory:

- $\mathbf{class}\,(A, quadrilateral) \leftarrow \mathbf{sides}\,(A, 4)$
- $\mathbf{class}\,(A, non\_quadrilateral) \leftarrow \mathbf{sides}\,(A, 3)$
- $\mathbf{class}\,(pentagon, non\_quadrilateral) \leftarrow \mathbf{sides}\,(A, 5)$
- $\mathbf{class}\,(hexagon, non\_quadrilateral) \leftarrow \mathbf{sides}\,(A, 6)$

Now let's compare the above theory to our method. In the next steps we show how the method proposed works for this problem.

1. **Creation of subsets of values.** This problem has a small number of discrete numeric values so we work on it as a categoric value which indicates the number of sides of each figure. We obtain two subsets of categories: $A = \{4\}$ and $B = \{3, 5, 6\}$.
2. **Creation of multivalues clauses.** For each subset of categoric values our method creates multivalue clauses, the corresponding pair is:
   $\mathbf{sidesA}\,(F) \leftarrow \mathbf{sides}\,(F, L)\,, \mathbf{member}\,(L, [4])$
   $\mathbf{sidesB}\,(F) \leftarrow \mathbf{sides}\,(F, L)\,, \mathbf{member}\,(L, [3, 5, 6])$
3. **Modification of the background knowledge.** In this step the new clauses are added to the background knowledge.
4. **Usage.** Finally this new background knowledge is used by adding literals *sidesA* and *sidesB* to the possibilities of constructing hypotheses by the ILP algorithms, in this example the final theory (with Aleph) is:
   $\mathbf{class}\,(A, quadrilateral) \leftarrow \mathbf{sides}\,(A, L)\,, \mathbf{member}\,(A, [4])\,.$
   $\mathbf{class}\,(A, no\_quadrilateral) \leftarrow \mathbf{sides}\,(A, L)\,, \mathbf{member}\,(A, [3, 5, 6])\,.$

4 Experiments

The databases analyzed were obtained from the UCI Repository [2], and each one was divided in ten folds to perform a cross validation analysis. In order to compare our method with univalue learning, we analyzed each problem with the following ILP systems:

- **Aleph.** This is a ILP system created by Ashwin Srinivasan [11], this system implements the Progol algorithm ([7]).
- **multivalues FOIL.** Adapted FOIL [9] with our method.

– **multivalues Aleph.** Adapted Aleph with our method.

For each problem, we analyzed: the number of rules for each theory, percentage of the covered examples, and the time of execution. In the last analysis we compare the results with Aleph and multivalue Aleph. All examples were performed on a modern multicore PC machine. In the next subsections we present the experiments that we performed.

4.1 Student Loan and Japanese Credit

For the Student Loan Problem the goal is to create a logic program which indicates if a student must pay a loan. The goal predicate is **no_payment_due** (**Student**) ←. The declared predicates in the background knowledge provide information about: gender, longest absense from school, school, employment, etc. This problem has two numeric attributes and two categoric attributes.

The second database is *Japanese Credit Screening Data Set* and contains information about people who were granted a bank credit. This database was generated from japanese enterprises which granted bank credits. The goal relation is **creditscreen** (**Person**) ←. In order to grant a credit is taken into account the following information: employment, type of good purchased for credit, gender, marital status, age, problematic region, etc. This database has five numeric attributes and one categoric attribute.

Results. The table 1 shows the results for these database, and we can see that the number of rules is decreased with our method. Regarding the accuracy, we can note that with the multivalues clauses the percentage of covered examples increases, hence it's neccesary to perform an study in order to can affirm that with our method the accuracy is improved.

Table 1. Student loan and Japanese credit results.

Student Loan	Avg	**Student Loan**	Avg
Number of rules	===	Percentage of covered examples	===
Aleph without multivalue clauses	9 rules	Aleph without multivalue clauses	71%
Aleph with multivalue clauses	6.2 rules	Aleph with multivalue clauses	89%
FOIL with multivalue clauses	5 rules	FOIL with multivalue clauses	87%
Time of performance	===		
Aleph without multivalue clauses	1.658 sec.		
Aleph with multivalue clauses	2.8868 sec.		
Japanese Credit	Avg	**Japanese Credit**	Avg
Number of rules	===	Percentage of covered examples	===
Aleph without multivalue clauses	17.8 rules	Aleph without multivalue clauses	79.87%
Aleph with multivalue clauses	14.7 rules	Aleph with multivalue clauses	82.23%
FOIL with multivalue clauses	10.1 rules	FOIL with multivalue clauses	96.66%
Time of performance	===		
Aleph without multivalue clauses	1.39 sec.		
Aleph with multivalue clauses	1.97 sec.		

5 Conclusions and future work

With the creation of the subsets of values (categoric or numeric) the ILP algorithms identify significant information which is contained in the examples and background knowledge. This information is the split point d, which, as seen from the results, we can use to reduce the number of rules for the theories induced. In order to affirm that the accuracy is improved with our method, we need to analyze more ILP databases with multivalue clauses.

Furthermore, not all values can be processed with our method. Only those attributes on literals whose values are applicable for all objects. For instance, an attribute like the name or surname is not applicable for all persons, hence it can not be used by our method. Instead a value like the age is applicable for all persons, therefore we can use it with our method. But this analisys is left to the user who selects which values are selected.

Regarding future work, there are many avenues to explore. We can perform some improvements in the implemented algorithm in order to improve the results obtained to create new clauses. We can go further by, on one hand, obtaining more than one split point thus dividing in more than two subsets the values. On the other hand, in addition to the multivalue clauses, we could also design a method which creates multivariate clauses. It might be productive to look into other decision tree inducers techniques to extract significative information.

Finally it would be interesting to implement all ILP algorithms in one system like weka [10] to allow the comparision on them to be more simple.

References

1. M. M. Bongard, *Pattern Recognition*, Hayden Book Co., Spartan Books., Rochelle Park, N.J., 1970.
2. A. Frank and A. Asuncion, *UCI machine learning repository*, 2010. http://archive.ics.uci.edu/ml.
3. H. Kim and W.-Y. Loh, *Classification trees with unbiased multiway splits*, Journal of the American Statistical Association, (2001), pp. 589–604.
4. N. Lavrac and S. Dzeroski, *Inductive Logic Programming: Techniques and Applications*, Ellis Horwood, New York, 1994.
5. W.-Y. Loh and Y.-S. Shih, *Split selection methods for classification trees*, Statistica Sinica, (1997), pp. 815–840.
6. T. M. Mitchell, *Machine Learning*, McGraw-Hill, 1997.
7. S. Muggleton, *Inverse entailment and progol*, New Generation Comput., (1995), pp. 245–286.
8. S. Muggleton and L. D. Raedt, *Inductive logic programming: Theory and methods*, Journal of Logic Programming, (1994), pp. 629–684.
9. R. Quinlan, *Learning logical definitions from relations*, Machine Learning, (1990), pp. 239–266.
10. E. F. Remco R. Bouckaert, *WEKA Manual for Version 3-6-0*, University of Waikato, Hamilton, New Zealand, 2008.
11. A. Srinivasan, *The Aleph Manual*, 2004. http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/.