Selective Propositionalization for Relational Learning

Érick Alphonse, Céline Rouveirol

Inference and Learning Group,
LRI - UMR 8623 CNRS
Bâtiment 490, Université Paris-Sud
91405 - Orsay Cedex (France)
{alphonse,celine}@lri.fr [*]

Abstract. A number of Inductive Logic Programming (ILP) systems have addressed the problem of learning First Order Logic (FOL) discriminant definitions by first reformulating the problem expressed in a FOL framework into a attribute-value problem and then applying efficient algebraic learning techniques. The complexity of such *propositionalization* methods is now in the size of the reformulated problem which can be exponential. We propose a method that selectively propositionalizes the FOL training set by interleaving boolean reformulation and algebraic resolution. It avoids, as much as possible, the generation of redundant boolean examples, and still ensures that explicit correct and complete definitions are learned.

1 Introduction

Learning relational concepts from examples stored in a multi-relational database has been identified as a challenge for Inductive Logic Programming (ILP) techniques by both KDD and ILP communities [3]. However, it is a well-known fact that the counterpart of learning in restrictions of FOL, even relational ones, is the dramatic complexity of the coverage test between a hypothesis and an example.

Here, we address discriminant concept learning in Datalog target concept languages[1]. In such languages, the exponential complexity of subsumption (classically θ-subsumption [9]) is inherent to the non determinacy of the computation of "matching" substitutions between a hypothesis and an example. This can happen when the Entity-Relationship schema of the target relational database contains 1-n or n-n associations.

While a number of specific biases have been developed directly in an FOL framework to control this indeterminacy by restricting the target concept language (see for instance, *ij*-determination [8]), a family of ILP methods (among others, LINUS [6], STILL [13], REPART [15], SP [5]) have addressed this problem by *propositionalizing* the ILP problem, i.e., by *reformulating* the ILP learning problem into an attribute value or even boolean one, which can then be

[*] This work has been partially supported by ESPRIT through LTR ILP^2 n. 20237.

[1] Horn clause languages without function symbols other than constants.

handled by learning techniques dedicated to this simpler formalism. Once the representation change has been performed, robust and efficient algorithms can be successfully applied, provided that the discriminant features of the FOL learning problem are preserved by propositionalization.

Propositionalizations in those systems all adopt the same schema: given a *pattern P*, FOL examples are reformulated into their (potentially multiple) matchings with P, yielding a tabular representation. Of course, the subsumption test being of exponential complexity in an unrestricted Datalog language, the size of the reformulated problem can be exponential [1] as well as highly redundant, and cannot be directly addressed as such for complex relational learning problems.

This paper presents a *selective propositionalization* that controls the size of the reformulated problem: instead of generating the whole boolean reformulation of the FOL problem before resolution, this method interleaves boolean reformulation and algebraic resolution. Information gathered during algebraic resolution is used to constrain the generation of the reformulated boolean problem to the boolean vectors that are useful for next refinement step(s) only. In doing so, it avoids, as much as possible, the generation of redundant boolean examples, enables partial storing of positive boolean instances only and still ensures that correct and complete definitions are learned.

2 Background

After [12, 15, 11], a learning problem can be decomposed into two subproblems, a *relational* (or structural) one and a *functional* one. To illustrate this decomposition, consider learning from examples stored in a multi-relational database. Here, learning from litterals representing the multiple foreign key links [14] among tuples of different relations is a structural learning problem, whereas learning on the other (mono-valued) attributes of those relations is a functional one.

Consequently, this paper focuses on relational learning, which is typically a non-determinate learning problem, within Datalog target concept language without constants and without restriction on the depth or level of "indeterminacy" of existential variables [8]. In such a language, the propositionalization process is described as follow:

Definition 1. *The pattern P is built from a seed positive example e, as the maximal generalization of e plus equality constraints between pairs of variables in the pattern which are satisfied by e (see example below). Each training example is then translated into a set of boolean vectors. For each matching σ_i of P variables onto constants of e, the attributes of the boolean vector associated to a FOL example indicate which constraints of the pattern (presence/absence of a literal, links between variables of the pattern) are satisfied by σ_i.*

Thus, the FOL search space is shifted to a boolean lattice ordered by boolean inclusion, denoted $\prec_b$. The search space of the reformulated problem is then that of concepts more general than or equal to the seed example: a partial mapping of P literals to FOL example literals yields a more general boolean vector than P, whereas a complete mapping yields a boolean vector equivalent to P. For instance, if E, CE are a positive and a negative example of the target concept and E' is the seed example, the obtained tabular representation is:

$E : c(a) \leftarrow p(a,b), p(b,c), q(c), q(a).$ $E' : c(a) \leftarrow p(a,b), q(b), q(a), r(c).$
$CE : c(a) \leftarrow p(a,b), p(b,c), q(b), q(c).$

P	$c(U)$	$p(V,W)$	$q(X)$	$q(Y)$	$r(Z)$	$U=V$	$U=Y$	$V=Y$	$W=X$
$\theta_{E,1}$	1	1	1	1	0	1	0	0	0
$\theta_{E,2}$	1	1	1	1	0	1	1	1	0
$\theta_{E,i}$	1	1	1	1	0	0	0	0	1
$\theta_{CE,1}$	1	1	1	1	0	1	0	0	1
$\theta_{CE,2}$	1	1	1	1	0	1	0	0	1
$\theta_{CE,j}$	1	1	1	1	0	0	0	1	1

Fig. 1. Excerpt of boolean representation of a FOL problem

Moreover, as pointed out in [15], the learning task is no longer to induce a Datalog concept consistent with **all** boolean vectors, but what is referred to as the multi-part problem[2]:

Definition 2. *(after [15]) The multi-part problem consists of finding a description that covers, for all FOL positive examples, at least one of their associated boolean vectors (completeness) and none of the boolean vectors associated to any FOL negative example (correctness).*

3 State of the Art

Although the reformulated problem can be delegated to efficient and robust algorithms (SP [5] with C4.5 [10], REPART with CHARADE [4]), the space complexity becomes intractable, as does the time complexity. As pointed out by [1], boolean learners working on the reformulated problem must deal with data of exponential size wrt the FOL problem. Indeed, following definition 1, each positive or negative FOL example is described by a set of boolean vectors (termed in the remainder of the paper *positive* and *negative* boolean vectors respectively), the cardinality of which is equal to its multiple matchings with the propositionalization pattern.
As far as we know, two learning systems have addressed this problem: STILL [13] and REPART [15]. For the former, propositionalization is performed through a stochastic selection of η example matchings (η is a system parameter) with the pattern, which allows for bounding the size of the reformulated problem, yielding a polynomial generalization process. To offset the imperfection of such generalizations as "standalone" classifiers, STILL learns a committee of them, that classifies unseen examples in a nearest neighbor-like way.

For the latter, restriction of the reformulated problem is performed through the choice of a relevant propositionalization pattern. The user/expert must provide a pattern as a (strong) bias which allows him to drastically decrease the matching space. The validity of the method relies on the assumption that the selected pattern preserves the discrimination information sought for. As the FOL learning problem is propositionalized before resolution, this system nevertheless has to cope with the size of the reformulated data.

We propose one alternative method in order to both reduce the data size of the reformulated problem and avoid data storing as much as possible.

[2] As noted by the authors, this learning problem is closely related to what Dietterich termed the multi-instance problem [2].

4 Selective Propositionalization

```
build $P$ from a seed positive example (see def. 1)
initialize $G$ as the universal element of the search space
**For each** $ce \in CE$ **do**
    **Repeat**
        Select $g \in G$
(1)     Compute a boolean vector $b$ from $ce$ (* $P \prec_b b \preceq_b g$ *)
        **If** $b$ is equal to $P$ **Then**
            no structural discrimination is possible
        **Else**
            Specialize $G$ to discriminate $b$ (* algebraic resolution *)
(2)         Evaluate each element of $G$ wrt positive example coverage
            Update $G$ (* beam search strategy *)
        **Endif**
    **Until** all elements of $G$ are correct
**Endfor**
**return**(G)
```

Fig. 2. Computation of n elements of G

The overall structure of our algorithm is quite classical. It is based on the Candidate Elimination Algorithm [7] and implements a *covering* method for learning disjunctive concepts. The algorithm computes a set of maximally general and correct solutions by a top-down search in a boolean search space. The two original ideas of the algorithm stem from the fact that the boolean examples handled by the algorithm are not generated before learning proceeds, but during resolution. Therefore, as opposed to classical propositionalization methods (see sec. 1) which compute as many boolean examples as the number of matchings between the pattern and FOL examples, this algorithm constructively exploits: **i)** information gathered during resolution to only generate boolean examples that are useful for (in)validating the current specialization step; **ii)** the partial ordering on the instance space in order to generate useful examples, that is, the "close to" most specific ones.

4.1 Exploiting Current Resolution Information

In classical propositionalization techniques, all (positive and negative) FOL examples are reformulated into their multiple matchings with a given pattern P. In contrast, our method only looks for boolean vectors that may invalidate the current hypothesis g and therefore yields a specialization of g. At each search step, it therefore attempts to build a negative boolean vector more specific than g, i.e., which contains at least all boolean attributes of g.

For a Datalog language, several tentative matchings may be necessary (in the worst case, an exponential number) in order to build a matching substitution σ. However, the benefit of selective propositionalization wrt a classical propositionalization, in terms of the matching space explored, is theoretically (and empirically, as shown in our first experiments, sec. 5) substantial: the space of matching substitutions to be searched is induced by literals belonging to g as opposed to P, that is, by relevant predicates wrt the current discriminant task.

4.2 Partial Ordering on the Instance Space

The size of the reformulated boolean problem is upper-bounded by the number of matching substitutions between the pattern and the FOL examples (positive and negative), but a large fraction of these boolean vectors are redundant (see in fig. 1 $\theta_{E,1}$ wrt $\theta_{E,2}$ and $\theta_{CE,1}$ wrt $\theta_{CE,2}$) in that they do not directly take part in the process of building a correct and complete discriminant solution. Such redundant data occur when propositionalizing both negative and positive examples (respectively, steps 1 and 2 of the algorithm).
Indeed, as far as negative examples are concerned, and after [12], there is a partial ordering (*nearest-miss*) of the negative instance space and it has been shown that only maximally specific negative examples wrt this partial ordering are sufficient for solving the discriminant learning problem. For positive examples, if we refer to definition 2, a FOL example is covered in the boolean search space if at least one of its corresponding boolean vectors is covered. Therefore, only the most specific ones wrt boolean inclusion are sufficient for our learning problem. After computing the matching substitution σ as stated above, the propositionalization will be as efficient as the extracted boolean vector is specific. We therefore complete σ by deterministically[3] matching literals of $\mathcal{P}$ with the FOL example. In doig so, we therefore cannot ensure that the extracted boolean vector is a most specific one, which would require an exponential complexity, but is only a "close to" most specific one.

5 Experimentations

The efficiency of our approach is evaluated by both percentages of boolean vectors computed and that of the matching space explored by our approach wrt classical propositionalization methods as presented in section 2. The former reflects the amount of non-redundant boolean vectors empirically computed by the selective propositionalization, that is the complexity of the learning problem resolution. As for the latter, it reflects the complexity of the selective propositionalization itself.

As a learning database, we have used a hard artificial problem derived from Michalski's trains involving an intractable number of data (about one hundred million) for classical propositionalization methods. As a result, we have obtained an amount of 0,0018% boolean vectors computed (with a standard deviation of 0,0017%), by exploring 1,62% of the whole matching space (with a standard deviation of 1,69%). As a corollary, learning methods implementing selective propositionalization are empirically about 62 times faster than classical propositionalization methods.

6 Conclusion

We have proposed an original propositionalization method which benefits from the advantages of both generate and test methods, using a "more general than or equal to" partial ordering, and from a sound and efficient algebraic specialization

[3] therefore, with polynomial time complexity.

operator. The selective propositionalization method has been validated on an artificial, yet complex relational problem, involving a huge matching space and seems well-suited for handling highly indeterminate FOL learning problem. The generation of a large amount of redundant data is avoided. On the other hand, the Version Space approach allows for storing just a few boolean vectors computed from positive FOL examples only.

Finally, this selective propositionalization technique can be adapted to any subsumption relation in the original FOL search space, and it can be combined with additional biases that can further improve the overall efficiency. For instance, user bias [15] can be incorporated in the pattern definition to further decrease the size of the matching space.

References

1. L. De Raedt. Attribute-value learning versus inductive logic programming : The missing link. In D. Page, editor, *Proc. of the 8th International Workshop on ILP*, pages 1–8. Springer Verlag, 1998.
2. T. Dietterich, R. Lathrop, and T. Lozano-Perez. Solving the multi-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1996.
3. U. Fayyad. Knowledge discovery in databases : An overview. In N. Lavrač and S. Džeroski, editors, *Proc. of the 7th International Workshop on ILP*, pages 1–16. Springer Verlag, 1997.
4. J.-G. Ganascia. A rule system learning system. In R. Bajcsy, editor, *Proc. of International Joint Conference of Artificial Intelligence*, pages 432–438. Morgan Kaufmann, 1993.
5. S. Kramer, B. Pfahringer, and C. Helma. Stochastic propositionalization of non-determinate background knowledge. In D. Page, editor, *Proc. of the 8th International Workshop on ILP*, pages 80–94. Springer Verlag, 1998.
6. N. Lavrač and S. Džeroski. *Inductive Logic Programming : techniques and Applications.* Ellis Horwood, 1994.
7. T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
8. S. Muggleton and C. Feng. Efficient induction in logic programs. In S. Muggleton, editor, *International Workshop on ILP*, pages 281–298. Academic Press, 1992.
9. G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5. Edinburgh University Press, 1970.
10. J. R. Quinlan. *C4. 5: Programs for Machine Learning.* MK, San Mateo, CA, 1993.
11. M. Sebag. Resource bounded induction and deduction in FOL. In *Proc. on Multi Strategy Learning*, 1998.
12. M. Sebag and C. Rouveirol. Induction of maximally general clauses consistent with integrity constraints. In S. Wrobel, editor, *Proc. of the 4th International Workshop on ILP*, pages 195–216, 1994.
13. M. Sebag and C. Rouveirol. Tractable induction and classification in first order logic via stochastic matching. In *15th Int. Join Conf. on Artificial Intelligence (IJCAI'97), Nagoya, Japan*, pages 888–893. Morgan Kaufmann, 1997.
14. S. Wrobel. An algorithm for multi-relational discovery of subgroups. In *Proc. of PKDD*, pages 78–87. Springer Verlag, 1997.
15. J.-D. Zucker and J.-G. Ganascia. Learning strcutural indeterminate clauses. In D. Page, editor, *Proc. of the 8th International Workshop on ILP*, pages 235–244. Springer Verlag, 1998.