Automatic Induction of Domain-related Information: Learning Descriptors Type Domains

Stefano Ferilli and **Floriana Esposito** and **Teresa M. A. Basile** and **Nicola Di Mauro** [1]

Abstract. Learning in complex contexts often requires pure induction to be supported by various kinds of meta-information. Providing such information is a critical, difficult and error-prone activity. This paper proposes an algorithm to automatically identify types from observations, and studies its performance and robustness.

1 Introduction

Learning in complex contexts often requires pure induction to be supported by a variety of techniques that can cope with different aspects of the learning task. In the current practice, it is in charge of the human expert to specify all the 'added-value' information needed by such techniques for being applicable. Providing it is a very difficult task, that requires a deep knowledge of the application domain, and is in any case an error-prone activity, since omissions and errors may take place. These considerations would make it desirable to develop procedures that can automatically generate such information starting from the same observations that are input to the learning process.

The next section presents a technique to automatically infer the description language. Then, Section 3 tests the proposed approach, even in the case of incomplete input information.

2 Inducing Descriptors Type Domains

An interesting issues is the identification of what *types* are used in the description language and their related *domains*. When using First Order Logic as a representation language, unary predicates represent possible values for properties. Hence, discovering the type domains for the properties in the language can be cast as the search for groups of unary predicates that semantically refer to the same attribute. Various learning systems in the literature (e.g., [13], [3], [12], [11], [4], [6] and [7]) can exploit meta-information of this kind to prune the search space and obtain more efficiency. Some attempts to automatically infer such information have already been carried out [9]. However, theories learned by many systems are constant-free, and allow only variables as terms.

Example 1. *Given a set of examples and descriptions in which the set of unary predicates in the description language is:* {high, large, white, low, small, blue, red, yellow}*, the system should understand that the values they represent define three domains referred to different types:* {white, blue, red, yellow} *(*color*),* {small, large} *(*size*), and* {high, low} *(*height*).*

In the following, we will assume that: all possible values of non-numeric properties are expressed by means of unary predicates; no unary predicate expresses a value that belongs to many types; no property is expressed by just the presence or absence of a corresponding predicate; all properties are applicable to any object that occurs in the descriptions.

The whole strategy is summarized in Algorithm 1. Since different values for the same attribute are mutually exclusive, a preliminary step cosists in testing for occurrence in the observations all the possible pairs of unary predicates .

Then, since *any* value in a given domain cannot co-occur in one object with *any* other value in the same domain, the problem becomes identifying groups of unary predicates whose elements are *couplewise* mutually exclusive. In particular, we are interested in maximal sets only. By mapping the problem onto a corresponding one in the graph context, we build an undirected graph G_e whose nodes are unary predicates in the description language, and where an edge connects two nodes if and only if they are mutually exclusive. In such a setting, the maximal sets we are looking for correspond to all the *maximal* cliques (i.e., cliques that cannot be further extended) in G_e.

Now, there can be 'spurious' groups of predicates with couplewise mutually exclusive elements even if they do not refer to a same attribute (e.g., a line is never too tall), but, in the end, the solution will include only groups that have no element in common. Again, this problem can be solved in the graph context by building an undirected graph G_d in which nodes are groups identified in the previous step as cliques of graph G_e, and an edge connects two nodes if and only if they are disjoint sets. Now, the solution will be made up by maximal groups of disjoint subsets, each of which corresponds to a maximal clique in G_d.

The clique in G_d will probably not be unique, in which case one must have a clue for choosing the right one. The intuition, in this case, is that any 'wrong' clique, in order to fulfill the mutual disjunction requirement, will have overall a number of values that is less than that of the correct solution, since the correct solution should be the only one containing all the possible values for each property (represented by a group), and hence the union of predicates in all of its components should be equal to the whole set of values for all possible attributes. In other words, the solution is actually a *partition* of the set of unary predicates.

3 Experimental Results

The proposed method was implemented in SICStus Prolog, and tested on various domains, covering all the possible cases of available observations and target types to be recognized.

The Scientific Papers dataset [5] is based on a representation language made up of predicates with various arities, of which unary predicates represent values belonging to many different domains

[1] Department of Computer Science, University of Bari, Italy email: {ferilli,esposito,basile,nicodimauro}@di.uniba.it

Algorithm 1 Identification of type domains

```
Require: Description language $L$
  $U := \{p \in L \mid p \text{ unary}\}$
  $E := \{(p,q) \in U \times U \mid \nexists X : p(X) \wedge q(X)\}$
  $G_e := (U,E)$
  $S := \{C \subseteq U \mid C \text{ clique in } G_e\}$
  $F := \{(p,q) \in S \times S \mid p \cap q = \emptyset\}$
  $G_d := (S,F)$
  $T := \{C \subseteq S \mid C \text{ clique in } G_d\}$
  return $argmax_{t \in T}(|\bigcup_{t_i \in t} t_i|)$
```

(*general* case). It includes 112 scientific papers, belonging to 4 different classes. The procedure found all the correct types: Width (7 values), Content (6 values), Vertical position (3 values), Horizontal position (3 values), Height (10 values).

The Family Relationships dataset [2] refers to a description language made up of predicates with various arities that describes a family tree, all whose unary predicates ({female, male}) belong to the same type (Sex), successfully retrieved by the algorithm).

The Tic Tac Toe dataset [1] description language is made up of unary predicates only, representing values of different types. It contains all possible instances of final game configurations, each reporting the status (blank, X, or O) of all 9 positions. The system correctly recognized these 9 types with the corresponding 3-valued domain.

Lastly, the Congressional Votes [8] dataset describes 435 Congressmen as being democrats or republicans according to their votes on 16 issues by means of 32 predicates, each representing the favorable or opposite vote on one of the 16 issues. It is particularly interesting because a certain amount of noise is present in the descriptions, in the form of unknown (omitted) votes. Nevertheless, the algorithm is able to correctly infer all the 16 types, each with its 2 descriptors (corresponding to the yes/no options).

To evaluate the effectiveness of the proposed algorithm in presence of a small amount of information, we focused on the Scientific Papers dataset, because it is the most complex among those considered. Various experiments were run, in which noise was progressively introduced in the dataset descriptions. For each fixed amount of noise to be introduced, 10 random corruptions of the dataset were performed, on which running the proposed algorithm. Then, the learned types were checked and categorized in one of the following categories (listed by decreasing desirability): *correct*, *incomplete* (i.e., missing some types or some values in some type domains, but without mixing values belonging to different types), *impossible* (when the algorithm autonomously recognized that the available information was too loose for getting to a correct solution), and *wrong* (when at least one of the identified types contained in its domain values actually belonging to different types).

A first experiment in this direction aimed at assessing how sensitive the algorithm is to the amount of observations provided to it. In this case, the dataset corruption consisted in progressively eliminating observations (examples) from it (remember that the initial size was 112). The amount of corruption ranged between 10% and 90% of the entire dataset. It is interesting to note that the algorithm never generated undesirable (i.e., impossible or wrong) type domains. Actually, up to 50% of the dataset it always gave correct and complete answers. After that threshold, completeness started decreasing, but even when 90% of the observations was dropped (i.e., only 12 paper descriptions were available) in 2 cases it succeeded in finding the correct and complete types. This should allow one to state that the system is effective also when provided with very few observations.

Then, the next question was how much noise could be present in the available knowledge in order for the system not to be misleaded in its task. For this purpose, all the available observations were corrupted by eliminating from them a progressively larger amount of information, ranging from 10% to 60%. The experimental outcomes suggest that the algorithm is more sensitive to partial descriptions than it was to a small number of observations. Indeed, in this case complete and correct types are induced only up to 20% of corruption, while accepting also incomplete types is ok up to 30%. Anyway, also after that threshold, the sum of desirable cases (i.e., correct and incomplete ones) far outperforms the number of undesirable ones. Only when 60% of each description in the dataset is dropped the number of wrong inductions becomes predominant, but interestingly it does not exceed half of the trials. This behaviour can be explained because the proposed algorithm heavily relies on co-occurrence of values for inducing the type domains. Thus, eliminating whole observations, but leaving complete the remaining ones, potentially still preserves many co-occurrences. On the contrary, dropping portions of each observation is likely to introduce false (supposed) incompatibilities among values that actually belong to different types.

4 Conclusions

Many learning systems in the literature exploit knowledge about the types used in the description language and their related domains to improve their performance. This paper proposed an algorithm to automatically identify this kind of meta-information from observations. Experimental evaluation in domains with different characteristics reveals encouraging performance and its robustness.

REFERENCES

[1] D. W. Aha. Incremental constructive induction: An instance-based approach. *Proceedings of the 8th International Workshop on Machine Learning*, pages 117–121. Morgan Kaufmann, 1991.

[2] H. Blockeel and L. De Raedt. Inductive database design. In *Foundations of Intelligent Systems*, volume 1079 of *Lecture Notes on Artificial Intelligence*, pages 376–385. Springer, 1996.

[3] R.M. Cameron-Jones and J.R. Quinlan. Efficient top-down induction of logic programs. *SIGART bulletin*, 5(1):33–42, 1994.

[4] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26(2):99–146, 1997.

[5] S. Ferilli, N. Di Mauro, T.M.A. Basile and F. Esposito. Incremental Induction of Rules for Document Image Understanding. In *AI*IA 2003: Advances in Artificial Intelligence*, volume 2829 of *Lecture Notes on Artificial Intelligence*, pages 176–188. Springer, 2003.

[6] P.A. Flach and N. Lachiche. Cooking up integrity constraints with primus. Preliminary Report CSTR-97-009, University of Bristol - Department of Computer Science, December 1997.

[7] P.A. Flach and N. Lachiche. Confirmation-guided discovery of first-order rules with Tertius. *Machine Learning*, 42(1/2):61–95, 2001.

[8] A. Kakas and F. Riguzzi. Abductive concept learning. *New Generation Computing*, 1999.

[9] E. McCreath and A. Sharma. Extraction of meta-knowledge to restrict the hypothesis space for ilp systems. In *Proceedings of the 8th Australian Joint Conference on Artificial Intelligence*, pages 78–82. World Scientific, 1995.

[10] R. S. Michalski. Inferential theory of learning. developing foundations for multistrategy learning. In R. S. Michalski and G. Tecuci, editors, *Machine Learning. A Multistrategy Approach*, volume IV, pages 3–61. Morgan Kaufmann, San Mateo, CA, U.S.A., 1994.

[11] K. Morik. Balanced cooperative modeling. *Machine Learning*, 11:217–235, 1993.

[12] S. Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3/4):245–286, 1995.

[13] E. Shapiro. Inductive inference of theories from facts. Technical Report 192, Computer Science Department, Yale University, 1981.