Representation shifts in learning

Céline Rouveirol *
celine@lri.fr

July 6, 1997

1 Introduction

Learning systems are very sensitive to the choice of the target concept language $\mathcal{L}_c$,, as the pre-requisite for learning to succeed is that the target concept definition belongs to the target concept language. Any restriction on the target concept language is known as *language bias*.

As a consequence, some works in the ML community, and more particularly in ILP have studied representation shifts for learning. There are several such representation changes. On the one hand, *predicate invention* or *constructive induction* is triggered by a learning failure, which demonstrates that the language bias is too strong for learning to succeed. For instance, there is no conjunctive formula of the target concept language that discriminates positive from negative examples. Partially automated techniques have been developed to locally extend $\mathcal{L}_c$ in this case. Another situation for predicate invention is when learning recursive programs : a recursive predicate has to be invented such that there exists a fintie axiomatisation of the theory, which has an infinite Herbrand Model. There are on the other hand, automatic representation changes, which do not require any user participation, the aim of which is to enhance the learning *search performance*.

2 Flattening

Flattening [4] is a fully automated representation change, dedicated to extend the search space of learning operators that do not handle function symbols.

Technically, flattening transforms formulas with function symbols (with structured terms) into logically equivalent formulas without function symbols. Its basic idea is to introduce for every function symbol of arity n a new predicate of arity $n+1$, where the first n arguments are the same as for the function and the last argument is the *result* of the function. Just to give a small example. Let E_x be the clause to flatten:

*Laboratoire de Recherche en Informatique, Bât. 490, Université Paris-Sud, 91405 - Orsay Cedex

E_x : $\quad member(blue, [blue]) \leftarrow .$

the corresponding flattened clause is:

E''_x: $\quad member(X, Z) \leftarrow consp(X, Y, Z) \wedge bluep(X) \wedge nilp(Y).$

with the three following clauses:

$$\begin{array}{ll} NIL: & nilp(nil) \leftarrow . \\ CONS: & consp(X, Y, cons(X, Y)) \leftarrow . \\ BLUE: & bluep(blue) \leftarrow . \end{array}$$

The flattened clause is logically equivalent to the initial structured clause, given the program of clauses NIL, $CONS$ and $BLUE$. The representation change is therefore reversible : by resolving E''_x (or any generalisation of it) with NIL, $CONS$ and $BLUE$, one obtains a clause with the initial function symbols again.

Learning operators, such as the dropping literal operator that were not able to generalise on structured terms may now do so. For instance, by dropping the literals $bluep(X)$ and $nilp(Y)$ in E''_x, and by unflattening the resulting clause, one gets the clause $G : member(X, [X|Y]) \leftarrow$ which was not reachable by dropping literal on the initial structured clause.

Internally working on flattened clauses thus extends the search space for the dropping literal operator. Moreover, as far as reasoning is concerned, flattening introduces existential variables for each subterm (here X and Y) of terms in the initial clause. Having an explicit access to subterms in the clauses allows reasoning (deduction, for for instance) on them. However, this “loss” of term structure may dramatically increase the complxity of the learning process. In particular, it proves to be harmful when applying, for instance, lgg [3], which has high complexity on input clauses with existential variables.

3 A FOL-to-attribute value representation shift

This second representation change aims at improving the performance of the search for discriminant hypotheses, by allowing robust attribute value techniques to be applied to a FOL discrimination problem. Let us consider the problem of building a maximally discriminant First Order Logic (FOL) hypothesis for a positive example $E+$ from a negative one $E-$, both expressed as ground clauses. A maximally general clause $\mathcal{C}$ is built from the positive example by turning every term of $E+$ into a different variable). This maximally general clause is then used as a “skeleton” for biaising the shift to an attribute value representation. The positive example is represented by a single substitution θ such that $\mathcal{C}.\theta = E+$ and the negative example is represented by all the substitutions σ_i such that $\mathcal{C}.\sigma_i \subseteq E-$. This transforms the initial training set $\{E+, E-\}$ into a set with a single positive attribute example θ, and a possibly exponential number of negative attribute value examples σ_i.

For coping with the potentially exponential complexity of discriminating two FOL examples, a declarative bias of any kind (syntactic, or stochastic as in [5]) can be applied in order to build only a subset of relevant σ_i. All the attribute value methods for building discriminant hypotheses can then be applied to this new example set, in particular those able to handle numbers. A resulting hypothesis θ_h at the attribute value level can then transformed into the FOL hypothesis $\mathcal{G}.\theta_h$ such that $\mathcal{G}$ is a mininal subset of $\mathcal{C}$ necessary to introduce the variables of θ_h. This repesentation change is therefore performed as a pre- and post-processing step of learning. As opposed to other representation changes from FOL to attribute value representation (see [1] or [6]), this representation change is dependent from examples, as it is guided by a "skeleton" extracted from the positive examples.

4 Learning in Databases

This last representation change aims at improving learning efficiency by focusing the search on interesting subparts of the search space. This work takes place in the framework of learning dependencies in databases. This learning task is usually weakly or even incorrectly specified, and this is why the user has to interact with the learning system in a trial-and-error loop in order to specify more and more the learning task and to get more and more adapted learning results [2].

The search space when learning dependencies, specially when the learning bias is too weak can be enormous, and may discourage the user to run a learning session. Based on works in the database community about logical formalisation of databases, we propose two different representations for hypotheses in the search space, that enables to focus search during learning.

Two granularity levels are proposed : in the large grain representation, namely *tuple-oriented representation*, each object is represented by a literal where the predicate symbol represents the database relation the object belongs to and the arguments represent the value of the object attributes. The target relation is represented by the head literal. For instance, the following clause

(1): $$\begin{aligned} train(T, eastbound) \leftarrow\ & car(B, T, 1, rectangle\_shape, D, E, F, G),\\ & car(I, T, 3, rectangle\_shape, J, K, L, M),\\ & load(O, I, triangle\_shape). \end{aligned}$$

states that trains going east have their first car of rectangular shape and their third with a load of triangular shape (the other attribute values are irrelevant to the learning problem).

The second representation, so called *domain-oriented* representation has a finer grain, as illustrated by the following clause equivalent to clause (1) in a domain oriented representation :

$$\begin{aligned} & direction(T, eastbound) \leftarrow \\ & \quad member\_train(A, T), range(A, 1), car\_shape(A, rectangle\_shape),\\ & \quad member\_train(B, T), Range(B, 3), car\_shape(B, rectangle\_shape),\\ & \quad member\_car(C, B), load\_shape(C, triangle\_shape). \end{aligned}$$

In a domain-oriented representation, each literal of a clause corresponds to an object attribute such as for instance $car\_shape(A, rectangle\_shape)$. Each object is represented by a distinct variable (A in the example). The literals are all binary. The predicate symbol is a database attribute name (*car_shape* in the example) and the first argument is the object variable. The second argument is a constant if the literal represents a restriction of the attribute to a value *rectangle_shape* in the example). It is a variable if it represents a join (the equality of two attribute restrictions). The head literal represents the restriction of the target attribute to the target value.

If hypotheses in the search space are encoded in a tuple oriented representation, dropping a literal in a hypothesis amounts to dropping an object in the hypothesis (a car or a load in a train in the example). Dropping a literal in a domain oriented clause amounts to drop an object attribute, meaning that the value of the attribute is irrelevant wrt the discrimination task (i.e., dropping the shape of the second car in a train).

It is clear that the search space induced for a given operator is much larger when using the domain-oriented representation than the tuple-oriented one : smaller generalisation/specialisation step can be performed in the domain oriented search space. As a direct consequence of the grain size of both representations, the consistency hypothesis usually done when learning is much weaker in a domain oriented representation than in a tuple-oriented one.

The proposed default strategy when learning dependencies from databases in such a configurable framework is therefore to explore the large grain space (the tuple oriented hypotheses space), in order to select relevant objects in a hypothesis, and then, once those relevant objects have been identified, to focus the search by shifting to a domain oriented representation. By doing so, a minimal number of hypotheses will be explored in the search space.

References

[1] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Ellis Horwood, 1994.

[2] C. Nedellec and C. Rouveirol. Focusing learning in relational databases by changing configuration of haiku.

[3] G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5. Edinburgh University Press, 1970.

[4] C. Rouveirol. Flattening and saturation : Two representation changes for generalization. *Machine Learning*, 14:219– 232, 1994. Special Issue on Representation Change in Learning.

[5] M. Sebag and C. Rouveirol. Tractable induction and classification in first order logic via stochastic matching. In *Proceedings of IJCAI'97.*

[6] J.-D. Zucker and J.-G. Ganascia. Representation changes for efficient learning in structural domains. In L Saitta, editor, *Proceedings of the 13^{th} International Conference on Machine Learning.*