Integrating Abduction and Induction in the Learning from Interpretation Setting

Evelina Lamma[1] and **Paola Mello**[2] and **Fabrizio Riguzzi**[1]
[1]Dipartimento di Ingegneria, Università di Ferrara, Via Saragat 1
44100 Ferrara, Italy,
{elamma,friguzzi}@ing.unife.it
[2]DEIS,Università di Bologna, Viale Risorgimento 2
40136 Bologna, Italy
pmello@deis.unibo.it

Abstract

In this paper we describe an approach for integrating abduction and induction in the ILP setting of learning from interpretations with the aim of solving the problem of incomplete information both in the background knowledge and in the interpretations. The approach is inspired by the techniques developed in the learning from entailment setting for performing induction from an incomplete background knowledge. Similarly to those techniques, we exploit an abductive proof procedure for completing the available background knowledge and input interpretations.

The approach has been implemented in a system called AICL that is based on the ILP system ICL. Preliminary experiments have been performed on a toy domain where knowledge has been gradually removed. The experiments show that AICL has a superior accuracy for levels of incompleteness between 5% and 20%.

1 Introduction

The integration of abduction and induction has recently received a lot of attention in the field of Inductive Logic Programming (ILP) [Muggleton and De Raedt, 1994]. A number of ILP systems combine abduction and induction in various ways: LAP [Lamma *et al.*, 1999], ACL [Kakas and Riguzzi, 2000], Progol 5.0 [Muggleton and Bryant, 2000], SOLDR [Yamamoto, 2000], CF-Induction [Inoue, 2001] and HAIL [Ray *et al.*, 2003].

However, all these systems are relative to the ILP setting of learning from entailment [Muggleton and De Raedt, 1994]. To the best of our knowledge, no attempt has been performed to integrate abduction and induction in the setting of learning from interpretations [De Raedt and Džeroski, 1994].

In this paper we propose an approach for integrating abduction and induction in the latter setting. In particular, we tackle a problem similar to the one examined in [Lamma *et al.*, 1999; Kakas and Riguzzi, 2000]: the incompleteness of available knowledge.

This is an important problem because in practice the knowledge acquisition process is rarely perfect: the acquired knowledge is very often incomplete in the sense that facts and rules may be missing from it.

In [Lamma *et al.*, 1999; Kakas and Riguzzi, 2000] the authors consider a learning problem where the background knowledge may be incomplete and they exploit abduction in order to complete the available knowledge. In practice, when testing the coverage of an example by a clause, the Prolog derivation is substituted by an abductive derivation. In this way, a positive example may be covered by abducing some positive or negative facts. Similarly, the system may avoid the coverage of a negative example by abducing some positive or negative facts.

When learning from interpretations, we can face the same incompleteness problem. In this case, the incompleteness may reside either in the background knowledge, in the interpretations or in both. This may cause a good clause to uncover a positive example or to cover a negative example. To this purpose, we exploit an abductive proof procedure in the testing of the coverage of interpretations by a clause, in order to abduce the facts that are missing from either the background and/or the interpretation. The asymmetry with respect the learning of entailment setting where only the background knowledge is incomplete is due to the fact that in that setting the information regarding each single example is contained in the background knowledge together with the general knowledge that applies to all examples. In the learning from interpretation setting the specific information regarding an example is stored in the associated interpretation, while general rules are stored in the background. So in practice both approaches complete the same kind of knowledge.

We thus present the algorithm AICL (Abductive ICL) that is based on ICL [De Raedt and Van Laer, 1995] and improves its ability of learning from incomplete interpretations.

The paper is organized as follows. In section 2 we recall some preliminaries. In section 3 we briefly describe the ICL system. Section 4 presents an example that will be used to explain AICL and will be the subject of the experiments. In section 5 we illustrate the AICL system. Section 6 reports on a set of preliminaries experiments for comparing the two systems. Finally in section 7 we present our conclusions and some directions for future work.

2 Preliminaries

A clause is a formula of the form

$$h_1 \vee h_2 \vee \ldots \vee h_n \leftarrow b_1, b_2, \ldots, b_m$$

where the h_i and b_i are logical atoms. The disjunction $h_1 \vee h_2 \vee \ldots \vee h_n$ is called the *head* of the clause and the conjunction $b_1 \wedge b_2 \wedge \ldots \wedge b_m$ is the called the *body*. Let us define the two functions $head(C)$ and $body(C)$ that, given a clause C, return respectively the head and the body of C. In some cases, we will use the functions $head(C)$ and $body(C)$ to denote the set of the atoms in the head or of the literals of the body respectively. The meaning of $head(C)$ and $body(C)$ will be clear from the context.

A definite clause is a clause where $n = 1$. A fact is a definite clause with an empty body ($n = 1, m = 0$).

A term (clause) is ground if it does not contain variables. The Herbrand universe $H(P)$ of a clausal theory P is the set of all the ground terms that can be constructed with the constant and function symbols appearing in P. The Herbrand base $H_B(P)$ of a clausal theory P is the set of all the atoms constructed with the predicates appearing in P and the terms of $H(P)$. A Herbrand interpretation is a subset of $H_B(P)$. In this paper we will consider only Herbrand interpretations and in the following we will drop the word Herbrand.

Let us now discuss how to ascertain the truth of clauses in an interpretation. A clause C is true in an interpretation I if for all grounding substitutions θ of C: $body(C)\theta \subset I \rightarrow head(C)\theta \cap I \neq \emptyset$. We also say I is a model for C, or C makes the interpretation I true, or even I is a true interpretation for C. If a clause C is not true in an interpretation I, we say that C is false in interpretation I or that I is not a model for C. An interpretation I is a model for a clausal theory T if and only if it is a model for all clauses in T. We also say that T is true in I. Therefore, it is sufficient for a single clause from T to be false in I in order for T to be false in I.

As observed by [De Raedt and Dehaspe, 1997], the truth of a clause C in a finite interpretation I can be tested by running the query $?-body(C), not\ head(C)$ on a database containing I, where $head(C)$ is interpreted as a disjunction (thus $not\ head(C)$ is a conjunction of negations). If the query succeeds, C is false in I. If the query finitely fails, C is true in I.

A Herbrand model for a definite clause theory P is an interpretation where each clause of P is true. The intersection of a set of Herbrand models is also a Herbrand model. The intersection of all the Herbrand models for P is the least Herbrand model. The semantics of definite clause theories is given in terms of the least Herbrand model. We denote the least Herbrand model of a definite clause theory P as $M(P)$.

Note that if P is a definite clause theory and I is a finite interpretation, $P \cup I$ is still a definite clause theory. The truth of clause C in the interpretation $M(P \cup I)$ can be tested by running the query $?-body(C), not\ head(C)$ against the logic program $P \cup I$. If the query succeeds, C is false in $M(P \cup I)$. If the query finitely fails, C is true in $M(P \cup I)$.

3 ICL

ICL solves the following learning problem:

Given

- a set P of interpretations;
- a set N of interpretations;
- a definite clause background theory B.

Find: a clausal theory H such that

- for all $p \in P$, $M(B \cup p)$ is a true interpretation of H;
- for all $n \in N$, $M(B \cup n)$ is a false interpretation of H;

Figure 1 shows the main loop of ICL. Figure 2 show the search for a clause of ICL.

4 Running Example

In this section we introduce a running example that will be used to explain the behaviour of AICL and that will provide a dataset for comparing ICL and AICL.

Consider a two bit multiplexer: it has two input pins and four output pins. The four output pins are numbered from 0 to 3. The behaviour of the multiplexer is the following: given values for the input pins, the output pin whose number is represented by the input pins is at 1, while the other output pins may assume either 0 or 1.

The aim is to learn how to distinguish a working multiplexer configuration from a faulty one. Each multiplexer configuration is completely described by the state of the six pins. Each pin can be at 0 or at 1. In total, we have 64 examples, 32 of which are positive (configurations of a working multiplexer) and 32 of which are negative (configurations of a faulty multiplexer).

We represent a multiplexer configuration using 12 nullary predicates, obtained by renumbering the pins from 1 to 6 (pins 1 and 2 are the input pins, pins 3, 4, 5 and 6 are the output pins). For example, the multiplexer configuration described by the bit string 010110 can be described by the following interpretation:

```
pin1at0.  pin2at1.  pin3at0.
pin4at1.  pin5at1.  pin6at0.
```

This is a positive example because output pin 4 is at 1.

A correct theory for distinguishing positive from negative configurations is the following:

```
pin3at1:-pin1at0,pin2at0.
pin4at1:-pin1at0,pin2at1.
pin5at1:-pin1at1,pin2at0.
pin6at1:-pin1at1,pin2at1.
```

Incompleteness is the interpretations in this case means that an interpretation does not contain any fact for some of the pins.

5 Abductive ICL

We modify the way in which ICL tests for the truth of a clause in an interpretation. Instead of using a standard Prolog proof procedure for testing the query $body(C), not\ head(C)$, we use an abductive proof procedure.

Consider a clause of the form

$$h_1 \vee h_2 \vee \ldots \vee h_n \leftarrow b_1, b_2, \ldots, b_m$$

The query that is tested is thus:

$$b_1, b_2, \ldots, b_m, not\ h_1, not\ h_2, \ldots, not\ h_n$$

```
Learn(P, N, B)
Initialize H := ∅
repeat until best clause C not found or N is empty
    find best clause C
    if best clause C found then
        (2) add C to H
        remove from N all interpretations that are false for C
return H
```

Figure 1: ICL covering algorithm

```
FindBestClause(P, N, B)
Initialize Beam := {false ← true}
Initialize BestClause := ∅
while Beam is not empty do
    Initialize NewBeam := ∅
    for each clause C in Beam do
        for each refinement Ref of C do
            (1) if Ref is better than BestClause and Ref
                is statistically significant then BestClause := Ref
            if Ref is not to be pruned then
                add Ref to NewBeam
                if size of NewBeam > MaxBeamSize then
                    remove worst clause from NewBeam
    Beam := NewBeam
return BestClause
```

Figure 2: ICL beam search algorithm

Suppose this query is tested against $B \cup p$ where p is a positive interpretation. If the interpretation is incomplete, it may happen that the query succeeds because one of the head atoms is false in $B \cup p$ when it should in fact be true. Suppose h_i is false because p is incomplete. By using an abductive proof procedure, we may abduce facts that make h_i true so that the query fails and the clause is true in the interpretation. The abduction is performed only if the abduced atoms are consistent with the integrity constraints.

Now consider an incomplete negative interpretation n. The query may fail against $B \cup n$ because one of body literals is false, so the clause is considered erroneously true in the interpretation. Suppose that b_j is false in $B \cup n$ because of the incompleteness of n. Then it could be useful to abduce facts that make b_j true so that the query succeeds and the clause is false in the interpretation. Again, the abduction of facts for making b_j true can be performed only if the facts are consistent with the integrity constraints.

More formally, ICL is modified in two points. The first is point (1) in function FindBestClause: in order to compare the current refinement with the best clause found so far, the refinement must be tested on the positive and negative interpretations, so that the heuristic and the likelihood ratio can be computed. The new function for testing a clause is represented in Figure 3.

In Figure 3 Derivation$(Goal, P)$ implements the Prolog derivation of a goal $Goal$ from a program P. It may succeed or fail, if it succeeds it returns a substitution θ for $Goal$. AbductiveDerivation$(Goal, P, \Delta_{in})$ implements the abductive derivation defined in [Kakas and Mancarella, 1990]. It may succeed or fail, if it succeeds returns a substitution θ for $Goal$ and a set of abduced literals Δ_{out} such that $\Delta_{out} \supseteq \Delta_{in}$.

In order to explain the behaviour of TestClause, consider the following example in which we want to test the clause C:

```
pin3at1 :- pin1at0,pin2at0.
```

over the incomplete positive interpretation p

```
pin1at0.  pin2at0.
pin4at1.  pin5at1.  pin6at0.
```

In this case the background knowledge B does not contain any clauses. However, it contains some integrity constraints, that are used by the abductive proof procedure: it contains the constraints that state that a pin can not be at the same time 0 and 1. One of these constraints is for example

```
:- pin1at0,pin1at1.
```

We first find the substitutions with which $body(C)$ is true in $p \cup B$. There is only one such substitution, the empty one. Thus $\Theta = \{\emptyset\}$. $covered$ is set to true and the outer cycle is entered. $Head$ is set to `pin3at1` and $found$ to false. Then the inner cycle is entered and an abductive derivation is started for the goal `pin3at1` from the theory $p \cup B$. Remember that the theory B contains the integrity constraints.

```
TestClause(P, N, B, C)
NP := 0 \* number of positive interpretations covered (C is true in them)*\
P' := ∅ \* set of covered positive interpretations *\
for each interpretation p ∈ P
    find the set Θ of all the substitutions θ such that
        Derivation(body(C), p ∪ B) succeeds
    Δ := ∅
    covered := true
    while Θ is not empty and covered
        remove the first element θ from Θ
        Head := head(C)θ
        found := false
        while there are literals in Head and not found
            remove the first literal L in Head
            if AbductiveDerivation(L, p ∪ B, Δ) succeeds returning Δ_out then
                found := true
                Δ := Δ_out
        if found = false then
            covered := false
    if covered then
        NP := NP + 1
        P' := P' ∪ {(p, Δ)}
NN := 0 \* number of negative interpretations not covered (C is false in them) *\
N' := ∅ \* set of non covered negative intepretations *\
for each interpretation n ∈ N
    find the set E of all the couples (θ, Δ) such that
        AbductiveDerivation(body(C), n ∪ B, ∅) succeeds
        returning θ as a substitution for Body and Δ
        as the set of abduced literals
    covered := true
    while E is not empty and covered
        remove the first element (θ, Δ) from E
        Head := head(C)θ
        add the facts of Δ to n
        call Derivation(not Head), n ∪ B)
        remove the facts of Δ from n
        if the derivation succeeds then
            covered := false
    if not covered then
        NN := NN + 1
        N' := N' ∪ {(n, Δ)}
return (NP, P', NN, N')
```

Figure 3: AICL test function

The abductive proof procedure tries to abduce `pin3at1` and succeeds because it is consistent with the integrity constraint `:- pin3at0,pin3at1` since `pin3at0` is not true in $p \cup B$.

Thus $found$ is set to true and Δ to $\{$`pin3at1`$\}$. The inner cycle terminates, the variable $covered$ remains at the value true and the outer cycle is terminated as well.

The value of the value $covered$ at the end of the outer cycle indicates that the example is covered.

Let us now consider the test of the same clause C over the negative interpretation n represented by

```
pin1at0.              pin3at0.
pin4at1.   pin5at1.   pin6at0.
```

In this case, an abductive derivation is started for the goal `pin1at0,pin2at0`. The derivation succeeds returning the empty substitution and $\Delta = \{$`pin2at0`$\}$. Thus $E = \{(\emptyset, \{$`pin2at0`$\})\}$. $covered$ is set to true. Then the cycle is entered. $Head$ is set to `pin3at1`. The fact contained in Δ is added to n and a derivation for `not pin3at1` is started. The derivation succeeds, $covered$ is set to false, the facts from Δ are removed from n, the cycle is terminated and the interpretation is not covered.

The second point in which ICL is modified is (2) in function Learn. The function FindBestClause not only returns the best clause found so far but it also returns the literals abduced for each interpretation during the test of the clause. The modified function Learn, besides adding the best clause C to the current theory H in point (2), also adds to each interpretation the facts abduced during the test of the coverage of the clause on that interpretation.

AICL has been implemented in Sicstus Prolog. In order to execute the function Derivation and AbductiveDerivation on a program containing an interpretation and the background knowledge, the Sicstus Prolog module system was used: each interpretation is loaded in a different module and the clauses of the background are asserted in all the modules.

In function TestClause the addition of the facts from Δ to the current interpretation is performed by asserting the facts in the corresponding module. Simiarly, the removal of the facts is performed by using the retract predicate.

6 Experiments

ICL and AICL were applied on the multiplexer dataset, containing 32 positive interpretations and 32 negative interpretations. A ten-fold cross-validation was performed. In order to test the performances of the two systems in the case of missing data, for each fold, facts from the interpretations were randomly chosen and removed. In particular, for each fold, different percentages of facts were removed from the training set: 5%, 10%, 15%, 20%, 25% and 30%. In this way we have obtained 7 training sets for each fold: one with the complete data and the other six with increasing missing information, from 5% to 30%. ICL and AICL were trained on the various training sets, the learned theories were tested on the testing set (from which no information was removed) and the accuracy was computed. The testing was performed by employing a Prolog derivation, i.e., abductive derivation was not used in testing. The accuracy is given by the number of covered positive examples plus the number of non covered negative example over the total number of examples.

When learning with ICL, the background knowledge was empty. When learning with AICL the background knowledge contained an abductive theory (T, A, IC) where T is empty, A contains all the 12 predicates used for describing the configurations and IC contains integrity constraints that state that a pin can not be at the same time 0 and 1.

The learning parameters for ICL were all left to their default values except the significance level which was set to 0, meaning that no significance test was performed. The same values have been used for AICL.

The accuracy on the testing set for each level of incompleteness has then been averaged over the ten folds. Figure 4 shows the value of the average accuracy as a function of the incompleteness level. As can be seen from the graph AICL outperforms ICL for the incompleteness levels from 5% to 20%. This shows that the abductions performed by AICL are frequently correct. Only when the level of incompleteness is particularly high (25%–30%) AICL has a performance similar to ICL. The average improvement of accuracy over all incompleteness levels is 11.2%.

7 Conclusions

We have proposed the algorithm AICL that modifies ICL in order to achieve a better performance on incomplete data. The modification is based on the use of an abductive proof procedure for testing the truth of clauses in the example interpretations.

AICL has been tested against ICL on a simple toy problem. Different levels of incompleteness of the data have been considered, from 5% to 30%. For the levels of incompleteness from 5% to 25% AICL reached a higher accuracy.

In the future, we plan to perform other experiments on larger domains in order to draw more grounded conclusions. In particular, we plan to apply AICL to the problem of learning the specification of protocols of interaction among agents from traces of their execution. In fact, these traces are very often incomplete due to the impossibility of recording every message exchanged between any two agents. Moreover, we would also like to investigate the adoption of other abductive proof procedures, as for example the IFF [Fung and Kowalski, 1997], the SCIFF [Alberti *et al.*, 2004] or the A-system [Kakas *et al.*, 2001], for completing the interpretations. These proof procedures are interesting because they provide a better handling of non ground abducibles.

References

[Alberti *et al.*, 2004] Marco Alberti, Marco Gavanelli, Evelina Lamma, Paola Mello, and Paolo Torroni. Abduction with hypothesis confirmation. In G. Rossi, editor, *Proceedings of the Convegno Italiano di Logica Computazionale (CILC-2004)*. University of Parma, June 2004.

[De Raedt and Dehaspe, 1997] L. De Raedt and L. Dehaspe. Clausal discovery. *Machine Learning*, 26(2–3):99–146, 1997.

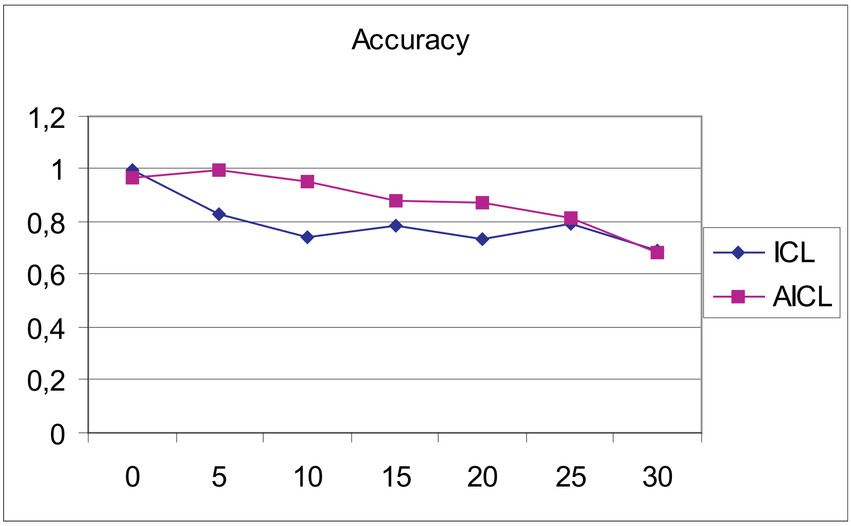

Figure 4: Accuracy as a function of the incompleteness level.

[De Raedt and Džeroski, 1994] L. De Raedt and S. Džeroski. First order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.

[De Raedt and Van Laer, 1995] L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 6th Conference on Algorithmic Learning Theory*, volume 997 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1995.

[Fung and Kowalski, 1997] T. H. Fung and R. A. Kowalski. The IFF proof procedure for abductive logic programming. *Journal of Logic Programming*, 33(2):151–165, November 1997.

[Inoue, 2001] K. Inoue. Induction, abduction, and consequence-finding. In Céline Rouveirol and Michèle Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming*, volume 2157 of *Lecture Notes in Artificial Intelligence*, pages 65–79. Springer-Verlag, September 2001.

[Kakas and Mancarella, 1990] A. C. Kakas and P. Mancarella. On the relation between truth maintenance and abduction. In *Proceedings of the 2nd Pacific Rim International Conference on Artificial Intelligence*, 1990.

[Kakas and Riguzzi, 2000] Antonis C. Kakas and Fabrizio Riguzzi. Abductive concept learning. *New Generation Computing*, 18(3), May 2000.

[Kakas *et al.*, 2001] A. C. Kakas, B. van Nuffelen, and M. Denecker. A-System: Problem solving through abduction. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, Seattle, Washington, USA (IJCAI-01)*, pages 591–596, Seattle, Washington, USA, August 2001. Morgan Kaufmann Publishers.

[Lamma *et al.*, 1999] Evelina Lamma, Paola Mello, Michela Milano, and Fabrizio Riguzzi. Integrating induction and abduction in logic programming. *Information Sciences*, 116(1):25–54, May 1999.

[Muggleton and Bryant, 2000] Stephen Muggleton and Christopher Bryant. Theory completion using inverse entailment. In J. Cussens and A. Frisch, editors, *Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 130–146. Springer-Verlag, 2000.

[Muggleton and De Raedt, 1994] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[Ray *et al.*, 2003] O. Ray, K. Broda, and A. Russo. Hybrid abductive inductive learning: a generalisation of Progol. In T. Horváth and A. Yamamoto, editors, *Proceedings of the 13th International Conference on Inductive Logic Programming*, volume 2835 of *Lecture Notes in Artificial Intelligence*, pages 311–328. Springer-Verlag, 2003.

[Yamamoto, 2000] A. Yamamoto. Using abduction for induction based on bottom generalization. In P. A. Flach and A. C. Kakas, editors, *Abductive and Inductive Reasoning, Essays on their Relation and Integration*, volume 18 of *Pure and Applied Logic*. Kluwer, 2000.