Overview of ILP Systems

Svetla Boytcheva

Department of Information Technologies, Faculty of Mathematics and Informatics, Sofia University, Bulgaria

Abstract: The current paper presents a brief overview of Inductive logic programming (ILP) systems. ILP algorithms are of special interest for machine learning, because most of them offer practical methods for extending the presentations used in algorithms that solve supervised learning tasks. The paper presents major approaches for solving supervised learning task, summarizes their features and classifies systems according different dimensions.
Keywords: *Inductive logic programming, Supervised Learning, Relational Learning*

1. Introduction

ILP algorithms are of special interest for machine learning, because most of them offer practical methods for extending the presentations used in algorithms that solve supervised learning tasks. According to languages used for presentation of examples, hypotheses and background knowledge (BK) we can separate these machine learning algorithms to two major classes: propositional (attribute-value) and relational. Relational languages are based on first-order logic and they are more expressive than propositional languages, because they allow more compact presentation of hypotheses, construction of recursive hypotheses, background knowledge usage. Thus relational representation is more convenient than attribute-value representation for many task domains, including: geography, mutogenesis, natural language processing, proteins' structure analyses, information extraction, mesh analyses, robotics, drugs and etc. The paper surveys ILP algorithms, focusing on major approaches for solving supervised learning task, summarizes their features and classifies systems according different dimensions.

2. Language Bias

ILP algorithms usually use one of the following relational languages:

- *general clauses* language
- *Horn clauses* language

Construction of the hypothesis in the language frameworks is not always possible, because of the following reasons:

- hypotheses space is huge and/or complex
- used language is not expressive enough

To solve this problem are used to types of bias - mechanism employed by a learner to constrain the search for hypotheses:

- language bias - determines the search space itself
- search bias - determines how the hypothesis space is searched .

There are two categories language bias

- the syntactic restrictions of the selected logic formalism;
- the vocabulary of predicate, function, variables and constant symbols: function-free clauses, ground clauses (e.g. without variables), non-recursive clauses, mode declarations (input/output) of the predicates' arguments

To represent examples, hypotheses and BK in the learning task are used examples' language (L_E), hypotheses language (L_H) and BK language (L_B).

Each of language restrictions mentioned above could be applied to each of these languages independently, or to all of them together (Table 1).

Table 1

System	Language bias					
	mode declarations (input/output) types of the predicates' arguments	function-free clauses	ground clauses	ground literals	non-recursive clauses	Horn clauses
LINUS				L_E	L_H	L_H
FOIL	L_E , L_H	L_E		L_B		L_H
MARKUS	L_E , L_B, L_H					L_H
FOIDL	L_E , L_H		L_E			L_H
GOLEM				L_E , L_B		L_H
LFP2				L_E , L_B		L_H
RICH		L_E , L_B			L_B	L_E , L_B, L_H

All ILP systems use some language bias. Mode declarations and learning of non-recursive clauses are necessary for narrowing search in the hypotheses space, but other language restrictions are imposed from the theory. For example, such a hypothesis does not exists in general case when both set of examples and BK set consists of Horn clauses.

3. Shift of Bias

To construct a hypothesis there are also used two types shift of bias:

- switch to a more expressive language (higher-order rules):
  - *second-order schema:* CIA [5], WiM [26] learns higher-order rule schemas by simply variablizing both the terms and the predicates of previously generated Horn clauses;
  - *higher-order rule schemas*: MODELER [36] keeps to each rule a set of its exceptions and this set increases enough generates a new predicate;
  - lambda-calculus:LILP (Lambda Inductive Logic Programming) [16].
- extend the given vocabulary by new predicates – predicate invention: MODELER [36], RINCON[35], CHILLIN[38], CIGOL[19], RICH[39].

Bias shift is used to construct more compact hypothesis, but usually the hypotheses space increases.

4. Characteristics of ILP systems

Incremental/ Non-incremental: This dimension describes the way the evidence (examples) is obtained. In non-incremental or empirical ILP, the evidence is given at the start and not changed afterwards, in incremental ILP, the examples are input one by one by the user, in a piecewise fashion. Non-incremental systems search typically either specific-to-general or general-to-specific. Incremental systems usually employ a mixture of these strategies as they may need to correct earlier induced hypotheses. Incremental ILP systems include: FORTE [29], LFP2 [34], MARVIN[32], RINCON [35] и CIGOL [19]. Empirical ILP systems include: GOLEM [20], FOIL [27], FOCL [22], MFOIL [6], ILP-R[25], RICH[39] and LINUS [7].

Interactive/Non-interactive: In interactive ILP, the learner is allowed to pose questions to an oracle (i.e. the user) about the intended interpretation.. Usually these questions query the user for the intended interpretation of an example or a clause. The answers to the queries allow to prune large parts of the search space (in the generic algorithm queries would normally be generated in the procedure Prune). Obviously, interactiveness implies incrementality. Most systems are non-interactive.. For example, interactive systems are: CIGOL [19], MARVIN [32], IRES [31] и ITOU [30].

Single/Multiple Predicate Learning: Single predicate learning systems are most popular ILP systems, but multiple predicate learning algorithms are more powerful. Although they are non efficient and hard. Recently interest to such systems growing: FORTE [29].

Theory Revision: Usually most of the systems have prestored BK, and systems keep it unchanged during the learning process, but there are some systems that allow theory revision. Although modifications of BK are possible, these systems observe the principle to stay most closely to the initial BK and to do minimum changes. Usually systems with theory revision are incremental multiple predicate learning systems. For example, MARVIN[32], CIGOL[19], M-ACL [11]. Theory revision systems often use many deductive and inductive rules, e.g. abduction combined with specialization and generalization M-ACL [11], ACL[12].

5. ILP Learning approaches

5.1. Algorithms using multiple representation

In these algorithms initially examples have relational representation and then they are transformed to new representation (usually propositional language). Thus, using these new examples' description algorithms can take advantages of some propositional learning algorithms. Finally the result hypotheses are transformed back to the initial representation. Thereby they avoid searching in the complex Horn clauses hypotheses space and construct compact hypothesis represented on the relational language. Algorithms WYL [8] and LINUS [7] use this approach.

In WYL initially examples are represented by relational language and then they are transformed to propositional language and hypothesis is created using decision trees. Finally the result hypothesis is transformed back to the relational language.

The current version of LINUS supports interfaces for working with propositional algorithms ASSISTANT [3], NEWGEM [17], and CN2[4]. LINUS has two modes:

- CLASS – corresponds to the propositional algorithm employed;
- RELATION – in this mode LINUS works as ILP system.

The basic principle of the transformation from first-order into propositional form is that all body literals which may possibly appear in a hypothesis clause (in the first-order formalism) are determined, thereby taking into account variable types. Each of these body literals corresponds to a boolean attribute in the propositional formalism.

One of the major defects of this approach is that these algorithms can not use BK, because they use proporsitional language for learning.

5.2. Searching in the hypothesis space

A lots of ILP algorithms belong to this group and use following search bias:

- *Uniformed search* (depth-first, breadth-first, iterative deepening): This is rarely used approach, because the huge hypothesis space. One of algorithms from this class is HYPER [2]. It learns logic programs by searching the space of complete hypotheses (i.e., sets of programs clauses), rather than performing repeated search for individual clauses.
- *Heuristic search* (best-first, hill-climbing, beam search)
  - for directing search
  - for stopping search (quality criterion)

FOIL [27] is one of the first successful empirical relational learning algorithms used this approach and on its base are developed many other algorithms.

Positive as well as negative examples are required for learning. FOIL induces concept definitions represented as function-free Horn clauses, optionally containing negated body literals. The background knowledge predicates are represented extensionally as sets of ground tuples. FOIL employs a heuristic search strategy (hill-climbing according to the information gain heuristics), which prunes vast parts of the hypothesis space. As its general search strategy, FOIL adopts a covering approach. For further control of the language bias, FOIL provides parameters limiting the total number and maximum depth of variables in a single clause. In addition, FOIL incorporates mechanisms for excluding literals which might lead to endless loops in recursive hypothesis clauses. FOIL stops adding literals to the hypothesis clause if the clause reaches a predefined minimum accuracy or if the encoding length of the clause exceeds the number of bits needed for explicitly encoding the positive examples it covers. This stop criterion prevents the induction of overly long and specific clauses in noisy domains.

Although search strategies of FOIL and its family algorithms makes them very efficient, they have considerable defect - these algorithms in the search process sometimes can prune searched hypotheses. To solve this problem are developed different modifications of FOIL:

- *Language bias*: **FOCL** [22] allows user-defined constraints which realize a declarative language bias (e.g. number of body literals in clauses) allow to restrict the search space.
- *Imperfect data handling*: HYDRA [1], MFOIL [6] The concept descriptions compete to classify test examples based on the likelihood ratios that are assigned to clauses of that concept description. This makes the algorithm more robust against noise.
- *Heuristics modification*:
  - CHAM [14] extends FOIL's information-gain heuristic with a syntactic measure of the “closeness" between a clause's input and existentially quantified variables with its output variables. This extension helps it to learn relations not learnable by FOIL.
  - MFOIL [6] uses beam-search with *m* –estimate heuristics function that takes into account the prior probabilities of examples, leading to a more reliable criterion for small example sets. The user-settable parameter *m* allows to control the influence of the prior probabilities
  - CLOG [15] the currently used gain function is user-defined.
  - ILP-R [25] It uses a non-myopic heuristic function called RELIEF. At the outer level, this learner uses a covering approach similar to the one used by FOIL. At the inner level, its top-down search for a consistent clause uses the RELIEF based heuristic for literal quality estimation.
- *Decision-trees*:
  - STRUCT [33] learns decision trees, where the root is the head of the target relation, each interior node is a literal, and paths through the tree encode Horn clauses.]
  - FFOIL [28] - the clauses found by FFOIL make up a decision list

- FOIDL [18] is a descendant of FOIL Unlike FFOIL, FOIDL generates the clauses in the decision list in reverse order.

▪ *Heuristic search algorithm*:
  - MARKUS [10] employs a covering strategy as FOIL, but it uses iterative deepening search.
  - MFOIL [6] uses beam-search.

▪ *Other features*:
  - *theory revision*: FORTE (First Order Revision of Theories from Examples) [29].
  - *inverse resolution operators*: FORTE [29].
  - *functional relations*: FFOIL [28] is specialized on learning functional relations. A functional relation is a relation where one or more arguments are distinguished as output arguments, and in any tuple of constants belonging to the relation the values of the output arguments are uniquely determined by the values of the other arguments.
  - *numerical arguments*: Handling numerical constraints in the normal ILP setting takes the form of induction of classification or regression rules that involve the use of real numbers, predicting a discrete or a real-valued class in the presence of background knowledge. FORS (First order regression system) [13] is an implementation of this idea, where numerical regression is focused on a distinguished continuous argument of the target predicate. This can be viewed as a generalization of the usual ILP problem.

5.3. Inverse resolution:

ILP systems use the following varieties of inverse resolution V- and W-operators (Table 1).: *absorbtion, identification, intra-construction, inter-construction, truncation, G1,G2.*

Table 1 Inverse resolution operators

System	Absorbtion	Inter-construction	Intra-construction	Truncation	G1	G2
MARVIN	X					
RINCON	X	X				
CIGOL	X		X	X		
IRES	X		X	X		
ITOU	X		X	X		
LFP2					X	X

MARVIN [32] was the first relational algorithm to incorporate this approach.. MARVIN is oracle-guided incremental algorithm. However, its concept description language is limited: it cannot learn clauses with existentially quantified variables and cannot invent new predicate descriptors.

RINCON [35] also is an incremental algorithm, but not oracle-guided. It uses intra-construction operator for inducing new predicate and after that apply absorbtion to replace some of literals with the head of newly generated predicate.

CIGOL [19] is oracle-guided incremental algorithm This is the first algorithm combining the three major inverse resolution operators. CIGOL's truncation operator is restricted to processing unit Horn clauses and the implementation of its other operators assume that one of the parent clauses is a unit clause. LFP2 [34] replaces CIGOL 's three operators with two more general operators that have no unit clause restrictions.

IRES [31] uses IRES system uses a flattening technique to simplify CIGOL 's operators and allow them to work with non-unit Horn clauses. ITOU [30] is descendant of IRES, and it uses the same operators like IRES, but extended with saturation.

5.4. Iinverse entailment

Inverse entailment approach was introduced by S. Muggleton [21]. The main difference between inverse entailment and inverse resolution is that in the first approach treats the problem of finding clauses from model-theoretical point of view, but the second approach treats this problem from proof-theoretical point of view. Only a few systems use inverse entailment approach: P-Progol [21] and its descendent Aleph.

5.5. Constructing RLGG (Relative least general generalization)

One of the characteristics of these systems is that they instead searching in the hypothesis space, tries to construct a clause that generalizes the set of examples. The first algorithm from this class was developed by Plotkin [23, 24], but unfortunately it was more theoretical than practical, because the number of literals in the constructed hypothesis increases exponentially and in some cases infinite.

GOLEM [20] is one of the "classical" algorithms using this approach. GOLEM is empirical algorithm and uses covering methods. It chooses random subset of the set of positive examples and constructs their RLGG. Between all constructed RLGG in such way, GOLEM chooses this one that covers greatest number positive examples and does not cover negative examples. On the next step GOLEM generalizes the best RLGG. This process continues until increasing the set of cover positive examples from the constructed RLGG stop. As a final step GOLEM reduces constructed RLGG by dropping irrelevant literals. Both the BK and examples consist only ground facts. There are also some restrictions to the hypothesis variables depth. GOLEM can not generate automatically new predicates.

RICH (Relative Implication of Horn clauses) [39] is also empirical algorithm, but in contracts of GOLEM both BK and examples consist function-free non recursive Horn clauses. To construct hypothesis RICH uses unification, anti-unification algorithms and some resolution steps. RICH can generate automatically new predicates.

6. Accuracy and Time characteristics

The following characteristics are measured in the classical chess and endgame domain "White King and Rook versus Black King", described in [40]. The results of the experiment are presented in the following table: The classification accuracy is given by the percentage of correctly classified testing instances and by the standard deviation (sd), averaged over 5 experiments.

Table 2

	100 training examples		1000 training examples	
System	Accuracy	Time	Accuracy	Time
CIGOL	77.2%	21.5hr	N/A	N/A
FOIL	90.8%	31.6 sec	99.4%	4.0 min
LINUS-ASSISTANT	98.1%	55.0 sec	99.7%	9.6 min
RICH	95.3%	53.9 sec	99.6%	8.3 min

Although LINUS is better than others algorithms in small and large training sets, it has one major defect - does not provide features for handling BK. From the rest algorithms RICH has better accuracy, but it is slower.

Summary

Although search strategies of FOIL and its family algorithms makes them very efficient, they have considerable defect - these algorithms in the search process sometimes can prune searched hypotheses.

Many inverse resolution algorithms increase the concept description language by constructing predictor descriptors (i.e., predicates), but are either limited to deduction or require an oracle to maintain reasonable efficiency.

Constructing RLGG methods employ additional constraints on the concept representation language (i.e., on existentially quantified variables). This trade off increases efficiency. However, efficient RLGG methods for automatically constructing descriptors have not yet been developed.

All of these algorithms are limited. For example, algorithms that use multiple representations cannot yet learn recursive relations. Information-gain directed algorithms cannot yet learn relations with function symbols. Efficient methods for automatically generating higher-order schemas without oracle guidance do not yet exist, except when learning is restricted to deductive inferencing. Most of RLGG methods cannot generate new descriptors.

References

[1] Ali K.M. and M.J. Pazzani. *Hydra: A noise-tolerant relational concept learning algorithm.* Proc. of IJCAI –93, pp. 1064-1071. Morgan Kaufmann, 1993.

[2] Bratko I.. *Refining complete hypotheses in ILP*. In Proc. of 9th International Workshop on Inductive Logic Programming, pp. 44-55. Springer, 1999.

[3] Cestnik, B., Kononenko, I., & Bratko, I.. *ASSISTANT-86: A knowledge-elicitation tool for sophisticated users*. In I. Bratko & N. Lavrač (Eds.), Progress in Machine learning. Bled, Yugoslavia: Sigma Press, 1987

[4] Clark, P. E., & Boswell, R. *Rule induction with CN2: Some recent improvements.* In Proceedings of the Fifth European Working Session on Learning pp. 151-163. Porto, Portugal: Springer-Verlag. 1991.

[5] De Raedt, L., & Bruynooghe, M.. *Constructive induction by analogy*. In Proc. of ICML -89, pp. 476-477. Morgan Kaufmann. 1989.

[6] Dzeroski. S. *Handling imperfect data in inductive logic programming*. In Procc. of the 4th Scandinavian Conference on AI, pp. 111-125. IOS Press, 1993.

[7] Džeroski S. and N. Lavrač. *Learning relations from noisy examples: An empirical comparison of LINUS and FOIL*. In L. Birnbaum and G. Collins, eds., Proc. of the 8th International Workshop on Machine Learning, pp. 399-402. Morgan Kaufmann, 1991.

[8] Flann, N. S., & Dietterich, T. G. *Selecting appropriate representations for learning from examples*. In Proceedings of the Fifth National Conference on Artificial Intelligence, pp. 460-466. Philadelphia, PA: Morgan Kaufmann, 1986.

[9] Giordana A. and F. Neri. Search-intensive concept induction. *Evolutionary Computation Journal*, 3(4):375-416, 1996.

[10] Grobelnik M.. *MARKUS: An optimized model inference system*. In C. Rouveirol, editor, Proc. of the ECAI-92 Workshop on Logical Approaches to ML, 1992.

[11] Kakas A., E. Lamma, and F. Riguzzi. *Learning multiple predicates*. In M. Lenzerini, editor, Proc. of AIMSA-98LNAI, 1480, pp. 303-316. Springer-Verlag, 1998.

[12] Kakas A.C. and F. Riguzzi, *Learning with Abduction*. Proc. in ILP97, Lecture Notes in Artificial Intelligence, Volume 1297, Springer-Verlag, 1997, pp. 181-189, 1997

[13] Karalic A., I. Bratko: First Order Regression. *Machine Learning*, 1997.

[14] Kijsirikul, B., Numao, M., & Shimura, M.. *Efficient learning of logic programs with non-determinate, non-discriminating literals.* In Proc. of the First International Workshop on Inductive Logic Programming, pp. 33-40. 1991

[15] Manandhar Suresh, Saso Dzeroski, and Tomaz Erjavec. *Learning Multilingual Morphology with CLOG*. In Proc. of ILP'98, Madison, Wisconsin, USA, 1998.

[16] Markov, Z. *A Functional Approach to ILP*, Proc. of ILP-95, 4-6 Sept. 1995, Leuven, Scientific report, Department of Computer Science, K.U. Leuven, pp. 267-280. 1995

[17] Mozetic, I., *NEWGEM: Program for learning from examples.* Technical documentation and user's guide. Reports of Intelligent Systems Group UIUCDCS-F-85-949, Department of Computer Science, University of Illinois. Urbana Champaign, IL, 1985.

[18] Mooney R.J. and M.E. Califf. Induction of first-order decision lists: Results on learning the past tense of English verbs. *Journal of Artificial Intelligence Research*, 3:1-24, 1995.

[19] Muggleton, S. and Buntine, *W. Machine invention of first-order predicates by inverting resolution.* In J.Laird editior, Proc. ICML-88,pp. 339-352, Morgan Kaufman, San Mateo, CA. 1988.

[20] Muggleton, S., & Feng, C. Efficient induction of logic programs. *Proc. of the First International Workshop on Algorithmic Learning Theory,* 368-381, Tokyo, Japan: Japanese Society for Artificial Intelligence. 1990.

[21] Muggleton, S. Inverse entailment and Progol. *New Generation Computing*, pp. 245–286, 1995

[22] Pazzani, M., & Kibler, D. *The utility of knowledge in inductive learning* (Technical Report 90-18). University of California, Irvine, Department of Information and Computer Science. 1990.

[23] Plotkin G.D.. A note on the inductive generalization. *Machine Intelligence*, 5:153-163, 1970.

[24] Plotkin G.D.. *Automatic Methods of Inductive Inference*. PhD thesis, Edinburg University, 1971.

[25] Pompe U.. *Restricting the hypothesis space, guiding the search, and handling the redundant information in ILP*. MSc Thesis, University of Ljubljana, Faculty of Computer Science and Informatics, Ljubljana, 1996.

[26] Popelinsky L., Stepankova 0.: WiM: *A Study on the Top-Down ILP Program.* FIMU-RS-95-03, Faculty of Informatics, 1995.

[27] Quinlan, J. R.. Learning logical definitions from relations. *Machine Learning,* 5, 239-266, 1990

[28] Quinlan. J.R. Learning first-order definitions of functions. *Journal of Artificial Intelligence Research*, 5:139-161, 1996.

[29] Richards B.L. and R.J. Mooney. Refinement of first-order Horn-clause domain theories. *Machine Learning,* 19(2):95-131, 1995.

[30] Rouveirol C. *Extensions of Inversion of Resolution applyied to Theory Completion*. Inductive Logic Programming, S, Muggleton (Ed.). Academic Press: London, pp. 63- 92, 1992.

[31] Rouveirol, C., & Puget, J. F. *Beyond inversion of resolution*. In Proceedings of the Seventh International Conference on Machine Learning, pp. 122-130. Austin, TX: Morgan Kaufmann. 1990.

[32] Sammut, C., & Banerji, R. B. Learning concepts by asking questions. In R. S. Michalski, J. G.Car-bonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. II). San Mateo, CA: Morgan Kaufmann. 1986.

[33] Watanabe, L., & Rendell, L. *Learning structural decision trees from examples*. In Proc. of IJCAI-91. Sydney, Australia,1991

[34] Wirth, R. *Completing logic programs by inverse resolution*. In Proceedings of the Fourth European Working Session on Learning pp. 239-250. Montpellier, France: Pitman. 1989.

[35] Wogulis, J. *A framework for improving efficiency and accuracy*. In Proc. of the Sixth International Workshop on Machine Learning, pp. 78-80. Morgan Kaufmann. 1989.

[36] Wrobel, S. *Automatic representation adjustment in an observational discovery system.* In Proceedings of the Third European Working Session on Learning, pp. 253-262. Glasgow, Scotland: Pitman. 1988

[37] Wirth, R., & O'Rorke, P.. Inductive completion of SLD proofs. *In Proceedings of the First In-Iternational Workshop on Inductive Logic Programming*, pp. 167-176, 1991

[38] Zelle J.M., R.J. Mooney, and J.B. Konvisser. *Combining top-down and bottom-up techniques in inductive logic programming*. In W.W. Cohen and H. Hirsh, editors, Proc. of ICML-94, pp. 343-351. Morgan Kaufmann, 1994.

[39] Boytcheva, S., Z. Markov, ***An Algorithm for inducing least generalization under relative implication***, In Proc. of the 15th International conference of Florida Artificial Intelligence Research Society (FLAIRS-2002), AAAI Press, 13-16 May 2002, Pensacola, Florida, USA, pp. 322-326, 2002.

[40] Muggleton S., Bain M., Hayes-Michie J. and D. Michie. *An experimental comparison of human and machine learning formalisms*. In Proc, Sixth International Workshop on Machine Learning, pp. 113-118, Morgan Kaufmann, San Mateo, CA, 1989.

Абстракт: Настоящата статия е кратък обзор на системите базирани на ИЛП (индуктивно логическо програмиране). Алгоритмите в ИЛП са от особен интерес за МС (машинното самообучение), защото повечето от тях предлагат практически методи за разширяване на представянето използвано при решаването на тези задачи. Статията представя основните подходи, които се използват в системите за решаването на задачите за МС, прави сравнение на техните основни характеристики, и представя класификации според различни критерии.