Meta-Interpretive Learning: achievements and challenges

Stephen H. Muggleton

Imperial College London, UK

Abstract. This invited talk provides an overview of ongoing work in a new sub-area of Inductive Logic Programming known as Meta-Interpretive Learning.

1 Introduction

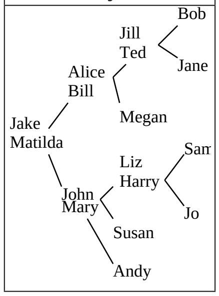
Meta-Interpretive Learning (MIL) [12] is a recent Inductive Logic Programming [7, 13, 14] technique aimed at supporting learning of recursive definitions. A powerful and novel aspect of MIL is that when learning a predicate definition it automatically introduces sub-definitions, allowing decomposition into a hierarchy of reuseable parts. MIL is based on an adapted version of a Prolog meta-interpreter. Normally such a meta-interpreter derives a proof by repeatedly fetching first-order Prolog clauses whose heads unify with a given goal. By contrast, a meta-interpretive learner additionally fetches higher-order meta-rules whose heads unify with the goal, and saves the resulting meta-substitutions to form a program. This talk will overview theoretical and implementational advances in this new area including the ability to learn Turing computabale functions within a constrained subset of logic programs, the use of probabilistic representations within Bayesian meta-interpretive and techniques for minimising the number of meta-rules employed. The talk will also summarise applications of MIL including the learning of regular and context-free grammars, [11], learning from visual represention [3] with repeated patterns, learning string transformations for spreadsheet applications, [6], learning and optimising recursive robot strategies [1] and learning tactics for proving correctness of programs [5]. The paper concludes by pointing to challenges which remain to be addressed within this new area.

2 Simple worked example

Suppose we machine learn a set of kinship relations such as those in Figure 1. If examples of the ancestor relation are provided and the background contains only father and mother facts, then a system must not only be able to learn ancestor as a recursive definition but also simultaneously *invent* parent to learn these definitions.

Although the topic of Predicate Invention was investigated in early Inductive Logic Programming (ILP) research [8, 18] it is still seen as hard and under-explored [14]. ILP systems such as ALEPH [17] and FOIL [15] have no predicate invention and limited recursion learning and therefore cannot learn recursive grammars from example sequences. By contrast, in [11] definite clause grammars were learned with predicate

Family Tree



Target Theory
$father(ted, bob) \leftarrow$
$father(ted, jane) \leftarrow$
$parent(X, Y) \leftarrow mother(X, Y)$
$parent(X, Y) \leftarrow father(X, Y)$
$ancestor(X, Y) \leftarrow parent(X, Y)$
$ancestor(X, Y) \leftarrow parent(X, Z), ancestor(Z, Y)$

First-order	Metalogical substitutions
Examples $ancestor(jake, bob) \leftarrow$ $ancestor(alice, jane) \leftarrow$	N/A
Background Knowledge $father(jake, alice) \leftarrow$ $mother(alice, ted) \leftarrow$	N/A
Instantiated Hypothesis $father(ted, bob) \leftarrow$ $father(ted, jane) \leftarrow$ $p1(X, Y) \leftarrow father(X, Y)$ $p1(X, Y) \leftarrow mother(X, Y)$ $ancestor(X, Y) \leftarrow p1(X, Y)$ $ancestor(X, Y) \leftarrow p1(X, Z),$ $ancestor(Z, Y)$	 $metasub(instance, [father, ted, bob])$ $metasub(instance, [father, ted, jane])$ $metasub(base, [p1, father])$ $metasub(base, [p1, mother])$ $metasub(base, [ancestor, p1])$ $metasub(tailrec, [ancestor, p1, ancestor])$

Fig. 1: Kinship example. *p1* invented, representing *parent*.

invention using Meta-Interpretive Learning (MIL). MIL [9, 10, 6] is a technique which supports efficient predicate invention and learning of recursive logic programs built as a set of metalogical substitutions by a modified Prolog meta-interpreter (see Figure 2) which acts as the central part of the ILP learning engine. The meta-interpreter is provided by the user with *meta-rules* (see Figure 3) which are higher-order expressions describing the forms of clauses permitted in hypothesised programs. As shown in Figure 3 each meta-rule has an associated Order constraint, which is designed to ensure termination of the proof. The meta-interpreter attempts to prove the examples and, for any successful proof, saves the substitutions for existentially quantified variables found in the associated meta-rules. When these substitutions are applied to the meta-rules they result in a first-order definite program which is an inductive generalisation of the examples. For instance, the two examples shown in the upper part of Figure 1 could be proved by the meta-interpreter in Figure 2 from the Background Knowledge BK by generating the Hypothesis H using the Prolog goal

$$\leftarrow prove([ancestor, jake, bob], [ancestor, alice, jane], BK, H).$$

Generalised meta-interpreter
$prove([], Prog, Prog).$
$prove([Atom\|As], Prog1, Prog2) :-$
$\quad metarule(Name, MetaSub, (Atom \text{ :- } Body), Order),$
$\quad Order,$
$\quad save\_subst(metasub(Name, MetaSub), Prog1, Prog3),$
$\quad prove(Body, Prog3, Prog4),$
$\quad prove(As, Prog4, Prog2).$

Fig. 2: Prolog code for the generalised meta-interpreter. The interpreter recursively proves a series of atomic goals by matching them against the heads of meta-rules. After testing the Order constraint $save\_subst$ checks whether the meta-substitution is already in the program and otherwise adds it to form an augmented program. On completion the returned program, by construction, derives all the examples.

Name	**Meta-Rule**	**Order**
Instance	$P(X, Y) \leftarrow$	$True$
Base	$P(x, y) \leftarrow Q(x, y)$	$P \succ Q$
Chain	$P(x, y) \leftarrow Q(x, z), R(z, y)$	$P \succ Q, P \succ R$
TailRec	$P(x, y) \leftarrow Q(x, z), P(z, y)$	$P \succ Q,$ $x \succ z \succ y$

Fig. 3: Examples of dyadic meta-rules with associated Herbrand ordering constraints. $\succ$ is a pre-defined ordering over symbols in the signature.

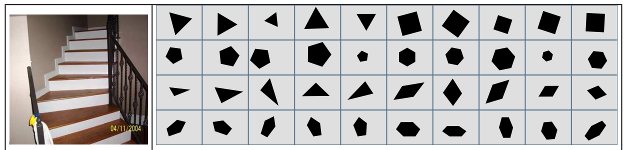
H is constructed by applying the metalogical substitutions in Figure 1 to the corresponding meta-rules found in Figure 3. Note that $p1$ is an invented predicate corresponding to *parent*.

Completeness of SLD resolution ensures that *all* hypotheses consistent with the examples can be constructed. Moreover, unlike many ILP systems, *only* hypotheses consistent with all examples are considered. Owing to the efficiency of Prolog backtracking MIL implementations have been demonstrated to search the hypothesis space 100-1000 times faster than state-of-the-art ILP systems [11] in the task of learning recursive grammars[1].

3 Vision applications

Figure 4 illustrates two applications in which MIL has been used to analyse images. The staircase learning in Figure 4a was based on data from Claude Sammut's group [4]. However, the original author's approach, using ALEPH was not entirely general since it does not involve recursion. Using MIL it was possible to learn a general recursive definition of a staircase using predicate invention. A staircase is represented as a set of ordered planes, where the background predicates *vertical* and *horizontal* describe

[1] Metagol$_R$ and Metagol$_{CF}$ learn Regular and Context-Free grammars respectively.



a) Staircase | b) Regular polyhedra

Fig. 4: MIL vision applications: a) learning a recursion definition of a staircase from a single image [11] and b) learning definition relating regular polygons [3].

stair(X,Y) :- a(X,Y).
stair(X,Y) :- a(X,Z), **stair(Z,Y)**.
a(X,Y) :- vertical(X,Z), horizontal(Z,Y).

Fig. 5: Definition of stiarcase learned in 0.08s on a laptop from single image. Note Predicate invention and recursion.

adjacent planes. The resulting hypothesis is shown in Figure 5, where a is an invented predicate corresponding to $step$. Due to its recursive form, this definition has shorter description length than those found by ALEPH. It is also general in its applicability and easily understood.

4 Challenges

A number of open challenges exist for Meta-Interpretive Learning. These include the following.

Generalise beyond Dyadic logic. The dyadic fragment of Prolog has provided an efficient approach to selecting a compact and efficient universal set of metarules [2] for MIL. However, many Prolog programs are more natural to represent when represented with more than two arguments.

Deal with classification noise. Most data sources for machine learning contain both classification and attribute noise. We are presently developing variants of the Metagol system which act robustly in the faace of such noise.

Active learning. Most forms of machine learning are *passive* in the sense that they take a given training data set and generate a model. Active learning involves proposing and testing instances which are classified either by a user of by carrying out experiments in the real world. We are developing probabilistic variants of Meta-Interpretive Learning [10] which could be adapted for efficient Active Learning.

Efficient problem decomposition. Finding efficient ways of decemposing the definitions in MIL is one of the hardest open problems in the field.

Meaningful hypotheses. In ongoing work [16] we are investigating the issues which are most important for improving the understandability of learned programs.

References

1. Cropper, A., Muggleton, S.: Learning efficient logical robot strategies involving composable objects. In: Proceedings of the 24th International Joint Conference Artificial Intelligence (IJCAI 2015). pp. 3423–3429. IJCAI (2015), http://www.doc.ic.ac.uk/s̃hm/Papers/metagolo.pdf
2. Cropper, A., Muggleton, S.: Logical minimisation of meta-rules within meta-interpretive learning. In: Proceedings of the 24th International Conference on Inductive Logic Programming. pp. 65–78. Springer-Verlag (2015), http://www.doc.ic.ac.uk/s̃hm/Papers/minmeta.pdf, lNAI 9046
3. Dai, W.Z., Muggleton, S., Zhou, Z.H.: Logical Vision: Meta-interpretive learning for simple geometrical concepts. In: Late Breaking Paper Proceedings of the 25th International Conference on Inductive Logic Programming. pp. 1–16. CEUR (2015), http://ceur-ws.org/Vol-1636
4. Farid, R., Sammut, C.: Plane-based object categorization using relational learning. ILP2012 MLJ special issue (2012)
5. Farquhar, C., Cropper, G.G.A., Muggleton, S., Bundy, A.: Typed meta-interpretive learning for proof strategies. In: Short Paper Proceedings of the 25th International Conference on Inductive Logic Programming. National Institute of Informatics, Tokyo (2015), http://www.doc.ic.ac.uk/s̃hm/Papers/typemilproof.pdf
6. Lin, D., Dechter, E., Ellis, K., Tenenbaum, J., Muggleton, S.: Bias reformulation for one-shot function induction. In: Proceedings of the 23rd European Conference on Artificial Intelligence (ECAI 2014). pp. 525–530. IOS Press, Amsterdam (2014), http://www.doc.ic.ac.uk/s̃hm/Papers/metabias.pdf
7. Muggleton, S.: Inductive Logic Programming. New Generation Computing 8(4), 295–318 (1991), http://www.doc.ic.ac.uk/s̃hm/Papers/ilp.pdf
8. Muggleton, S., Buntine, W.: Machine invention of first-order predicates by inverting resolution. In: Proceedings of the 5th International Conference on Machine Learning. pp. 339–352. Kaufmann (1988), http://www.doc.ic.ac.uk/s̃hm/Papers/cigol.pdf
9. Muggleton, S., Lin, D.: Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. In: Proceedings of the 23rd International Joint Conference Artificial Intelligence (IJCAI 2013). pp. 1551–1557 (2013), http://www.doc.ic.ac.uk/s̃hm/Papers/metagold.pdf
10. Muggleton, S., Lin, D., Chen, J., Tamaddoni-Nezhad, A.: Metabayes: Bayesian meta-interpretative learning using higher-order stochastic refinement. In: Zaverucha, G., Costa, V.S., Paes, A.M. (eds.) Proceedings of the 23rd International Conference on Inductive Logic Programming (ILP 2013). pp. 1–17. Springer-Verlag, Berlin (2014), http://www.doc.ic.ac.uk/s̃hm/Papers/metabayeslong07.pdf, lNAI 8812
11. Muggleton, S., Lin, D., Pahlavi, N., Tamaddoni-Nezhad, A.: Meta-interpretive learning: application to grammatical inference. Machine Learning 94, 25–49 (2014), http://www.doc.ic.ac.uk/s̃hm/Papers/metagolgram.pdf
12. Muggleton, S., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning 100(1), 49–73 (2015), http://www.doc.ic.ac.uk/s̃hm/Papers/metagolDMLJ.pdf
13. Muggleton, S., Raedt, L.D.: Inductive logic programming: Theory and methods. Journal of Logic Programming 19,20, 629–679 (1994), http://www.doc.ic.ac.uk/s̃hm/Papers/lpj.pdf
14. Muggleton, S., Raedt, L.D., Poole, D., Bratko, I., Flach, P., Inoue, K.: ILP turns 20: biography and future challenges. Machine Learning 86(1), 3–23 (2011), http://www.doc.ic.ac.uk/s̃hm/Papers/ILPturns20.pdf
15. Quinlan, J.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)

16. Schmid, U., Zeller, C., Besold, T., Tamaddoni-Nezhad, A., Muggleton, S.: How does predicate invention affect human comprehensibility? In: Russo, A., Cussens, J. (eds.) Proceedings of the 26th International Conference on Inductive Logic Programming (ILP 2016). Springer-Verlag, Berlin (2016), http://www.doc.ic.ac.uk/s̃hm/Papers/compinv.pdf, in Press
17. Srinivasan, A.: The ALEPH manual. Machine Learning at the Computing Laboratory, Oxford University (2001)
18. Stahl, I.: Constructive induction in inductive logic programming: an overview. Tech. rep., Fakultat Informatik, Universitat Stuttgart (1992)