Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Learning Efficient Logic Programs

Andrew Cropper
Imperial College London, United Kingdom
a.cropper13@imperial.ac.uk

Abstract

Most logic-based machine learning algorithms rely on an Occamist bias where textual simplicity of hypotheses is optimised. This approach, however, fails to distinguish between the efficiencies of hypothesised programs, such as quick sort ($O(n \ log \ n)$) and bubble sort ($O(n^2)$). We address this issue by considering techniques to minimise both the resource complexity and textual complexity of hypothesised programs. We describe an algorithm proven to learn optimal resource complexity robot strategies, and we propose future work to generalise this approach to a broader class of logic programs.

1 Introduction

In machine learning, the goal of a search procedure is to find a hypothesis that is consistent with examples. If multiple hypotheses are consistent then Occam's razor suggests selecting the simplest one. Most logic-based machine learning algorithms rely on an Occamist bias to select hypotheses which minimise textual complexity. This approach, however, fails to distinguish between the efficiencies of hypothesised programs, such as quick sort ($O(n \ log \ n)$) and bubble sort ($O(n^2)$). Clearly, learning efficient logic programs is valuable in many domains, such as program synthesis and robotics.

We address this issue by considering techniques to minimise both the resource complexity and textual complexity of hypothesised logic programs. Our main contribution, thus far, is the introduction of a framework for minimising resource complexity in robot strategy problems and the demonstration of a learning algorithm proven to learn optimal resource complexity robot strategies.

2 Related work

Logic-based machine learning literature has addressed efficiently learning logic programs [Ahlgren and Yuen, 2013]. Likewise, in AI, planning literature has addressed developing efficient planners [Xing *et al.*, 2006]. By contrast, we want to learn efficient logic programs which are optimal with respect to an objective function by which the quality of a program is measured. A common objective function, based on Occam's razor, is the length of the hypothesis. In planning this is often the number of actions required to execute a plan. However, if certain actions are costly, we may prefer a hypothesis which minimises the overall cost of the actions. Action costs have been used in answer set programming to learn optimal plans [Eiter *et al.*, 2003; Yang *et al.*, 2014]. By contrast, we want to learn recursive logic programs involving predicate invention. This includes learning robot strategies where a strategy is a mapping from a set of initial situations to a set of goal situations. Some machine learning approaches support the construction of strategies [Laird, 2008; Otero, 2005], including approaches which employ heuristic policy optimisation [Sutton and Barto, 1998]. However, we are unaware of any learning approach that provides a provable convergent means for finding optimal strategies, nor of any approach that generalises to a broad class of logic programs which we propose.

3 Completed work

Our approach is an extension of meta-interpretive learning (MIL) [Muggleton *et al.*, 2014; 2015], a form of inductive logic programming based on an adapted Prolog meta-interpreter. Whereas a standard Prolog meta-interpreter attempts to prove a goal by repeatedly fetching first-order clauses whose heads unify with a given goal, a MIL learner attempts to prove a goal by repeatedly fetching higher-order metarules (higher-order expressions which describe the forms of clauses permitted, i.e. the declarative bias) whose heads unify with a given goal. The resulting meta-substitutions are saved in an abduction store and can be re-used in later proofs. Following the proof of a set of goals, a hypothesis is formed by projecting the meta-substitutions onto their corresponding metarules, allowing for a form of inductive programming which supports predicate invention and the learning of recursive theories.

We have developed Metagol$_O$, a variant of Metagol$_D$ [Muggleton *et al.*, 2015], an implementation of the MIL framework which learns programs within the Datalog subset H^2_2, where H^i_j consists of definite Datalog programs with predicates of adicity at most i and at most j literals in the body of each clause. The key difference between Metagol$_O$ and Metagol$_D$ is the search procedure. Metagol$_D$ uses iterative deepening to ensure that the first hypothesis returned contains the minimal number of clauses. By contrast, Metagol$_O$ uses iterative descent to minimise both the resource complexity and textual complexity of hypotheses. To explain this, we first define resource complexity:

Definition 1 (Resource complexity) *If E^+ is a set of positive examples and $H \in \mathcal{H}$ is a hypothesis in the hypothesis space, then the resource complexity of hypothesis H on example set E^+ is:*

$$r(H, E^+) = \sum_{e \in E+} r(H(e))$$

where $r(H(e))$ is the sum of resource costs of the predicates in applying the program H to example e.

To find the hypothesis with minimal resource complexity, we employ a search procedure named iterative descent, which works as follows: starting at iteration 1, we search for a hypothesis H_1 with the minimal number of clauses and we do not enforce a maximum resource bound. Since the hypothesis space is exponential in the length of the hypothesis [Lin *et al.*, 2014], the hypothesis H_1 is the most tractable to learn. The hypothesis H_1 gives us an upper bound on the resource complexity from which to descend. At iteration $i > 1$, we search for a hypothesis H_i with the minimal number of clauses but we enforce a maximum resource bound set to $r(H_{i-1}, E^+) - 1$. This ensures that any returned hypothesis H_i has a lower resource complexity than any hypothesis H_j, where $j < i$. If a hypothesis H_i exists, the search continues at $i+1$ until we converge on the optimal hypothesis. In [Cropper and Muggleton, 2015], we prove convergence of this search procedure and we describe experiments on two robot strategy problems (a robot postman and a robot mail sorter) where the efficiencies of learned programs are in agreement with the theoretical optimal predictions.

4 Conclusions and future work

By focusing on robot strategies, we have made an initial attempt at learning logic programs with optimal resource complexity. We intend to generalise this approach to a broader class of programs. For example, sorting algorithms provide a classic domain for discovering optimal algorithms, where, for instance, one may want to minimise the number of element swaps in a strategy. Other potential domains include learning proof tactics and learning game tactics. The ultimate goal is to generalise our approach to learn general logic programs, including normal logic programs.

The approach taken in this paper can be generalised in several ways. For example, the use of dyadic datalog programs could be generalised by using a richer set of metarules. For the immediate future, however, we intend to remain in the H_2^2 fragment, known to be Turing expressive [Muggleton *et al.*, 2015], which allows us to represent problems as robot strategies. For example, sorting algorithms can clearly be solved with a single-tape Turing machine. We can, therefore, represent this task as a robot strategy problem, where a robot manipulates symbols on a strip of tape. This strategy can be optimised, for instance, by minimising the number of manipulations required. Having learned optimal solutions for a single-tape Turing, we can then generalise to multi-tape Turing machines.

To summarise, we believe that the ideas proposed in this paper open exciting avenues in a variety of AI domains for understanding the value of machine learning efficient logic programs.

References

[Ahlgren and Yuen, 2013] John Ahlgren and Shiu Yin Yuen. Efficient program synthesis using constraint satisfaction in inductive logic programming. *The Journal of Machine Learning Research*, 14(1):3649–3682, 2013.

[Cropper and Muggleton, 2015] Andrew Cropper and Stephen H Muggleton. Learning efficient logical robot strategies involving composable objects. In *Proceedings of the 24th international joint conference on Artificial Intelligence*, 2015. To appear.

[Eiter *et al.*, 2003] Thomas Eiter, Wolfgang Faber, Nicola Leone, Gerald Pfeifer, and Axel Polleres. Answer set planning under action costs. *Journal of Artificial Intelligence Research*, pages 25–71, 2003.

[Laird, 2008] J. E. Laird. Extending the soar cognitive architecture. *Frontiers in Artificial Intelligence and Applications*, pages 224–235, 2008.

[Lin *et al.*, 2014] D. Lin, E. Dechter, K. Ellis, J.B. Tenenbaum, and S.H. Muggleton. Bias reformulation for one-shot function induction. In *Proceedings of the 23rd European Conference on Artificial Intelligence (ECAI 2014)*, pages 525–530, Amsterdam, 2014. IOS Press.

[Muggleton *et al.*, 2014] S.H. Muggleton, D. Lin, N. Pahlavi, and A. Tamaddoni-Nezhad. Meta-interpretive learning: application to grammatical inference. *Machine Learning*, 94:25–49, 2014.

[Muggleton *et al.*, 2015] S.H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. *Machine Learning*, 2015. To appear.

[Otero, 2005] R. Otero. Induction of the indirect effects of actions by monotonic methods. In *Proceedings of the Fifteenth International Conference on Inductive Logic Programming (ILP05)*, volume 3625, pages 279–294. Springer, 2005.

[Sutton and Barto, 1998] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[Xing *et al.*, 2006] Zhao Xing, Yixin Chen, and Weixiong Zhang. Optimal strips planning by maximum satisfiability and accumulative learning. In *Proceedings of the International Conference on Autonomous Planning and Scheduling (ICAPS)*, pages 442–446, 2006.

[Yang *et al.*, 2014] Fangkai Yang, Piyush Khandelwal, Matteo Leonetti, and Peter Stone. Planning in answer set programming while learning action costs for mobile robots. *AAAI Spring 2014 Symposium on Knowledge Representation and Reasoning in Robotics (AAAI-SSS)*, 2014.