The 29^{th} ILP conference, 3–5 Sep 2019, Plovdiv, Bulgaria.

Louise: A Meta-Interpretive Learner for Efficient Multi-clause Learning of Large Programs

Stassa Patsantzis and Stephen H. Muggleton
e.patsantzis17@imperial.ac.uk, s.muggleton@imperial.ac.uk.

Imperial College London

Abstract. We present Louise, a new Meta-Interpretive Learner that performs efficient multi-clause learning, implemented in Prolog. Louise is efficient enough to learn programs that are too large to be learned with the current state-of-the-art MIL system, Metagol. Louise learns by first constructing the most general program in the hypothesis space of a MIL problem and then reducing this "Top program" by Plotkin's program reduction algorithm. In this extended abstract we describe Louise's learning approach and experimentally demonstrate that Louise can learn programs that are too large to be learned by our implementation of Metagol, Thelma.

1 Introduction

In Meta-Interpretive Learning (MIL) [3, 2] a learner searches a space of hypotheses consisting of sets of first order definite datalog clauses. This search is expensive and the state-of-the-art MIL learner, Metagol [1] can not effectively learn theories larger than a handful of clauses in size[1]. Our new MIL-learner, Louise[2], implemented in Prolog, avoids a classical search of the space of hypotheses and instead learns by constructing the most general program that explains the training examples given the background knowledge, and then logically reducing this "Top program" to remove redundant clauses. The result is an efficient learner that can learn hypotheses that are too large to be learned by Metagol.

	blood_relative/2		*relative*/2	
	Time	Clauses	Time	Clauses
Thelma	76.509sec.	11	2hrs (Timed out)	5 (Timed out)
Louise	0.094sec.	11	46.169sec.	29

Table 1: Experiment results

[1] In practice this means 5 to a dozen clauses, depending on the problem.
[2] https://github.com/stassa/louise

Top program construction
Generalisation of positive examples
$m(chain, grandfather, father, father).$ $m(chain, grandfather, father, mother).$ $m(chain, grandfather, father, parent).$ $m(chain, grandfather, parent, father).$ $m(chain, grandfather, parent, mother).$ $m(chain, grandfather, parent, parent).$
Specialisation by negative examples
$m(chain, grandfather, father, father).$ $m(chain, grandfather, father, mother).$ $m(chain, grandfather, father, parent).$
Hypothesis construction
Top proram unfolding
$m(grandfather, A, B) : -m(father, A, C), m(father, C, B).$ $m(grandfather, A, B) : -m(father, A, C), m(mother, C, B).$ $m(grandfather, A, B) : -m(father, A, C), m(parent, C, B).$
Top proram reduction
$m(grandfather, A, B) : -m(father, A, C), m(parent, C, B)$
Reduction excapsulation
$grandfather(A, B) : -father(A, C), parent(C, B).$

Table 2: Top program construction and reduction for "grandfather".

2 Framework

Algorithm 1 Learning by Top program construction and reduction

Given: A MIL Problem, $\mathcal{T} = \{E, B, \mathcal{M}, \mathcal{H}\}$.
Return: A hypothesis, $H \in \mathcal{H}$.

1: Encapsulate $\{E, B, \mathcal{M}\}$ and expand $\mathcal{M}$.
2: Construct the Top prorgam for $\mathcal{T}$, $\mathcal{S}$.
3: Reduce $\mathcal{S}$ to $\mathcal{S}'$.
4: Excapsulate $\mathcal{S}'$ as H.
5: Return H

Louise's learning procedure is described in Algorithm 1. Construction of the Top program is defined in Algorithm 2. Reduction of the Top program is achieved by application of Gordon Plotkin's program reduction algorithm, defined in Plotkin's doctoral thesis, [4], as Theorem 3.3.1.2. Louise's learning procedure is illustrated in table 2 with an example for the "grandfather" relation.

A MIL problem $\mathcal{T} = \{E, B, \mathcal{M}, \mathcal{H}\}$ consists of ground definite clause examples, $E = \{E^+, E^-\}$, definite clause definitions of background knowledge predicates, B, second order definite clause *metarules*, $\mathcal{M}$, and the corresponding

Algorithm 2 Top program construction

Given: An encapsulated MIL problem, $\mathcal{T} = \{E^+, E^-, B, \mathcal{M}, \mathcal{H}\}$.
Return: A set $\mathcal{S}$ of metasubstitutions μ/M in all solutions of $\mathcal{T}$.

```
Generalise(E+, B, M) → S+
Specialise(E−, B, M, S+) → S−
Return S = S−

procedure Generalise(E+, B, M)
Initialise: Let S+ = ∅
    for e+ ∈ E+ do
        for M ∈ M do
            Set S+ = S+ ∪ {μ/M : μM ∪ B ∪ M ⊨ e+}
        end for
    end for
    Return S+
end procedure

procedure Specialise(E−, B, M, S+)
Initialise: Let S− = S+
    for μ/M ∈ S− do
        if ∃e− ∈ E−: μM ∪ B ∪ M ⊨ e− then
            Set S− = S− \ {μ/M}
        end if
    end for
    Return S−
end procedure
```

search space of definite datalog hypotheses, $\mathcal{H}$. Louise begins learning by *encapsulating* a MIL problem. Briefly, to encapsulate a literal means to replace it by a new literal of an encapsulation predicate (m, in Louise) whose arguments are the predicate symbol and arguments of the original literal. Encapsulation allows a compact representation of the Top program as a set of first order atoms representing *metasubstitutions* of the existentially quantified variables in a metarule. In algorithm 2 a metasubstitution μ of the existentially quantified variables of a metarule $M \in \mathcal{M}$ is represented as μ/M. The set of metasubstitutions in the Top program is unfolded to produce a set of definite clause instances of the metarules in $\mathcal{M}$ before program reduction. The reduced Top program is then *exacpsulated* to form the learned hypothesis. Excapsulation is the opposite process of encapsulation.

3 Experiments

We compare the performance of Louise to Thelma[3], our Prolog implementation of Metagol, on two kinship problems. The target theory for *blood_relative*/2

[3] https://github.com/stassa/thelma

consists of 16 clauses of the *Identity* metarule, $P(X,Y) \leftarrow Q(X,Y)$. Each clause maps one of 16 definitions of kinship relations to $blood\_relative/2$. The target theory for $relative/2$ cannot be expressed in the background knowledge and metarules we allow for the experiment, forcing the two systems to learn a larger hypothesis. We did not know what this larger hypothesis would be in advance of our experiments. We set a time limit of 2 hours for both experiments. Results of the experiments are listed in table 1. Both learners compress the target theory for $blood\_relative/2$ to 11 clauses. However, Thelma takes a significantly longer time to learn the shorter theory than Louise and times out on the longer theory. Louise learns both theories in under a minute.

4 Conclusions and future work

Unlike Metagol and Thelma, Louise is not yet capable of predicate invention because of the left-recursive nature of its representation of metarules. However, Louise can learn some programs for which predicate invention is normally required by performing episodic learning and unfolding metarules onto each other to produce *extended* metarules.

In future work we intend to implement predicate invention by use of a TP operator for bottom-up evaluation, to avoid issues with left-recursion in metarules. We also intend to test Louise on larger problems, in particular, machine vision problems.

References

1. Andrew Cropper and Stephen H. Muggleton. Metagol system, 2016.
2. Stephen Muggleton and Dianhuan Lin. Meta-Interpretive Learning of Higher-Order Dyadic Datalog : Predicate Invention Revisited. *Machine Learning*, 100(1):49–73, 2015.
3. Stephen H. Muggleton, Dianhuan Lin, Niels Pahlavi, and Alireza Tamaddoni-Nezhad. Meta-interpretive learning: Application to grammatical inference. *Machine Learning*, 94(1):25–49, 2014.
4. Gordon Plotkin. *Automatic Methods of Inductive Inference*. PhD thesis, The University of Edinburgh, 1972.