Inductive Programming
Lecture 2
Domain-Specific Languages and Background Knowledge

Stephen Muggleton
Department of Computing
Imperial College, London and
University of Nanjing

9th October, 2023

Papers for this lecture

Paper2.1: S.H. Muggleton, D. Lin, and A. Tamaddoni-Nezhad. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning, 100(1):49-73, 2015.

Paper2.2: A. Cropper and S.H. Muggleton. Learning higher-order logic programs through abstraction and invention. In Proceedings of the 25th International Joint Conference Artificial Intelligence (IJCAI 2016), pages 1418-1424. IJCAI, 2016.

Motivation

- Inductive Programming
- Simple programs
- Support repetitive tasks
- Few examples provided by human
- Weak learning bias implies many examples
- Strong learning bias requires few examples

Probably Approximately Correct (PAC) learnability model

PAC-learning (Valiant, 1984) Defines a class of polynomial-time learning algorithms which, when given sufficient training examples, have high Probability of choosing a hypothesis which is Approximately Correct on unseen examples.

Formal definition Polynomial-time learning algorithm A is PAC for hypothesis and example space $\mathcal{H}$ and $\mathcal{E}$ respectively iff $\forall$ prob bounds $\epsilon, \delta \in [0,1]$, hypothesis $H \in \mathcal{H}$, prob distribution $\mathcal{D}_{\mathcal{E}}$ and sample size m $\exists$ polynomial function p such that E randomly sampled from $\mathcal{D}_{\mathcal{E}}^m$ and $m < p(\frac{1}{\epsilon}, \frac{1}{\delta}, ln(\mathcal{H}))$ and $H = A(E)$ implies $Pr(\text{Error}(H, \mathcal{D}_{\mathcal{E}}) > \epsilon) < 1 - \delta$.

Blumer bound - Learning from few examples

PAC algorithm Assume PAC algorithm with $m = |E|$, $\mathcal{H}$, ϵ, δ.

Blumer bound (JACM, 1989) $m \geq \frac{(ln|\mathcal{H}| + ln\frac{1}{\delta})}{\epsilon}$

Significance of Blumer Ohm's Law of Machine Learning.

Blumer 1 m is $O(\frac{ln|\mathcal{H}|}{\epsilon})$

Blumer 2 ϵ is $O(\frac{ln|\mathcal{H}|}{m})$.

Learning Few examples requires $ln|\mathcal{H}|$ small.

Strong Bias in IP DSLs, Background knowledge, Meta-logical constraints.

Meta-Interpretive Learning (MIL)

MIL An Inductive Programming approach in which recursive logic programs can be induced incrementally from a small number of examples together with background predicates and metarules.

Formal definition Given input (B, M, E^+, E^-) where background B is a logic program, metarules M are higher-order clauses and examples E^+, E^- are ground atoms. An MIL algorithm returns a logic program hypothesis H such that $M \models H$ and $H \cup B \models E^+$ and $H \cup B \not\models E^-$.

Meta-interpreter (Paper2.1)

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

Metarules

Name	**Meta-Rule**	**Order**
PreCon	$P(x, y) \leftarrow Q(x), R(x, y)$	$P \succ Q, P \succ R$
PostCon	$P(x, y) \leftarrow Q(x, y), R(y)$	$P \succ Q, P \succ R$
Chain	$P(x, y) \leftarrow Q(x, z), R(z, y)$	$P \succ Q, P \succ R$
TailRec	$P(x, y) \leftarrow Q(x, z), P(z, y)$	$P \succ Q,$ $x \succ z \succ y$

H^2_2 hypothesis space

Hypothesis space H^2_2 definite clauses with at most two body atoms and at most predicate arity of two.

Size hypothesis space $\mathcal{H}$ is $O(|M|^n p^{3n})$ given M metarules, n clauses, p predicate symbols.

Log hypothesis space size $ln(|\mathcal{H}|) = n(ln(M) + 3ln(p))$.

Sample complexity (Blumer) For fixed M, p we have m is $O(\frac{n}{\epsilon})$.

Logical form of Metarules

General form

$$
\begin{aligned}
P(x,y) &\leftarrow Q(x,y)\\
P(x,y) &\leftarrow Q(x,z), R(z,y)
\end{aligned}
$$

Meta-rule general form is

$$\exists P, Q, ..\forall x, y, ..P(x, ..) \leftarrow Q(y, ..), ..$$

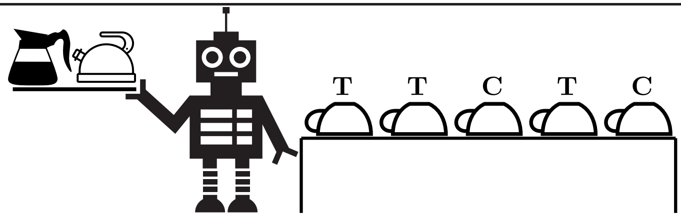
Supports predicate/object invention and recursion.

Hypothesis language is datalog logic programs in H^2_2, which contain predicates with arity at most 2 and has at most 2 atoms in the body.

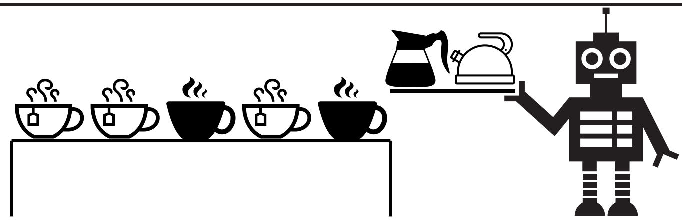
Metagol$_D$ implementation

- Ordered Herbrand Base [Knuth and Bendix, 1970; Yahya, Fernandez and Minker, 1994] - guarantees termination of derivations. Lexicographic + interval.
- Episodes - sequence of related learned concepts, reduces $\prod_i |H_i|$ to $\sum_i |H_i|$.
- Iterative deepening search $H_0, .., H_n$ returns $h_n \in H_n$ where n is number of clauses in h_n and n is minimal consistent hypothesis.
- Log-bounding (PAC result) - $log_2 n$ clause definition needs n examples.
- Github implementation - https://github.com/metagol/metagol.
- PHP interface - http://metagol.doc.ic.ac.uk.

Inductive Programming task
Robotic Waiter



Initial state



Final state

Metagol$_{\mathbf{D}}$ (Paper2.1)
First-order background knowledge
Recursive solution

```
f(A,B):-f3(A,B),at_end(B).
f(A,B):-f3(A,C),f(C,B).
f3(A,B):-f2(A,C),move_right(C,B).
f2(A,B):-turn_cup_over(A,C),f1(C,B).
f1(A,B):-wants_tea(A),pour_tea(A,B).
f1(A,B):-wants_coffee(A),pour_coffee(A,B).
```

Metagol$_{\mathbf{AI}}$ (Paper2.2)
Higher-order background knowledge
Abstraction and Invention solution

Shorter program

```
f(A,B):-until(A,B,at_end,f3).
f3(A,B):-f2(A,C),move_right(C,B).
f2(A,B):-turn_cup_over(A,C),f1(C,B).
f1(A,B):-ifthenelse(A,B,wants_tea, pour_tea, pour_coffee).
```

Alternation of Abstraction and Invention steps

	$\rightarrow$	**Abstract**	$\rightarrow$	**Invent**	$\rightarrow$	**Abstract**
f		until		f3,f2,f1		ifthenelse

Abstraction and Invention - Robot example

Higher-order definition
until(S1,S2,Cond,Do) $\leftarrow$ Cond(S1)
until(S1,S2,Cond,Do) $\leftarrow$ not(Cond(S1)), Do(S1,S2)

Abstraction
f(A,B) $\leftarrow$ until(A,B,at_end,f3)

Invention
f3(A,B) $\leftarrow$ f2(A,C),move_right(C,B)

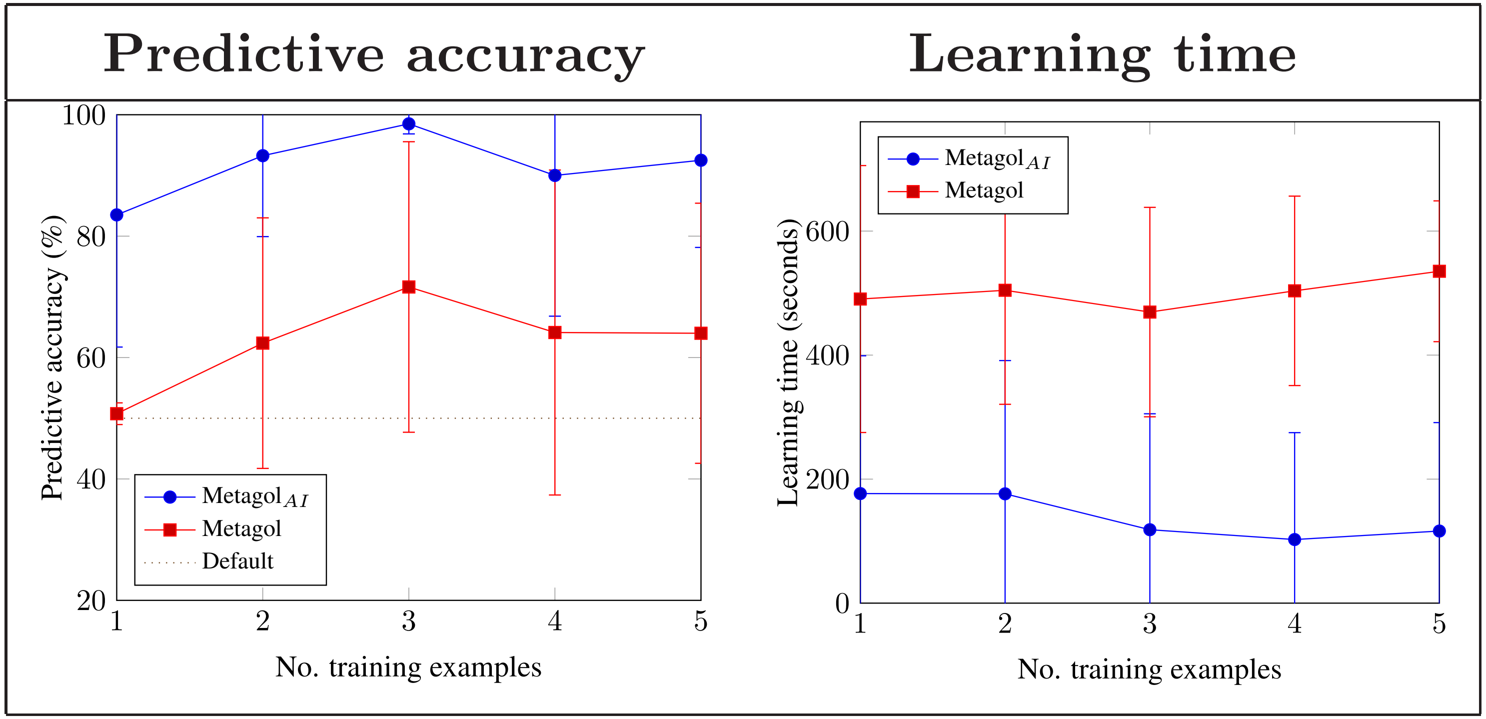
Metagol$_{AI}$ (Paper2.2)

https://github.com/metagol/metagol

Addition clause for meta-interpreter

```
prove_aux(Atom,H1,H2):-
    background((Atom:-Body)),
    prove(Body,H1,H2).
```

Results - Waiter



Blumer 2 ϵ is $O(\frac{n}{m})$

n is minimum consistent program size

Summary

- Inductive Programming - Complex programs, Few examples.
- Blumer bound - error decreases with log hypothesis space.
- Meta-Interpretive Learning and Metagol.
- First-order background knowledge - eg. move_right/2.
- Metarules - eg Chain.
- Second-order background knowledge - eg. until/4.
- Blumer bound - Abstraction and Invention decreased example requirement.