Hypothesis Space Processing for Efficient Rule Learning Through Inductive Logic Programming

David M. Cerna

August 31^{st}- September 5^{th} 2024

AITP-25

Czech Academy of Sciences

dynatrace

Inductive Logic Programming (ILP)

- **ILP?** symbolic machine learning.
- Introduced in the early 90s (Muggleton, 1991).
- **Goal:** Form an explanatory hypothesis using:
  1) Positive and negative **evidence**
  2) Provided **background knowledge**

Background Knowledge (BK)		Evidence
$\texttt{mother}(a,b).$	$\texttt{father}(g,b).$	$\texttt{grandparent}(a,d)^{+}.$
$\texttt{mother}(a,c).$	$\texttt{father}(g,c).$	$\texttt{grandparent}(g,d)^{+}.$
$\texttt{mother}(b,d).$	$\texttt{father}(f,d).$	$\texttt{grandparent}(a,h)^{+}.$
$\texttt{mother}(e,f).$	$\texttt{father}(c,h).$	$\texttt{grandparent}(g,h)^{+}.$
		$\texttt{grandparent}(a,e)^{-}.$

- $\texttt{mother}(a,b)$ denotes a is a mother of b.

Grandparent Example

- From *mother/2* and *father/2* we learn a **logic program** for

  X is a grandparent of Y

- One solution is

$$\texttt{grandparent}(X,Y)\text{:-}\texttt{mother}(X,Z),\texttt{mother}(Z,Y)$$
$$\texttt{grandparent}(X,Y)\text{:-}\texttt{mother}(X,Z),\texttt{father}(Z,Y)$$
$$\texttt{grandparent}(X,Y)\text{:-}\texttt{father}(X,Z),\texttt{mother}(Z,Y)$$
$$\texttt{grandparent}(X,Y)\text{:-}\texttt{father}(X,Z),\texttt{father}(Z,Y)$$

- A popular learning paradigm is Learning from Entailment:

$$BK, \mathbf{H} \models E^{+} \qquad BK, \mathbf{H} \not\models E^{-}$$

- **Goal:** Find **H**.

A "Better" Grandparent

BK	E^+	E^-
$\mathtt{mom}(a,b).\ \mathtt{dad}(e,b).$	$\mathrm{gp}(a,d).\ \mathrm{gp}(e,d).$	$\mathrm{gp}(a,b).\ \mathrm{gp}(b,c).$
$\mathtt{mom}(a,c).\ \mathtt{dad}(e,c).$	$\mathrm{gp}(a,f).\ \cancel{\mathrm{gp}(e,f).}$	$\mathrm{gp}(c,f).\ \mathrm{gp}(d,f).$
$\mathtt{mom}(b,d).\ \mathtt{dad}(c,f).$		

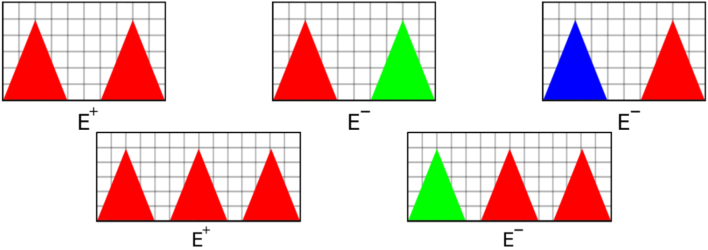
Learning is less brittle with the parent predicate

$$\mathrm{gp}(X,Y)\text{:-}\mathtt{mom}(X,C),\mathtt{mom}(C,Y)$$
$$\mathrm{gp}(X,Y)\text{:-}\mathtt{mom}(X,C),\mathtt{dad}(C,Y)$$
$$\mathrm{gp}(X,Y)\text{:-}\mathtt{dad}(X,C),\mathtt{mom}(C,Y)$$
$$\cancel{\mathrm{gp}(X,Y)\text{:-}\mathtt{dad}(X,C),\mathtt{dad}(C,Y)}$$

$$\mathrm{gp}(X,Y)\text{:-}\mathbf{p}(X,C),\mathbf{p}(C,Y)$$
$$\mathbf{p}(X,Y)\text{:-}\mathtt{mom}(X,Y)$$
$$\mathbf{p}(X,Y)\text{:-}\mathtt{dad}(X,Y)$$

- ISSUE: Searching through a larger hypothesis space.

Negation: Even Better Generalizaion



- Optimal Solution:

$$f(S) \text{ :- } scene(S), \textbf{not } inv_1(S).$$
$$inv_1(S) \text{ :- } cone(S, P), \textbf{not } red(P).$$

- There does not exists a cone in the scene that is not red.
- *Generalisation Through Negation and Predicate Invention* AAAI-24 (D. Cerna, A. Cropper)

Higher-order: Shorter and General Programs

- Higher-order definitions, larger space $\Rightarrow$ smaller programs:

$$\mathtt{map}(P, [], []).$$
$$\mathtt{map}([H1|T1], [H2|T2], P) \text{:-} P(H1, H2), \mathtt{map}(T1, T2).$$

- First-order dropLast:

$$\mathtt{dropLast}(A, B) \text{:-}\ \mathtt{empty}(A), \mathtt{empty}(B).$$
$$\mathtt{dropLast}(A, B) \text{:-}\ \mathtt{con}(A, B, C), \mathtt{reverse}(C, E), \mathtt{tail}(E, F), \mathtt{reverse}(F, G), \mathtt{con}(B, G, H), \mathtt{dropLast}(D, H).$$

- Higher-order dropLast:

$$\mathtt{dropLast}(A, B) \text{:-}\ \mathtt{map}(inv, A, B).$$
$$\mathtt{inv}(A, B) \text{:-}\ \mathtt{reverse}(A, C), \mathtt{tail}(C, D), \mathtt{reverse}(D, B).$$

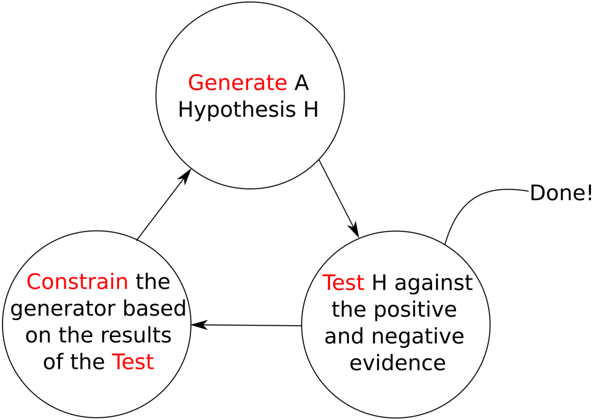
- *Learning Higher-Order Logic Programs From Failures*, IJCAI-22 (S. Purgal, D. Cerna, C. Kaliszyk)

Classical Approaches to Learning from Entailment

- **Top down:** (Foil,TILDE) overly general guess and try to specialize it.
  - Assume the empty program and add literals.
- **Bottom up:** (Progol, Aleph) overly specific guess and try to generalize it.
  - Assume much of the BK and remove/generalize literals.
- Unsuccessfully learning: Predicate Invention, Recursion, Higher-order definitions.
- Modern Approach: Meta-learning, i.e. encoding.
- Example meta-learners:
  - **MAXSYNTH** (Hocquette *et al., 2024*), **NOPI** (Cerna and Cropper, 2024), **Joiner**(Hocquette *et al., 2024*), **Disco** (Cropper and Hocquette, 2023), **Hopper** (Purgal, Cerna, and Kaliszyk, 2022), **Popper** (Cropper and Morel, 2021,2022), **Apperception Engine** (Evans *et al.*, 2021)
  - **δILP** (Evans and Grefenstette, 2018), **Metagol** (Muggleton *et al.*, 2015), **ILASP** (Law *et al.*, 2014)

Popper: Learning from Failures

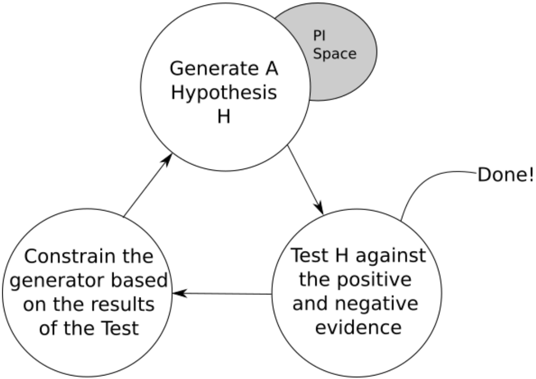
- Counterexample-guided synthesis:
- **Main Loop:**



- Uses a Multishot ASP solver.

Popper with PI

- The main difference is a larger search space.



- Larger search space.
- Full of redundancy. Efficient search is needed.

Pointless rules

- Literals in the body of rules are may imply each other.

$$r_1: \quad h(A) \text{ :- } odd(A), int(A).$$
$$r_2: \quad h(A) \text{ :- } odd(A).$$

- Observe, $odd(A) \Rightarrow int(A)$ and $int(A) \Rightarrow odd(A)$.
- Thus, keeping r_1 in the search space is pointless.
- Furthermore, **“specializations”** of r_1 cannot be optimal.
- Scare quotes: propositional subsumption relation.

Reducible Rules

Definition (**Captured literal**)

Let r be a rule, $l \in body(r)$, and $vars(l) \subseteq vars(body(r) \setminus \{l\}) \cup vars(head(r))$. Then l is *r-captured*.

Definition (**Reducible**)

Let r be a rule, B be BK, $l \in body(r)$ be r-captured, and $B \models r \leftrightarrow r \setminus \{l\}$. Then r is *reducible*.

- The generator can prune **"Specializations"** of reducible rules.
- Can we relax the logical equivalence?

Pointless rules

- No implications present in the following rule:

$$
\begin{aligned}
r_1 : \quad & h(A) \text{ :- } odd(A), lt(A, 10). \\
r_2 : \quad & h(A) \text{ :- } odd(A). \\
E^- \ &= \ \{h(1), h(2), h(3)\}
\end{aligned}
$$

- Observe, $lt(A, 10)$ accepts all of E^-, i.e. r_1 and r_2 accept the same members of E^-.
- Thus, keeping r_1 in the search space is pointless.
- Furthermore, **"specializations"** of r_1 cannot be optimal.
- Such rules are referred to as indiscriminate.

Indiscriminate Soundness

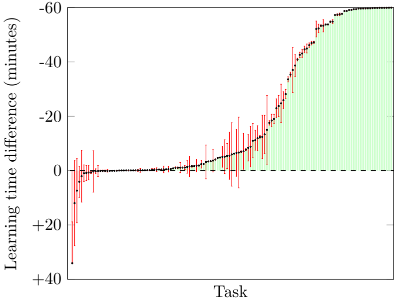
Definition (Indiscriminate)

Let r be a rule, B be BK, E^- be negative examples with the same predicate symbol as the head of r, $l \in body(r)$ be r-captured, and for all $e \in E^-$, $B \models (r \rightarrow e) \leftrightarrow (r \setminus \{l\} \rightarrow e)$. Then r is *indiscriminate*.

Proposition (Indiscriminate soundness)

Let B be BK, E^- be negative examples, h_1 be a hypothesis, r_1 be a **basic rule** *in h_1, $h_2 \subseteq h_1$, $r_2 \in h_2$, $r_2 \subseteq r_1$, and r_2 be indiscriminate with respect to B and E^-. Then h_1 is not optimal.*

Improvements and code



Algorithm 2: Finding pointless rules.

```
def pointless(h, neg, bk):
  for rule in h:
    if not basic(rule, h):
      continue
    head, body = rule
    for literal in body:
      body′ = body-literal
      if not captured(head, body′, literal):
        continue
      if reducible(bk, body′, literal):
        return true
      if indiscriminate(bk, neg, rule, head, body′):
        return true
  return false

def reducible(bk, neg, body′, literal):
  rule′ = (⊥, body′ ∪ {¬literal})
  return unsat(bk, rule′)

def indiscriminate(bk, neg, rule, head, body′):
  rule′ = (head, body′)
  s1 = neg_covered(bk, neg, rule)
  s2 = neg_covered(bk, neg, rule′)
  return s1 == s2
```

- Reducible and Indiscriminate rules are found during search.
- **Datasets:** 1D-arc, IMDB, Zendo, IGGP.
- Can we add preprocessing to the **generator**?

Preprocessing: Recall Reducible rules

- No implications present in the following rule:

$$
\begin{aligned}
BK &= \{edge(a,b), edge(b,c), edge(c,a)\} \\
r_1: \quad & h \text{ :- } edge(A,B), edge(B,C), edge(C,D), edge(D,E). \\
r_2: \quad & h \text{ :- } edge(A,B), edge(B,C), edge(C,A). \\
\theta &= \{D \mapsto A, E \mapsto B\}
\end{aligned}
$$

- Observe, that $r_1 \preceq_\theta r_2$ as $r_1\theta = r_2$, $|r_1| > |r_2|$, and $B \models r_1 \rightarrow r_2$ follows from $r_1 \preceq_\theta r_2$.
- Furthermore, $B \models r_2 \rightarrow r_1$ follows from *edge/2* being a bijective mapping from $\{a, b, c\}$ to itself.
- That is **"specializations"** of r_2 by adding an edge literal cannot be optimal.
- Such rules are referred to as Recall Reducible.

Beyond Definite Clause Subsumption

Definition (**Recall reducible**)

Let B be BK, r_1 be a rule, $r_1 \preceq_\theta r_2$, $|r_1| > |r_2|$, and $B \models r_1 \leftrightarrow r_2$. Then r_1 is recall reducible.

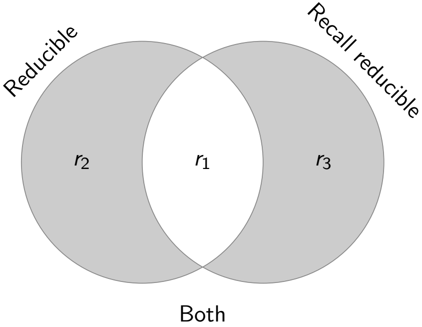
Proposition (**Recall specialisation**)

Let B be BK, r_1 be recall reducible, and $r_1 \subseteq r_2$. Then r_2 is recall reducible.

Proof.

We show that recall reducibility is equivalent to recognizing pigeonhole arguments in the definitions of the BK. □

Comparision to Reducible/Indiscriminate



r_1 : h :- $leq(B, C), succ(A, B), succ(A, C).$

r_2 : h :- $nat(A), succ(A, B).$

r_3 : h :- $succ(A, B), succ(A, C), odd(B), prime(C).$

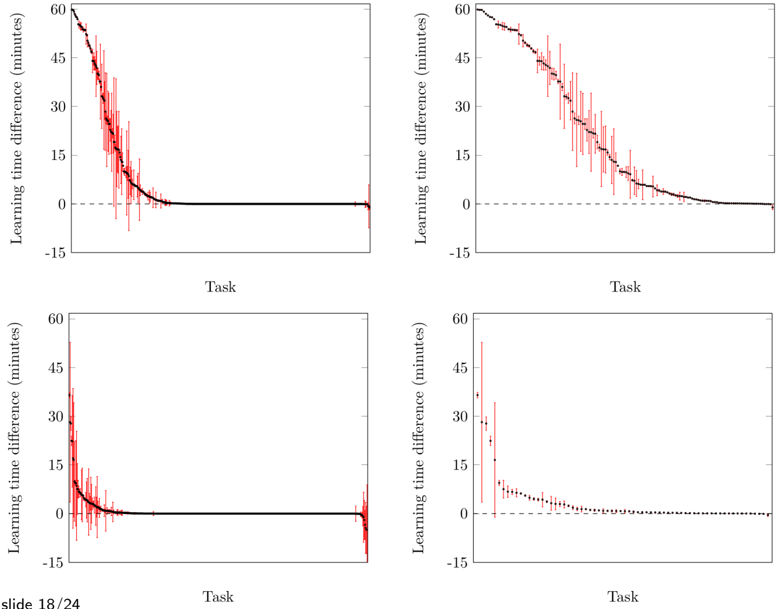
r_4 : h :- $leq(B, B), succ(A, B).$

r_5 : h :- $succ(A, B), succ(A, C).$

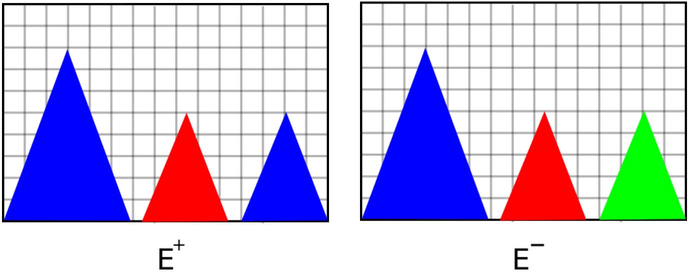
r_6 : h :- $succ(A, B).$

r_7 : h :- $succ(A, C), odd(C), prime(C).$

Results Preprocessing: Reducible and Recall (10 seconds)



Symmetries Breaking for ILP



$$r_1 = zendo(A) \leftarrow piece(A,B),\ size(B,C),\ blue(B),\ small(C)$$

$$r_2 = zendo(A) \leftarrow piece(A,C),\ size(C,B),\ blue(C),\ small(B)$$

- r_1 and r_2 are **variants**.
- How to quickly (and efficiently) recognize this?

Precise Problem

Definition (**Hypothesis space reduction problem**)

Given an ILP input $(E, B, \mathcal{H})$, the *hypothesis space reduction problem* is to find $\mathcal{H}' \subseteq \mathcal{H}$ such that if $\mathcal{H}$ contains an optimal hypothesis then there exists an optimal hypothesis $h \in \mathcal{H}'$.

- For example, removing body variants.

Definition (**Body variant**)

A rule r' is a *body-variant* of a rule r if there exists a bijective renaming σ from $body\_vars(r)$ to $body\_vars(r')$ such that $r\sigma = r'$.

Proposition (**Body-variant hardness**)

The body-variant problem is GI-hard.

- When can we solve the variant problem efficiently?

Ordering Variable Arguments

Definition (**Witnessed**)

Let r be a rule, v be a variable, $l_1 \in body(r)$ such that $v \in skipped(l_1)$, and $l_2 \in body(r)$ such that $v \in vars(l_2)$ and $l_2 <_{lex} l_1$. Then we say that l_1 is *v-witnessed* in r.

- In $p(A, C, E)$ the variables B and D are skipped.
- In r_1, $p(A, E)$ is not C-witnessed.
- In r_2, $p(B, D)$ is C-witnessed and $p(C, E)$ is E-witnessed.

$$r_1 : \quad h(A, B) \text{ :- } p(A, E), p(B, C), p(C, D).$$
$$r_2 : \quad h(A, B) \text{ :- } p(A, C), p(B, D), p(C, E).$$

Definition (**Safe variable**)

Let r be a rule and $v \in body\_vars(r)$ such that for all $l \in body(r)$, where $v \in skipped_k(l)$, l is v-witnessed in r. Then v is *safe*.

Soundness modulo Variants

Proposition (Soundness)

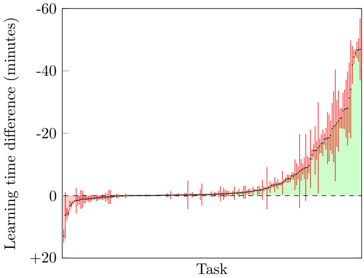
For every rule r there exists a rule r' such that r' is a body-variant of r and all variables in r' are safe.

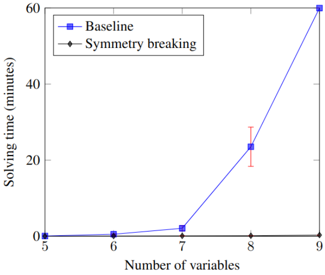
Proof.

Proof by induction. Selecting the smallest unsafe variable x and construct a substitution that, when applied to r, results in a rule where the smallest unsafe variable is larger than x. □

$$
\begin{aligned}
r_1 :\quad & h(A, B) \text{ :- } p(A, E), p(B, C), p(\mathbf{C}, D). \\
& \quad \sigma_1 = \{E \mapsto C, C \mapsto F\}\{F \mapsto E, E \mapsto D\} \\
r_2 :\quad & h(A, B) \text{ :- } p(A, C), p(B, E), p(E, \mathbf{D}). \\
& \quad \sigma_2 = \{E \mapsto D, D \mapsto F\}\{F \mapsto E, E \mapsto D\} \\
r_3 :\quad & h(A, B) \text{ :- } p(A, C), p(B, D), p(D, E).
\end{aligned}
$$

Experiments





- (right) Tested on a single hard tasks from *trains* dataset.

Future work

- **Simple:**
  - put all this work together and find more optimizations.
  - Apply to predicate invention.
- Papers can be found on ArXiv:
  - *Efficient Rule Induction by Ignoring Pointless Rules* (A. Cropper, D. M. Cerna)
  - *Honey, I Shrunk the Hypothesis Space (Through Logical Preprocessing)* (A. Cropper, F. Gouveia, D. M. Cerna)
  - *Symmetry Breaking for Inductive Logic Programming* (A. Cropper, D. M. Cerna, M. Järvisalo)