Inductive logic programming

On ILP with $\perp$

Simon Jacquet

Faculty of Computer Science
Unamur

October 27th 2021

ILP approach

$$B \wedge H \models E \quad (1)$$
$$B \wedge \overline{E} \models \overline{H} \quad (2)$$
$$B \wedge \overline{E} \models \overline{\bot} \quad (3)$$
$$\overline{\bot} \models \overline{H} \quad (4)$$
$$H \models \bot \quad (5)$$

- From (3), we build $\overline{\bot}$
- From (5), we build H

Subsumption

Atom subsumption

Let θ be a substitution of the form $\{v_1/t_1, ..., v_n/t_n\}$.
Let F be an arbitrary atom.
$F\theta$ is the atom F where each of its variable v_i have been replaced by t_i.

Clause subsumption

Let C and D be clauses.
C subsumes D iff $\exists\theta = \{v_1/t_1, ..., v_n/t_n\}, C\theta \subseteq D$
We note $C \preceq D$
It is worth noting that if $C \preceq D$, then $C \rightarrow D$

Subsumption example

A	All men are mortal	mortal(X)
C	Socrates is mortal	mortal(socrates)

- Let $\theta = \{X/\text{socrates}\}$

$$\begin{array}{rccc} & A\theta & = & C \\ \Leftrightarrow & A & \preceq & C \end{array}$$

Computing $\perp$ (i)

$$
\begin{aligned}
B \wedge H &\models E &(1)\\
B \wedge \overline{E} &\models \overline{H} &(2)
\end{aligned}
$$

- We suppose H and E to be single Horn clauses

$$
\begin{aligned}
E &\equiv A \leftarrow B_1, \ldots, B_n\\
&\equiv A \vee \neg B_1 \vee \ldots \vee \neg B_n\\
\overline{E} &\equiv \neg E\\
&\equiv \neg A \wedge B_1 \wedge \ldots \wedge B_n\\
&\equiv \leftarrow A, \quad B_1 \leftarrow, \quad \ldots, \quad B_n \leftarrow
\end{aligned}
$$

- The negation of a Horn clause is a set of ground unit clauses

Negation of a Horn clause

$$
\begin{array}{rcll}
E & \equiv & \text{mortal}(X) \leftarrow \text{man}(X) & \forall X \\
 & \equiv & \text{mortal}(X) \vee \neg\text{man}(X) & \forall X \\
\overline{E} & \equiv & \neg E & \\
 & \equiv & \neg\text{mortal}(X) \wedge \text{man}(X) & \exists X \\
 & \equiv & \neg\text{mortal}(cst) \wedge \text{man}(cst) &
\end{array}
$$

- The negation of a Horn clause is a set of unit clauses
- The unit clauses do not have variables $\Rightarrow$ they are ground

Computing $\bot$ (ii)

$$
\begin{aligned}
B \wedge H &\models E & (1)\\
B \wedge \overline{E} &\models \overline{H} & (2)\\
B \wedge \overline{E} &\models \overline{\bot} & (3)\\
\overline{\bot} &\models \overline{H} & (4)\\
H &\models \bot & (5)
\end{aligned}
$$

- $\overline{\bot}$ is the conjunction of ground literals which are true in all models $B \wedge \overline{E}$
  - ☞ the smallest Herbrand model of $B \wedge \overline{E}$
- $\bot = \neg\overline{\bot}$
- H can be found by considering the clauses that θ-subsume $\bot$ (5)

Model

Model

In first order logic, a model M consists of:

- D: universe
- For each n-ary relation constant p an n-ary relation p^D over D
- For each n-ary function constant f an n-ary function f^D over D
- For each object constant c an element c^D from D

Herbrand model

Ground

A term or atom is ground iff it has no variable

Herbrand universe

The Herbrand universe, noted U_H is the set of all ground terms

Herbrand base

The Herbrand base, noted B_H is the set of all ground atoms

Herbrand model

Model in which the universe is a Herbrand universe

Computing $\bot$: example 1

$$B \wedge \overline{E} \quad \models \quad \overline{\bot} \tag{3}$$

$$\begin{array}{lll}
B & \text{animal}(X) \leftarrow \text{pet}(X). & \forall X \\
 & \text{pet}(X) \leftarrow \text{dog}(X). & \forall X \\
E & \text{nice}(X) \leftarrow \text{dog}(X). & \forall X
\end{array}$$

Computing $\bot$: example 1

$$B \wedge \overline{E} \quad \models \quad \overline{\bot} \tag{3}$$

$$\begin{array}{lll}
B & \text{animal}(X) \leftarrow \text{pet}(X). & \forall X \\
 & \text{pet}(X) \leftarrow \text{dog}(X). & \forall X \\
E & \text{nice}(X) \leftarrow \text{dog}(X). & \forall X \\
\overline{E} & \neg\text{nice}(X) \wedge \text{dog}(X). & \exists X
\end{array}$$

Computing $\bot$: example 1

$$B \wedge \overline{E} \quad \models \quad \overline{\bot} \tag{3}$$

B	$\text{animal}(X) \leftarrow \text{pet}(X).$	$\forall X$
	$\text{pet}(X) \leftarrow \text{dog}(X).$	$\forall X$
E	$\text{nice}(X) \leftarrow \text{dog}(X).$	$\forall X$
$\overline{E}$	$\neg\text{nice}(X) \wedge \text{dog}(X).$	$\exists X$
$\overline{\bot}$	$\neg\text{nice}(X) \wedge \text{dog}(X) \wedge \text{pet}(X) \wedge \text{animal}(X).$	$\exists X$

Computing $\bot$: example 1

$$B \wedge \overline{E} \quad \models \quad \overline{\bot} \tag{3}$$

B	$\text{animal}(X) \leftarrow \text{pet}(X).$	$\forall X$
	$\text{pet}(X) \leftarrow \text{dog}(X).$	$\forall X$
E	$\text{nice}(X) \leftarrow \text{dog}(X).$	$\forall X$
$\overline{E}$	$\neg\text{nice}(X) \wedge \text{dog}(X).$	$\exists X$
$\overline{\bot}$	$\neg\text{nice}(X) \wedge \text{dog}(X) \wedge \text{pet}(X) \wedge \text{animal}(X).$	$\exists X$
$\bot$	$\text{nice}(X) \vee \neg\text{dog}(X) \vee \neg\text{pet}(X) \vee \neg\text{animal}(X).$	$\forall X$

Computing $\bot$: example 1

$$B \wedge \overline{E} \quad \models \quad \overline{\bot} \tag{3}$$

$$\begin{array}{lll}
B & \text{animal}(X) \leftarrow \text{pet}(X). & \forall X \\
 & \text{pet}(X) \leftarrow \text{dog}(X). & \forall X \\
E & \text{nice}(X) \leftarrow \text{dog}(X). & \forall X \\
\overline{E} & \neg\text{nice}(X) \wedge \text{dog}(X). & \exists X \\
\overline{\bot} & \neg\text{nice}(X) \wedge \text{dog}(X) \wedge \text{pet}(X) \wedge \text{animal}(X). & \exists X \\
\bot & \text{nice}(X) \vee \neg\text{dog}(X) \vee \neg\text{pet}(X) \vee \neg\text{animal}(X). & \forall X \\
 & \text{nice}(X) \leftarrow \text{dog}(X), \text{pet}(X), \text{animal}(X). & \forall X
\end{array}$$

Computing $\perp$: example 2

$$B \wedge \overline{E} \quad \models \quad \overline{\perp} \tag{3}$$

B white($swan1$).

E $\leftarrow$ black($swan1$).

Computing $\perp$: example 2

$$B \wedge \overline{E} \quad \models \quad \overline{\perp} \tag{3}$$

B white($swan$1).

E $\leftarrow$ black($swan$1).

$\neg$black($swan$1).

Computing $\bot$: example 2

$$B \wedge \overline{E} \quad \models \quad \overline{\bot} \tag{3}$$

B white($swan$1).

E $\leftarrow$ black($swan$1).

$\neg$black($swan$1).

$\overline{E}$ black($swan$1).

Computing $\perp$: example 2

$$B \wedge \overline{E} \quad \models \quad \overline{\perp} \tag{3}$$

B white($swan$1).

E $\leftarrow$ black($swan$1).

$\neg$black($swan$1).

$\overline{E}$ black($swan$1).

$\overline{\perp}$ black($swan$1), white($swan$1).

UNIVERSITÉ DE NAMUR

Computing $\bot$: example 2

$$B \wedge \overline{E} \models \overline{\bot} \tag{3}$$

B white($swan1$).

E $\leftarrow$ black($swan1$).

$\neg$black($swan1$).

$\overline{E}$ black($swan1$).

$\overline{\bot}$ black($swan1$), white($swan1$).

$\bot$ $\leftarrow$ black($swan1$), white($swan1$).

$\perp_i$: introduction

- In general, $\perp$ has infinite cardinality
- To deal with this, we introduce $\perp_i$
  - $\perp_i \preceq \perp$
    - ☞ property we impose on $\perp_i$
  - $\Box \preceq H \preceq \perp_i \preceq \perp$
    - ☞ when searching for H, we will limit ourselves to those who respect this equation
- To ensure we have a valid $\perp_i$, we will use equation 3 under another form

$$B \wedge \overline{E} \models \overline{\perp} \tag{3}$$

$$B \wedge \overline{E} \wedge \perp \models \Box \tag{3a}$$

$$B \wedge \overline{E} \wedge \perp_i \models \Box \tag{3b}$$

Modes (i)

Mode declaration

- modeh(n,atom) | modeb(n,atom)
- n: $* \mid \geq 1$, called the recall
- atom, a ground atom,
- terms in this atom are either (i) constants, (ii) a function of terms, (iii) +type, -type or #type (type is a constant)
- modeh(*,f(+int,-int)), modeb(*,d(+int,-int)) are modes

Instantiation

Let m be a mode declaration, $a(m)$ is the atom of m with place-markers (iii) replaced by distinct variables.

$$m = \text{modeh}(*, f(+\text{int}, -\text{int}))$$
$$a(m) = f(A, B)$$

Modes (ii)

$C \in \mathcal{L}(M)$

- Let $C = h \leftarrow b_1, \ldots, b_n$, a definite clause
- Let M be a definite mode language
- $C \in \mathcal{L}$ iff
  1. h (resp. b_i) is the atom of a modeh (resp. modeb) declaration, with $+$type and -type replaced by variables, and $\#$type replaced by a ground term
  2. every variable $+$type in atom b_i corresponds to either (i) a variable $+$type in h or (ii) a variable -type in b_j, $1 \leq j < i$

Example

With $M = \begin{array}{ll} modeh(*, f(+int, -int)) & modeb(*, d(+int, -int)) \\ modeb(*, f(+int, -int)) & modeb(*, m(+int, +int, -int)) \end{array}$

The clause $f(A, B) \leftarrow d(A, C), f(C, D), m(A, D, B)$ is in $\mathcal{L}(M)$.

$\bot_i \in \mathcal{L}_i(M)$

Depth $d(v)$

$$d(v) = \begin{cases} 0, & \text{if } v \text{is in the head of } C \\ (\max_{u \in U_v} d(u)) + 1, & \text{otherwise} \end{cases}$$

where U_v are the variables in atoms in the body of C containing v.

$C \in \mathcal{L}_i(M)$

- $C \in \mathcal{L}(M)$
- All variables in C have depth $\leq i$

- $\bot$ is the most-specific definite clause such that $B \land \bot \land \overline{e} \vdash_h \Box$
- $\bot_i$ is the most-specific definite clause in $\mathcal{L}_i(M)$ such that $\bot_i \preceq \bot$

Algorithm to construct $\perp_i$ (i)

Prior knowledge

- h, a natural number: depth bound on deduction
- i, a natural number: variable distance
- B, a set of Horn clauses: background knowledge
- e, a definite clause: single sample from examples
- M, a set of mode declarations

Initialisation

- hash, a hash function: Terms $\rightarrow N$
- $\overline{e}$, a logic program: $\overline{a} \wedge b_1 \wedge ... \wedge b_n$
- $\perp_i$, a logic program
- InTerms, a set of terms
- k, a natural number

Algorithm to construct $\perp_i$ (ii)

Algorithm 1: Construct $\perp_i$ - Part 1

```
Get m ∈ M, modeh such that a(m) ⪯ a with substitution θ_h
if ∄m then
    return □
a_h ← a(m)
for v/t ∈ θ_h do
    if v corresponds to #type in m then
        Replace v by t in a_h
    else
        Replace v by v_k in a_h, with k = hash(t)
    if v corresponds to +type in m then
        Add v to InTerms
Add a_h to ⊥_i
```

Algorithm to construct $\perp_i$ (iii)

Algorithm 2: Construct $\perp_i$ - Part 2

for $k \leftarrow 1, \ldots, i$ **do**
- **forall** *modeb* $m \in M$ **do**
  - Let $\{v_1, \ldots, v_n\}$ be the variables corresponding to $+$type in $a(m)$
  - Let T_i be the set of all terms of the type associated with v_i in m
  - $T(m) \leftarrow T_1 \times \ldots \times T_n$
  - **forall** $\langle t_1, \ldots, t_n \rangle \in T(m)$ **do**
    - $a_b \leftarrow a(m)$
    - $\theta \leftarrow \{v_1/t_1, \ldots, v_n/t_n\}$
    - **if** *Prolog succeeds on goal* $a_b\theta$ **then**
      - Let Θ_b be the set of answer substitutions
      - **forall** $\theta_b \in \Theta_b$ **do**
        - **forall** $v/t \in \theta_b$ **do**
          - **if** v *corresponds to* $\#$*type in* m **then**
            - Replace v by t in a_b
          - **else**
            - Replace v by v_k in a_b, with $k = \text{hash}(t)$
          - **if** v *corresponds to -type* **then**
            - Add v to InTerms
- Add $\overline{a_b}$ to $\perp_i$

return $\perp_i$

Algorithm for search $\square \preceq C \preceq \bot_i$

Algorithm 3: Construct C

input: $h, B, e, \bot_i$

Open $\leftarrow \{\langle \square, \emptyset, 1 \rangle\}$, Closed $\leftarrow \emptyset$

while *True* **do**

  $s \leftarrow$ best(Open)

  Open $\leftarrow$ Open $-\{s\}$, Closed $\leftarrow$ Closed $\cup \{s\}$

  if $\neg$ *prune(s)* **then**

    Open $\leftarrow$ (Open $\cup\ \rho(s)$) $-$ Closed

  if *terminated(Closed,Open)* **then**

    return best(Closed)

  if *Open* $= \emptyset$ **then**

    print 'no compression'

    return $\langle e, \emptyset, 1 \rangle$

- ρ is the refinement operator

Cover set algorithm

Algorithm 4: Cover set algorithm

input: h, i, B, M, E

forall $e \in E$ **do**

  Construct $\perp_i$ for e using Algorithms 1 and 2

  Construct state s from $\perp_i$ using Algorithm 3

  Let C' be the unflattening of $C(s)$

  $B \leftarrow B \cup C'$

  $E \leftarrow E - \{e : e \in E,\ B \wedge \overline{e} \vdash_h \emptyset\}$