B4M36SMU

Inductive Logic Programming
Learning from Entailment

Monday 24th April, 2017

ILP Settings

- last time – learning from interpretations
  - $o \models \gamma \; \forall o \in O^+$
  - $o \not\models \gamma \; \forall o \in O^-$
- this lab – learning from entailment
  - $T \vdash o \; \forall o \in O^+$
  - $T \nvdash o \; \forall o \in O^-$

θ-Subsumption

$\gamma_1 \leq_\theta \gamma_2$, read as γ_1 *θ-subsumes* γ_2, if and only if there exists a substitution θ such that $\gamma_1\theta \subseteq \gamma_2$. Then, γ_1 is a generalization of γ_2 and γ_2 is a specialization of γ_1 under θ-subsumption.

Which $\leq_\theta$ relations hold within following pairs:

- $\gamma_1 = p(X, X)$
  $\gamma_2 = p(Y, Z)$
- $\gamma_1 = p(X, a)$
  $\gamma_2 = p(b, Y)$
- $\gamma_1 = \neg p(W, X) \vee \neg p(X, Y) \vee \neg p(Y, Z)$
  $\gamma_2 = \neg p(T, T)$

θ-Subsumption and $\vdash$

Which relations, $\gamma_1 \leq_\theta \gamma_2$ and $\gamma_1 \vdash \gamma_2$, hold within following pairs:

- $\gamma_1 = apple(X) \Longleftarrow fruit(X), round(X)$
  $\gamma_2 = apple(X) \Longleftarrow fruit(X), round(X), red(X)$

- $\gamma_1 = p(succ(X)) \Longleftarrow p(X)$
  $\gamma_2 = p(succ(succ(X))) \Longleftarrow p(X)$

Reduced Clause

Two clauses γ_1 and γ_2 are logically equivalent, $\gamma_1 \sim \gamma_2$, if $\gamma_1 \leq_\theta \gamma_2$ and $\gamma_2 \leq_\theta \gamma_1$.

A clause γ is reduced if there is no $\gamma' \sim \gamma$ such that $|\gamma'| < |\gamma|$.

Compute reduced clauses for following examples:

- $\neg p(X, Y, W) \vee \neg m(Y, Z, Q) \vee \neg p(X, U, Q) \vee \neg m(U, V, W)$
- $\neg p(X, Y) \vee \neg m(Z, Y) \vee \neg m(Y, Y) \vee \neg r(Z, Y)$

Least General Generalization

$LGG(\gamma_1, \gamma_2) = \gamma$

$\gamma \leq_\theta \gamma_1$

$\gamma \leq_\theta \gamma_2$

and for any γ' s.t. $\gamma' \leq_\theta \gamma_1$, $\gamma' \leq_\theta \gamma_2$ holds that

$\gamma' \leq_\theta \gamma$

$$LGG(\gamma_1, \gamma_2) = \bigvee_{l_1 \in \gamma_1, l_2 \in \gamma_2} LGG(l_1, l_2)$$

$$LGG(p(t_1, t_2, \dots), p(t'_1, t'_2, \dots)) = p(LGG(t_1, t'_1), LGG(t_2, t'_2), \dots)$$

$$LGG(f(t_1, \dots), f'(t'_1, \dots)) = \begin{cases} X & f \neq f' \\ f(LGG(t_1, t'_1), \dots) & otherwise \end{cases}$$

$$LGG(f(\dots), X) = X'$$

$$LGG(X, Y) = X'$$

where p is a predicate, f is a functor, t is a term, c is a constant and X is a variable.

Compute LGG

- $\gamma_1 = goal(X) \Longleftarrow path(A, B), path(B, X), gold(an(A), f(2), X)$
  $\gamma_2 = goal(a) \Longleftarrow path(a, B), gold(poss(B, C), f(2), a)$

- $\gamma_1 = \neg e(A, B) \vee \neg e(B, A)$
  $\gamma_2 = \neg e(A, B) \vee \neg e(B, C) \vee \neg e(C, A)$

Assignment 3

Submission

- brute system
- deadline: 9 May 2017, 23:59
- source codes – Python implementation (3.5) – and a PDF report (in one archive)
- 12 points can be obtained
- mandatory and optional part (see next slide)

Task and Scoring – mandatory part (6 points)

- implement LGG algorithm into *lggAgent.py*
- the reduction step does not have to be implemented

Assignment 3, cont'd

Task and Scoring – optional part (6 points)

- use learned clause to classify observations in the form of interpretations – implement *classifier.py* (2 points)
- a) construct your own set of clauses and their lgg such that the final lgg is a reduced clause (1 point)
- b) construct your own set of clauses and their lgg such that the final lgg is not a reduced clause (1 point)
- c) find reduced clause of the clause from b) (1 point)
- provide mapping for variable-tuple in a) and b), e.g. by subscripts or tables
- For a) and b), use starting set of clause with at least two clauses. If at least one of these sets has more than three clauses, then you will obtain another 1 point.
- write all of the clauses from a-c to your PDF report
- do not use examples from the textbook or labs
- do not use clauses which have trivial lgg, e.g. empty clause