Artificial Intelligence

8. Inductive Logic Programming

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute of Economics and Information Systems
& Institute of Computer Science
University of Hildesheim
http://www.ismll.uni-hildesheim.de

Inductive Logic Programming (ILP)

Given some **positive examples** for a **target predicate** P, say

$$\text{daughter}(\text{mary}, \text{ann})$$
$$\text{daughter}(\text{eve}, \text{tom})$$

and some **negative examples**

$$\neg\text{daughter}(\text{tom}, \text{ann})$$
$$\neg\text{daughter}(\text{eve}, \text{ann})$$

as well as some **descriptive predicates** Q of the entities envolved

$$\text{female}(\text{ann})$$
$$\text{female}(\text{eve})$$
$$\text{parent}(\text{ann}, \text{mary})$$
$$\text{parent}(\text{tom}, \text{eve})$$

find a **hypothesis definition / rule** of P in terms of Q that

1. covers all the positive examples,
2. does not cover any negative example, and
3. is sufficient general.

Trivial Solutions

$$\text{daughter}(X, Y)$$

covers all positive examples,
but unfortunately also all negative examples.

$$\text{false}$$

covers no negative example,
but unfortunately also no positive example.

$$(X = \text{mary} \wedge Y = \text{ann}) \vee (X = \text{eve} \wedge Y = \text{tom}) \rightarrow \text{daughter}(X, Y)$$

covers all positive examples,
covers no negative example,
but unfortunately does not generalize (new examples will fail).

Two principal approaches:

- top-down: generalization of decision trees (FOIL).
- inverse deduction (inverse resolution).

1. Inductive Logic Programming

2. FOIL

3. Inverse Resolution

FOIL

First Order Inductive Learner (FOIL; Quinlan 1990).

Idea:

- iteratively build rules that cover
  - some positive examples,
  - but no negative ones.

  Once a rule has been found, remove the positive examples covered and proceed.

- to build a rule:
  - add literals to the body until no negative example is covered
  - if literals introduce new variables,
    extend example tuples by all possible constants.

FOIL / Algorithm (1/2)

Algorithm 4.1 (FOIL – the covering algorithm)

Initialize $\mathcal{E}_{cur} := \mathcal{E}$.
Initialize $\mathcal{H} := \emptyset$.
repeat {covering}
  Initialize clause $c := T \leftarrow$.
  Call the $SpecializationAlgorithm(c, \mathcal{E}_{cur})$
    to find a clause c_{best}.
  Assign $c := c_{best}$.
  Post-process c by removing irrelevant literals to get c'.
  Add c' to $\mathcal{H}$ to get a new hypothesis $\mathcal{H}' := \mathcal{H} \cup \{c'\}$.
  Remove positive examples covered by c' from $\mathcal{E}_{cur}$ to get
    a new training set $\mathcal{E}'_{cur} := \mathcal{E}_{cur} - covers_{ext}(\mathcal{B}, \{c'\}, \mathcal{E}^+_{cur})$.
  Assign $\mathcal{E}_{cur} := \mathcal{E}'_{cur}, \mathcal{H} := \mathcal{H}'$.
until $\mathcal{E}^+_{cur} = \emptyset$ or encoding constraints violated.

Output: Hypothesis $\mathcal{H}$.

[Lavrac/Dzeroski 1994]

FOIL / Algorithm (2/2)

Algorithm 4.2 (FOIL – the specialization algorithm)

Initialize local training set $\mathcal{E}_i := \mathcal{E}_{cur}$.
Initialize current clause $c_i := c$.
Initialize $i := 1$.
while $\mathcal{E}_i^- \neq \emptyset$ or *encoding constraints violated* **do**
  Find the best literal L_i to add to the body of $c_i = T \leftarrow Q$
    and construct $c_{i+1} := T \leftarrow Q, L_i$.
  Form a new local training set $\mathcal{E}_{i+1}$ as a set of extensions of
    the tuples in $\mathcal{E}_i$ that satisfy L_i.
  Assign $c := c_{i+1}$.
  Increment i.
endwhile

Output: Clause c.

[Lavrac/Dzeroski 1994]

FOIL / Example

Current clause c_1 : $daughter(X,Y) \leftarrow$					
$\mathcal{E}_1$	$(mary, ann)$	$\oplus$		$n_1^{\oplus} = 2$	$I(c_1) = 1.00$
	(eve, tom)	$\oplus$		$n_1^{\ominus} = 2$	
	(tom, ann)	$\ominus$	$L_1 = female(X)$		
	(eve, ann)	$\ominus$	$Gain(L_1) = 0.84$	$n_1^{\oplus\oplus} = 2$	
Current clause c_2 : $daughter(X,Y) \leftarrow female(X)$					
$\mathcal{E}_2$	$(mary, ann)$	$\oplus$		$n_2^{\oplus} = 2$	$I(c_2) = 0.58$
	(eve, tom)	$\oplus$		$n_2^{\ominus} = 1$	
	(eve, ann)	$\ominus$	$L_2 = parent(Y,X)$		
			$Gain(L_2) = 1.16$	$n_2^{\oplus\oplus} = 2$	
Current clause c_3 : $daughter(X,Y) \leftarrow female(X), parent(Y,X)$					
$\mathcal{E}_3$	$(mary, ann)$	$\oplus$		$n_3^{\oplus} = 2$	$I(c_3) = 0.00$
	(eve, tom)	$\oplus$		$n_3^{\ominus} = 0$	

[Lavrac/Dzeroski 1994]

FOIL / Example

$Current\ clause\ c_1:\ daughter(X,Y) \leftarrow$				
$\mathcal{E}_1$	$(mary, ann)$	$\oplus$		$n_1^{\oplus} = 2$
	(eve, tom)	$\oplus$		$n_1^{\ominus} = 2$
	(tom, ann)	$\ominus$	$L_1 = parent(Y, Z)$	
	(eve, ann)	$\ominus$		$n_1^{\oplus\oplus} = 2$
$Current\ clause\ c_2:\ daughter(X,Y) \leftarrow parent(Y,Z)$				
$\mathcal{E}_2$	$(mary, ann, mary)$	$\oplus$		$n_2^{\oplus} = 4$
	$(mary, ann, tom)$	$\oplus$		
	(eve, tom, eve)	$\oplus$		
	(eve, tom, ian)	$\oplus$		
	$(tom, ann, mary)$	$\ominus$		$n_2^{\ominus} = 4$
	(tom, ann, tom)	$\ominus$		
	$(eve, ann, mary)$	$\ominus$		
	(eve, ann, tom)	$\ominus$		

[Lavrac/Dzeroski 1994]

Literal Selection

Let $n_i^{\oplus}$ be the number of positive examples in step i, $n_i^{\ominus}$ be the number of negative examples in step i.

information:

$$I(c_i) := -\log_2 \frac{n_i^{\oplus}}{n_i^{\oplus} + n_i^{\ominus}}$$

If the new literal does not introduce new variables,
$n_{i+1}^{\oplus} \leq n_i^{\oplus}$ and $n_{i+1}^{\ominus} \leq n_i^{\ominus}$.
But if new variables are introduced, this may not hold anymore.
Denote by $n_i^{\oplus\oplus}$ the number of positive tuples in $\mathcal{E}_i$ represented by at least one tuple in $\mathcal{E}_{i+1}$.

weighted information gain:

$$\text{WIG}(L_i, c_i) := \text{WIG}(c_{i+1}, c_i) := n_i^{\oplus\oplus}(I(c_i) - I(c_{i+1}))$$

Select the literal with the highest weighted information gain.

1. **Inductive Logic Programming**

2. **FOIL**

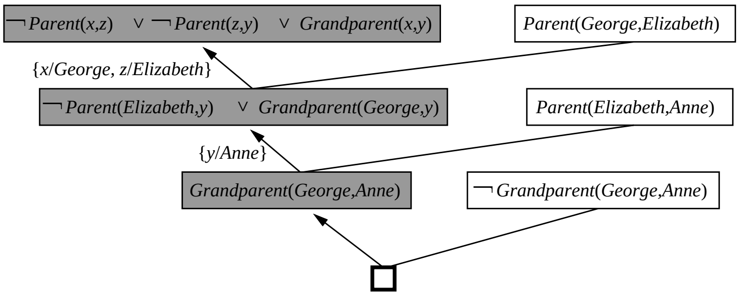
3. **Inverse Resolution**

Resolution:
Given clauses C_1 and C_2, infer resolvent C.

$$C_1 := C_1' \cup \{R\}, C_2 := C_2' \cup \{\neg R'\}, R\theta = R'\theta \quad \rightsquigarrow \quad C := C_1'\theta \cup C_2'\theta$$

Inverse resolution:
Given resolvent C and clause C_1, infer clause C_2.

$$C_1 := \{R\} \quad \rightsquigarrow \quad C_2 := \{\neg R'\} \cup C', \quad R'\theta = R\theta, C'\theta = C\theta$$



Inverse resolution is a search,
as there may be many pairs of clauses leading to resolvent C:

$\neg$Parent(Elizabeth, Anne) $\vee$ Grandparent(George, Anne)
$\neg$Parent(z, Anne) $\vee$ Grandparent(George, Anne)
$\neg$Parent(z, y) $\vee$ Grandparent(George, y)
...

Many techniques available for narrowing search space:

- eliminate redundancies, e.g., by generating only the most specific hypothesis.
- restrict proof strategy, e.g., to linear proofs.
- restrict representation language, e.g., to Horn clauses.
- use different inference method, e.g., model checking or ground propositional clauses.