

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

# Rule Based Learning

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

- 1 Rules
  - Predicate
  - Horn-clauses
- 2 Learning of Predicates
  - Decision Trees
  - Sequential Covering
- 3 Inductive Logic Programming
  - Top-Down
  - Bottom-Up

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

- 1 Rules
  - Predicate
  - Horn-clauses
- 2 Learning of Predicates
  - Decision Trees
  - Sequential Covering
- 3 Inductive Logic Programming
  - Top-Down
  - Bottom-Up

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Predicate

**Predicate** — Logical expressions

Depends in the values of attributes

Rainy  $\wedge$  Cold

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Horn-clauses

## Horn Clauses

**Horn Clauses** — First Order Logic

Can include **variables**

$$\text{sibling}(x, y) : \text{parent}(z, x) \wedge \text{parent}(z, y)$$

More expressional power than predicates

Foundation for e.g. *Prolog*

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Horn-clauses

Why would one like to learn rules?

- Can be transformed into "natural language"
- Explaining

Drawbacks?

- Hard to handle noise and uncertainty

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

- 1 Rules
  - Predicate
  - Horn-clauses
- 2 Learning of Predicates
  - Decision Trees
  - Sequential Covering
- 3 Inductive Logic Programming
  - Top-Down
  - Bottom-Up

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Decision Trees

Disjunction of conjunctions

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Sequential Covering

Sequential Covering (Sekvensiell täckning)

**Idéa underlying Sequential Covering**

Do not attempt to explain everything at once!

- 1 Construct **one** rule which
  - Matches many positive examples
  - Does not match any negative examples
- 2 Remove the positive examples which matched
- 3 Repeat until all positive examples are removed

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Sequential Covering

How does one find a good rule?

**Greedy Search** — Select attributes which matches as many positive examples as possible

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Sequential Covering

Beam search

Alternative to greedy search  
A few alternative search paths are kept

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Sequential Covering

Result from the search: several rules

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

- 1 Rules
  - Predicate
  - Horn-clauses
- 2 Learning of Predicates
  - Decision Trees
  - Sequential Covering
- 3 Inductive Logic Programming
  - Top-Down
  - Bottom-Up

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

## Inductive Logic Programing

ILP — Inductive Logic Programming  
Automatic construction of Horn Clauses from training data

**Horn Clauses**

- More expressional power than predicates
- Variables in the definitions

Example

$$\begin{aligned} \text{grandfather}(x, y) : & \text{parent}(x, z) \\ & \wedge \text{male}(x) \\ & \wedge \text{parent}(z, y) \end{aligned}$$

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

## Inductive Logic Programing

How can a program automatically find the rule

$$\text{grandfather}(x, y) : \text{parent}(x, z) \wedge \text{male}(x) \wedge \text{parent}(z, y)$$

from examples like this

$$\begin{aligned} \text{grandfather}(\text{Sven}, \text{Pär}) & \mapsto \text{False} \\ \text{grandfather}(\text{Pär}, \text{Sven}) & \mapsto \text{True} \\ \text{grandfather}(\text{Lisa}, \text{Sven}) & \mapsto \text{False} \end{aligned}$$

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

- **Top-down**  
Use Sequential Covering  
Start from a general rule and specialize
- **Bottom-up**  
Start from one example and construct explaining rules

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Top-Down

Top-Down technique to find **one rule**

- Start being maximally general

$$\text{grandfather}(x, y) : \text{True}$$

- Generate possible specializations

$$\begin{aligned} \text{grandfather}(x, y) : & \text{parent}(x, y) \\ \text{grandfather}(x, y) : & \text{parent}(x, z) \\ \text{grandfather}(x, y) : & \text{female}(x) \\ \text{grandfather}(x, y) : & \text{male}(x) \end{aligned}$$

- Keep the one which best describes data
- If the rule matches negative examples: **specialize it**

Rules 000      Learning of Predicates 00000      Inductive Logic Programing 0000

Top-Down

How does one specialize a rule?

- Append a conjunction factor

$$\begin{aligned} \text{grandfather}(x, y) : & \text{parent}(x, z) \wedge \text{male}(x) \\ \text{grandfather}(x, y) : & \text{parent}(x, z) \wedge \text{male}(z) \end{aligned}$$

- Choose the best

Repeat this until no negative examples match

How are new possible factors generated?

- Test **all known rules** with "old" variables
- Introduce **new variables**, in all but one position
- Use **equality** and inequalities
- Use **type information** for pruning

grandfather( $x, y$ ): parent( $x, y$ )  
parent( $y, x$ )  
male( $x$ )  
male( $y$ )

grandfather( $x, y$ ): parent( $x, z$ )  
parent( $z, x$ )  
parent( $y, z$ )  
parent( $z, y$ )

grandfather( $x, y$ ):  $x = y$   
 $x \neq y$   
 $x < y$

Bottom Up techniques

### Inverse Resolution

Use methods from automatic deduction "backwards"

- Start from **one example**
- Build an explaining rule
- Remove data that is explained
- Repeat