

Rule Based Learning

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

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

Predicate

Predicate — Logical expressions

Depends in the values of attributes

Rainy \wedge Cold

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*

Why would one like to learn rules?

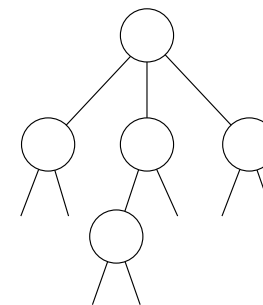
- Can be transformed into "natural language"
- Explaining

Drawbacks?

- Hard to handle noise and uncertainty

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

Decision Trees



Disjunction of conjunctions

1 Rules

- Predicate
- Horn-clauses

2 Learning of Predicates

- Decision Trees
- Sequential Covering

3 Inductive Logic Programming

- Top-Down
- Bottom-Up

Inductive Logic Programming

ILP — Inductive Logic Programming

Automatic construction of Horn Clauses from training data

Horn Clauses

- More expressional power than predicates
- Variables in the definitions

Example

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

Inductive Logic Programming

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

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

• Top-down

Use Sequential Covering
Start from a general rule and specialize

• Bottom-up

Start from one example and construct explaining rules

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

- Start being maximally general

grandfather(x, y) : **True**

- Generate possible specializations

grandfather(x, y) : parent(x, y)
 grandfather(x, y) : parent(x, z)
 grandfather(x, y) : female(x)
 grandfather(x, y) : male(x)

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

How does one specialize a rule?

- Append a conjunction factor

grandfather(x, y) : parent(x, z) ∧ male(x)
 grandfather(x, y) : parent(x, z) ∧ male(z)

- 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 ≠ y
 x < 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