Rule Induction

Rule Induction

- **Given:** Set of positive and negative examples of some concept
  - **Example:** *(x_1, x_2, … , x_n, y)*
  - *y:* **concept** (Boolean)
  - x_1, x_2, … , x_n: **attributes** (assume Boolean)
- **Goal:** Induce a set of rules that cover all positive examples and no negative ones
  - **Rule:** $x_a \wedge x_b \wedge \ldots \Rightarrow y$ (x_a: Literal, i.e., x_i or its negation)
  - Same as **Horn clause**: *Body* ⇒ *Head*
  - Rule *r* **covers** example *x* iff *x* satisfies body of *r*
- ***Eval(r):*** Accuracy, info gain, coverage, support, etc.

Learning a Single Rule

```
head ← y
body ← Ø
repeat
   for each literal x
      r_x ← r with x added to body
      Eval(r_x)
      body ← body ^ best x
until no x improves Eval(r)
return r
```

Learning a Set of Rules

```
R ← Ø
S ← examples
repeat
   learn a single rule r
    R ← R U { r }
   S ← S − positive examples covered by r
until S = Ø
return R
```

First-Order Rule Induction (a.k.a. Inductive Logic Programming)

- *y* and x_i are now predicates with arguments
  E.g.: *y* is Ancestor(x,y), x_i is Parent(x,y)
- Literals to add are predicates or their negations
- Literal to add must include at least one variable already appearing in rule
- Adding a literal changes # groundings of rule
  E.g.: Ancestor(x,z) ^ Parent(z,y) ⇒ Ancestor(x,y)
- *Eval(r)* must take this into account
  E.g.: Multiply by # positive groundings of rule
  still covered after adding literal

MLN Structure Learning

- Generalizes feature induction in Markov nets
- Any inductive logic programming approach can be used, but . . .
- Goal is to induce any clauses, not just Horn
- Evaluation function should be likelihood
- Requires learning weights for each candidate
- Turns out not to be bottleneck
- Bottleneck is counting clause groundings
- Solution: Subsampling

MLN Structure Learning

- **Initial state:** Unit clauses or hand-coded KB
- **Operators:** Add/remove literal, flip sign
- **Evaluation function:**
  Pseudo-likelihood + Structure prior
- **Search:** Beam search, shortest-first search