

Inductive Logic Programming

Devendra Laulkar

Learning Rules

- Decision tree algorithms learn rules in form of propositional rules

IF (sky = 'Sunny' AND airtemp = 'Warm')
THEN EnjoySport = yes

First Order Logic

- Extension of first order logic
- Quantification
- $\forall x P(x)$, $\exists x Q(x)$
- Predicates can have **variables**
- First order logic is more expressive than propositional logic

Relations among values

- IF father(Y, X) and female(Y)
THEN daughter(X, Y)
- Above rule cannot be effectively represented in propositional logic

Examples of rules

- $\text{sister}(X, Y) :- \text{father}(X, XF) \wedge \text{father}(Y, XF) \wedge \text{female}(Y).$
- $\text{can-reach}(N1, N2) :- \text{linked-to}(N1, N2).$
 $\text{can-reach}(N1, N2) :- \text{linked-to}(N1, N3) \wedge \text{can-reach}(N3, N2).$
- $\text{ancestor}(X, Y) :- \text{parent}(X, Y).$
 $\text{ancestor}(X, Y) :- \text{parent}(X, Z) \wedge \text{ancestor}(Z, Y)$

Input

- Set of ground facts.

father(jane, bob)

father(tom, bob)

father(bob, victor)

female(jane)

male(victor)

male(bob)

male(tom)

Training Data

- brother(jane, tom) +
- brother(bob, victor) -
- brother(tom, bob) -

Output

- Logic program

```
brother(X, Y) :- father(X, XF), father(Y,  
    XF), male(Y).
```

Demo of Prolog

- Family Relationships
- List processing

Terminology

- Variables : X, Y
- Constants : bob, jane, victor
- Predicates : mother, male, female
- Literal is a predicate applied to any set of terms
Eg. female(sharon), father(Y, Z).
- Clause is a disjunction of literals
 $L_1 \vee L_2 \vee \dots \vee L_n$
Eg. parent(X, Y) :- father(X, Y).

Horn Clause

- Clause of form

$$H :- (L1 \wedge L2 \wedge \dots \wedge Ln)$$

- Equivalent to

$$H \vee \sim L1 \vee \sim L2 \vee \dots \vee \sim Ln$$

Sequential Covering Algorithms

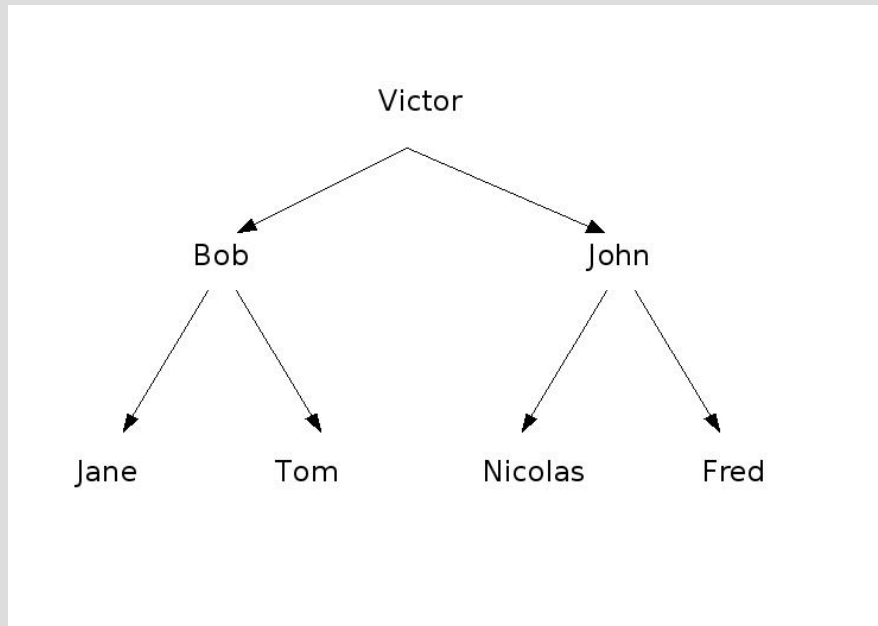
- Learn one rule at a time
- The rule must cover some positive examples and reject almost all negative examples

Learning a single rule

- Start with most general clause
- Specialize it by adding literals
- Search through the literal space for specializing the current clause

- General to Specific search

Example



- brother(bob, john)
+
- brother(john, bob)
+
- brother(jane, tom)
+
- brother(nicolas, fred)
+
- brother(fred, nicolas)
+
- brother(victor

Literal Space

- General clause
brother(X,Y) :-

- Possible literals

equal(X,Y) father(X,Y) father(Y, X)
male(X) male(Y) father(X, Xf)
father(Y, Yf)

Selecting Literal

- Adding literal to clause must cover few positive tuples and reject negative tuples

equal(X,Y) father(X,Y) father(Y,X) – Covers no + tuples, hence not selected

male(X) and male(Y) cover positive as well as negative examples

female(X) and female(Y) reject many + examples

Literal selected is father(X, Xf)

Selecting Literals...

- New variable Xf is introduced.
- Literal space increases, literals like $\text{male}(Xf)$, $\text{femal}(Xf)$ are added.
- Again the literals are checked.
- $\text{brother}(X, Y) \text{ :- father}(X, Xf)$

Selecting Literals...

- Next `father(Y, Xf)` covers all + tuples and rejects all but one – tuple (tom,jane,bob)
- `brother(X,Y) :- father(X, Xf) , father(Y, Xf)`

Selecting Literals...

- Literal `male(Y)` now covers all + tuples and rejects the - tuple, completing the clause
- `brother(X,Y) :- father(X,Xf) , father(Y, Xf) ,
male(Y).`

Algorithm

```
Learn_One_Rule(Target_Predicate, Predicates,
  Examples)
Pos, Neg = Positive and Negative training
  examples
NewRule = Rule that predicts target predicate
  with no preconditions
while Neg do
  candidate_literals = Generate literals which
    can be added to NewRule based on Predicates
  best_literal = Select best literal based on
    a performance measure
  Add best_literal to NewRule
  Neg = Subset of Neg that satisfies NewRule
end
```

Types of Literal

- Two types of literals : Gainful and Determinate
- Gainful literals eliminate – tuples.
- Determinate literals add new variables.

Eg. $\text{male}(Y)$ is gainful

$\text{father}(X, XF)$ is determinate.

Performance Measure

- Information Content of Data Set

$$I(T) = - \log_2 (P^+ / (P^+ + P^-))$$

- After adding a literal, the new data set will be generated
- Information Gain = $I(T_2) - I(T_1)$
- Select literal with highest gain

Learning Multiple Clauses

- Apply Learn One Rule repeatedly
- Remove positive tuples covered by each clause
- When all positive tuples are covered, return the learned rules.

FOIL

```
theory := null
remaining = all + tuples of target relation R
while remaining is not empty
  /* grow a new clause – Learn new rule */
  clause := R(A, B, ...) :-
    while clause covers – tuples of R
      Find appropriate literal
      Add L to right hand side of clause
      Remove + tuples covered by clause from
remaining
      Add clause to theory
```

Examples of FOIL

- Examples of FOIL

Advantages of FOIL

- Can generate rules as function free Horn Clauses
- Can find recursive definitions
- Does not require an oracle

Problems with FOIL

- Noisy data, Missing Values
- Adding Background knowledge increases complexity
- Handling Recursion
- Cannot discover new predicates
- Requires training data set of + and – tuples
- Short sighted Greedy algorithm
- Handling of continuous data

Cohen's Algorithm

- Specific to General Approach
- Add all possible possible to literals to clause.
- Check out training data, remove literals that do not match the training data
- Interactive algorithm.
- Can be considered as FIND – S algorithm for first order logic.

Thats it

devendralaulkar@yahoo.com