SGN-13000/SGN-13006 Introduction to Pattern Recognition and Machine Learning (5 cr)

Learning Sets of Rules

Joni-Kristian Kämäräinen

September 2016

Department of Signal Processing
Tampere University of Technology

Material

- Lecturer's slides and blackboard notes
- T.M. Mitchell. *Machine Learning*. McGraw-Hill, 1997: Chapter 10
- Computer examples

Contents

Introduction

Learning sets of rules

1. One of the most expressive and human readable representations for learned hypothesis is a set of IF-THEN rules.
2. IF-THEN rules are analog to *propositional logic*
3. Many applications in automatic assessing of chemical agents and biocomputing $\rightarrow$ *explaining phenomena!*

Deductive decisions from a set of rules: PROLOG

Example (relations.pl)

```
% Literals (kind of training set)
mother(annikki,joni).
mother(annikki,kalle).
mother(helvi,annikki).
mother(alice, robin).
father(robin,joni).
father(teuvo,annikki).
father(john,robin).
father(joni,aaro).
father(joni,reko).
grandmother(helvi,joni).
```

Query: mother(annikki,X).

Learning disjunctive sets of rules

Learning disjunctive sets of rules

Sequential covering algorithms

Learning disjunctive sets of rules

Learning disjunctive sets of rules

Method 1: Learn decision tree, convert to rules

Learning disjunctive sets of rules

Method 1: Learn decision tree, convert to rules

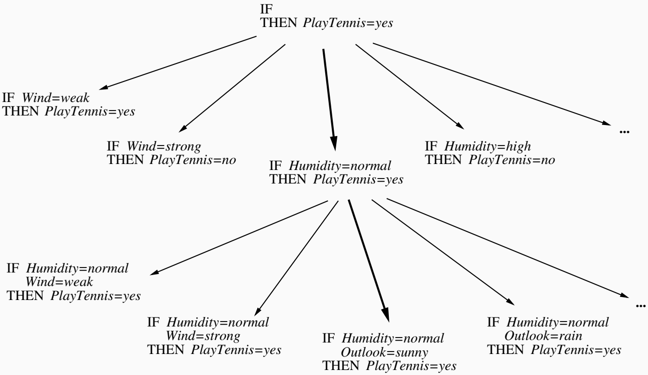
Method 2: Sequential covering algorithm:

1. *Learn one rule* with high accuracy, any coverage
2. Remove positive examples covered by this rule
3. Repeat

Sequential covering algorithm

Sequential-covering($Target\_attribute$, $Attributes$, $Examples$, $Threshold$)

$Learned\_rules \leftarrow \{\}$
$Rule \leftarrow$ learn-one-rule($Target\_attribute$, $Attributes$, $Examples$)
while performance($Rule$, $Examples$) $> Threshold$ **do**
    $Learned\_rules \leftarrow Learned\_rules + Rule$
    $Examples \leftarrow Examples - \{$examples correctly classified by $Rule\}$
    $Rule \leftarrow$ learn-one-rule($Target\_attribute$, $Attributes$, $Examples$)
end while
$Learned\_rules \leftarrow$ sort $Learned\_rules$ accord to performance over $Examples$
return $Learned\_rules$


IF
THEN PlayTennis=yes
IF Wind=weak
THEN PlayTennis=yes
IF Wind=strong
THEN PlayTennis=no
IF Humidity=normal
THEN PlayTennis=yes
IF Humidity=high
THEN PlayTennis=no
...
IF Humidity=normal
Wind=weak
THEN PlayTennis=yes
IF Humidity=normal
Wind=strong
THEN PlayTennis=yes
IF Humidity=normal
Outlook=sunny
THEN PlayTennis=yes
IF Humidity=normal
Outlook=rain
THEN PlayTennis=yes
...

Learning First-Order Rules

Learning first order rules

- Can learn sets of (recursive) rules such as

  $Ancestor(x, y) \leftarrow Parent(x, y)$
  $Ancestor(x, y) \leftarrow Parent(x, z) \wedge Ancestor(z, y)$

- Learned rules are PROLOG programs
- Finding rules: Inductive Logic Programming (ILP)

Example: First-order rules in PROLOG

Example (relations.pl (cont.))

```
% Literals (training set)
mother(annikki,joni).
mother(annikki,kalle).
mother(helvi,annikki).
mother(alice, robin).
father(robin,joni).
father(teuvo,annikki).
father(john,robin).
father(joni,aaro).
father(joni,reko).
grandmother(helvi,joni).

% Clauses
grandfather(X,Y) :-
father(X,Z),
father(Z,Y).
grandfather(X,Y) :-
father(X,Z),
mother(Z,Y).
```

Example: Classifying Web pages

[Slattery, 1997]

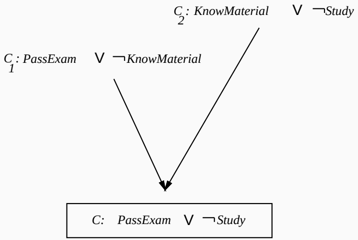
course(A) $\leftarrow$
has-word(A, instructor),
Not has-word(A, good),
link-from(A, B),
has-word(B, assign),
Not link-from(B, C)

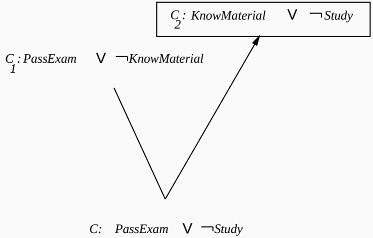
Train: 31/31, Test: 31/34

Learning First-Order Rules

Inverting resolution

Inverting Resolution





Learning First-Order Rules

FOIL

FOIL(*Target_predicate*, *Predicates*, *Examples*)

```
Pos ← positive Examples
Neg ← negative Examples
while Pos do {Learn a NewRule}
    NewRule ← most general rule possible
    NewRuleNeg ← Neg
    while NewRuleNeg do {Add a new literal to specialize NewRule}
        Candidate_literals ← generate candidates
        Best_literal ← argmax_{L∈Candidate_literals} Foil_Gain(L, NewRule)
        add Best_literal to NewRule preconditions
        NewRuleNeg ← subset of NewRuleNeg that satisfies NewRule
        preconditions
    end while
    Learned_rules ← Learned_rules + NewRule
    Pos ← Pos − {members of Pos covered by NewRule}
end while
return  Learned_rules
```

Specializing Rules in FOIL

Learning rule: $P(x_1, x_2, \ldots, x_k) \leftarrow L_1 \ldots L_n$

Candidate specializations add new literal of form:

- $Q(v_1, \ldots, v_r)$, where at least one of the v_i in the created literal must already exist as a variable in the rule.
- $Equal(x_j, x_k)$, where x_j and x_k are variables already present in the rule
- The negation of either of the above forms of literals

Information Gain in FOIL

$$Foil\_Gain(L, R) \equiv t \left(\log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right)$$

Where

- L is the candidate literal to add to rule R
- p_0 = number of positive bindings of R
- n_0 = number of negative bindings of R
- p_1 = number of positive bindings of $R + L$
- n_1 = number of negative bindings of $R + L$
- t is the number of positive bindings of R also covered by $R + L$

Note

- $-\log_2 \frac{p_0}{p_0+n_0}$ is optimal number of bits to indicate the class of a positive binding covered by R

Summary

Summary

1. Can be suitable for some novel application fields of machine learning yet to be exploited: biocomputing, scientific expert systems etc.
2. First-order Horn clauses (predicate logic) is a poweful knowledge representation that can be used in logical decision making with the PROLOG language and interpreter
3. The clauses can be automatically learnt from examples using the FOIL algorithm