Inductive Logic Programming

Fabrizio Riguzzi
University of Ferrara

Targeted Mailing

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	yes
ann	29	f	wa	no
steve	31	m	va	no

If Age<30 and Address=ca then Resp=yes

Multi Table

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	no
ann	29	f	wa	yes
steve	31	m	va	no

article

Name	Category	Size	Price
bike_1	sport	l	1000
jacket_2	clothing	l	150
tent_2	outdoor	m	250

transaction

Name	Article	Quantity
john	bike_1	2
ann	jacket_2	1
steve	bike_1	1
john	tent_2	1
ann	bike_1	3

The customer will respond if she/he has bought an item of category clothing

Join

customer $\triangleright\!\triangleleft$ transaction $\triangleright\!\triangleleft$ article

Name	Age	Sex	Address	Article	Quantity	Category	Size	Price	Resp
john	35	m	ca	bike_1	2	sport	l	1000	no
john	35	m	ca	tent_2	1	outdoor	m	250	no
mary	25	f	ca						no
ann	29	f	wa	jacket_2	1	clothing	l	150	yes
ann	29	f	wa	bike_1	3	sport	l	1000	yes
steve	31	m	va	bike_1	2	sport	l	1000	no

Replicate

Replicate attributes

Name	Age	Sex	Address	Article1	Quantity1	Category1	Size1	Price1
john	35	m	ca	bike_1	2	sport	l	1000
mary	25	f	ca					
ann	29	f	wa	jacket_2	1	clothing	l	150
steve	31	m	va	bike_1	2	sport	l	1000

Article2	Quantity2	Cateogory2	Size2	Price2	Resp
tent_2	1	outdoor	m	250	no
					no
bike_1	3	sport	l	1000	yes
					no

Logic

customer

Name	Age	Sex	Address	Resp
john	35	m	ca	no
mary	25	f	ca	no
ann	29	f	wa	yes
steve	31	m	va	no

article

Name	Category	Size	Price
bike_1	sport	l	1000
jacket_2	clothing	l	150
tent_2	outdoor	m	250

transaction

Name	Article	Quantity
john	bike_1	2
ann	jacket_2	1
steve	bike_1	1
john	tent_2	1
ann	bike_1	3

$$respond(Customer) \leftarrow$$
$$transaction(Customer, Article, \_Quantity),$$
$$article(Article, Category, \_Size, \_Price),$$
$$Category = clothing.$$

Outline of the Talk

- Predictive ILP
  - Learning from entailment
    - Bottom-up systems: Golem
    - Top-down systems: FOIL, Progol
  - Learning from interpretations
    - ICL, Tilde
- Descriptive ILP
  - Claudien
- Probabilistic ILP
  - ALLPAD
- Applications

Predictive ILP

- Aim:
  - classifying instances of the domain, i.e.
  - predicting the class
- Two settings:
  - Learning from entailment
  - Learning from interpretations

Learning from Entailment

- Given
  - A set of positive example E^+
  - A set of negative examples E^-
  - A background knowledge B
  - A space of possible programs $\mathcal{H}$
- Find a program $P \in \mathcal{H}$ such that
  - $\forall e^+ \in E^+$, $P \cup B \models e^+$ (completeness)
  - $\forall e^- \in E^-$, $P \cup B \not\models e^-$ (consistency)

Mailing Example

- Positive examples $E^+ = \{respond(ann)\}$
- Negative examples
  $E^- = \{respond(john), respond(mary), respond(steve)\}$
- Background $B =$ facts for relations $customer$, $transaction$ and $article$
  $customer(john, 35, m, ca).$
  $customer(mary, 25, f, ca).$
  $customer(ann, 29, f, wa). \ldots$
  $transaction(john, bike\_1, 2).$
  $transaction(ann, jacket\_2, 1). \ldots$
  $article(bike\_1, sport, l, 1000).$
  $article(jacket\_2, clothing, 1, 150). \ldots$

Mailing Example

- Space of programs $\mathcal{H}$: programs containing clauses with
  - in the head $respond(Customer)$
  - in the body a conjunction of literals from the set
    $\{customer(Customer, Age, Sex, Address),$
    $transaction(Customer, Article, Quantity),$
    $article(Article, Category, Price),$
    $Age = constant, Sex = constant, \ldots\}$

Definitions

- $covers(P, e) = true$ if $B \cup P \models e$
- $covers(P, E) = \{e \in E | covers(P, e) = true\}$
- A theory P is more general than Q if
  $covers(P, U) \supseteq covers(Q, U)$
- If $B \cup P \models Q$ then P is more general than Q
- A clause C is more general than D if
  $covers(\{C\}, U) \supseteq covers(\{D\}, U)$
- If $B, C \models D$ then C is more general than D
- If a clause covers an example, all of its generalizations will ($covers$ is antimonotonic)
- If a clause does not cover an example, none of its specializations will

Theta Subsumption

- C θ-subsumes D ($C \geq D$) if there exists a substitution θ such that $C\theta \subseteq D$ [Plotkin 70]
- $C \geq D \Rightarrow C \models D \Rightarrow B, C \models D \Rightarrow C$ is more general than D
- $C \models D \not\Rightarrow C \geq D$

Examples of Theta Subsumption

- $C1 = grandfather(X, Y) \leftarrow father(X, Z)$
- $C2 = grandfather(X, Y) \leftarrow father(X, Z), parent(Z, Y)$
- $C3 = grandfather(john, steve) \leftarrow father(john, mary), parent(mary, steve)$
- $C1 \geq C2$ with $\theta = \emptyset$
- $C1 \geq C3$ with $\theta = \{X/john, Y/steve, Z/mary\}$
- $C2 \geq C3$ with $\theta = \{X/john, Y/steve, Z/mary\}$

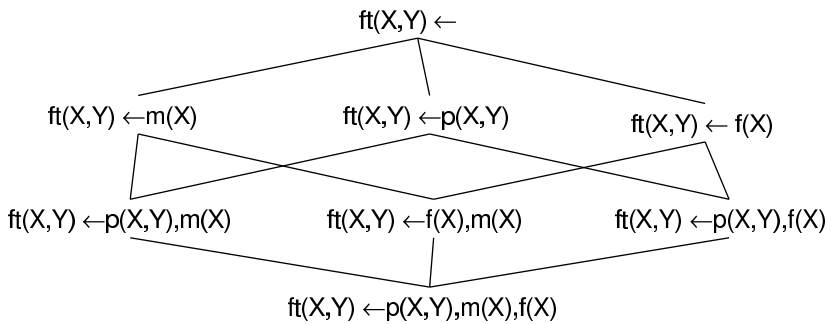
In Practice

- Logical consequence is undecidable
- Coverage test: SLD or SLDNF resolution
- Generality order:
  - θ-subsumption because it is decidable (even if NP-complete)

Properties of Theta Subsumption

- θ-subsumption induces a lattice in the space of clauses
- Every set of clauses has a least upper bound (lub) and a greatest lower bound (glb)
- This is not true for the generality relation based on logical consequence

Lattice



Least General Generalization

- $lgg(C, D) =$ least upper bound in the θ-subsumption order
- An algorithm exists which has complexity $O(s^2)$ where s is the size of the clauses
- Example:

$C = father(john, mary) \leftarrow parent(john, mary), male(john)$
$D = father(david, steve) \leftarrow parent(david, steve), male(david)$
$lgg(C, D) = father(X, Y) \leftarrow parent(X, Y), male(X)$

- For a set of n clauses the complexity is $O(s^n)$

Relative Subsumption

- θ subsumption does not take into account background knowledge
- $C \geq D \Leftrightarrow \models \forall(C\theta \rightarrow D)$
- Relative Subsumption [Plotkin 71]: C θ subsume D relative to background B ($C \geq_B D$) if there exists a substitution θ such that $B \models \forall(C\theta \rightarrow D)$

Relative Least General Generalization

- Relative Least General Generalization (rlgg): lgg with respect to relative subsumption.
- It does not exists in the general case of B a set of Horn clauses
- It exists in the case that B is a set of ground atoms and can be computed in this way:
- $rlgg((H1 \leftarrow B1), (H2 \leftarrow B2)) =$
  $lgg((H1 \leftarrow B1, B), (H2 \leftarrow B2, B))$
- If B is a set of clauses, we can compute the h-easy model

Relative Least General Generalization

- Example

$C1 = father(john, mary)$

$C2 = father(david, steve)$

$B = \{parent(john, mary), parent(david, steve),$
$parent(kathy, ellen), female(kathy),$
$male(john), male(david)\}$

$rlgg(C1, C2) = father(X, Y) \leftarrow parent(X, Y), male(X)$

Outline of the Talk

- Predictive ILP
  - Learning from entailment
    - **Bottom-up systems: Golem**
    - Top-down systems: FOIL, Progol
  - Learning from interpretations
    - ICL, Tilde
- Descriptive ILP
  - Claudien
- Probabilistic ILP
  - ALLPAD
- Applications

Bottom-up Systems

- Covering loop
- Search for a clause from specific to general

Learn(E, B)
$P := 0$
repeat /* covering loop */
    $C :=$GenerateClauseBottomUp(E, B)
    $P := P \cup \{C\}$
    Remove from E the positive examples covered by P
until Sufficiency criterion
return P

Golem [Muggleton, Feng 90]

- Bottom-up system
- Generalization by means of rlgg
- Sufficiency criterion: $E^+ = \emptyset$

Golem

GolemGenerateClause(E, B)
select randomly some couples of examples from E^+
compute their rlgg
let C be the rlgg that covers most positive examples
    while covering no negative
repeat
    randomly select some examples from E^+
    compute the rlgg between C and each selected example
    let C be the rlgg that covers most positive examples
        while covering no negative
    remove from E^+ the examples covered by C
while the set of examples covered by C increases
remove literals from the body of C until C covers
    some negative examples
return C

Outline of the Talk

- Predictive ILP
  - Learning from entailment
    - Bottom-up systems: Golem
    - **Top-down systems: FOIL, Progol**
  - Learning from interpretations
    - ICL, Tilde
- Descriptive ILP
  - Claudien
- Probabilistic ILP
  - ALLPAD
- Applications

Top-down Systems

- Covering loop as bottom-up systems
- Search for a clause from general to specific

Top-down Systems

GenerateClauseTopDown(E,B)
$Beam := \{p(X) \leftarrow true\}$
$BestClause := null$
repeat /* specialization loop */
    Remove the first clause C of $Beam$
    compute $\rho(C)$
    score all the refinements
    update $BestClause$
    add all the refinements to the beam
    order the beam according to the score
    remove the last clauses that exceed the dimension d
until the Necessity criterion is satisfied
return $BestClause$

Typical Stopping Criteria

- Sufficiency criteria:
  - $E^+ = \emptyset$
  - GenerateClauseTopDown returns $null$
  - a disjunction of the above
- Necessity criteria
  - the number of negative examples covered by $BestClause$ is 0
  - the number of covered negative examples covered by $BestClause$ is below a threshold
  - $Beam$ is empty
  - a disjunction of the above

Refinement Operator

- $\rho(C) = \{D | D \in L, C \geq D\}$
- where L is the space of possible clauses
- A refinement operator usually generates only minimal specializations
- A typical refinement operator applies two syntactic operations to a clause
  - applies a substitution to the clause
  - adds a literal to the body

Heuristic Functions

- Notation: $n^+(C), n^-(C)$ number of positive and negative examples covered by clause C
- $n(C) = n^+(C) + n^-(C)$
- Accuracy: $Acc = p(+|C)$ (more accurately Precision), $p(+|C)$ can be estimated by
  - relative frequency: $p(+|C) = \frac{n^+(C)}{n(C)}$
  - m-estimate: $p(+|C) = \frac{n^+(C)+mp(+)}{n(C)+m}$, where $p(+) = n^+/n$
  - Laplace: m-estimate with $m = 2, p = 0.5$ $p(+|C) = \frac{n^+(C)+1}{n(C)+2}$

Heuristic Functions

- Coverage: $Cov = n^+(C) - n^-(C)$
- Informativity: $Inf = \log_2(Acc)$
- Weighted relative accuracy: $WRAcc = p(C)(p(+|C) - p(+))$

Coverage Tests

- How to test coverage of an example e for a clause $C = h \leftarrow Body$?
- Intensional Coverage: try to derive e from $B \cup P \cup \{C\}$
- Extensional coverage:
  - let θ be the mgu of e and h,
  - try to derive $Body\theta$ from $B \cup E^+$

FOIL [Quinlan 90]

- Top-down system with
  - Dimension of the beam: 1
  - Heuristic: (approximately) weighted gain of Inf: $H = n(C')(Inf(C') - Inf(C))$
  - Extensional coverage
  - Refinement operator: addition of a literal or unification
  - Sufficiency criterion: $E^+ = \emptyset$
  - Necessity criterion: $n^-(BestClause) = 0$

Progol [Muggleton 95]

- Top-down system with
  - Dimension of the beam: user defined
  - Heuristic: Compression: $Comp = n^+(C) - n^-(C) - |C|$
  - Intensional coverage
  - Refinement operator: see next slides
  - Sufficiency criterion: $E^+ = \emptyset$
  - Necessity criterion: $Beam = \emptyset$ or a maximum number of iterations of the loop is reached

Progol Refinement Operator

- Progol refinement operator
  - adds a literal from the most specific clause $\perp$ after having substituted some of the constants with variables

How to Obtain $\bot$

$B = \{parent(john, mary), male(john), parent(david, steve),$
$male(david), parent(kathy, ellen), female(kathy)\}$
$e = father(john, mary)$

$$B \wedge C \models e$$

$$B \wedge \overline{e} \models \overline{C}$$

- $\overline{C}$ is a set of ground skolemized facts
- $B \wedge \overline{e} =$
  $\{parent(john, mary), male(john), parent(david, steve),$
  $male(david), parent(kathy, ellen), female(kathy),$
  $\neg father(john, mary)\}$

How to Obtain $\bot$

$B \wedge \overline{e} = \{parent(john, mary), male(john), parent(david, steve),$
$male(david), parent(kathy, ellen), female(kathy),$
$\neg father(john, mary)\}$

- Let $\overline{\bot}$ be the potentially infinite conjunction of ground literals true in the model of $B \wedge \overline{e}$

$\overline{\bot} = B \wedge \overline{e}$
$\bot = father(john, mary) \leftarrow parent(john, mary), male(john),$
$parent(david, steve), male(david),$
$parent(kathy, ellen), female(kathy)$

Progol Refinement Operator

- $\overline{\bot}$ is the potentially infinite conjunction of ground literals true in the model of $B \wedge \overline{e} \Rightarrow$

$$\overline{\bot} \models \overline{C} \Rightarrow$$

$$C \models \bot$$

- We can find some of the solutions of the learning problem by looking for clauses that subsume $\bot$
- Example:
  $C = father(X, Y) \leftarrow parent(X, Y), male(X)$
- C θ-subsumes
  $\bot = father(john, mary) \leftarrow$
  $parent(john, mary), male(john), parent(david, steve),$
  $male(david), parent(kathy, ellen), female(kathy)$

Progol Refinement Operator

- $\bot$ can have infinite cardinality
  - Progol uses a limit to the depth of the derivations with which $\bot$ is built
- Moreover, restrictions to constrain the space of clauses that subsume $\bot$
  - limit to the depth of variables in C
  - mode declarations

Mode Declarations

- $modeh(n, atom)$ or $modeb(n, atom)$
- n is the maximum number of times that atom can be added to C
- $atom$ is of the form $p(Term_1, \ldots, Term_n)$
- $Term_i$ is of the form
  - $+type$: input variable of type $type$
  - $-type$: output variable of type $type$
  - $\#type$: constant of type $type$

Examples of Mode Declarations

$modeh(1, plus(+int, +int, -int))$
$modeb(*, append(-list, +list, +list))$
$modeb(1, append(+list, [+any], -list)$
$modeb(4, (+int > \#int))$

Outline of the Talk

- Predictive ILP
  - Learning from entailment
    - Bottom-up systems: Golem
    - Top-down systems: FOIL, Progol
  - **Learning from interpretations**
    - ICL, Tilde
- Descriptive ILP
  - Claudien
- Probabilistic ILP
  - ALLPAD
- Applications

Learning from Interpretations

- Aim: learning a classifier for logical interpretations
- Classifier: a set of disjunctive clauses
- Disjunctive clause
  $C = h_1 \vee h_2 \vee \ldots \vee h_n \leftarrow b_1, b_2, \ldots, b_m$
- $head(C) = \{h_1, h_2, \ldots, h_n\}$
- $body(C) = \{b_1, b_2, \ldots, b_m\}$
- $body^+(C) =$ set of positive literals of $body(C)$
- $body^-(C) =$ set of atoms of negative literals of $body(C)$
- Interpretation = set of ground atoms.

Learning from Interpretations

- Set of clauses as a classifier
  - an interpretation is positive if all the clauses are true in the interpretation
  - an interpretation is negative if there exists at least one clause that is false in it
- A clause C is true in an interpretation I if for all grounding substitutions θ of C:
  $I \models body(C)\theta \rightarrow head(C)\theta \cap I \neq \emptyset$
  or
  $body^+(C)\theta \subseteq I \wedge body^-(C)\theta \cap I = \emptyset \rightarrow head(C)\theta \cap I \neq \emptyset$

Test of the Truth of a Clause

- Range restricted clause: all the variables in the head appear in the body
- Range restricted clause C, finite interpretation I: run the query $?- body(C), not\ head(C)$ against a logic program containing I
- If $C = h_1 \vee h_2 \vee \ldots \vee h_n \leftarrow b_1, b_2, \ldots, b_m$ then the query is $?- b_1, b_2, \ldots, b_m, not\ h_1, not\ h_2, \ldots, not\ h_n$
- If the query succeeds, C is false in I. If the query fails, C is true in I [De Raedt, Bruynooghe 93]

Example

- $I = \{female(liz), male(richard), gorilla(liz), gorilla(richard)\}$
- $C = male(X) \vee female(X) \leftarrow gorilla(X)$: the clause is true in I because the query $?- gorilla(X), not\ male(X), not\ female(X)$ fails
- $C = male(X) \leftarrow gorilla(X)$: the clause is false in I because the query $?- gorilla(X), not\ male(X)$ succeeds with $\theta = \{X/liz\}$.

Learning from Interpretations

- **Given**
  - a space of possible clausal theories $\mathcal{H}$
  - a set P of interpretations
  - a set N of interpretations
- **Find**: a clausal theory $H \in \mathcal{H}$ such that
  - for all $p \in P$, $p \models H$
  - for all $n \in N$, $n \not\models H$
- Less expressive than learning from entailment: no recursive definitions

Test with Background

- Background: a normal program B
- Truth of a clause C in the interpretation $M(B \cup I)$ where M is the model according to the chosen semantics and I is an interpretation (i.e. $B \cup I \models C$)
- Range restricted clause C, normal program B containing only range restricted clauses, interpretation I: run the query $? - body(C), not\ head(C)$ against the logic program $B \cup I$.
- If the query succeeds, C is false in $M(B \cup I)$ ($B \cup I \not\models C$). If the query fails, C is true in $M(B \cup I)$ ($B \cup I \models C$)

Learning from Int. with Background

Given

- a space of possible clausal theories $\mathcal{H}$
- a set P of interpretations
- a set N of interpretations
- a background theory B

Find: a clausal theory $H \in \mathcal{H}$ such that

- for all $p \in P$, $B \cup p \models H$
- for all $n \in N$, $B \cup n \not\models H$

Generality Relation

- $cover(\{C\}, e) = true$ if $e \models C$
- $C \geq D \Rightarrow C \models D \Rightarrow D$ is more general than C
- the relation is reversed
- Example:

$false \leftarrow true$
$false \leftarrow gorilla(X)$
$female(X) \leftarrow gorilla(X)$
$female(X) \vee male(X) \leftarrow gorilla(X)$

ICL [De Raedt, Van Laer, 95]

- Dual version of a top down entailment algorithm:
  - coverage loop is performed on negative examples
- Updates CN2 to first order

ICL(P, N, B)
$H := \emptyset$
repeat
    $C :=$FindBestClause(P, N, B)
    if $C \neq null$ then
        add C to H
        remove from N all interpretations that are false for C
until $C = null$ or N is empty
return H

ICL FindBestClause

FindBestClause(P, N, B)
$Beam := \{false \leftarrow true\}$, $BestClause := null$
while $Beam$ is not empty do
    $NewBeam := \emptyset$
    for each clause C in $Beam$ do
        for each refinement Ref of C do
            if Ref is better than $BestClause$ and Ref is statistically significant then
                $BestClause := Ref$
            if Ref is not to be pruned then
                add Ref to $NewBeam$
                if size of $NewBeam > MaxBeamSize$ then
                    remove worst clause from NewBeam
    $Beam := NewBeam$
return $BestClause$

ICL Heuristics

- $n(\overline{C})$= number of interpretations (positive and negative) where C is false
- $n^-(\overline{C})$= number of negative interpretation where C is false
- $H(C) = p(-|\overline{C}) = \frac{n^-(\overline{C})+1}{n(\overline{C})+2}$ = precision over negative class

Tilde [Blockeel, De Raedt 98]

- Updates C4.5 to first-order
- Solve a slightly different learning problem
- **Given**
  - a space of possible theories $\mathcal{H}$
  - a set of classes C
  - a set E of classified interpretations (couples (I, c))
  - a background theory B
- **Find**: a theory $H \in \mathcal{H}$ such that $\forall (I, c) \in E$
  - $B \cup I \cup H \models c$
  - $\forall c' \in C \setminus \{c\}, : B \cup I \cup H \not\models c'$
- Less expressive than learning from entailment: excludes recursive definitions

Example

$E =$

```
Machine 1: {worn(gear), worn(engine), sendback}
Machine 2: {ok}
Machine 3: {worn(gear), fix}
Machine 4: {worn(engine), sendback}
Machine 5: {worn(gear), worn(chain), fix}
```

$B =$

```
replaceable(gear).
replaceable(wheel).
replaceable(chain).
not_replaceable(engine).
not_replaceable(control_unit).
```

Example

```
worn(A) ?
+--yes: not_replaceable(A) ?
|       +--yes: [sendback] [6.0/6.0]
|       +--no:  [fix] [6.0/6.0]
+--no:  [ok] [3.0/3.0]
```

Equivalent Prolog program (FODL):

```
class(sendback) :- worn(A),not_replaceable(A), !.
class(fix) :- worn(A), !.
class(ok).
```


Tilde Algorithm

Tilde(E)
T:=GrowTree($E, true$)
return Prune(T)

GrowTree(E, Q: query)
$Q_b :=$ OptimalSplit($\rho(Q), E$)
if StopCrit(Q_b, E) then
    return $leaf(majority\_class(E))$
else
    $conj := Q_b - Q$
    $E_1 := \{I \in E | \leftarrow Q_b$ succeeds in $B \cup I\}$
    $E_2 := \{I \in E | \leftarrow Q_b$ fails in $B \cup I\}$
    $T := inode(conj,$
        GrowTree(E_1, Q_b),GrowTree(E_2, Q))
    return T

Outline of the Talk

- Predictive ILP
  - Learning from entailment
    - Bottom-up systems: Golem
    - Top-down systems: FOIL, Progol
  - Learning from interpretations
    - ICL, Tilde
- **Descriptive ILP**
  - Claudien
- Probabilistic ILP
  - ALLPAD
- Applications

Descriptive ILP

- Discovering regularities, patterns
- Example tasks:
  - finding association rules
  - clustering
  - subgroup discovery

Claudien [De Raedt, Dehaspe 97]

- Learning problem: Given
  - a space of possible clausal theories $\mathcal{H}$
  - a set P of interpretations
  - a background theory B
- **Find**: a clausal theory $H \in \mathcal{H}$ such that
  - $\forall p \in PB \cup p \models H$
  - H is maximally specific

Example

$p1 = \{female(liz), male(richard),$
$gorilla(liz), gorilla(richard)\}$
$p2 = \{female(ginger), male(fred),$
$gorilla(ginger), gorilla(fred)\}$
If $\mathcal{H}$ is restricted to range-restricted, constant-free clauses a solution is:
$gorilla(X) \leftarrow female(X)$
$gorilla(X) \leftarrow male(X)$
$male(X) \vee female(X)$
$\leftarrow male(X), female(X)$

Claudien Algorithm

ClausalDiscovery(E, B)
$H := \emptyset$
$Beam := \{false \leftarrow true\}$
while $Beam$ is not empty do
    delete from $Beam$ the first clause C
    if C is true on E then
        $H := H \cup \{C\}$
    else
        for all $C' \in \rho(C)$ for which not prune(C') do
            $Beam := Beam \cup \{C'\}$
return H

Outline of the Talk

- Predictive ILP
  - Learning from entailment
    - Bottom-up systems: Golem
    - Top-down systems: FOIL, Progol
  - Learning from interpretations
    - ICL, Tilde
- Descriptive ILP
  - Claudien
- **Probabilistic ILP**
  - ALLPAD
- Applications

Probabilistic ILP

- Learning a probabilistic logic program
- Formalisms:
  - BLP: ayesian Logic Programs
  - CLP(BN): CLP over Bayesian Networks
  - MIA: Meta-Interpreter Approach
  - SLP: Stochastic Logic Programs
  - LBN: Logical Bayesian Networks
  - LPAD: Logic Programs with Annotated Disjunctions

LPADs [Vennekens et al. 04]

- A **Logic Program with Annotated Disjunctions** consists of a set of formulas of the form

$$(h_1 : p_1) \vee (h_2 : p_2) \vee \ldots \vee (h_n : p_n) \leftarrow b_1, b_2, \ldots b_m$$

- The p_i are real numbers in the interval $[0, 1]$ such that $\sum_{i=1}^{n} p_i = 1$.
- Instance: a ground normal program obtained by selecting one atom from each grounded clause
- They define a probability distribution π_P over
  - instances: product of probabilities of heads selected
  - interpretations and formulas: probability of ϕ is the sum of the probabilities of instances where ϕ is true

Example

$mother(m, c) \quad father(f, c)$

$cg(m, 1, w) \quad cg(m, 2, w)$

$cg(f, 1, p) \quad cg(f, 2, w)$

$(cg(X, 1, A) : 0.5) \vee (cg(X, 1, B) : 0.5) \leftarrow$
$\quad mother(Y, X), cg(Y, 1, A), cg(Y, 2, B)$

$(cg(X, 2, A) : 0.5) \vee (cg(X, 2, B) : 0.5) \leftarrow$
$\quad father(Y, X), cg(Y, 1, A), cg(Y, 2, B)$

$color(X, purple) \leftarrow cg(X, \_N, p)$

$color(X, white) \leftarrow cg(X, 1, w), cg(X, 2, w)$

Learning LPADs

- Learning problem:
  Given:
  - a set E of couples $(I, \pi(I))$ where I is an interpretation and $\pi(I)$ is its associated probability
  - a space of possible LPAD $\mathcal{H}$

  Find:
  - an LPAD $P \in \mathcal{H}$ such that $\forall (I, \pi(I)) \in E \;\; \pi_P^*(I) = \pi(I)$
- Instead of a set of couples $(I, \pi(I))$, the input of the learning problem can be a multiset E' of interpretations.

ALLPAD [Riguzzi 06]

- Three phases:
  - find all the ground clauses that satisfy a number of constraints
  - compute the probabilities in the head
  - solve an optimization problem

Outline of the Talk

- Predictive ILP
  - Learning from entailment
    - Bottom-up systems: Golem
    - Top-down systems: FOIL, Progol
  - Learning from interpretations
    - ICL, Tilde
- Descriptive ILP
  - Claudien
- Probabilistic ILP
  - ALLPAD
- **Applications**

Applications

- Biology
- Chemistry
- Engineering
- Various

Algorithm Evaluation

- Notation:
  - $n^+(P)$ number of positive examples covered by P
  - $n^-(\overline{P})$ number of negative examples not covered by P
  - $n = |E|$
- Accuracy:

$$Acc(P) = \frac{n^+(P) + n^-(\overline{P})}{n}$$

Structure Activity Relationships (SARs)

- Predicting the activity of a compound on humans based on its chemical structure and properties
  - Drugs: whether they are effective
  - Compounds, drugs: whether they are toxic

Description of Chemical Compounds

Basic structure:
$atom(compound, atom, element, atomType, charge)$
e.g. $atom(d2, d2\_1, c, 22, 0.067)$
$bond(compound, atom1, atom2, bondType)$
e.g. $bond(d2, d2\_1, d2\_2, 7)$
Structures:
$benzene(compound, listOfAtoms)$
e.g. $benzene(d4, [d4\_6, d4\_1, d4\_2, d4\_3, d4\_4, d4\_5])$
$phenanthrene(compound, listOfListsOfAtoms))$
$nitro(compound, listOfAtoms)$
...
Properties:
$polar(atom, polarity)$
$polar(d2\_1, polar3)$
...

SAR

- Drugs against Alzheimer's disease
  - Golem: not significantly different from propositional, comprehensibility [King et al. 95]
- Drugs for inhibition of E. Coli Dihydrofolate Reductase
  - Golem: not significantly different from propositional, comprehensibility [King et al. 95]
- Predicting carcinogenicity
  - Progol: 72% highest machine accuracy [Srinivasan et al. 97]

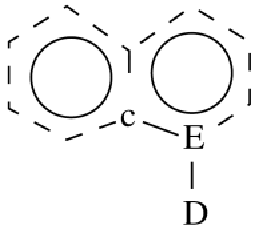
SAR

- Predicting mutagenicity
  - regression friendly compounds
    - FOIL: 82% [Srinivasan et al 95]
    - ICL: 86.2% [Van Laer et al. 97]
    - Progol: 88% [Srinivasan et al 95]
    - Claudien: found alternative explanations [De Raedt, Dehaspe 97]
  - regression unfriendly compounds
    - Progol: 85.7% [King et al. 96]

Progol on Mutagenesis

$active(A) \leftarrow$
$atom(A, B, c, 27, C),$
$bond(A, D, E, 1), bond(A, E, B, 7)$

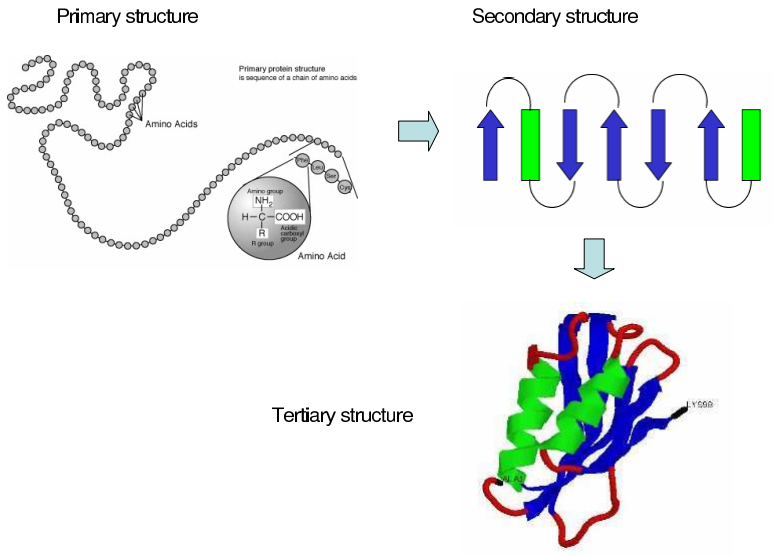
A carbon atom of type 27 merges two six-membered aromatic rings.
A bond of type 7 is an aromatic bond.
This rule identifies compounds of two fused six-membered aromatic rings, one of which has a further single bond with an atom of any type.



Biology

- Description of the binding sites (pharmacophores) of ACE inhibitors (hypertension drug) and an HIV-protease inhibitor (an anti-AIDS drug)
  - Progol: rediscovered a pharmacophore found by experts [Finn et al. 98]
- Biological classification of river water quality
  - Golem: comprehensibility [Dzeroski et al. 94]
  - Claudien: intuitive rules [De Raedt, Dehaspe 97]

Proteins



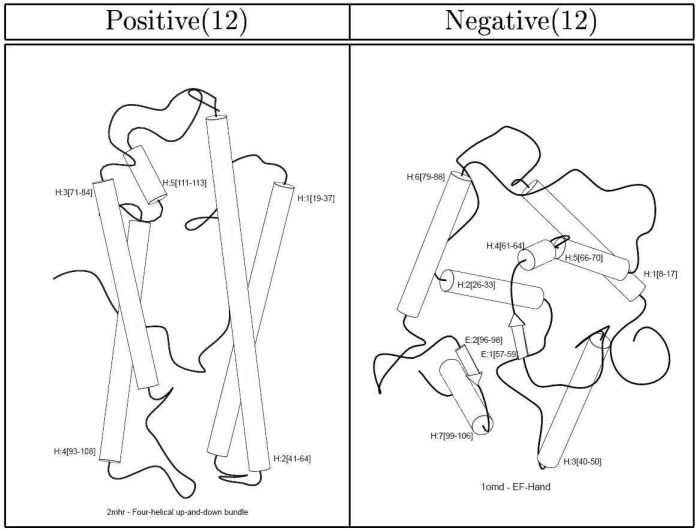
Protein Secondary Structure

- Predicting protein secondary structure from the amino-acid sequence
- Structures
  - helices, of various types and length
  - strands, of various orientations and length
- Results:
  - Golem: 80% [Muggleton et al. 92]
  - FOIL: 65% [Quinlan, Cameron-Jones 95]

Protein Tertiary Structure

- Predicting the tertiary structure of proteins by classifying them into one of the SCOP classes
- Proteins represented as a sequence of secondary structure elements
- Results:
  - Progol: 78.28% [Turcotte et al. 01]
  - ALLPAD: 85.67% [Riguzzi 06]

Protein Tertiary Structure



Chemistry

- Identification of the structure of diterpene from spectral information
  - FOIL: 78.3% [Dzeroski et al. 96]
  - Tilde: 90.4% [Blockeel, De Raedt 98]
- Predicting the half-time of aqueous biodegradation of a compound from its chemical structure
  - ICL: 58.1% [Van Laer et al. 97]
- Judging whether a molecule is a musk
  - Tilde: 79.4% [Blockeel, De Raedt 98]

Engineering

- Learning rules for finite element mesh design
  - Claudien: 34% from pos. ex. only [De Raedt, Dehaspe 97]
  - ICL: 66.5% [Van Laer et al. 97]
  - Golem: 78% [Dolsak et al. 94]

Various

- Identifying document components
  - FOIL: 96.3%-100% [Quinlan, Cameron-Jones 95]
- Recovering program loop invariants from program traces
  - Claudien: found true invariants [De Raedt, Dehaspe 97]

Conclusions

- Active Research Field
- New directions:
  - upgrading propositional systems (probabilistic models, support vector machines, reinforcement learning, neural networks,...)
  - downgrading first order systems (to make it more efficient in special cases)
  - multi relational data mining

Pointers

- ILPnet2
  - http://www.cs.bris.ac.uk/∼ILPnet2/
  - http://www-ai.ijs.si/∼ilpnet2/
- KDnet http://www.kdnet.org/
- Books:
  - [Lavrac, Dzeroski 94]: freely available in pdf on the web
  - [Bergadano et al. 96]
  - [Dzeroski, Lavrac 01]

Bibliography

- [Bergadano et al. 96] F. Bergadano and D. Gunetti, Inductive Logic Programming - From Machine Learning to Software Engineering, MIT Press, 1996
- [Blockeel, De Raedt 98] H. Blockeel and L. De Raedt, Top-down Induction of First-order Logical Decision Trees, Artificial Intelligence, 101, 1998
- [Bratko, Muggleton 95] I. Bratko and S.H. Muggleton, Applications of Inductive Logic Programming, Communications of the ACM, 38(11):65-70, 1995
- [Cameron-Jones et al. 94] R. M. Cameron-Jones and J. Ross Quinlan, Efficient Top-down Induction of Logic Programs, SIGART, 5, 1994
- [De Raedt, Bruynooghe 93] L. De Raedt and M. Bruynooghe, A Theory of Clausal Discovery, Proceedings of the 13th International Joint Conference on Artificial Intelligence, 1993
- [De Raedt, Dehaspe 97] L. De Raedt and L. Dehaspe Clausal Discovery, Machine Learning, 26, 1997.
- [De Raedt, Van Laer 95] L. De Raedt and W. Van Laer, Inductive Constraint Logic, Proceedings of the 6th Conference on Algorithmic Learning Theory, 1995

Bibliography

- [Dolsak et al. 94] B. Dolsak, I. Bratko and A. Jezernik Finite Element Mesh Design: An Engineering Domain for ILP Application, Proceedings of the 4th International Workshop on Inductive Logic Programming, 1994
- [Dzeroski et al. 94] S. Dzeroski, L. Dehaspe, B. Ruck and W. Walley, Classification of river water quality data using machine learning, Proceedings of the 5th International Conference on the Development and Application of Computer Techniques to Environmental Studies, 1994
- [Dzeroski et al. 96] S. Dzeroski, S. Schulze-Kremer, K. Heidtke, K. Siems and D. Wettschereck, Applying ILP to diterpene structure elucidation from C NMR spectra, Proc. 6th International Workshop on Inductive Logic Programming, 1996
- [Dzeroski, Lavrac 01] S. Dzeroski and N. Lavrac, editors, Relational Data Mining Springer, Berlin, 2001
- [Finn et al. 98] P. Finn, S. Muggleton, D. Page and A. Srinivasan. Pharmacophore discovery using the inductive logic programming system Progol. Machine Learning, 30:241-271, 1998
- [King et al. 95] R. D. King, A. Srinivasan and M. J. E. Sternberg, Relating chemical activity to structure: an examination of ILP successes. New Gen. Comput., 1995

Bibliography

- [King et al. 96] R. D. King, S. H. Muggleton, A. Srinivasan and M. Sternberg, Structure-activity relationships derived by machine learning: the use of atoms and their bond connectives to predict mutagenicity by inductive logic programming, Proceedings of the National Academy of Sciences, 93:438-442, 1996
- [Lavrac, Dzeroski 94] N. Lavrac and S. Dzeroski, Inductive Logic Programming Techniques and Applications, Ellis Horwood, 1994
- [Muggleton 95] S. H. Muggleton, Inverse Entailment and Progol, New Gen. Comput., 13:245-286, 1995
- [Muggleton 99] S.H. Muggleton, Scientific knowledge discovery using Inductive Logic Programming. Communications of the ACM, 42(11):42-46, 1999
- [Muggleton, De Raedt 94] S.H. Muggleton and L. De Raedt, Inductive logic programming: Theory and methods, Journal of Logic Programming, 19,20:629-679, 1994
- [Muggleton, Feng 90] S. H. Muggleton and C. Feng, Efficient induction of logic programs, Proceedings of the 1st Conference on Algorithmic Learning Theory, 1990

Bibliography

- [Muggleton et al. 92] S. Muggleton, R. D. King, and M. J. E. Sternberg Predicting protein secondary structure using inductive logic programming, Protein Engineering, 5:647–657, 1992
- [Plotkin 70] G.D. Plotkin, A note on inductive generalisation, Machine Intelligence 5, Edinburgh University Press, 1970
- [Plotkin 71] G.D. Plotkin, Automatic Methods of Inductive Inference, PhD thesis, Edinburgh University, 1971
- [Quinlan 90] J. R. Quinlan, Learning logical definitions from relations, Machine Learning, 5:239– 266, 1990
- [Quinlan 91] J. R. Quinlan, Determinate literals in inductive logic programming, Proceedings of Twelfth International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 1991
- [Quinlan, Cameron-Jones 93] J. R. Quinlan and R. M. Cameron-Jones, FOIL: A Midterm Report, Proceedings of the 6th European Conference on Machine Learning, Springer-Verlag, 1993

Bibliography

- [Quinlan, Cameron-Jones 95] J. R. Quinlan, and R. M. Cameron-Jones, Induction of Logic Programs: FOIL and Related Systems, New Generation Comput. 13(3&4): 287-312, 1995
- [Riguzzi 06] F. Riguzzi, ALLPAD: Approximate Learning Logic Programs with Annotated Disjunctions, Inductive Logic Programming, 2006
- [Srinivasan et al. 97] A. Srinivasan, R.D. King, S.H. Muggleton and M. Sternberg. Carcinogenesis predictions using ILP, Proceedings of the Seventh International Workshop on Inductive Logic Programming, pages 273-287, 1997
- [Srinivasan et al. 95] A. Srinivasan, S.H. Muggleton and R.D. King, Comparing the use of background knowledge by inductive logic programming systems, Proceedings of the Fifth International Inductive Logic Programming Workshop, 1995
- [Turcotte et al. 01] M. Turcotte, S. Muggleton and M. J. E. Sternberg, The effect of relational background knowledge on learning of protein three-dimensional fold signatures, Machine Learning, 43(1/2):81–95, 2001
- [Van Laer et al. 97] W. Van Laer, L. De Raedt and S. Dzeroski, On Multi-class Problems and Discretization in Inductive Logic Programming, 10th International Symposium on Foundations of Intelligent Systems, ISMIS, 1997

Bibliography

- [Vennekens et al. 04] J.Vennekens, S. Verbaeten and M. Bruynooghe, Logic programs with annotated disjunctions, Proceedings of the Twentieth International Conference on Logic Programming, 2004