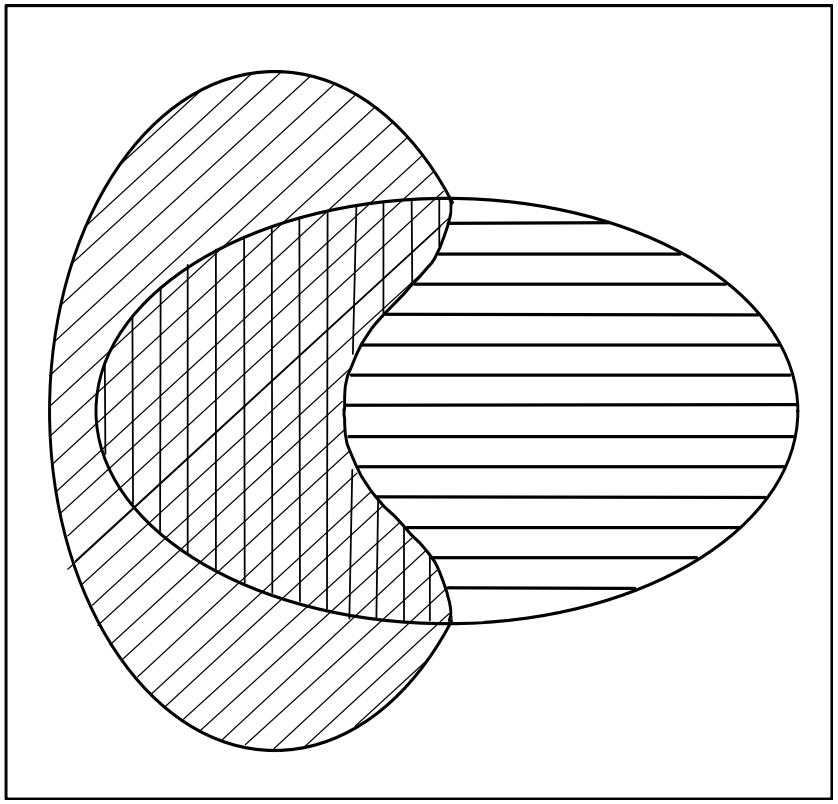
Introduction to Inductive Logic Programming

Luboš Popelínský
Fakulta informatiky
Masarykova universita v Brně,
Botanická 68a, 602 00 Brno

popel@fi.muni.cz
http://www.fi.muni.cz/~popel
http://www.fi.muni.cz/kd

In collaboration with Olga Štěpánková

Goal of inductive machine learning

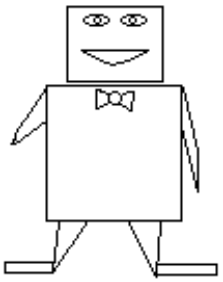
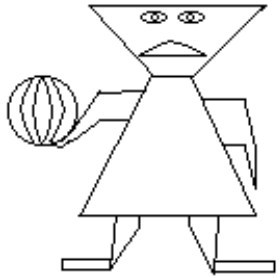
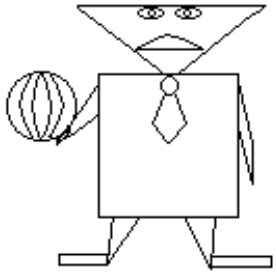
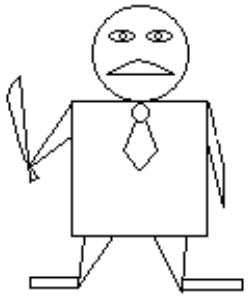


Given a finite subset of positive and negative examples $\mathbf{E}^+$ and $\mathbf{E}^-$, find a general description **H** of the whole set (a concept) so that

- **H** fits $\mathbf{E}^+$ best, and
- discriminates also other positive and negative examples

Example:

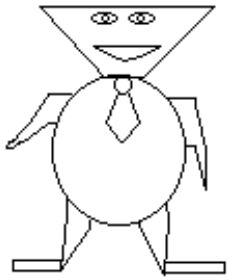
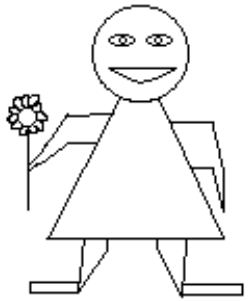
Can we recognize robots after short experience?

friendly

unfriendly



Example:
Robots and an atribute-value description

head	*smile*	*neck*	*body*	*In hand*	*friendly*
Circle	ne	Tie	Rectangle	Sword	no
Rectangle	ano	Butterfly	Rectangle	Nothing	yes
Circle	ne	Butterfly	Circle	Sword	yes
Triangle	ne	Tie	Rectangle	Ball	no
Circle	ano	Nothing	Triangle	Flower	no
Triangle	ne	Nothing	Triangle	Ball	yes
Triangle	ano	Tie	Circle	Nothing	no
Circle	ano	Tie	Circle	Nothing	yes

Example: hypothesis and testing

In the form of a decision tree

```
if  neck = butterfly   then  yes
          =  nothing   then
                       if  head  = triangle then yes
                                           else no
          = tie        then
                       if  body = rectangle  then  no
                                             else
                                             if head = circle then  yes
                                                               else  no
```

head	**smile**	**neck**	**body**	**in hand**	**friendly**
circle	no	tie	circle	sword	yes
triangle	yes	nothing	rectangle	nothing	yes

Example: hypothesis and testing (cont.)

Using a relation of equality

if neck = body then yes
else no

head	**smile**	**neck**	**body**	**in hand**	**friendly**
circle	no	tie	circle	sword	yes
triangle	yes	nothing	rectangle	nothing	no

Both trees classify the learning examples in the same way but they differ on testing set.

How may hypotheses can we have for a given learning set **E**?

- **Fact:** a number of concepts is much more that a number of possible hypotheses.
- **Consequence**: mostly we have to accept a hypothesis that is only "almost correct" (semi-optimal).
- Even in that case, we usualy have more then one hypothesis for a given **E**.

Hypothesis selection and **Ockham razer**

William of Ockham
(c. 1285- c.1349)
Franciscan Monk
Philosopher

Williamu of Ockham :
„*Entia non sunt multiplicanda praeter necessitatem*“,

- „ ***A number of entities should not be more than is necessary.***“
- **Einstein**: „… *but also not less*.“

Example 2: a need for domain knowledge

Pø.	139	319	854	468	349	561	756	789	987	256	189	354
Kl.	+	-	-	+	+	-	-	+	-	+	+	-

What hypothesis language to use?

E.g. attributes **c1**, **c2** and **c3** describe the 1st, 2nd and 3rd digit

Example 2: a need for domain knowledge

Př.	139	319	854	468	349	561	756	789	987	256	189	354
Kl.	+	-	-	+	+	-	-	+	-	+	+	-

Not enough

because classification depends on an order of digits

a relation of ordering is needed
= (apriori) domain knowledge

Result: **if c1 < c2 & c2 < c3 then '+'.**

Example 3: parity

Pø.	C1	C2	C3	C4	C5	C6	C7	C8	Kl.
1	1	0	1	1	0	0	0	0	-
2	1	0	1	1	0	0	0	1	+
3	1	1	1	1	0	0	0	1	-
4	0	1	1	1	0	0	0	1	+

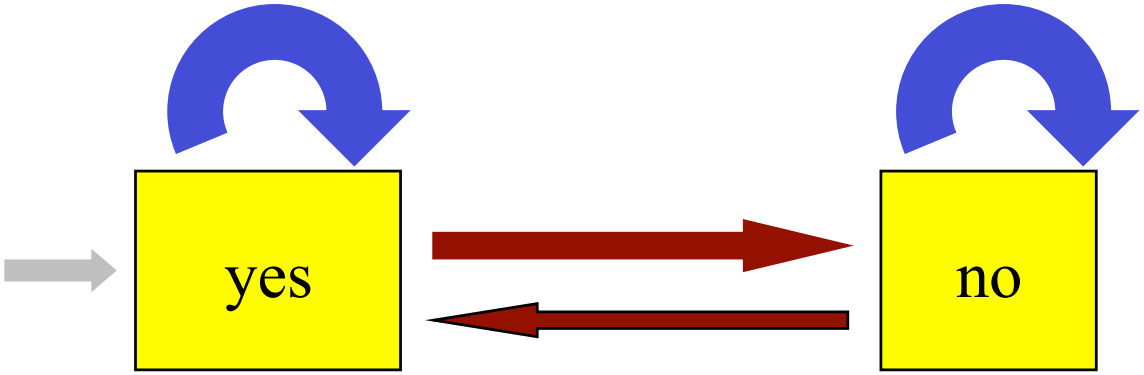
No attribute can be removed – all are important

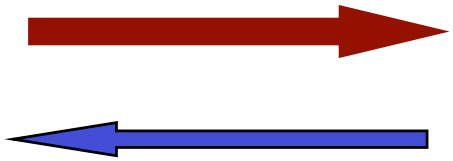
A decision tree would be too complex –

Why not an automaton?

Recursion is useful

Automaton that decides
„odd number of 1“





Input = **1**

Input = **0**

Selection of a good hypothesis

depends on a right

- language for representation of examples
- hypothesis language
- domain knowledge

When an attribute-value representation is insufficcient?

- Examples do not have a uniform description (e.g. are of a different length)
- A structure of examples is important
- Domain knowledge is (multi-)relational

Inductive logic programming: Basic task

(Muggleton94)

A set of positive E+ and negative E- examples
Domain knowledge B (a logic program)

goal: to find a logic program P that together with B covers
(almost all) positive examples and
not cover (almost no) negative example

+: much more flexible
data of any structure can be processed

-: some effort needed
more time consuming(even though << NeuroN)

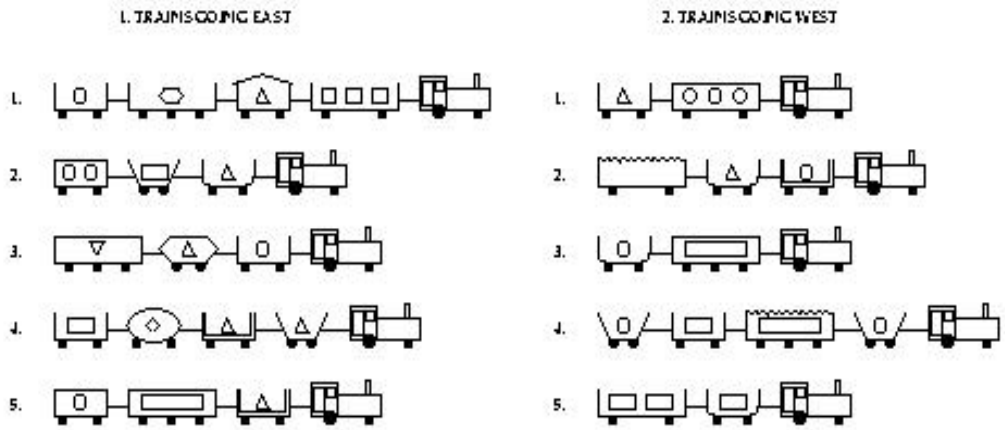
Inductive logic programming: Example

Example: find a path in an oriented graph

path(X,Y) :- edge(X,Y).
path(X,Y) :- path(X,U),edge(U,Y).

edge(1,2). edge(1,3). edge(2,3). edge(2,4). …
= domain knowledge

East-West Trains (1)



East-West Trains (2)

```
eastbound(east1).
eastbound(east2).
eastbound(east3).
eastbound(east4).
eastbound(east5).

eastbound(west6).
eastbound(west7).
eastbound(west8).
eastbound(west9).
eastbound(west10).
```

```
% eastbound train 1
short(car_12). closed(car_12).
long(car_11). open_car(car_11).
…
shape(car_11,rectangle). shape(car_12,rectangle).
…
load(car_11,rectangle,3). load(car_12,triangle,1).
…
wheels(car_11,2).  wheels(car_12,2).
…
has_car(east1,car_11).  has_car(east1,car_12).
has_car(east1,car_13). has_car(east1,car_14).
```

East-West Trains (3)

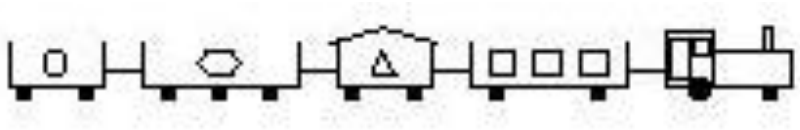
```
:- modeh(1,eastbound(+train)).
:- modeb(*,has_car(+train,-car)).
:- modeb(1,short(+car)).
:- modeb(1,load(+car,#shape,#int)).
...
:- determination(eastbound/1,has_car/2).
:- determination(eastbound/1,short/1).
:- determination(eastbound/1,load/3).
...
:- set( … ).
```

```
?- [aleph].
?- read_all(train).
?- induce.
…
   [Rule 1] [Pos cover = 5 Neg cover = 0]

   eastbound(A) :-
         has_car(A,B), short(B), closed(B).

         Actual
        +     -          Accuracy = 1.0
     + 5     0     5
Pred - 0     5     5     [time taken] [0.07]
       5     5    10     [total clauses constructed] [100]
```

East-West Trains (4)



[bottom clause][literals] [25][saturation time] [0.01]

```
eastbound(A) :-
   has_car(A,B), has_car(A,C), has_car(A,D), has_car(A,E),
   short(B), short(D), closed(D), long(C), long(E), open_car(B), open_car(C), open_car(E),
   shape(B,rectangle), shape(C,rectangle), shape(D,rectangle), shape(E,rectangle),
   wheels(B,2), wheels(C,3), wheels(D,2), wheels(E,2),
   load(B,circle,1), load(C,hexagon,1), load(D,triangle,1), load(E,rectangle,3).
```

[reduce]

```
eastbound(A).                                  [5/5]
eastbound(A) :- has_car(A,B).                  [5/5]
eastbound(A) :-
      has_car(A,B), short(B).                  [5/5]
…
eastbound(A) :-
     has_car(A,B),wheels(B,3).                 [3/1]
eastbound(A) :-
     has_car(A,B), closed(B).                  [5/2]
eastbound(A) :- has_car(A,B),
     load(B,triangle,1).                       [5/2]
...
```

…

```
eastbound(A) :- has_car(A,B), closed(B), shape(B,rectangle).
eastbound(A) :-   has_car(A,B), closed(B), wheels(B,2).
eastbound(A) :-has_car(A,B), closed(B), load(B,triangle,1). [2/0]
```

…

```
eastbound(A) :- has_car(A,B), short(B), closed(B).    [5/0]
```

Specialization and generalization

A formula G is a **specialization of** a formula G iff

F is a logical consequence of G

G |= F (any model of G is also a model of F).

Specialization operator (refinement operator)

assigns to a clause a set of all its specializations

Most of ILP systems use two basic operations of specialization

binding two variables

spec(path(X, Y)) = path(X, X)

adding a goal into a clause body

spec(path(X,Y)) = (path(X,Y):-edge(U,V))

and also

substitution a variable with a constant

spec(number(X)) = number(0)

substitution a variable with a most general term

spec(number(X) = number(s(Y)) .

Generic algorithm

```
QH :=  initialize(B; E-, E- ) ;

while not(stopping_kriterium(QH)) do
   remove H from QH ;
   choose_rules r1,…,rk from R ;
   applying r1,…,rk to H create a set H1;
   QH := (QH-H) ∪_H1 ;
   remove_some_elements from QH ;

choose_hypothesis P from QH
```

Example: path in a graph

Learning set

positive examples : path(1,2). path(1,3). path(1,4). path(2,3).

negative examples: path(2,1). path(2,5).

Specialization (refinement) tree

path(X,Y).

path(X,X). path(X,Y) :- edge(Z,U). path(X,Y):-path(Z,U).

path(X,Y) :- edge(X,U). path(X,Y):-path(X,U).

path(X,Y) :- edge(X,Y). …

path(X,Y):-path(X,U),edge(V,W).

path(X,Y):-path(X,U),edge(X,W).

…

path(X,Y):-path(X,U),edge(U,W).

…

path(X,Y):-path(X,U),edge(U,Y).

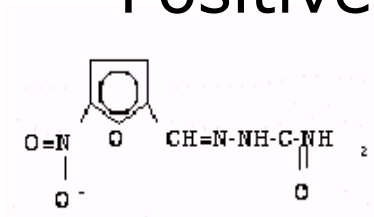
Succesful ILP applications

Bioinformatics

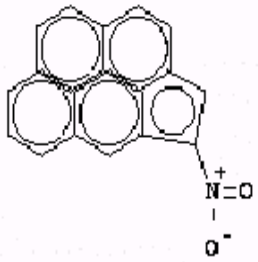
- **Structure Activity Relationships** (SAR): given
  - Chemical structure and
  - Empirical data about toxicity/ mutageneticity/ effect in terapy

- **Goal: to find a reason for the effect**

Positive

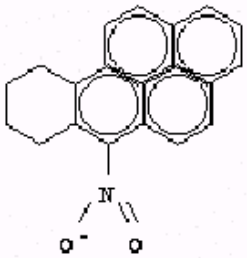


nitrofurazone

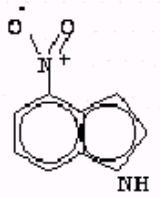


4-nitropenta[cd]pyrene

Negative

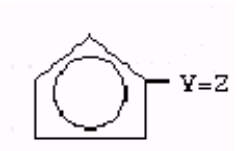


6-nitro-7,8,9,10-tetrahydrobenzo[a]pyrene

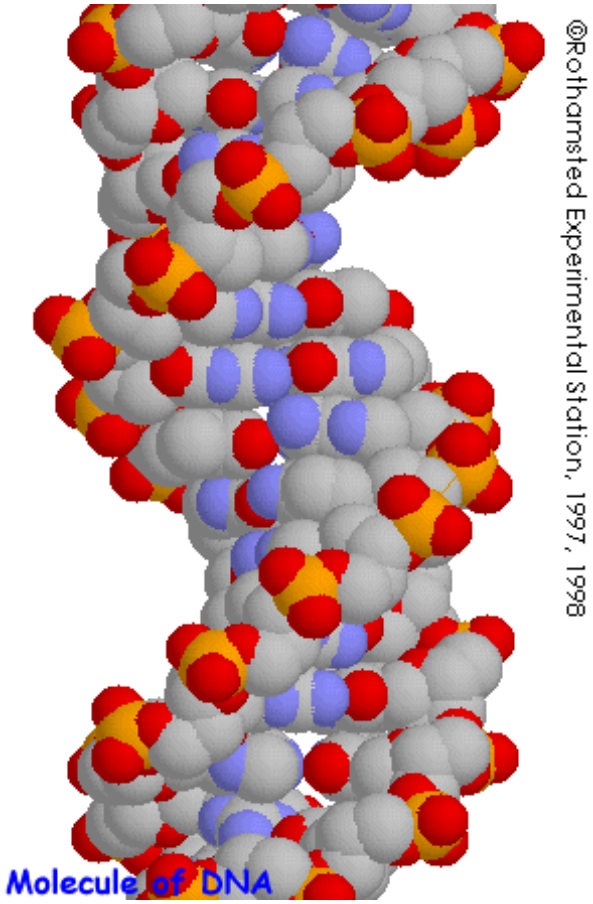


4-nitroindole

Result: structural indicator



Bioinformatics - prostor. uspořádání bílkovin



Bílkoviny = řetězce aminokyselin tvořících složité prostor. útvary.

- Posloupnost aminokyselin = **primární struktura**.
- *Lze předpovědet prostorovou strukturu molekuly na základě info. o její primární struktuře?*
- **Interpretace NMR spektra** - rozdělení do 23 strukturních typů. Klasické metody - 80% uspěšnost, **ILP 90%** - odpovídá výkonu zkušeného odborníka

Bioinformatika - karcinogenicita

- 230 aromatických a heteroaromatických dusíkatých sloučenin

  188 sloučenin (lze je dobře klasifikovat regresí v rámci atributové reprezentace)

  + 42 RU sloučenin (regression-unfriendly skupina).

- Na RU skupině se prokázaly výhody relační reprezentace:

  Hypotéza navržená PROGOLem dosahovala přesnosti 88%

  zatímco klasické metody asi o 20 % méně.