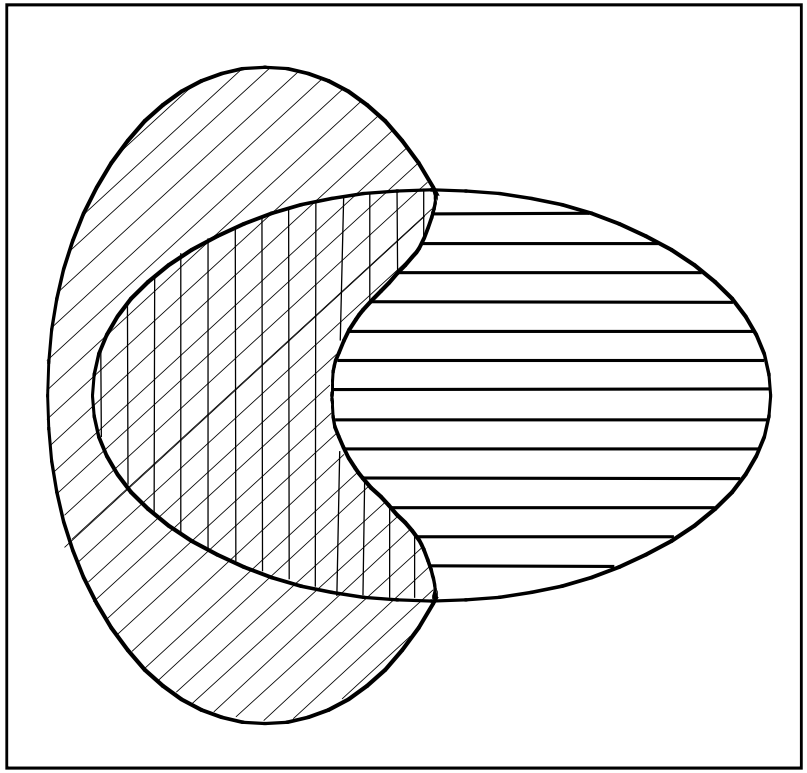
Induktivní logické programování

společně s Olgou Štěpánkovou
Katedra kybernetiky, FEL ČVUT Praha
step@labe.felk.cvut.cz

Cíl induktivního strojového učení



Na základě omezeného vzorku příkladů $\mathbf{E}^+$ a $\mathbf{E}^-$, charakterizovat (popsat) zamýšlenou skupinu objektů (koncept) tak, aby

- popis co nejlépe odpovídal právě prvkům z $\mathbf{E}^+$
- byl použitelný pro určení i objekty mimo E

Příklad 2: význam doménové znalosti

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

Jaký jazyk pro popis dat zvolit?

Např. atributy **c1**, **c2** a **c3**, které označují první, druhou a třetí číslici ve trojici.

Příklad 2: význam doménové znalosti

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

Jakýjazyk pro popis dat zvolit?

Např. atributy **c1**, **c2** a **c3**, které označují první, druhou a třetí číslici ve trojici.

Klasifikace souvisí s uspořádáním číslic
relace uspořádání = apriorní (nebo doménová) znalost

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

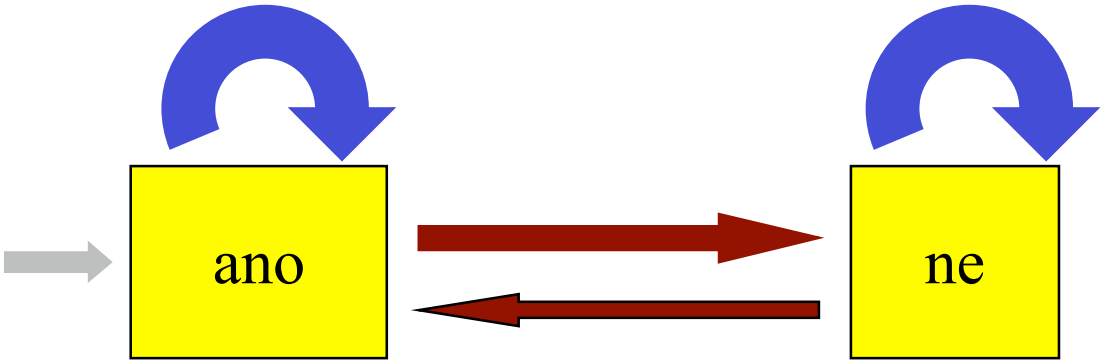
Příklad 3: parita

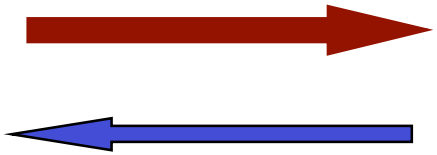
Př.	C1	C2	C3	C4	C5	C6	C7	C8	Kl.
1	1	0	1	1	0	0	0	0	-
2	1	0	1	1	0	0	0	1	+
3	1	1	1	1	0	0	0	1	-
4	0	1	1	1	0	0	0	1	+

Není atribut, který by bylo možné vynechat - na všech záleží !!!
Rozhodovací strom by byl velmi komplikovaný –

Pročn e automat?
Význam formátu hypotézy – výhodná by byla rekurze !!

Automat rozhodující úlohu „**sudý počet symbolů 1**“





Na vstupu je symbol **1**

Na vstupu je symbol **0**

Podmínky nalezení dobré hypotézy

Záleží na vhodné volbě

- jazyka reprezentace příkladů
- jazyka pro formulaci hypotéz
- doménové znalosti

Kdy nestačí atributové vyjádření?

- popis příkladů nemá uniformí tvar (def. obor tvoří slova různé délky)
- struktura jednotlivých příkladů má rozhodující charakter
- doménová znalost je výrazně relační

Induktivní logické programování(1)

- příklady jsou složeny z různého počtu elementů, mezi nimiž jsou vztahy, které jsou podstatné pro příslušnost (např. koncept “oblouk” v doméně objektů tvořených z dětských kostek)
- není možné (nebo je velmi nepřirozené) popisovat všechny trénovací příklady jednotně prostřednictvím jediného souboru atributů (universum tvoří slova různé délky)
- potřebná apriorní znalost má výrazně relační charakter (např. rodič(X,Y), hrana v grafu atd.)

Induktivní logické programování(2)

ILP

formuluje hypotézy pro zkoumané koncepty pomocí jazyka predikátové logiky

hypotézu tvoří konečná množina klauzulí odpovídající logickému programu - nejčastěji v Prologu

rozšíření o modální operátory; deskripční logika; pravděpodobnostní ILP

Induktivní logické programování(3)

(Muggleton94)

množina pozitivních E+ a negativních E- příkladů
doménová znalost B (logický program)

cíl: najít logický program P, který spolu s B pokrývá
téměř všechny pozitivní příklady a
nepokrývá téměř žádný z negativních příkladů

výhody: flexibilnější (doménová znalost, proměnná délka kontextu,pořadí slov)
nevýhoda: výpočty časově náročnější (i když << NeuroN)

Induktivní logické programování(4)

P (výsledek učení) i B (doménová znalost)
se skládají z logických formulí

$A :- A_1 , \ldots , A_n ,$

kde A, A_i jsou literály, čárka znamená logickou konjunkci, :- implikaci

Příklad: cesta v orientovaném grafu

cesta(X,Y) :- hrana(X,Y). = program P
cesta(X,Y) :- cesta(X,U),hrana(U,Y).

hrana(1,2). hrana(1,3). hrana(2,3). hrana(2,4). … = doménová znalost

Základní úloha ILP

Pro dané množiny pozitivních a negativních příkladů E^+ a E^- a množinu axiomů B takových, že

Apriorní bezespornost: $\forall e \in E^- : B \not\vdash e$

Apriorní nutná podmínka: $\exists e \in E^+ : B \not\vdash e$

hledáme P takové, že

Aposteriorní úplnost: $\forall e \in E^+ : B \cup P \vdash e$

Aposteriorní bezespornost: $\forall e \in E^- : B \cup P \not\vdash e$

Specializace a generalizace

hypotéza F je **specializací** G, právě když F je logickým důsledkem G

G |= F (libovolný model G je i modelem F).

Specializační operátor

přiřazuje každé klauzuli množinu jejích specializací.

Většina ILP systémů používá dvě základní operace specializace

ztotožnění 2 proměnných

spec(cesta(X, Y)) = cesta(X, X)

přidání podcíle do těla formule

spec(cesta(X,Y)) = (cesta(X,Y):-hrana(U,V))

nahrazení proměnné konstantou

spec(číslo(X)) = číslo(0)

nahrazení proměnné složeným termem

spec(číslo(X) = číslo(s(Y)) .

Generický algoritmus ILP

```
QH :=  inicializuj(B; E⁻, E⁻ ) ;

while not(kriterium_ukončení(QH)) do
   vyjmi H z QH ;
   zvol_odvozovací_pravidla r1,…,rk z R ;
   aplikací r1,…,rk na H vytvoř množinu H1;
   QH := (QH-H) ∪ H1 ;
   zruš_některé_prvky z QH ;

vyber_hypotézu P z QH
```

Příklad: Cesta v grafu

Učicí množina

Pozitivní příklady : cesta(1,2). cesta(1,3). cesta(1,4). cesta(2,3).

Negativní příklady: cesta(2,1). cesta(2,5).

Specializační strom

cesta(X,Y).

cesta(X,X). cesta(X,Y) :- hrana(Z,U). cesta(X,Y):-cesta(Z,U).

cesta(X,Y) :- hrana(X,U). cesta(X,Y):-cesta(X,U).

cesta(X,Y) :- hrana(X,Y). …

cesta(X,Y):-cesta(X,U),hrana(V,W).

cesta(X,Y):-cesta(X,U),hrana(X,W).

…

cesta(X,Y):-cesta(X,U),hrana(U,W).

…

cesta(X,Y):-cesta(X,U),hrana(U,Y).

Systémy

Aleph (dříve P-Progol), Oxford University
FOIL (Quinlan 1993)
MIS (Shapiro 1981), Markus (Grobelnik 1992), WiM (1994)

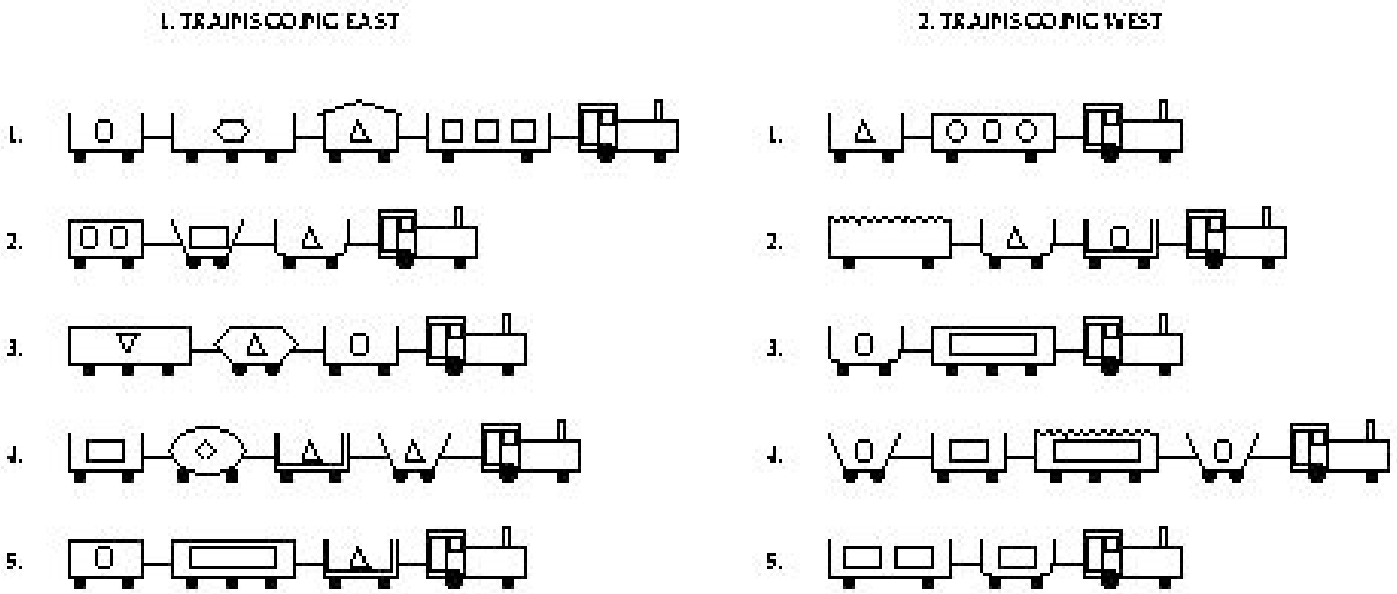
RAP (Blaťák 2003) učení častých vzorů
Tilde + WARMR = ACE (Blockeel, De Raedt 1998)

Další systémy: http://www-ai.ijs.si/~ilpnet2/systems/

Aleph

- vyber z učicí množiny jeden nebo více pozitivních příkladů
- najdi jejich nejmenší generalizaci vhledem k dané doménové znalosti
- z literálů vyskytujících se v této generalizaci vytvoř pomocí heuristického hledání(metodou shora-dolů, od nejkratší klauzule) takovou klauzuli, která nejlépe pokrývá pozitivní příklady a je co nejméně nekonzistentní - pokrývá minimum negativních příkladů
- tuto klauzuli přidej k dosud nalezeným
- odstraň všechny příklady, které jsou nově pokryty dosud nalezeným řešením (tzv. pokrývací paradigma)
- celý proces opakuj tak dlouho, dokud nejsou všechny pozitivní příklady (případně až na malý počet) pokryty a není pokryt žádný negativní (případně až na malý počet) pokryt

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)

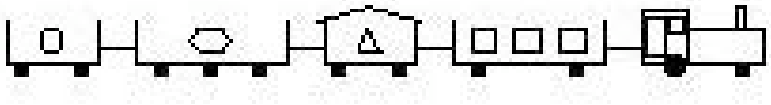
```
:- modeh(1,eastbound(+train)).
:- modeb(*,has_car(+train,-car)).
:- modeb(1,short(+car)).
:- modeb(1,load(+car,#shape,#int)).
...
:- determination(eastbound/1,has_car/2).
:- determination(eastbound/1,short/1).
:- determination(eastbound/1,load/3).
...
:- set( … ).
```

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

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

            Actual
          +     -
      +   5     0      5      Accuracy = 1.0
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]
```

Úspěšné ILP aplikace

- A., kde ILP dosáhlo mimořádně dobrých výsledků, které vzbudily zájem odborné veřejnosti nejen v komunitě, která se věnuje strojovému učení, ale i v kruzích odborníků z oblasti aplikace

- A., které jsou nezvyklé z hlediska použití metod strojového učení.

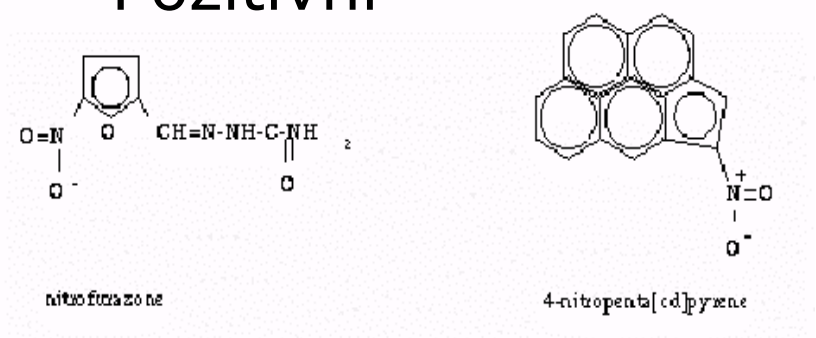
- Bioinformatika, medicína, životní prostředí
- Technika
- Zpracování přirozeného jazyka

Bioinformatika - úloha SAR

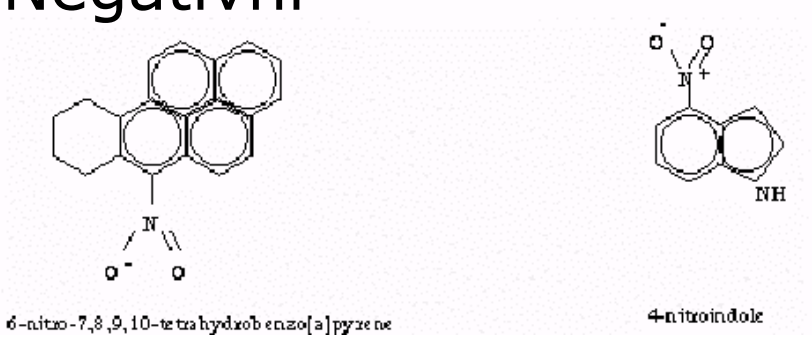
- **Structure Activity Relationships** (SAR): je známa
  - chem.struktura látky a
  - empirické údaje o její toxicitě/ mutageneticitě/ terapeutickém účinek.

- **Co je příčinou pozorovaného chování**?

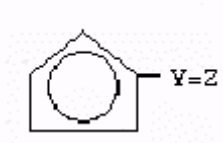
Pozitivní



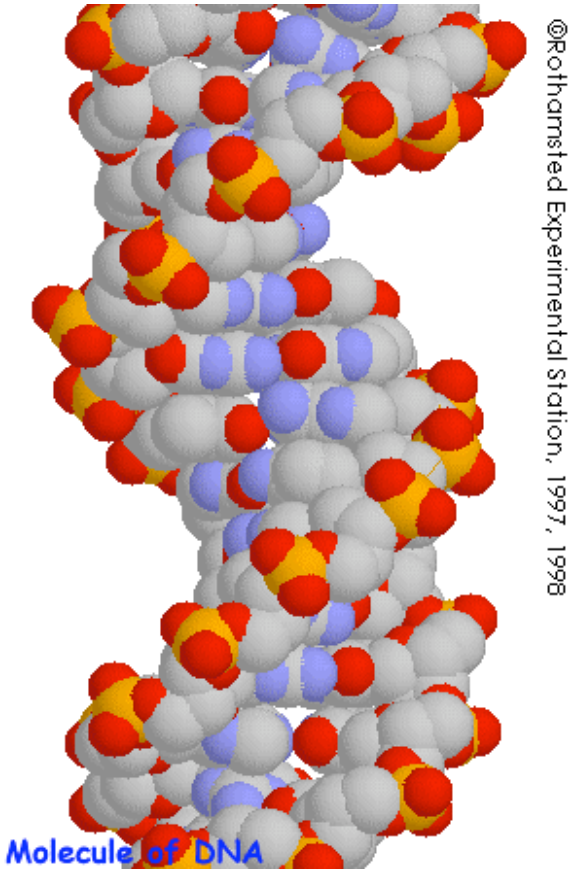
Negativní



Výsledek: strukturární indikátor



Bioinformatika - 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.

Morfologická desambiguace češtiny

Učicí data

jednoznačně/víceznačně označkovaná
selektivní vzorkování (Nepil et al.01)
bez ručního značkování (Šmerk03)

Doménová znalost

délka kontextu – počet slov nutných pro klasifikaci
pozice slov v kontextu
predikáty popisující vlastnosti slov a jejich kategorií
p(Kontext, PodčástKontextu, Predikát)

Příklad: se - buď zvratné zájmeno nebo předložka

```
zájmeno(Left,Right) :-
  p(PravýKontext, nejbližších_slov(1), vždy(k6)),
  p(LevýKontext,nejbližších_slov(2),někdy([k5,aI,eA])).
```