

Induktivní logické programování

společně s Olgou Štěpánkovou

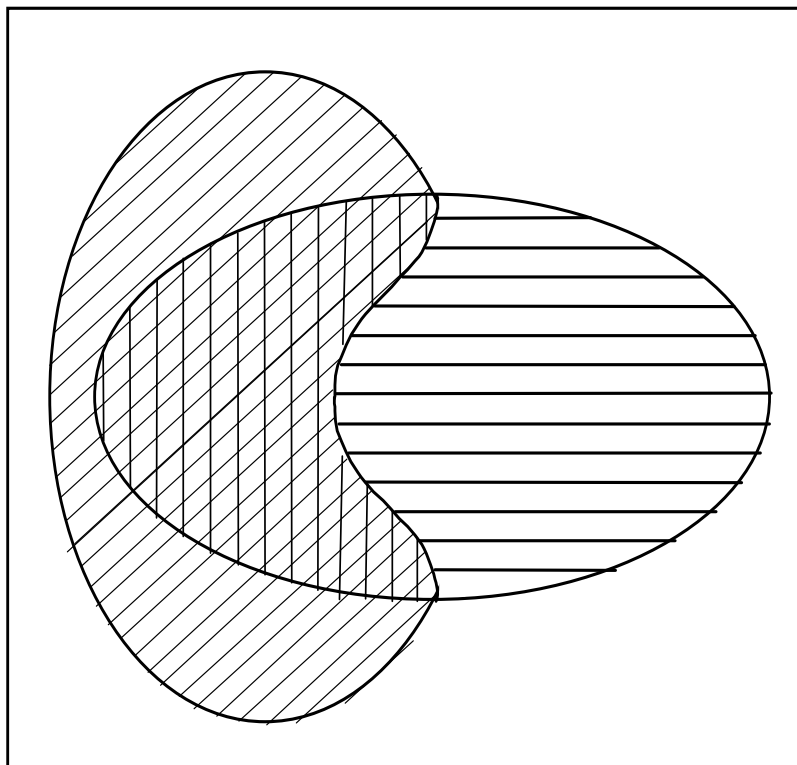
Katedra kybernetiky, FEL ČVUT Praha

step@labe.felk.cvut.cz

ILP a indukativní strojové učení

- Strojové učení: cíle, pojmy a metody
- Problémy reprezentace
- Meze klasických metod - příklady
- Princip ILP a generický ILP algoritmus
- Existující systémy a příklady použití
- Zajímavé aplikace

Cíl induktivního strojového učení

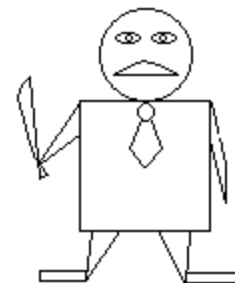
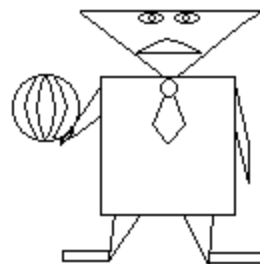
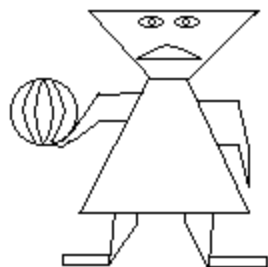
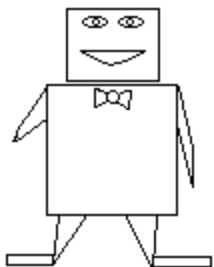


- Na základě omezeného vzorku příkladů E^+ a E^- , charakterizovat (popsat) zamýšlenou skupinu objektů (koncept) tak, aby
- popis co nejlépe odpovídal právě prvkům z E^+
 - byl použitelný pro určení i objekty mimo E

Základní pojmy

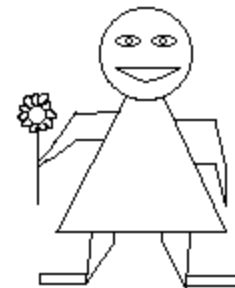
- Necht' Ω je definiční obor konceptu \mathbf{K} , tj. $\mathbf{K} \subseteq \Omega$.
- E je množina trénovacích příkladů (opět $E \subseteq \Omega$) doplněná klasifikací cl těchto příkladů, tedy $cl: E \rightarrow \{ano, ne\}$.
- E^+ jsou prvky E s klasifikací *ano*
- E^+ a E^- tvoří disjunktní rozklad (pokrytí) množiny E

Příklad 1 „počítačová hra“. Můžeme se naučit roboty rozlišit na základě krátké zkušenosti?

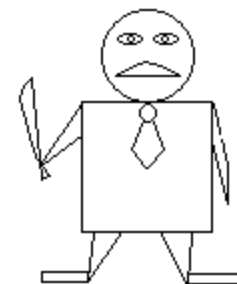
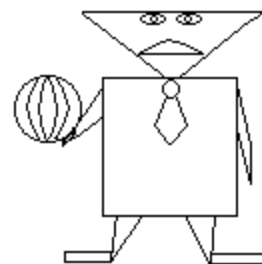
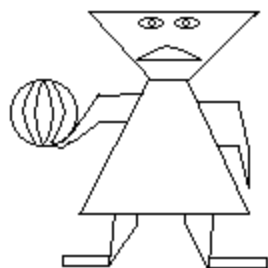
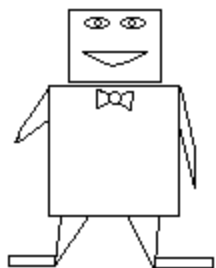


přátelští

nepřátelští

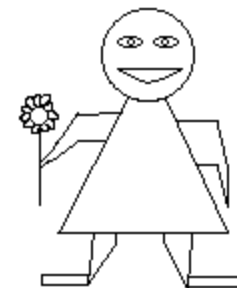


Příklad 1 „počítačová hra“. Můžeme se naučit roboty rozlišit na základě krátké zkušenosti?



přátelští

nepřátelští



Příklad 1: Roboti a atributový popis

<i> tvar hlavy </i>	<i> úsměv </i>	<i> ozdoba krku </i>	<i> tvar těla </i>	<i> předmět v ruce </i>	<i> přátelský </i>
Kruh	ne	kravata	čtverec	šavle	ne
Čtverec	ano	motýlek	čtverec	nic	ano
Kruh	ne	motýlek	Kruh	šavle	ano
Trojúhelník	ne	kravata	čtverec	balón	ne
Kruh	ano	nic	trojúhelník	květina	ne
Trojúhelník	ne	nic	trojúhelník	balon	ano
Trojúhelník	ano	kravata	Kruh	nic	ne
Kruh	ano	kravata	Kruh	nic	ano

Příklad 1: hypotéza a její testování

H1 ve tvaru rozhodovacího stromu

```
if ozdoba_krku(r) = motýlek then „přátelský robot“
  = nic then
    if tvar_hlavy(r) = trojúhelník then „přátelský robot“
      else „nepřátelský robot“
    = kravata then
      if tvar_těla(r) = čtverec then „nepřátelský robot“ else
        if tvar_hlavy(r) = kruh then „přátelský robot“
          else „nepřátelský robot“
```

Tvar hlavy	ús měv	Ozdoba krku	Tvar těla	předmět	Přátelský?
kruh	ne	kravata	kruh	šavle	ano
trojúhelník	ano	nic	čtverec	nic	ano

Příklad 1: hypotéza a její testování

H2 s relací rovnosti

if tvar_hlavy (r) = tvar_těla(r) then „přátelský robot“
else „nepřátelský robot“

Tvar hlavy	úsměv	Ozdoba krku	Tvar těla	předmět	Přátelský?
kruh	ne	kra vata	kruh	šavle	ano
troj úhelník	ano	nic	čtverec	nic	ne

H1 i **H2** klasifikují správně data v trénovacích příkladech, ale liší se na testovací množině

Hypotéza - pokus o formální popis konceptu

Pro popis příkladů i hypotéz potřebujeme **jazyk**. Hypotéza je pak vyjádření v odpovídajícím jazyce pomocí formule $\varphi(\mathbf{X})$ s jedinou volnou proměnnou \mathbf{X} .

Definujme extenzi \mathbf{Ext}_φ hypotézy $\varphi(\mathbf{X})$ vzhledem k definičnímu oboru Ω = množina všech prvků Ω , které splňují φ , tj. $\mathbf{Ext}_\varphi = \{ \mathbf{o} \in \Omega : \varphi(\mathbf{o}) \text{ platí} \}$

Vlastnosti hypotéz

- hypotéza φ je **úplná**, pokud $\mathbf{E}^+ \subseteq \mathbf{Ext}_\varphi$ ψ je **korektní** vzhledem k množině
- h. ψ je **konsistentní**, pokud není splněna pro žádný negativní trénovací příklad, tj. $\mathbf{Ext}_\psi \cap \mathbf{E}^- = \emptyset$
- h. ψ je **korektní**, pokud je úplná a konsistentní

Kolik korektních hypotéz lze navrhnout pro danou trénovací množinu E ?

- **Fakt:** možných konceptů je nesrovnatelně víc než možných hypotéz
- **Důsledek:** pro většinu konceptů se musíme smířit s popisem pomocí hypotézy, která je pouze "skoro správná".
- Ani pro "skoro správné" hypotézy neplatí, že pro danou E existuje jediná vhodná h .

Výběr hypotézy a Ockhamova břitva



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

- Williamu of Ockham** navrhl srovnávací měřítko pro kvalitu navržených hypotéz: „*Entia non sunt multiplicanda praeter necessitatem*“,
- „*Entit by nemělo být víc, než kolik je nezbytně nutné*“
 - **Einstein:** „... *ale také ne méně.*“

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.	+	-	-	+	+	-	-	+	-	+	+	-

Jaký jazyk pro popis dat zvolit?

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

Klasifikace souvisí s uspořádáním čísel
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	+

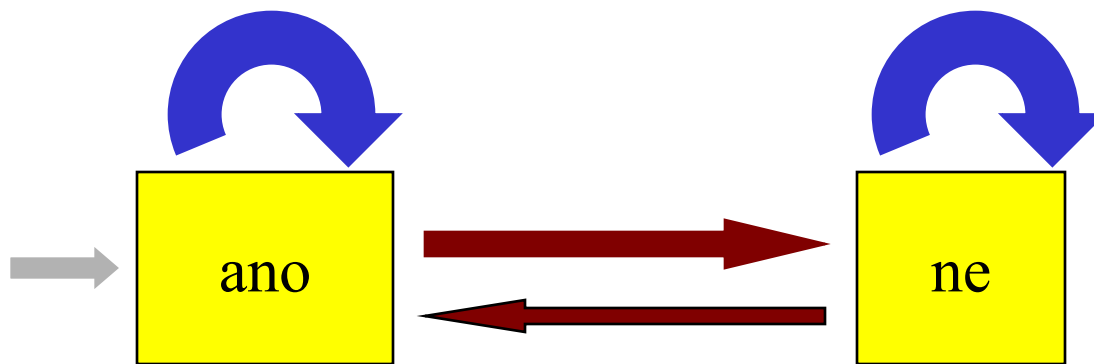
Není a tribut, který by bylo možné vynechat - na všech záleží !!!

Rozhodovací strom by byl velmi komplikovaný –

Proč není automat?

Význam formální 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 uniformní 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 nepřírozené) 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 \ll NeuroN)

Induktivní logické programování(4)

P (výsledek učení) i B (doménová znalost)

se skládají z logických formulí

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

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
 $G \models F$ (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

$$\text{spec}(\text{cesta}(X, Y)) = \text{cesta}(X, X)$$

přidání podcíle do těla formule

$$\text{spec}(\text{cesta}(X, Y)) = (\text{cesta}(X, Y) : \text{-hrana}(U, V))$$

nahrazení proměnné konstantou

$$\text{spec}(\text{číslo}(X)) = \text{číslo}(0)$$

nahrazení proměnné složeným termem

$$\text{spec}(\text{číslo}(X)) = \text{číslo}(s(Y)) .$$

Generický algoritmus ILP

$QH := \text{inicializuj}(B; E^-, E^-) ;$

while not($kriterium_ukončení(QH)$) **do**

$vyjmi$ H z QH ;

$zvol_odvozovací_pravidla$ r_1, \dots, r_k z R ;

$aplikací$ r_1, \dots, r_k na H vytvoř množinu H_1 ;

$QH := (QH - H) \cup H_1$;

$zruš_některé_prvky$ z QH ;

$vyber_hypotézu$ P z QH

Systemy

Aleph (dřívě P-Progol), Oxford University

FOIL (Quinlan 1993)

MIS (Shapiro 1981), Markus (Grobelnik 1992), WiM (1994)

RAP (Blat'ák 2003) učení častých vzorů

Tilde + WARMR = ACE (Blockeel, De Raedt 1998)

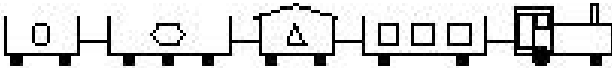



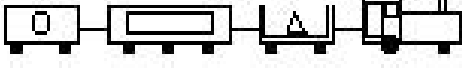
Další systémy: <http://www-ai.ijs.si/~ilpnet2/systems/>

Aleph




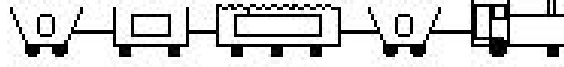
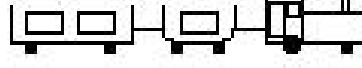
- vyber z učicí množiny jeden nebo více pozitivních příkladů
- najdi jejich nejmenší generalizaci vzhledem k dané doménové znalosti
- z literálů vyskytujících se v této generalizaci vytvoř pomocí heuristického hledání (metodou shora-dolů, od nejkratší klauzule) takovou klauzuli, která nejlépe pokrývá 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)

1. TRAINS GOING EAST

1. 
2. 
3. 
4. 
5. 

2. TRAINS GOING WEST

1. 
2. 
3. 
4. 
5. 

East-West Trains (2)



```
eastbound(east1).      % eastbound train 1
eastbound(east2).      short(car_12). closed(car_12).
eastbound(east3).      long(car_11). open_car(car_11).
eastbound(east4).      ...
eastbound(east5).      shape(car_11,rectangle). shape(car_12,rectangle).
                        ...
eastbound(west6).      load(car_11,rectangle,3). load(car_12,triangle,1).
eastbound(west7).      ...
eastbound(west8).      wheels(car_11,2). wheels(car_12,2).
eastbound(west9).      ...
eastbound(west10).     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]

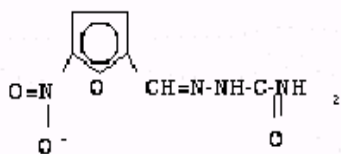
Úspěšné ILP aplikace

- 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

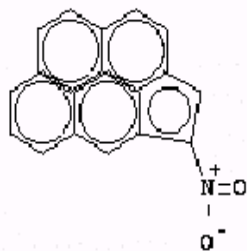
Bioinformatika - úloha SAR

- **Structure Activity Relationships (SAR):** je známa
 - chem.struktura látky a
 - empirické údaje o její toxicitě/ mutagenicitě/ terapeutickém účinek.
- **Co je příčinou pozorovaného chování?**

Pozitivní

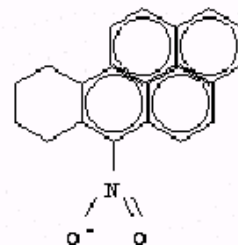


nitrofurazone

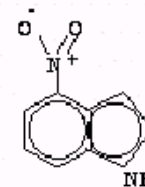


4-nitropenta[cd]pyrene

Negativní



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

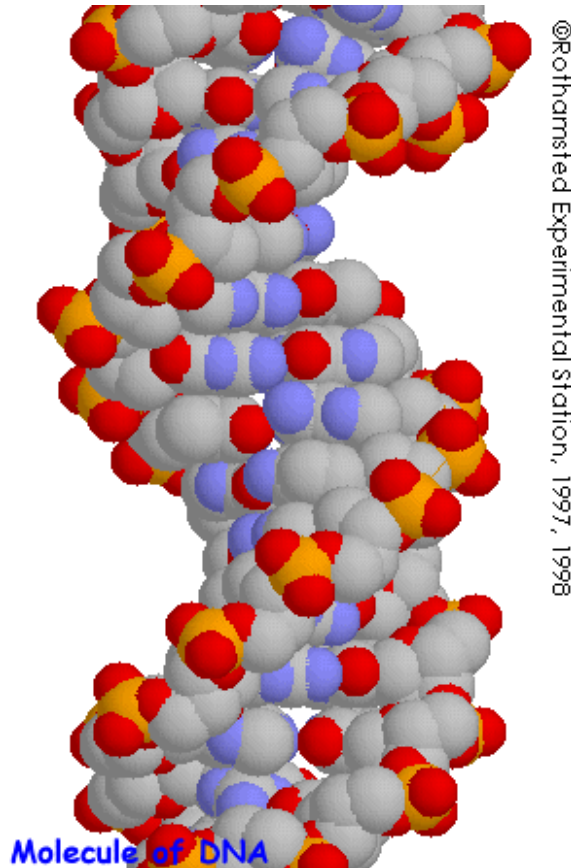


4-nitroindole

Výsledek: strukturální indikátor



Bioinformatika - prostor. uspořádání bílkovin



Bílkoviny = řetězce aminokyselin tvořících složité prostor. útvary.

- Posloupnost aminokyselin = **primární struktura**.
- *Lze předpovědět prostorovou strukturu molekuly na základě info. o její primární struktuře?*
- **Interpretace NMR spektra** - rozdělení do 23 strukturních typů. Klasické metody - 80% úspěšnost, **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 označovaná
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 zvrtné zájmeno nebo předložka

zájmeno (Left, Right) :-

```
p(PravýKontext, nejbližších_slov(1), vždy(k6)),  
p(LevýKontext, nejbližších_slov(2), někdy([k5, aI, eA])).
```