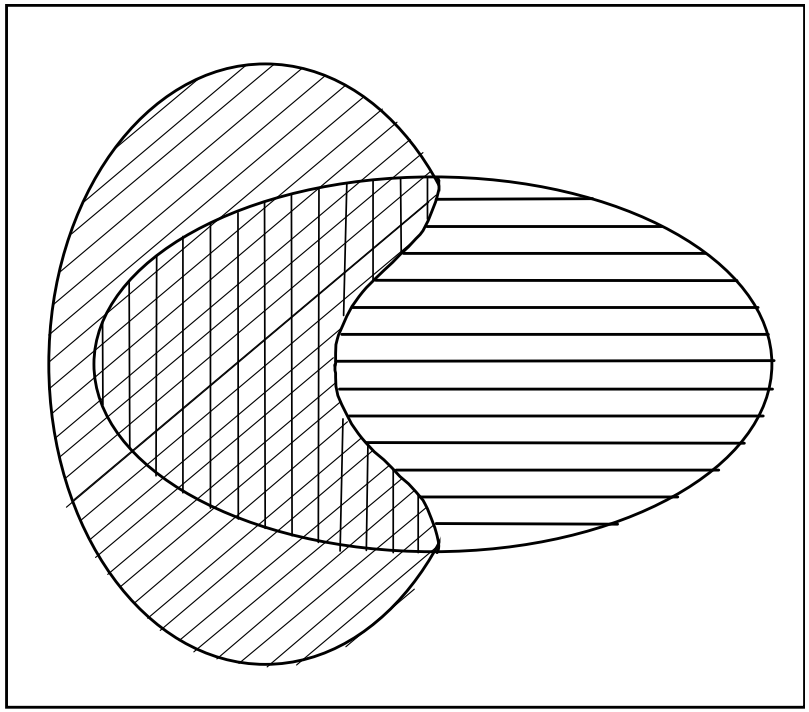
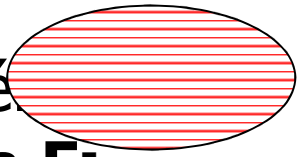
ILP a induktivní strojové učení

Osnova

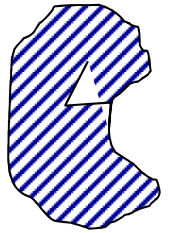
- Strojové učení: cíle, pojmy a metody
- Problémy reprezentace
- Meze klasických metod - příklady
- Princip ILP a generický ILP algoritmus
- Existující systémy a příklady použití
- Zajímavé aplikace

Cíl induktivního strojového učení





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

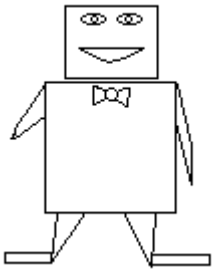
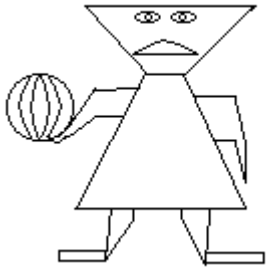


- co nejlépe odpovídal právě jen prvkům z $\mathbf{E^+}$
- byl použitelný pro určení i objekty mimo **E**

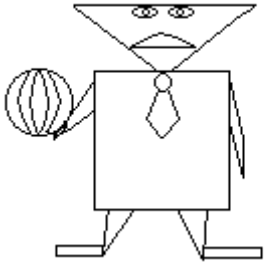
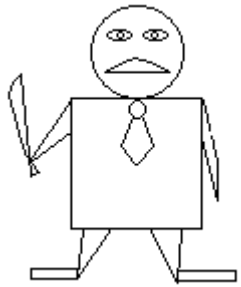
Základní pojmy

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

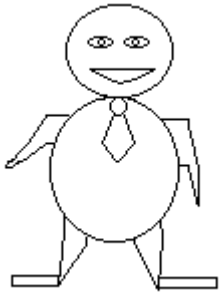
Příklad 1 „počítačová hra“. Můžeme se naučit roboty rozlišit na základě krátké zkušenosti?

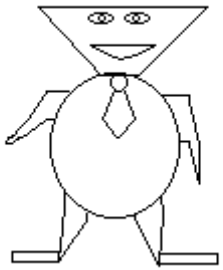
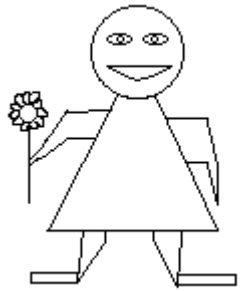



přátelství

nepřátelství

Příklad 1: Roboti a atributový popis

tvar hlavy	*úsměv*	*ozdoba krku*	*tvar těla*	*předmět v ruce*	*přátelský*
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	úsměv	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	kravata	kruh	šavle	ano
trojúhelník	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 pokus charakterizovat prvky konceptu v odpovídajícím jazyce pomocí formule φ**(X)** s jedinou volnou proměnnou **X**.

Definujme **pokrytí** (extenzi) $\mathbf{Ext}_{\varphi}$ hypotézy φ**(X)** vzhledem k definičnímu oboru Ω jako množinu všech prvků Ω, které splňují φ, tj.

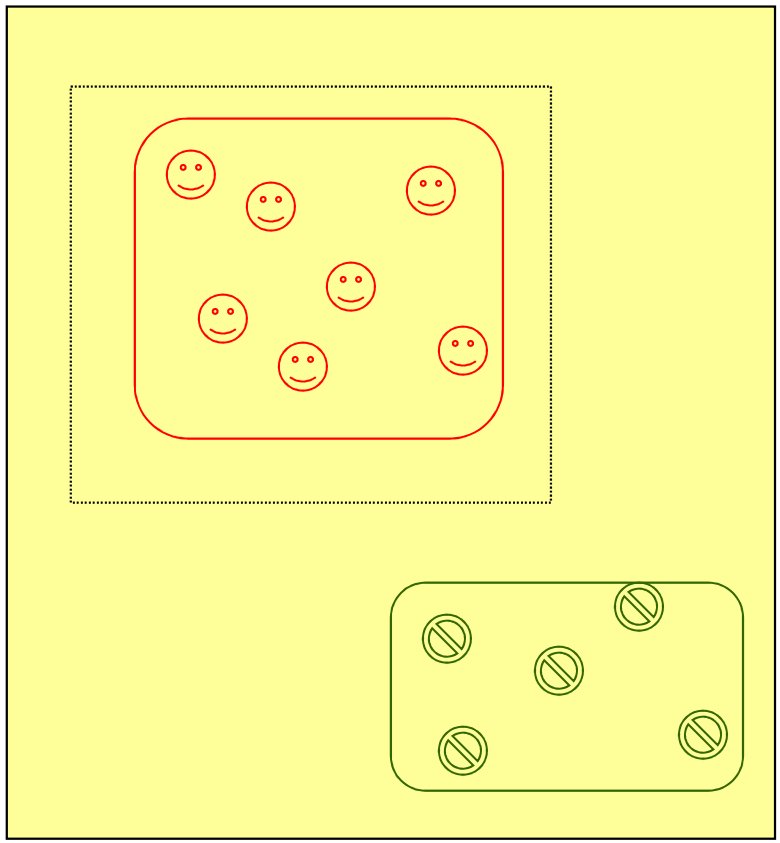
$$\mathbf{Ext}_{\varphi} = \{\mathbf{e} \in \Omega : \varphi(\mathbf{e}) \text{ platí } \}$$

Vlastnosti hypotéz

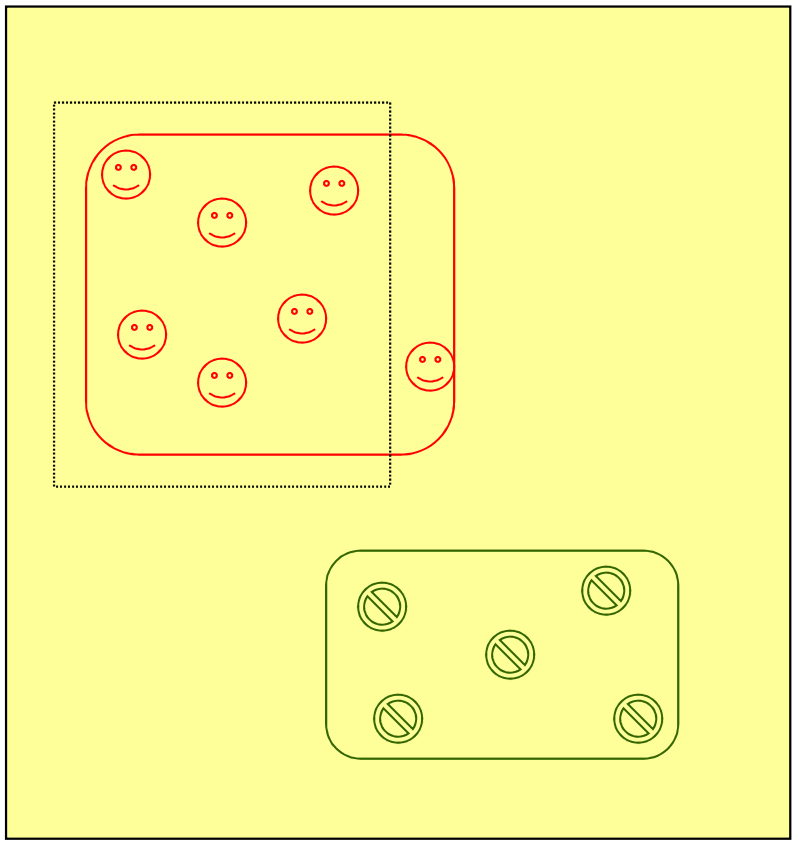
- hypotéza φ je **úplná**, pokud $\mathbf{E}^{+} \subseteq \mathbf{Ext}_{\varphi}\ \psi$
- h. ψ je **konsistentní** (bezesporná), pokud není splněna pro žádný negativní trénovací příklad, tj. $\mathbf{Ext}_{\psi} \cap \mathbf{E}^{-} = \varnothing$

Konzistentní hypotéza

úplná (complete)

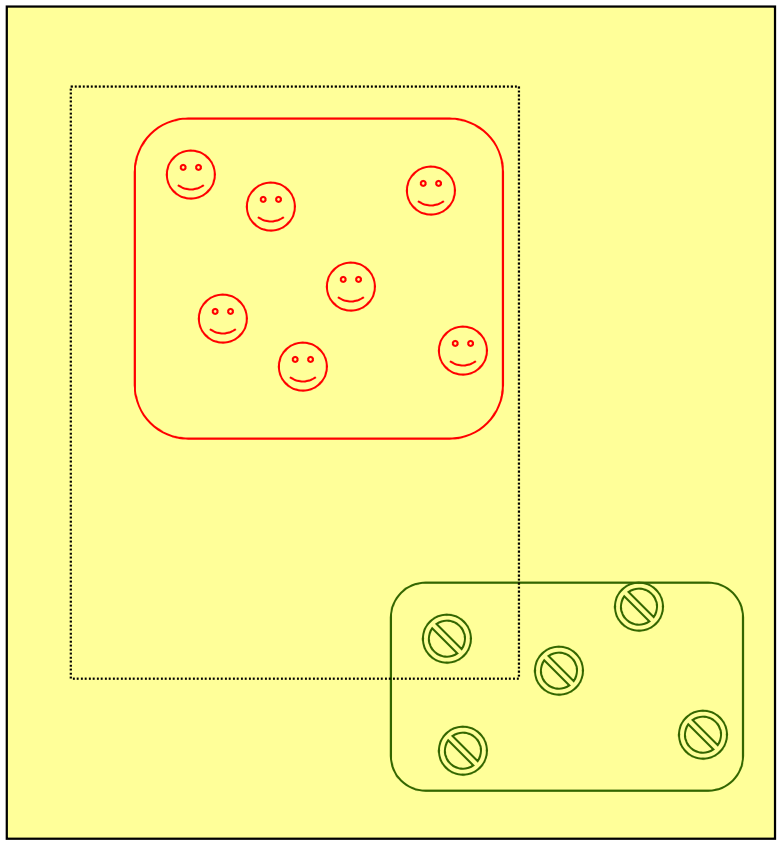


neúplná (incomplete)

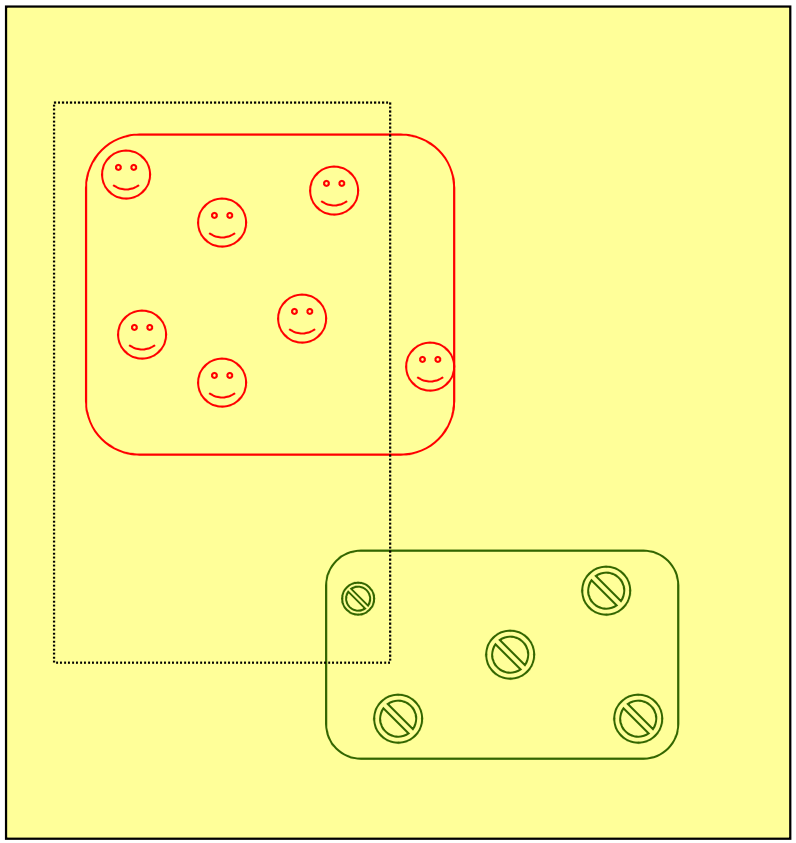


Nekonzistentní hypotéza

úplná (complete)



neúplná (incomplete)



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**

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.“*

Nature Inspired Technologies Group

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.
POZOR: hodnoty atributů samotné nestačí (viz příklady 1 a 2)

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

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

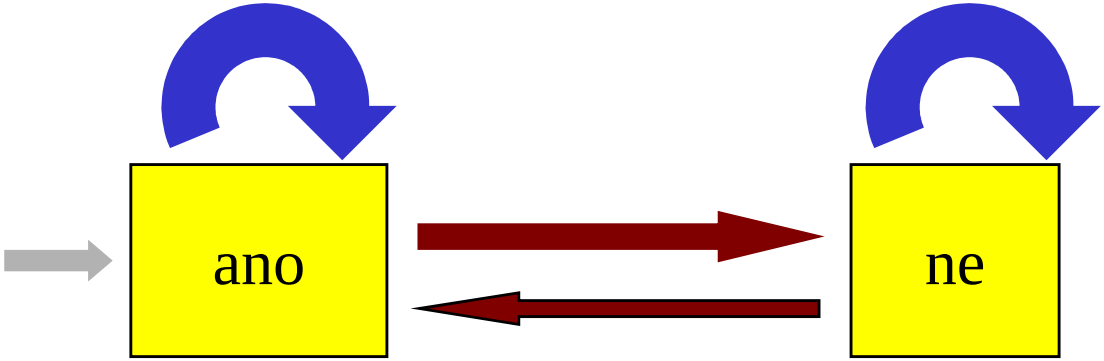
Příklad 3: parita

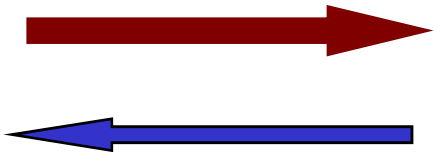
Př.	C1	C2	C3	C4	C5	C6	C7	C8	Kl.
1	1	0	1	1	0	0	0	0	-
2	1	0	1	1	0	0	0	1	+
3	1	1	1	1	0	0	0	1	-
4	0	1	1	1	0	0	0	1	+

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

Proč ne automat?
Význam formátu hypotézy – výhodná by byla rekurze !!

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



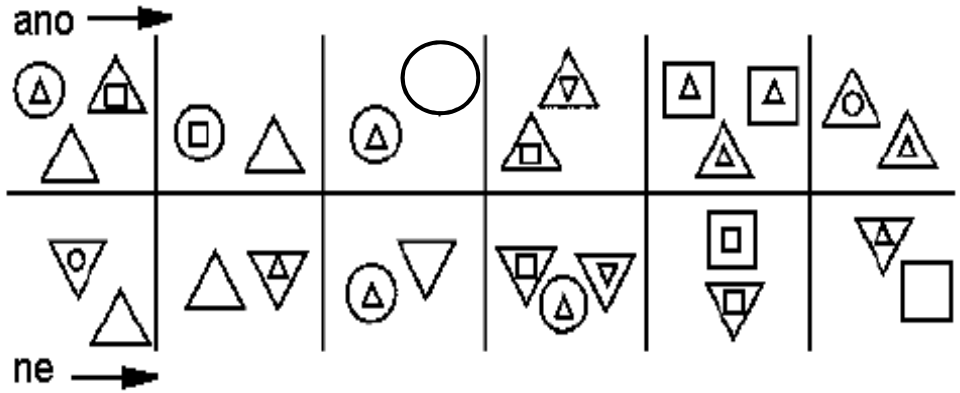


Na vstupu je symbol **1**

Na vstupu je symbol **0**

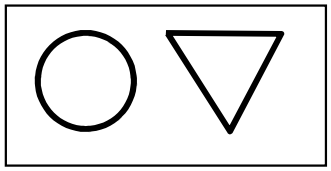
Příklad 4: Strukturovaný problém

Jak popsat takové zadání pomocí atributů?



Příklad 4: Strukturovaný problém (i)

- Kdyby každá skupina objektů obsahovala jen pevný počet prvků



Object 1	Pointing	Object 2	Pointing	Class
circle	N/A	triangle	down	positive

- Obtíž a: Přehození objektů 1 & 2?
  - Exponenciální nárůst počtu reprezentací téže entity (*zrcadlový obrázek*, atd.)

Příklad 4: Strukturovaný problém (ii)

- **Oblíž b**: Relace pro popis prostorových vztahů
  - nárůst atributů s hodnotou **false** nebo **N/A**

object 1	object 2	1 left of 2	2 left of 1	1 above 2	2 above 1	1 inside 2	2 inside 2	.....
circle	triangle	true	false	false	false	false	false	.....

- **Obtíž c**: Různý počet prvků ve skupině

object 1	object 2	object 3	object 4	object 5	.....	object MAX	1 left of 2	1 above 2
circle	triangle	rectangle	N/A	N/A	.....	N/A	true	false
rectangle	circle	triangle	circle	circle	.....	triangle	false	false

  - nárůst počtu prázných atributů
  - nárůst objemu celé tabulky
- Potřebujeme silnější jazyk pro representaci!

Podmínky nalezení dobré hypotézy

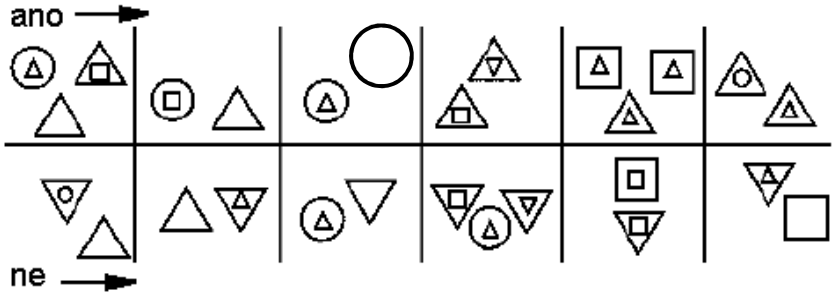
Záleží na vhodné volbě

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

Kdy nestačí atributové vyjádření?

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

Jazyk pro ILP



- **Predikáty:**

  group(X), in_group(e1,c1).

  circle(Z), square(Z),

  triangle (t3,up).

- **Popis první skupiny objektů**

  group(e1).

  circle(c1). triangle(t1,up). triangle(t2,up).

  triangle(t3,up). square(s1).

  in_group(e1,c1). in_group(e1,t1). in_group(e1,t2).

  inside(t3,c1). inside(s1,t2).

- **Vhodné hypotézy**

  positive(X) :- group(X), in_group(X,Y1), triangle(Y1,up), in_group(X,Y2), triangle(Y2,up).

  negative(X) :- group(X), in_group(X,Y1), triangle(Y1,down).

Motivace pro vznik ILP

- **Pokus vyrovnat se s omezeními klasických technik strojového učení a řešit pozorované obtíže**
  - při návrhu reprezentace úlohy: trénovací data
  - spojené s potřebou vyjádřit znalost z prostředí úlohy, t.j. doménovou znalost (background k.)
- **Logické programy umožňují použít jednotnou representaci pro**
  - příklady,
  - doménovou znalost,
  - formulaci hypotéz

ILP a jeho vlastnosti

Induktivní Logické Programování (1)

- **Příklady obvykle bývají:**
  - složeny z *různého počtu elementů*, mezi nimiž jsou vztahy, které jsou podstatné pro klasifikaci (např. koncept "inside" v Příkladu 4)
  - *není možné jednotně* (nebo je velmi nepřirozené) *popisovat* všechny trénovací příklady prostřednictvím jediného souboru atributů (universum tvoří slova různé délky), např. „group"
- **Potřebná apriorní znalost** má výrazně relační charakter (např. rodič(X,Y), hrana v grafu atd.)
- **Hypotéza** je formulována pomocí jazyka logiky prvního řádu, přesněji hypotézu tvoří konečná množina klauzulí odpovídající programu v použitém systému logického programování (nejčastěji Prologu)

Induktivní logické programování (2)

Indukce v logice:

Background Knowledge + Examples => **Theory**

Vstupy (Muggleton94)

- trénovací množina: pozitivní $\mathbf{E}^+$ a negativní $\mathbf{E}^-$ příklady
- doménová znalost **B** (logický program)

cíl: najít logický program **P** takový, že (**P + B**) pokrývají *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(3)

Příklad: popis konceptu „cesta" v orientovaném grafu

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

trénovací příklady: cesta(1,3). cesta(1,4). ...

Výsledný program P (hypotéza):

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

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

Pro jednoduchost se v konečných doménách uvádějí někdy

Základní úloha ILP

Pro dané množiny pozitivních a negativních příkladů $\mathbf{E}^+$ a $\mathbf{E}^-$ a množinu axiomů **B** takových, že

Apriorní bezespornost: $\forall \mathbf{e} \in \mathbf{E}^- : \mathbf{B} \mid\!/\!- \mathbf{e}$

Apriorní nutná podmínka: $\exists \mathbf{e} \in \mathbf{E}^+ : \mathbf{B} \mid\!/\!- \mathbf{e}$

(tj. $\mathbf{E}^+$ není plně pokryto **B**)

hledáme **P** takové, že

Aposteriorní úplnost: $\forall \mathbf{e} \in \mathbf{E}^+ : \mathbf{B} \cup \mathbf{P} \mid\!- \mathbf{e}$

Aposteriorní bezespornost: $\forall \mathbf{e} \in \mathbf{E}^- : \mathbf{B} \cup \mathbf{P} \mid\!/\!- \mathbf{e}$

Principy použité v ILP

Na prostoru hypotéz je možné definovat částečné uspořádání odpovídající pojmům "**generalizace**" a "**specializace**":

- θ-subsumpce (→ svaz, lattice)
- implikace

Postupy prohledávání tohoto prostoru a odpovídající **operátory**

- **sdola-nahoru** (zobecnění, generalizace)
- **shora-dolů** (zjemnění, specializace)
  - uprav klauzuli použitím substituce
  - přidej do těla klauzule další literál

Heuristiky použité při prohledávání

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 *spec*

přiřazuje každé klauzuli množinu jejích specializací.

Většina ILP systémů používá dvě **základní operace specializace**

- **úprava proměnných**

  ztotožnění 2 proměnných
  spec(cesta(X, Y)) = cesta(X, X)
  nahrazení proměnné konstantou
  spec(číslo(X)) = číslo(0)
  nahrazení proměnné složeným termem
  spec(číslo(X) = číslo(s(Y)) .

- **přidání podcíle do těla formule**

Generický algoritmus ILP

pracující s množinou R odvozovacích pravidel pro úpravu hypotéz

QH := *inicializuj*(B; E^+, E^-) ; /*návrh výchozí hypotézy*/

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

vyber hypotézu **H** z **QH** ;

zvol_odvozovací_pravidla $\mathbf{r_1,\ldots,r_k}$ z **R** ;

aplikací $\mathbf{r_1,\ldots,r_k}$ na **H** vytvoř množinu $\mathbf{H_1}$;

QH := (QH-H) + $\mathbf{H_1}$;

zruš_některé_prvky z **QH** ; /*prořezávání*/

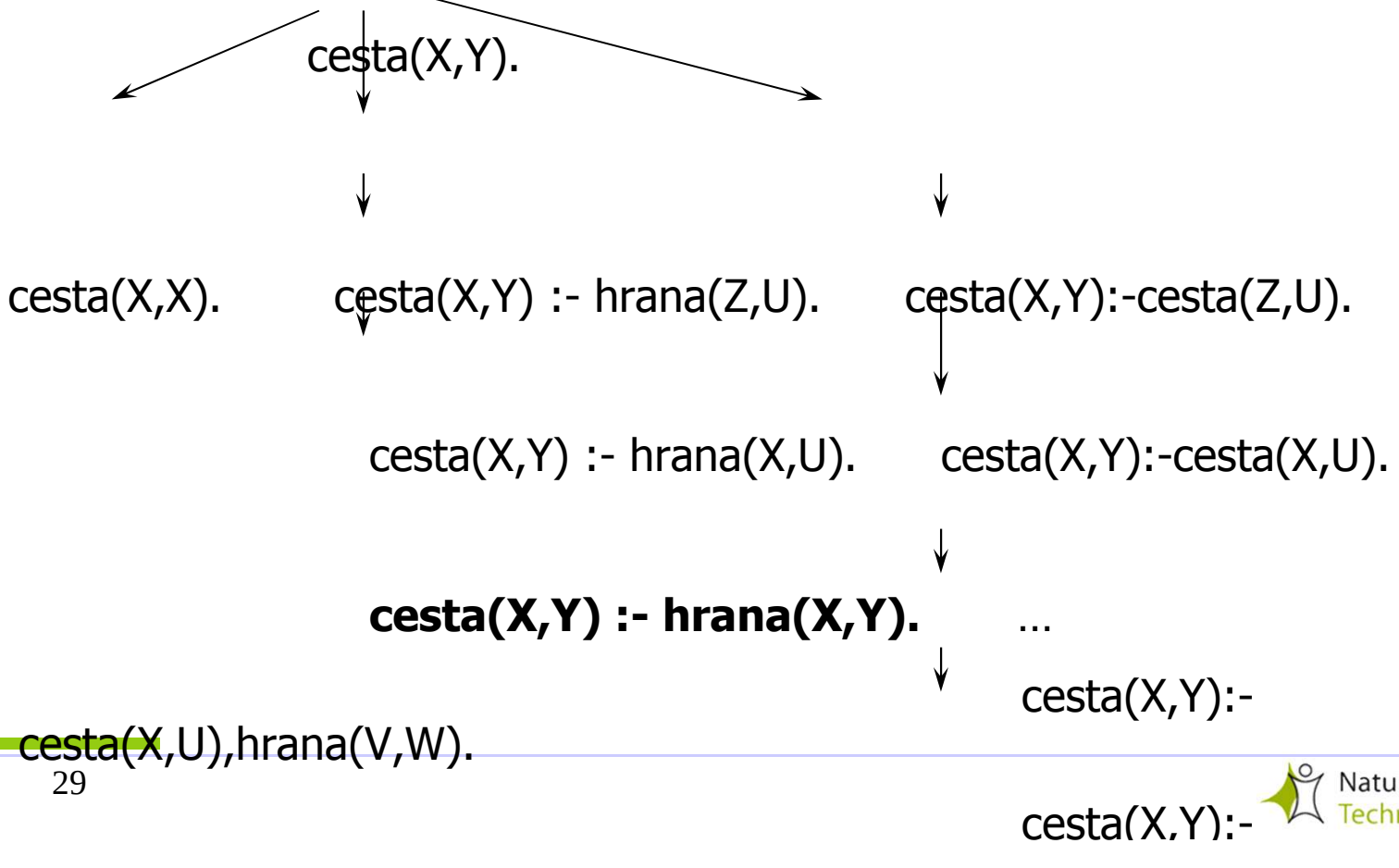
Příklad: Cesta v grafu

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



Systémy

Aleph (dříve P-Progol), Oxford University

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

FOIL (Quinlan 1993)

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

RSD (Železný 2002) hledání zajímavých podskupin

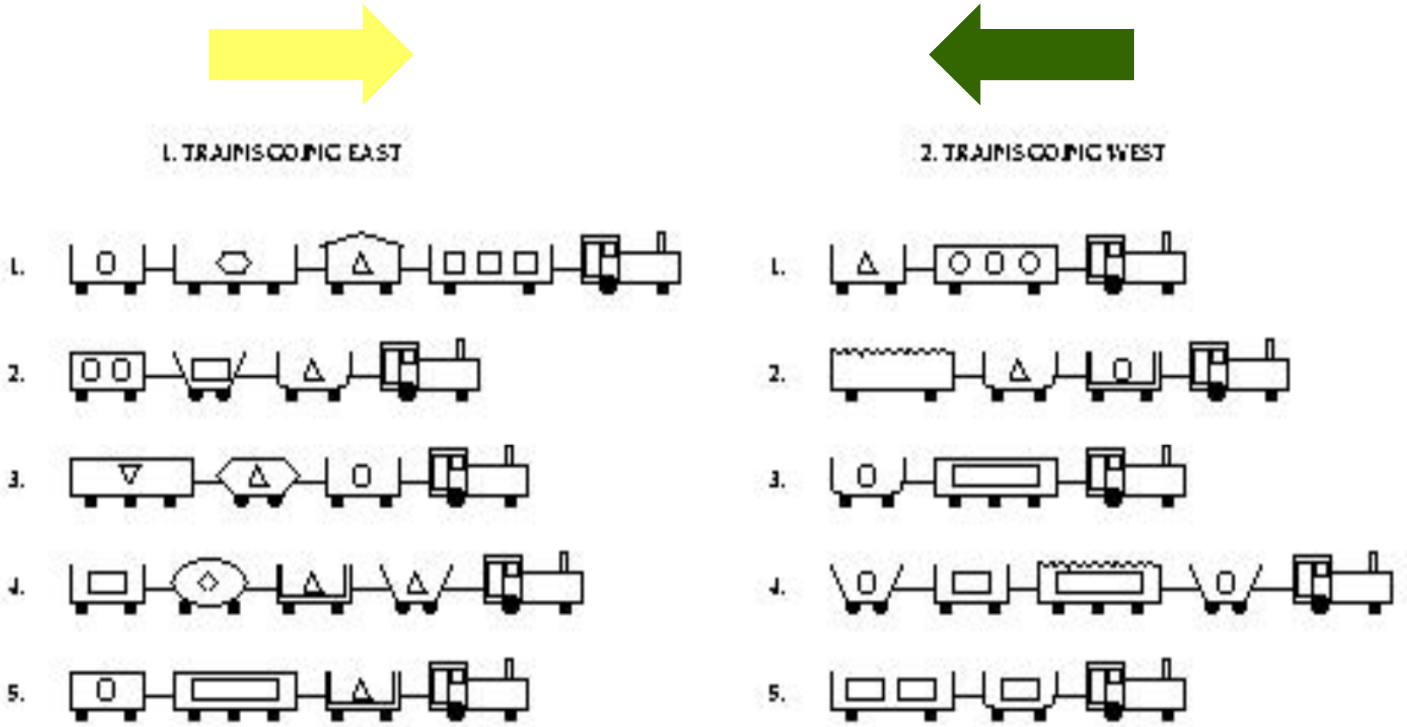
Brno FI MU

INDEED (Nepil 2003) učení ze strukturovaných (lingvistických) dat

Aleph

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

East-West Trains (1)



Nature Inspired Technologies Group

East-West Trains (2)

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

% eastbound train 1

short(car_12). closed(car_12).

shape(car_12,rectangle).
load(car_12,triangle,1).

wheels(car_12,2).

…

long(car_11). open_car(car_11). …

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

shape(car_11,rectangle).
load(car_11,rectangle,3).

wheels(car_11,2).

…

has_car(east1,car_11).

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).
```

```
?- [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

East-Westbound Trains

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

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).

…

…

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

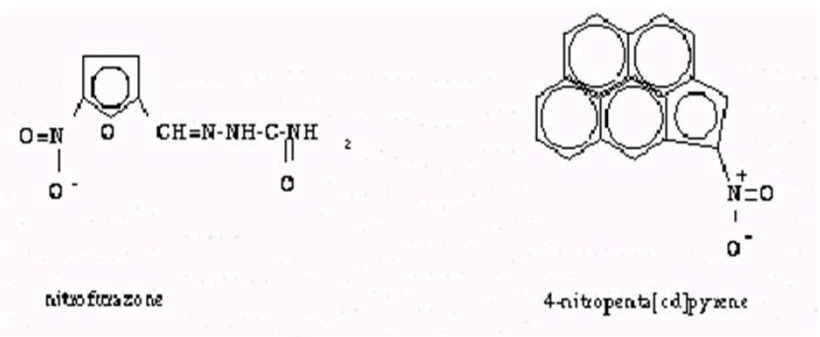
- A., které jsou nezvyklé z hlediska použití metod strojového učení.

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

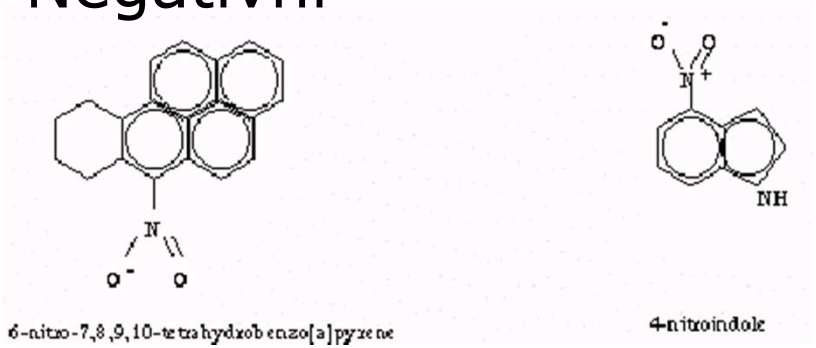
Bioinformatika - úloha SAR

- Hypotéza **Structure Activity Relationship** (SAR) říká, že molekuly s podobnou strukturou mají i podobné chování. Někdy tomu tak není. Skupina látek s podobnou strukturou, ve které jsou s velmi různými účinky. Známe:
  - chem.struktura látky a
  - empirické údaje o její toxicitě/ mutageneticitě/ terapeutickém účinek.
- **Co je příčinou pozorovaného chování**?

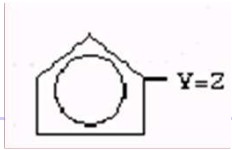
Pozitivní



Negativní



Výsledek: strukturární indikátor



Nature Inspired Technologies Group

Bioinformatika - karcinogenicita

- 230 aromatických a heteroaromatických dusíkatých sloučenin

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

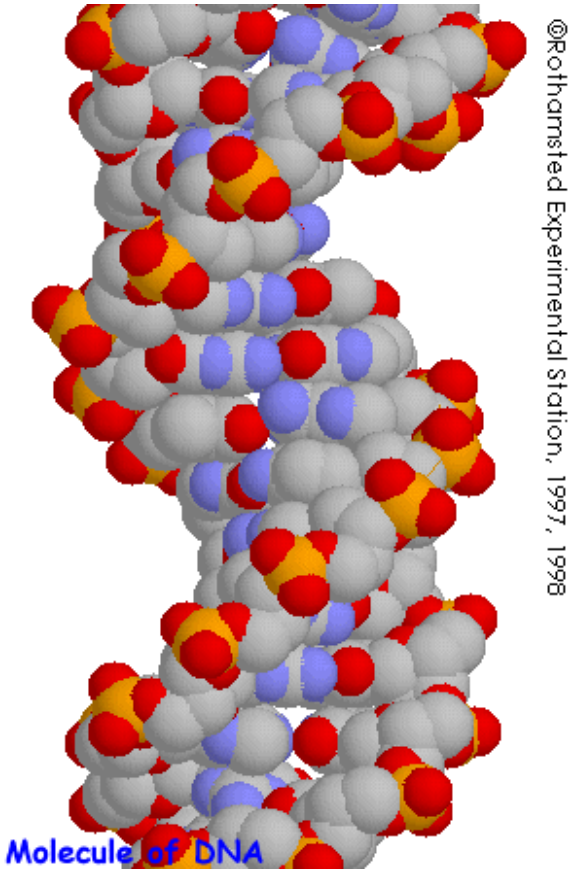
+ 42 RU sloučenin (regression-unfriendly skupina).

- Na RU skupině se prokázaly výhody relační reprezentace:

Hypotéza navržená PROGOLem dosahovala přesnosti 88%

zatímco klasické metody asi o 20 % méně.

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

Predikce událostí v telefonní ústředně

- Trénovací příklady

```
...
incoming(date(10,18),time(13,37,29),[0,6,4,8,2,5,6,8,4,9],[3,2],t
   ransfer([[1,6],[1,2]],transfer([[2,8],[2,6]],talk))).
...
```

- Nalezená pravidla

```
...
incoming(D,T,EX,31,transfer([10|R],RES)) :-
        day_is(monday,D),branch(EX,[5,0]).
...
```

proměnné
(zastoupí jakýkoliv objekt)

relace apriorní znalosti

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) :

Zdroje

Oxford University http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/

ILPNet2 http://www.cs.bris.ac.uk/%7EILPnet2/index.html

Kurs ILP http://www.fi.muni.cz/usr/popelinsky/lectures/ilp/