Další témata DM a strojového učení

Osnova

- Co plyne z odhadů PAC pro strojové učení?
- Relevance atributů
- Problémy reprezentace
- Meze klasických metod - příklady
- Princip ILP a generický ILP algoritmus
- Existující systémy a příklady použití
- Zajímavé aplikace

Některá zajímavá pozorování vyplývající z teorie **PAC**

Víme, že je-li mohutnost **H** konečná, je třeba mít k dispozici alespoň **1/ ε *(ln |H| + ln (1/δ))** příkladů: pokud učící algoritmus navrhuje hypotézu **h**, která je se všemi příklady konzistentní, pak pravděpodobnost toho, že „chyba **h** je menší než **ε** (**h** je **ε-skoro správná)**“ je větší než (1- δ).

- **Je možné** naučit se koncept z omezené množiny trénovacích příkladů skoro správně i pokud možných hypotéz je nekonečně !
- Při učení se hypotéz z konečné množiny možných h. bývá počet potřebných příkladů funkcí $\mathbf{n^k}$, kde **n** je počet atributů použitých v popisu příkladů (rozhodovací strom hloubky **k**) - ***lze dosáhnout snížení tohoto čísla?***

- ***Změna reprezentace trénovacích příkladů. Používané metody:***
  - Snížení počtu atributů prostřednictvím nových **odvozených atributů,** které jsou funkcí nějaké skupiny původních atributů (NN, statistické řešení – Support Vector Machine)
  - Eliminace zbytečného zavádění nových atributů (např. při převodu multirelační úlohy na atributovou)
  - zjišťování **relevance**

Relevance Based (RB) učení: metoda pro **výběr relevantních atributů**

- Pokusme se využít doménovou znalost pro nalezení a eliminaci nepodstatných atributů. *Pokud hodnota některého atributu je funkcí jiného atributu, pak jeden z nich lze vynechat.*
- **Determinace úlohy** je taková podmnožina μ množiny všech atributů, která není v rozporu s trénovací množinou, tj. nenastane případ, kdy 2 objekty jsou pomocí atributů z μ popsány stejně, ale je jim přiřazena odlišná klasifikace.
- Zajímá nás **minimální determinace**, tj. taková, že žádná její podmnožina není determinací.

Hledání minimální determinace pro „vodivost“

Podmínky nesplňují např. {materiál, číslo_objektu} *ani* {teplota, objem}.

Naopak determinací jsou {teplota, materiál} nebo {váha, teplota, objem}.

Číslo objektu	Váha	Teplota	materiál	Objem	Vodivost
1	12	26	měď	3	0,59
1	12	100	měď	3	0,57
2	18	26	olovo	3	0,05
3	12	26	olovo	2	0,05
3	12	100	olovo	2	0,04
4	24	26	olovo	4	0,05

Algoritmus pro hledání minimální determinace

- Systematické prohledávání všech podmnožin atributů (od nejkratších) a jejich ověřování na trénovacích datech.
- Je zřejmé, že má-li nejkratší minimální determinace délku **p**, musíme předem prohledat všechny kratší podmnožiny. Jde o kombinace **p** prvků z **n**, což řádově odpovídá $\mathbf{n^p}$. Pozor tedy - jedná se opět o NP úlohu!

Ovšem nalezení minimální determinace může zásadně přispět k výraznému zjednodušení výchozí úlohy. Proto je těmto metodám stále věnována pozornost. *Má význam používat např. heuristický nebo randomizovaný přístup*.

Cesta k návrhu úspěšného modelu pro studovaná data

- **Model** = přibližný popis relace implicitně definované trénovacími daty; **tvoří se** např. **metodami strojového učení**
- **Jak vybírat vhodnou metodu pro zpracovávaná data?** Metoda není to nejpodstatnější!

  Holte,R.C.: ***Very Simple Classification Rules Perform Well on Most Commonly Used Datasets***, ML Vol.11, pp.63-91, Kluwer Ac.Publ. 1993

  Důležité je předejít přeučení!!!
- Teorie PAC učení upozorňuje, že pro techniky strojového učení je zásadní respektovat vztah mezi počtem **trénovacích příkladů** a počtem **použitých atributů**
  - **důraz na hledání relevantních atributů**
  - **předzpracování dat, vizualizace, ...**

Kolik korektních hypotéz lze navrhnout pro danou trénovací množinu **E**?

- **Fakt:** možných konceptů je obvykle nesrovnatelně víc než možných hypotéz. *To je nejlépe vidět na příkladu spočetného definičního oboru **D**, který popisujeme pomocí nanejvýš spočetně atributů **A**. Konceptů je v tomto případě tolik, co podmnožin **D**, tedy **nespočetně**. Naopak hypotéz je nanejvýš tolik, kolik je konečných slov v abecedě **A**, tedy **spočetně**. Neexistuje tedy prosté zobrazení z množiny všech konceptů na množinu všech hypotéz a **jen některé koncepty lze přesně popsat pomocí nějaké hypotézy** !*
- **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 hypotéza !

Klíčovou roli hraje volba vhodné reprezentace - jak pro popis příkladů, tak pro jazyk, ve kterém tvoříme hypotézy !

Podmínky nalezení dobré hypotézy

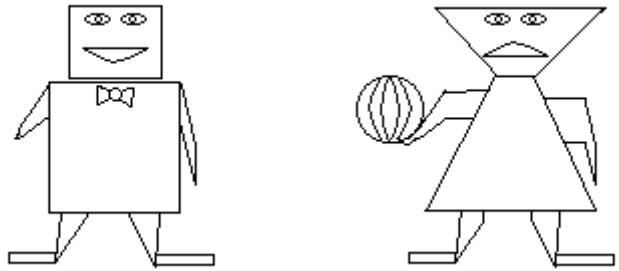
Záleží na vhodné volbě

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

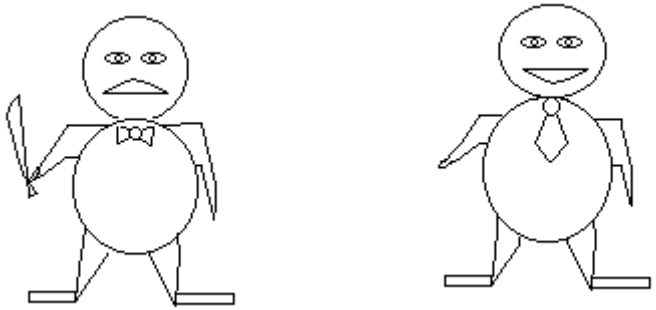
Kdy nestačí atributové vyjádření?

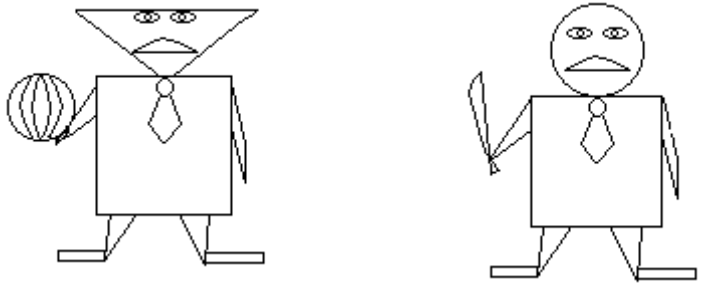
- popis příkladů nemá uniformní tvar (definiční obor tvoří slova různé délky)
- struktura jednotlivých příkladů má rozhodující charakter
- doménová znalost je výrazně relační

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

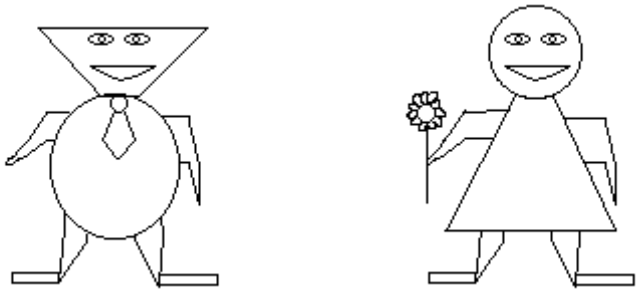


přátelství





nepřátelství



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. nechť atributy **c1**, **c2** a **c3** označují první, druhou a třetí číslici ve trojici. Zkuste navrhnout strom, který popisuje uvedenou klasifikaci – bude velmi košatý a zcela nesrozumitelný!

Klasifikace totiž souvisí s uspořádáním číslic

relace uspořádání "**<**" = apriorní (nebo doménová) znalost

Použijeme-li "**<**" pro tvorbu hypotézy, je výsledek zcela jasný:

if c1 < c2 & c2 < c3 then '+'.

Příklad 3: parita

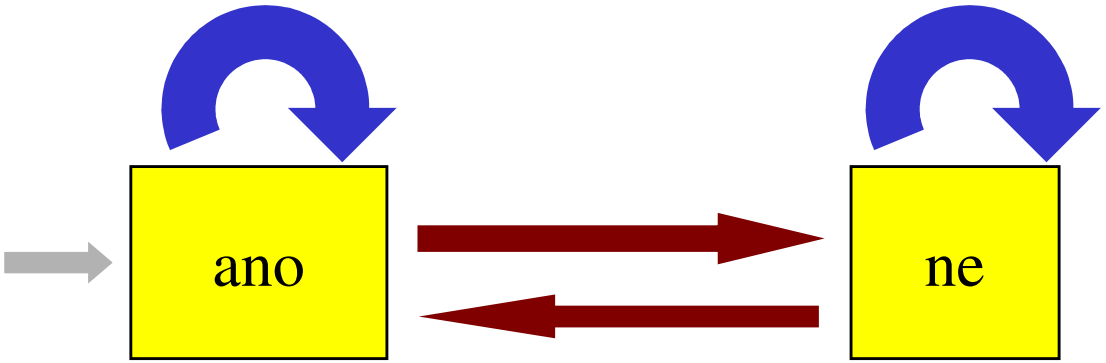
Př.	C1	C2	C3	C4	C5	C6	C7	C8	Kl.
1	1	0	1	1	0	0	0	0	-
2	1	0	1	1	0	0	0	1	+
3	1	1	1	1	0	0	0	1	-
4	0	1	1	1	0	0	0	1	+

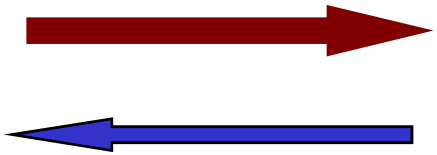
Není atribut, který by bylo možné vynechat - na všech záleží !!!
Rozhodovací strom by byl velmi komplikovaný –

Proč nepoužít automat?

Význam formátu hypotézy – v tomto případě by byla výhodná rekurze !!

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



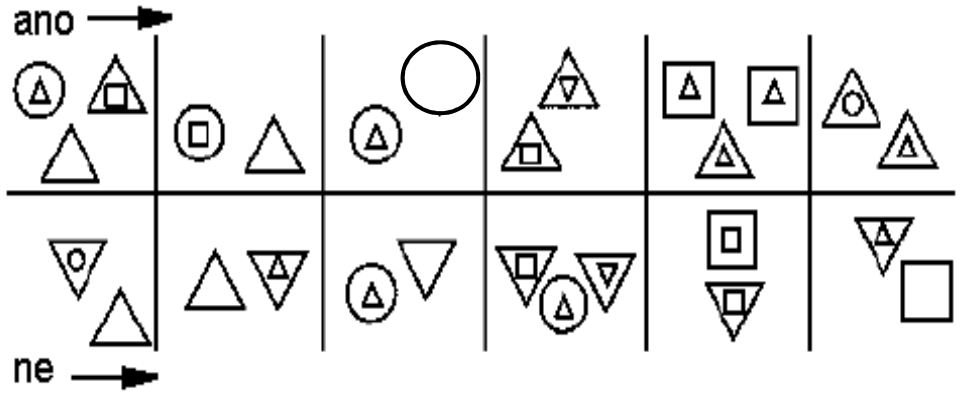


Na vstupu je symbol **1**

Na vstupu je symbol **0**

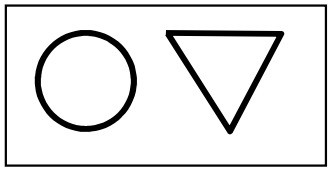
Příklad 4: Strukturovaný problém

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**: Co když skupiny objektů jsou chápány jako množiny, tj. přehození objektů 1 & 2 reprezentuje tutéž skupinu?
  - 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
  - *Co když neznáme maximální přípustný počet objetů?*
- Potřebujeme silnější jazyk pro representaci!

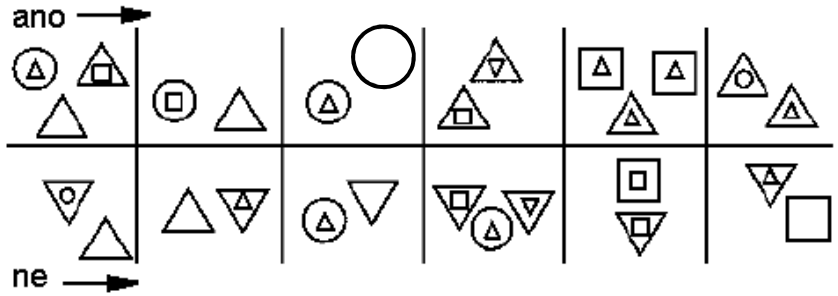
Jazyk pro ILP

- **Predikáty:**

  group(X), in_group(X,Y).

  circle(Z), square(Z),

  triangle (W,up).



- **Popis první skupiny objektů**

  group(e1). circle(c1). square(s1).

  triangle(t1,up). triangle(t2,up). triangle(t3,up).

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

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

- **Vhodné hypotézy,** např. hypotézu „*Skupina* **X** *patří do třídy* NE , *pokud* **X** *obsahuje objekt* **Y1**, *který má tvar trojúhelníka orientovaného dolů*.“ lze popsat takto:

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

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

Motivace pro vznik ILP

- **Pokus vyrovnat se s omezeními klasických technik strojového učení a řešit pozorované obtíže**
  - při návrhu reprezentace úlohy (popis trénovacích dat)
  - 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“ může obsahovat 1-10 objektů
- **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 (pro popis doménové znalosti, 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 jen pozitivní příklady a využívá se předpoklad uzavřeného světa (CWA).

Základní úloha ILP

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

Apriorní bezespornost: $\forall \mathbf{e} \in \mathbf{E}^- : \mathbf{B} \not\vdash \mathbf{e}$ (**e** není dokazatelné z **B**)

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

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

hledáme **P** takové, že

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

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

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**

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

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

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

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

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

vyber_hypotézu **P** z **QH**

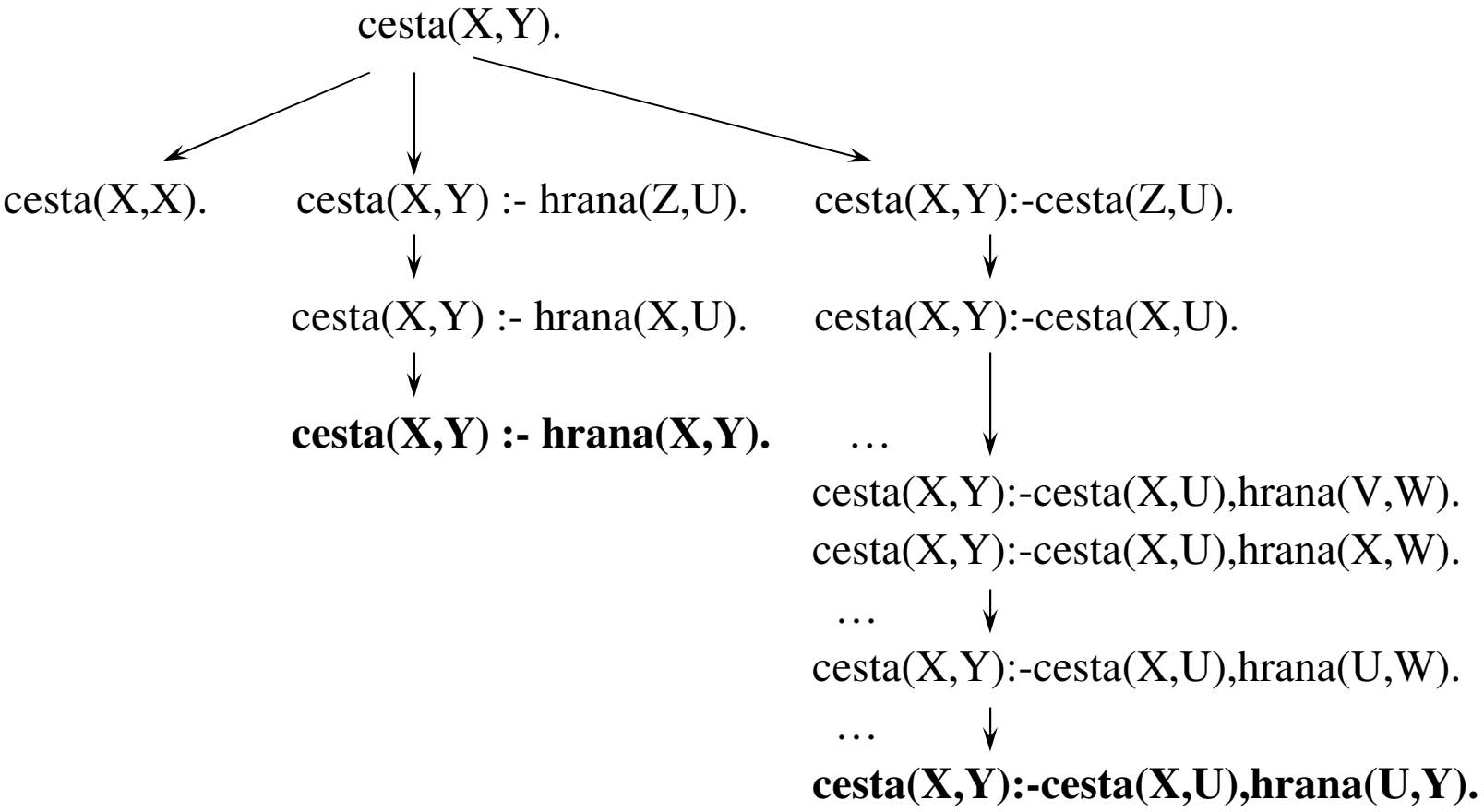
Příklad: Cesta v grafu

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



Systémy pro práci s relačními daty

ILP systémy, viz http://www-ai.ijs.si/~ilpnet2/systems/

Aleph (dříve P-Progol), Oxford University

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

FOIL (Quinlan 1993)

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

***ČVUT*: RSD** (Železný 2002) hledání zajímavých podskupin

Brno FI MU: INDEED (Nepil 2003) učení ze strukturovaných (lingvistických) dat , RAP (Blaťák 2003) učení častých vzorů

Další systémy , např.

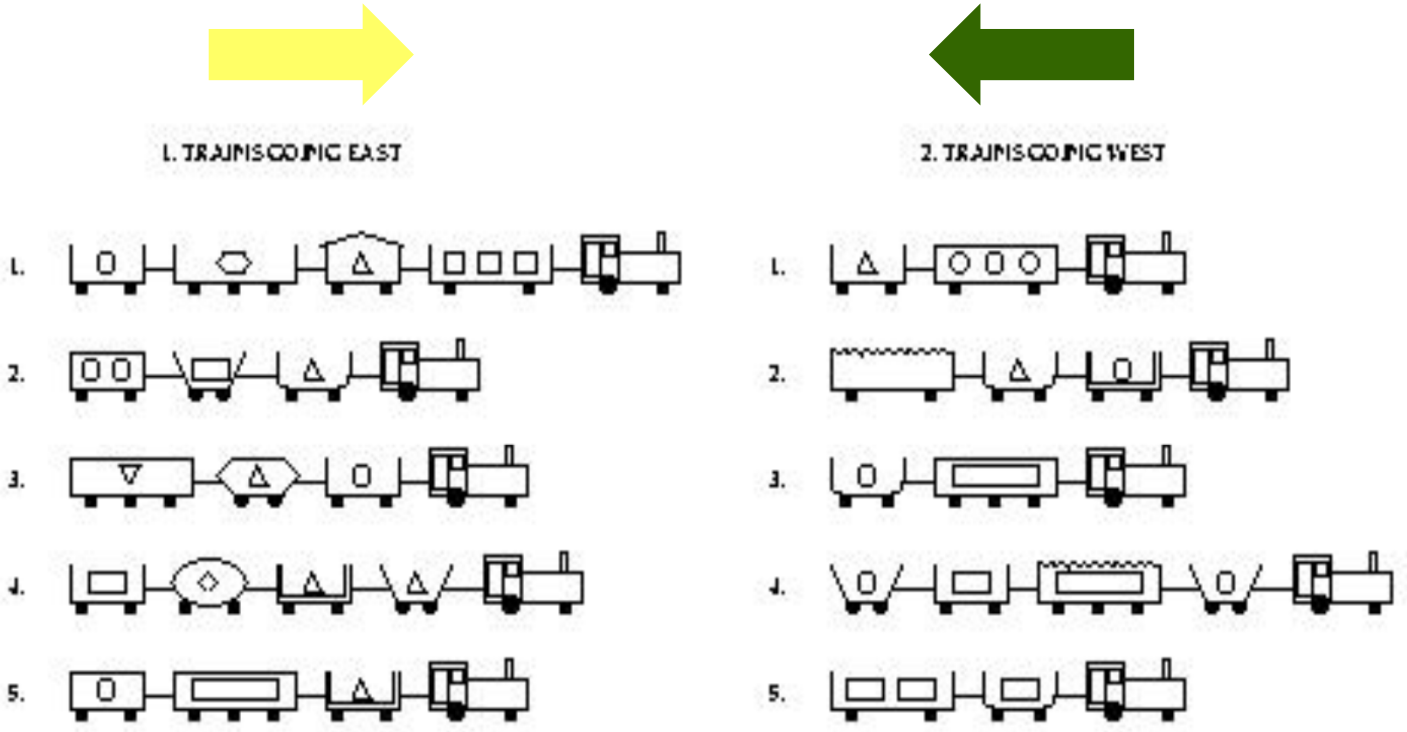
SAFARI: http://www.kiminkii.com/company.html (nový komerční produkt pro **Multi-Relational Data Mining**)

DATACONDA, http://www.dataconda.net/ (free download)

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 počet) pokryty a není pokryt žádný negativní (případně až na malý počet) pokryt

East-West Trains (1)



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 počet) pokryty a není pokryt žádný negativní (případně až na malý počet) pokryt

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 počet) pokryty a není pokryt žádný negativní (případně až na malý počet) pokryt

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).
shape(car_12,rectangle). load(car_12,triangle,1).
wheels(car_12,2).
…
long(car_11). open_car(car_11). …
shape(car_11,rectangle). load(car_11,rectangle,3).
wheels(car_11,2).
…
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( … ).
```

#shape: dosaď konstantu daného typu

* : počet různých dosazení (zde lib.př.č.)

```
?- [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
          5     5    10     [time taken] [0.07]
                             [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

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

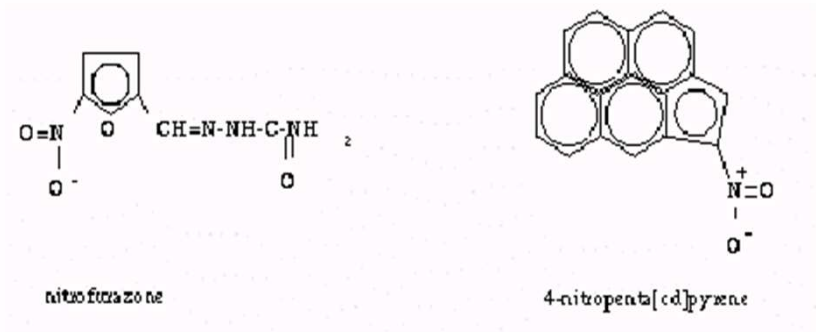
- Aplikace, které jsou nezvyklé z hlediska použití metod strojového učení.

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

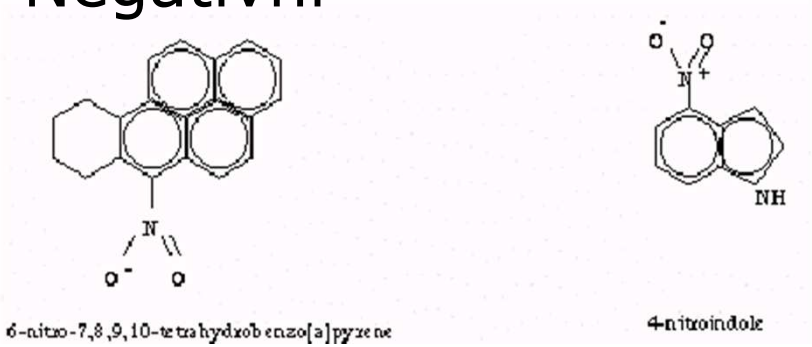
Bioinformatika - úloha SAR

- 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**?

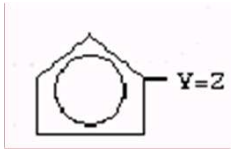
Pozitivní



Negativní



Výsledek: strukturární indikátor



Bioinformatika - karcinogenicita

- 230 aromatických a heteroaromatických dusíkatých sloučenin

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

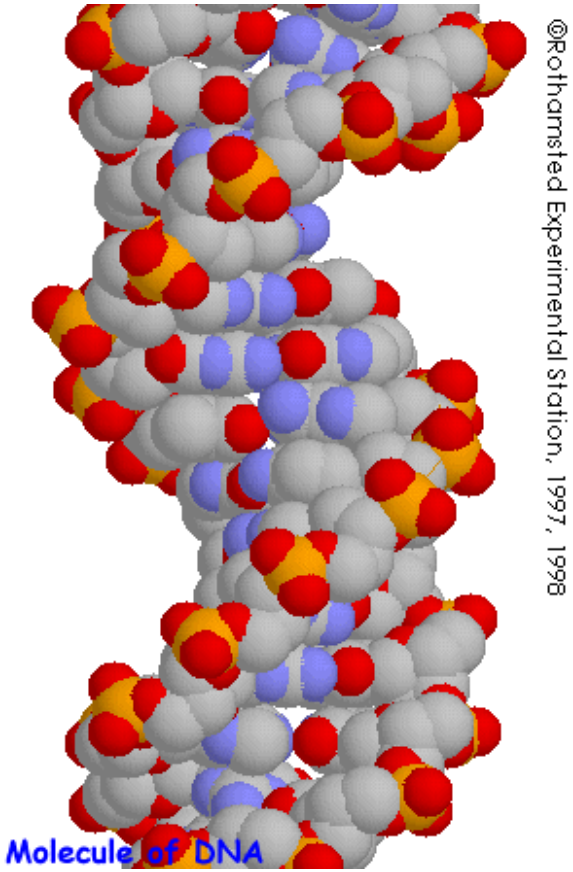
  + 42 RU sloučenin (regression-unfriendly skupina).

- 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 výkonu zkušeného odborníka

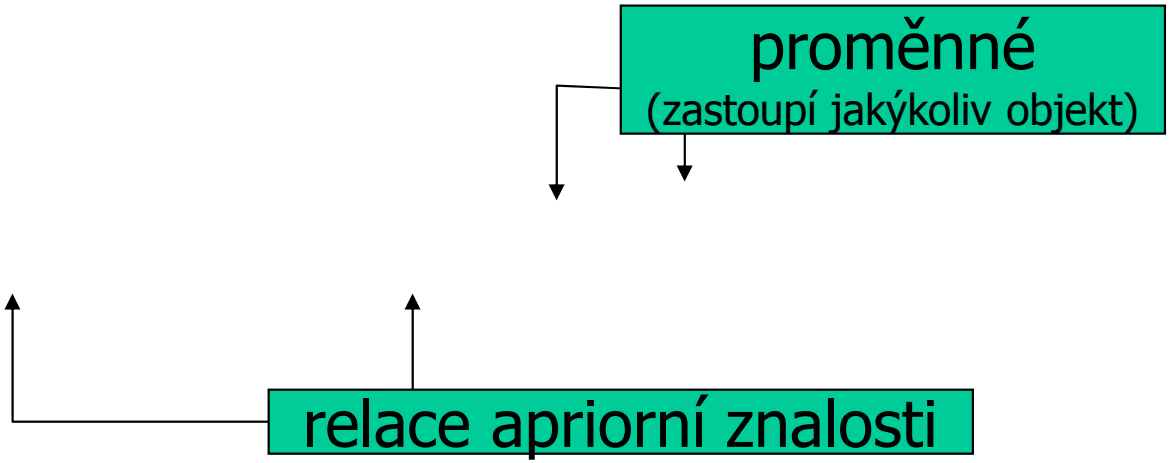
Predikce událostí v telefonní ústředně

- 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]).
...
```



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])).
```

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/

http://www.dataconda.net/

Samorani, M. 2015. **Dataconda: A software Framework for Mining Relational Databases.** *The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications*

SAFARI
http://www.kiminkii.com/company.html