Induktivní logické programování

tj. učení logického programu z dat

Opakování pojmů z logiky a log. prog.

- jazyk: Konstanty, proměnné, Predikáty, funkce
- termy, literály, positivní a negované literály
- klauzule je disjunkce literálů
- základní literál, klauzule = bez proměnných
- Hronovská klauzule = má právě jeden positivní literál
- logický program je množina Hornovských klauzulí
- substituce, unifikace

Motivace

Mějme údaje o osobách, atributy Name, Mother, Father, Male, Female, cílový atribut Daughter.

Abychom se byli schopni aspoň nějak naučit cíl pomocí rozhodovacího stromu (nebo pravidel Sequential covering), potřebujeme v jednom učícím příkladu data o dvou lidech a hodnotu Daughter, např.

⟨ Name1=Sharon, Mother1=Loise, Father1=Bob,
Male1=False, Female1=True,
Name2=Bob, Mother2=Nora, Father2=Victor,
Male2=True, Female2=False, Daughter12=True ⟩

i tak jsme schopni se naučit jen specifická pravidla typu:

IF (Father1=Bob) & (Name2=Bob) & (Female1=True)
THEN (Daughter12=True)

Ale takovéto pravidlo se na nových datech použije zřídka, pokud vůbec.

V predikátové logice jsme schopni napsat obecně:

IF Father(y,x) & Female(y) THEN Daughter(x,y)

Podobně GrandDaughter, Ancestor.

FOIL

- Jeden z mnoha programů na učení logických programů
- velmi podobný učení pravidel postupným pokrýváním
  Sequential covering
- učí Hornovské klauzule se dvěma vyjímkami:
  - nejsou dovoleny funkční symboly v literálech (i tak je prostor k prohledávání dost velký)
  - dovoluje i negované literály v těle, tj. nejen Hornovské klauzule.
- Naučil se rekurzivní definici Quicksortu, rozpoznávat legální a nelegální pozice v šachu.
- Pokrývá jen pozitivní příklady, provádí hill–climbing. ne beam search

Algoritmus **FOIL(Cíl, Predikáty,** *Data***)**

Pos ← positivní příklady v *Data*
Neg ← negativní příklady v *Data*
Pravidla ← {}
while *Pos*, **do** % nauč se nové pravidlo
    NovePravidlo ←pravidlo s prázdným tělem predikující Cíl
    NegPr ← *Neg*
    while *NegPr*, **do** % přidej literál k NovePravidlo
        NoveLiteraly ← k *NovePravidlo* vytvoř kandidáty z *Predikáty* %bude
        Favorit ← *Literál* ∈ *NoveLiteraly*, mající maximální
            FOIL–Gain(*Literál*, *NovePravidlo*, *Data*, Cíl)
        NovePravidlo ← *NovePravidlo* ∪ *Favorit*
        NegPr ← {*k* ∈ *NegPr* splňující tělo *NovePravidlo*}
    Pravidla ← *Pravidla* ∪ {*NovePravidlo*}
    Pos ← *Pos*\ příklady pokryté *Favorit*–em
return *Pravidla*

Generování kandidátů

Předpokládejme, že tvoříme kandidáty–literály k pravidlu R

$$P(x_1, x_2, \ldots, x_k) \leftarrow L_1, L_2, \ldots, L_n$$

Uvažujeme každý literál L_{n+1} vzniklý jednou z variant:

- $Q(v_1, \ldots, v_r)$, kde Q je predikát z Predikáty, v_i jsou proměnné a aspoň jedna z nich se vyskytuje mezi proměnnými v pravidle (hlavě nebo tělu).
- $Equal(x_j, x_i)$, kde se obě proměnné x_i, x_j vyskytují v pravidle
- negace literálu z předchozích bodů.

Příklad

Učíme predikát $GrandDaughter(x, y)$, další predikáty jsou $Father$ a $Female$.

- První návrh pravidla je $GrandDaughter(x, y) \leftarrow$.
- Zkoušíme přidat každý z literálů:
  $Equal(x, y), Female(x), Female(y), Father(x, y), Father(y, x),$
  $Father(x, z), Father(z, x), Father(y, z), Father(z, y)$ a negaci každého z nich.
- Řekněme, že FOIL vybere $Father(y, z)$ a generuje pravidlo:
  $GrandDaughter(x, y) \leftarrow Father(y, z)$
- Pro další rozšíření uvažuji všechny výše uvedené literály a navíc $Female(z), Equal(z, x), Equal(z, y), Father(z, w), Father(w, z)$ a jejich negace.

- FOIL vybere (např.) $Father(z, x)$ a po ještě jedné iteraci $Female(y)$ a vytvoří pravidlo:
  $GrandDaughter(x, y) \leftarrow Father(y, z) \& Father(z, x) \& Female(y)$
- Toto pravidlo nepokrývá žádný negativní příklad, tedy uzavřeme tvorbu pravidla, přidáme ho do množiny pravidel a odstraníme positivní příklady pokryté tímto pravidlem. Pokud zbyly positivní příklady, učíme další pravidla.

Příklad

Předpokládejme data:

GrandDaughter(Victor, Sharon) Father(Sharon, Bob) Father(Tom,Bob)

Female(Sharon) Father(Bob,Victor)

Předpokládáme uzavřený svět, tj. negaci všech literálů o GrandDaughter, Father, Female, které tu nejsou uvedeny.

Máme tedy čtyři konstanty–jména.

Pravidlo $GrandDaughter(x, y) \leftarrow$ pokrývá všech 16 možných přiřazení, jedno positivní $GrandDaughter(Victor, Sharon)$ a 15 negativních.

Výběr literálu

FOIL–Gain($Literál, NovePravidlo, Data$, Cíl)$=t\left(log_2 \frac{p_1}{p_1+n_1} - log_2 \frac{p_0}{p_0+n_0}\right)$

kde:

p_0	počet positivních přiřazení proměnným v pravidle R (bez nového literálu)
n_0	počet negativních přiřazení proměnným v pravidle R
p_1	počet positivních přiřazení proměnným v pravidle R' (s novým literálem)
n_1	počet negativních přiřazení proměnným v pravidle R'
t	počet positivních přiřazení pravidlu R, která jsou pokrytá i pravidlem R'

Negativním přiřazením se myslí takové přiřazení, že pro něj v Datech cílový predikát neplatí.

Positivní přiřazení R je pokryto R', pokud aspoň jedno rozšíření o přidané proměnné je přiřazením v R'.

Funkce FOIL–Gain má přímou interpretaci v teorii informace. Logaritmy představují minimální počet bitů potřebných ke kódování cílové třídy pomocí pravidla R, resp. R'. *t* je počet přiřazení, pro která se redukce aplikuje, čili FOIL–Gain(L,R) představuje redukci bitů potřebných ke klasifikaci positivních přiřazení R, pokud přidáme literál L.

Rozšíření

Rekurzivní pravidla
Pokud do Predikáty přidáme i cílový predikát, můžeme se učit i rekurzivní pravidla. Musíme jen přidat kontrolu nekonečně vnořené rekurze.

FOIL byl schopen se naučit některé rekurzivní predikáty.

Šum
Na datech bez šumu můžeme přidávat literály do té doby, až není pokryt žádný negativní příklad. Na datech s šumem používáme kombinaci přesnosti, pokrytí a složitosti pravidla. FOIL používá princip minimální délky zápisu. FOIL navíc používá prořezávání (post–prunning).

Učení bez učitele

(Unsupervised learning)

Analýza nákupního koše

Apriori algoritmus

- Které věci lidé často kupují zároveň?
- Hodně velká data, i co do počtu sloupců – sortimentu zboží.
- Zadám práh, minimální četnost kombinace tzv.**support**, která mě ještě zajímá.
- Při i–tém průchodu daty počítám pro všechny možné kandidáty délky i jejich četnost.
- Kandidátem do dalšího průchodu jsou jen ty množiny správné délky, že **všechny jejich podmnožiny** mají nadprahovou četnost.
- Množiny mohu přepsat na pravidla tak, že z možných kandidátů vyberu ta pravidla, která mají velkou přesnost (confidence).