Relacijsko učenje in odkrivanje enačb

Ljupčo Todorovski

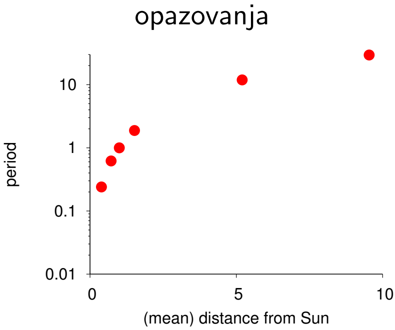
Univerza v Ljubljani, Fakulteta za upravo
Institut Jožef Stefan, Odsek za tehnologije znanja (E8)

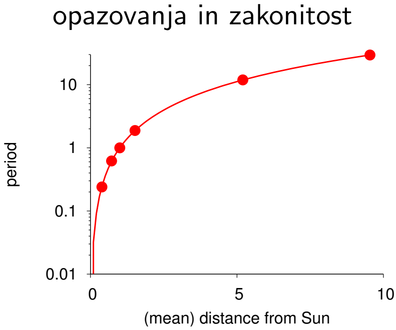
Maj 2019

Odkrivanje enačb: Keplerjev zakon

Rekonstrukcija Keplerjevega tretjega zakona iz podatkov

$$d^3/p^2 = const$$





Pregled predavanja

Operatorji izostritve

- Pomembne lastnosti
- Izčrpno in hevristično iskanje
- Odkrivanje enačb

Propozicionalizacija

- Poizvedbe kot generatorji spremenljivk
- Pretvorba relacijskega v običajno strojno učenje
- Odkrivanje enačb

Definicija

Operator izostritve, *refinement operator* ρ

$$\rho(p) = \{q : p \preceq q\}$$

Od splošnega k specifičnemu (lahko tudi od specifičnega k splošnemu)

Relacija splošnosti: p je *bolj splošna* od q, $p \preceq q$

$$coverage(q, S) \subseteq coverage(p, S)$$

- rečemo tudi, da je q *bolj specifična* od p
- S je množica učnih primerov
- $coverage(p, S) = \{e \in S : covers(p, e)\}$

Pregled lastnosti operatorjev izostritve

- Idealnost: nasledniki so najbolj splošne poizvedbe
- Popolnost: iz najbolj splošne poizvedbe z zaporedjem izostritev pridemo do katerokoli druge poizvedbe *vsaj na en način*
- Optimalnost: isto kot zgoraj, a *na točno en način*

- Monotonost in anti-monotonost

Definicija

Monotonost Boolove funkcije f glede na relacijo splošnosti $\preceq$

$$\forall p, q, S : (p \preceq q) \wedge f(p, S) \Rightarrow f(q, S)$$

- p in q sta hipotezi (poizvedbi)
- S je množica učnih primerov (Herbrandove interpretacije)

Anti-monotonost

$$\forall p, q, S : (p \preceq q) \wedge f(q, S) \Rightarrow f(p, S)$$

Primer monotone funkcije

Stopnja pokritosti $c(p,S)$ množice S z hipotezo p

$$c(p,S) = |coverage(p,S)|/|S|$$

Boolova funkcija $f_c(p,S)$: stopnja pokritosti je *največ* ϵ

$$f_c(p,S) \equiv c(p,S) \leq \epsilon$$

- Definicija splošnosti $p \preceq q \Rightarrow coverage(q,S) \subseteq coverage(p,S)$
- Očitno velja $coverage(q,S) \subseteq coverage(p,S) \Rightarrow |coverage(q,S)| \leq |coverage(p,S)|$
- Torej monotonost $p \preceq q \wedge f_c(p,S) \Rightarrow f_c(q,S)$

Primer anti-monotone funkcije

Boolova funkcija $f_c(p, S)$: stopnja pokritosti je *vsaj* ϵ

$$f_c(p, S) \equiv c(p, S) \geq \epsilon$$

- Definicija splošnosti $p \preceq q \Rightarrow coverage(q, S) \subseteq coverage(p, S)$
- Očitno velja $coverage(q, S) \subseteq coverage(p, S) \Rightarrow |coverage(q, S)| \leq |coverage(p, S)|$
- Torej anti-monotonost $p \preceq q \wedge f_c(q, S) \Rightarrow f_c(p, S)$

Algoritem od splošnega k specifičnemu

```
function G2S(Init, Stop, Cond, S)
   Q = Init
   R = ∅
   while not Stop do
      izberi p iz Q:  Q = Q \ {p}
      if Cond(p, S) then
         R = R ∪ {p}
      Q = Q ∪ ρ(p)
   return R
```

Kaj če je *Cond* anti-monotona Boolova funkcija?

Rezanje preiskovalnega prostora

Če je *Cond* anti-monotona funkcija

$$\forall p, q \in \rho(p), S : \neg Cond(p, S) \Rightarrow \neg Cond(q, S)$$

- Če trenutno obravnavana hipoteza p ne izpolnjuje pogoja
- Potem nobena njena izostritev $q \in \rho(p)$ ne bo izpolnjevala pogoja

Priložnost za rezanje

$\rho(p)$ dodamo Q le v primeru, da velja $Cond(p, S) = \top$

Algoritem od splošnega k specifičnemu z rezanjem

function $G2SPrune(Init, Stop, Cond, S)$
    $Q = Init$
    $R = \emptyset$
    while not $Stop$ **do**
        izberi p iz Q: $Q = Q \setminus \{p\}$
        if $Cond(p, S)$ **then**
            $R = R \cup \{p\}$
            $Q = Q \cup \rho(p)$
    return R

Primerjaj rdečo vrstico z algoritmom iz prosojnice 9.

Definicija

Optimalen operator izostritve ρ

$$\forall p \, \exists! \; p_0, p_1, p_2, \ldots p_n = p$$

- $p_0 = \top$
- $p_i \in \rho(p_{i-1}), i = 1, 2, \ldots n$
- Vsako hipotezo p lahko izpeljemo *samo na en način*

Primer optimalnega operatorja za množice postavk

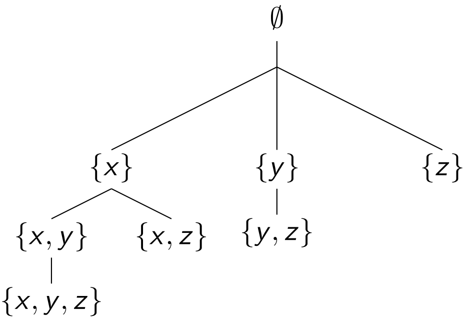
Dodajanje elementov v nekem vrstnem redu

$$\rho_O(I) = I \cup \{j\}, \forall i \in I : i \ll j$$

- $\ll$ je arbitrarna ureditev elementov v U
- U množica vseh možnih elementov (postavk)

Grafična ponazoritev ρ_O za $U = \{x, y, z\}$

$$x \ll y \ll z$$



Izčrpno iskanje

function $G2SExhaustive(Init, Stop, Cond, S)$
  $Q = Init$
  $R = \emptyset$
  while not $Stop$ **do**
    izberi p iz Q: $Q = Q \setminus \{p\}$
    if $Cond(p, S)$ **then**
      $R = R \cup \{p\}$
    $Q = Q \cup \rho(p)$
  return R

ρ je *optimalen* operator izostritve

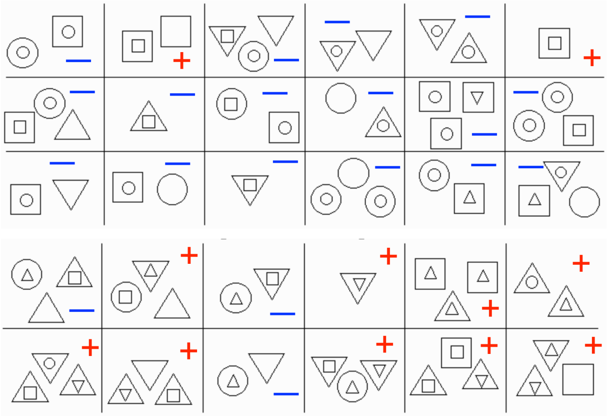
Primer: Iskanje pogostih množic relacijskih postavk

function $RFI(S, \epsilon)$
  $Q = \{(\top, 0)\}$
  while $Q \neq \emptyset$ **do**
    for $s \in S$ **do**
      for $(p, c) \in Q$ **do**
        if $covers(p, s)$ **then** $p = p + 1$
    $R = \{p : (p, c) \in C \wedge c \geq \epsilon\}$
    $Q = \bigcup_{p \in R \wedge (p,c) \in Q} \rho(p)$
  return Q

- Operator izostritve: θ-subsumpcija (prejšnja predavanja)
- Za zagotavljanje optimalnosti dodatni testi sintaktičnih variant

BONGARD: 30 primerov za dvojiško razvrščanje

Primere oštevilčimo od leve proti desni, nato navzdol



Grafična ponazoritev izvajanja *RFI* na BONGARD

Na tabli.

Definicija

Idealen operator izostritve ρ

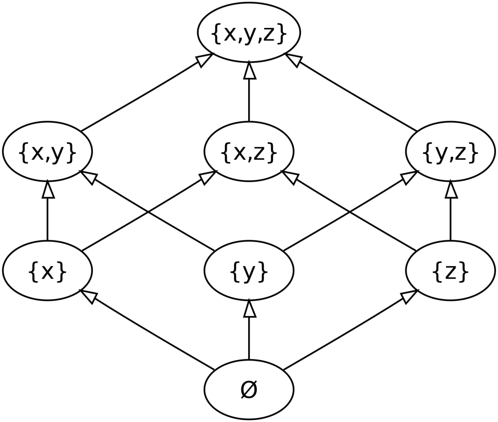
$$\rho(p) = \min(\{q : p \preceq q\})$$

- Izostritev p gre le en korak naprej
- Tvori le neposredne sosede p v Hassejevem diagramu

Primer idealnega operatorja za množice postavk

$$\rho_I(I) = I \cup \{j\}, j \notin I$$

Grafična ponazoritev ρ_I za $U = \{x, y, z\}$



Hevristično iskanje: iskanje s snopom, *beam search*

function $G2SBeam(f, S, b)$
  $Q = \{(\top, f(\top, S))\}$
  while $\top$ **do**
    $\mathrm{R} = \mathrm{Q}$
    izberi prvi (p, v) iz R: $R = R \setminus \{p\}$
    for $q \in \rho(p)$ **do**
      $R = R \cup \{(q, f(q, S))\}$
    uredi R po padajočih vrednosti $v : (p, v) \in R$
    ohrani le b prvih elementov R
    if $R = Q$ **then return** R

- f poljubna funkcija, ki za podano hipotezo p in podatkovno množico S vrne realno število
- ρ_I je *idealen* operator izostritve
- $b = 1$: algoritem vzpon po hribu, *hill climbing*

Definicija naloge

Podano

- Podatkovna množica S: $X_i : D_i = \mathbb{R}, i = 1, 2, \ldots p$; $Y : D_Y = \mathbb{R}$
- Definicija prostora $\mathcal{E}$ aritmetičnih izrazov $E(X)$ iz spremenljivk X

Najdi enačbo oblike $Y = E(X)$ za katero

$$\min_{E(X)\in\mathcal{E}} \sum_{(\boldsymbol{x},y)\in S} (y - E(\boldsymbol{x}))^2$$

Definicija prostora možnih enačb in gramatike

$$
\begin{array}{lcl}
E & \rightarrow & E + F \mid E - F \mid F \\
F & \rightarrow & F \cdot T \mid F / T \mid T \\
T & \rightarrow & const \mid V \mid (E) \\
 & & \\
V & \rightarrow & X_1 \mid X_2 \mid \ldots \mid X_p
\end{array}
$$

Definicija gramatike

Gramatike uporabljamo za definiranje sintakse jezikov.

Kontekstno neodvisna (*context-free*) gramatika $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{P}, SN)$

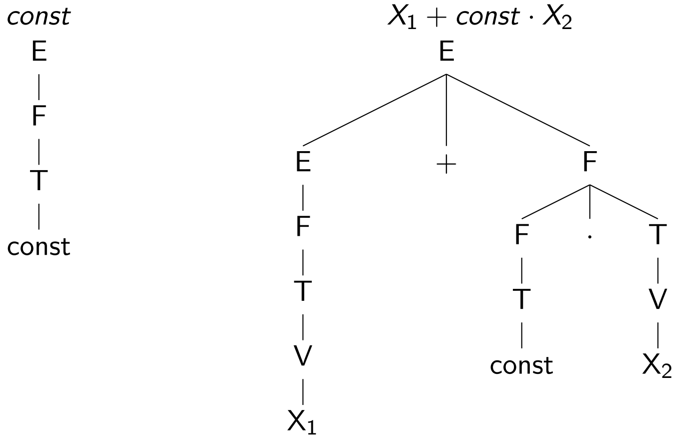
- $\mathcal{N}$ je množica *nekončnih* simbolov, *nonterminals*
- $\mathcal{T}$ je množica *končnih* simbolov, *terminals*
- $\mathcal{P}$ je množica produkcij oblike $A \to W$
  - A je ne-končni simbol $A \in \mathcal{N}$
  - B je niz končnih in nekončnih simbolov $W \in (\mathcal{N} \cup \mathcal{T})^*$
- SN je začetni simbol, $SN \in \mathcal{N}$

Univerzalna gramatika za poljuben aritmetični izraz

- $\mathcal{N} = \{E, F, T, V\}$
- $\mathcal{T} = \{+, -, \cdot, /, const, (,), X_1, \ldots X_p\}$
- $\mathcal{P}$ podana na prosojnici 22
- $SN = E$

Kako izpeljemo aritmetični izraz s pomočjo gramatike?

Primera dreves izpeljave za *const* in $X_1 + const \cdot X_2$



Od izpeljave do izraza: preberemo končna vozlišča z leve proti desni.

Definicija drevesa izpeljave, *parse tree*

Oznake vozlišč

- Notranja vozlišča imajo oznake iz $\mathcal{N}$
- Končna vozlišča imajo oznake iz $\mathcal{T}$
- Oznaka korenskega vozlišča je začetni simbol $SN \in \mathcal{N}$

Nasledniki vozlišča n z oznako A imajo oznake $A_1, A_2, \ldots, A_k$

Če obstaja produkcija $A \to A_1 A_2 \ldots A_k \in \mathcal{P}$.

Algoritem za tvorjenje dreves izpeljave

function GenerateTree(G, A)
  if $A \in \mathcal{T}$ **then return** $leaf(A)$
  izberi produkcijo $A \rightarrow A_1 A_2 \ldots A_k \in \mathcal{P}$
  for $i = 1$ **to** k **do**
    $t_i = GenerateTree(G, A_i)$
  return $tree(node(A), \{t_1, t_2, \ldots, t_k\})$

Začetni klic: $GenerateTree(G, SN)$

Operatorji izostritve za drevesa izpeljave

Rabimo

- Urediti izpeljave od *splošnega* proti *specifičnemu*
- V tem primeru od *enostavnih* izpeljav proti *zapletenim*

- Rešitev: urejanje dreves izpeljave po globini.

Globina produkcije

Globine končnih simbolov $T \in \mathcal{T}$

$$d(T) = 0$$

Globine nekončnih simbolov $N \in \mathcal{N}$

$$d(N) = \min_{P \in \mathcal{P} : P = N \to W} d(P)$$

Globina produkcije $P = A \to A_1 A_2 \ldots A_k$

$$d(P) = 1 + \max_{i=1}^{k} d(A_i)$$

Globina najbolj plitvega drevesa izpeljave s korenskim vozliščem A.

Globine produkcij za univerzalno gramatiko

$\mathcal{T}, \mathcal{N}, \mathcal{P}$	$d = 0$	$d = 1$		$d = 2$		$d = 3$		$d = 4$
$\forall t \in \mathcal{T}$	0	0	0	0	0	0	0	0
E							3	3
F					2	2	2	2
T			1	1	1	1	1	1
V			1	1	1	1	1	1
$E \to E + F$								4
$E \to E + F$								4
$E \to F$						3	3	3
$F \to F \cdot T$						3	3	3
$F \to F / T$						3	3	3
$F \to T$				2	2	2	2	2
$T \to const$		1	1	1	1	1	1	1
$T \to E$								4
$T \to V$				2	2	2	2	2
$V \to X_i$		1	1	1	1	1	1	1

Urejeni seznami produkcij po globini

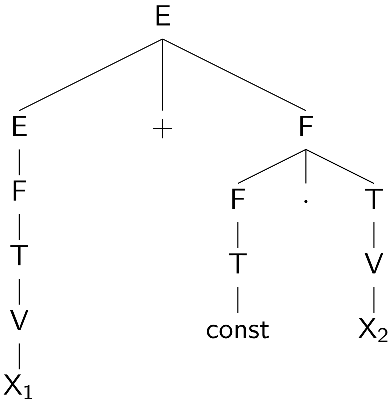
1. $V : V \to X_1 \ll V \to X_2 \ll \ldots \ll V \to X_p$
2. $T : T \to const \ll T \to V \ll T \to (E)$
3. $F : F \to T \ll F \to F \cdot T \ll F \to F/T$
4. $E : E \to F \ll E \to F + T \ll E \to E - F$

Optimalen operator izostritve

1. Izberi najbolj desno nekončno vozlišče v drevesu izpeljav
   - Naj bo oznaka vozlišča N, uporabljena produkcija $N \to W$
   - Ob vračanju poišči najbolj desno nekončno vozlišče levo od N
   - Če ob vračanju ni več takih vozlišč, $\rho_O(PT) = \emptyset$
2. Poišči naslednico $N \to W'$ produkcije $N \to W$
   - Če take produkcije ni, se vrni na korak 1
   - Zamenjaj produkcijo $N \to W$ z naslednico $N \to W'$
3. Vozlišča, ki so nastala zaradi W' rekurzivno zaključi z najbolj enostavnimi (plitvimi) produkcijami brez predhodnic
4. Dobljeno drevo PT' je rezultat izostritve, $\rho_O(PT) = \{PT'\}$

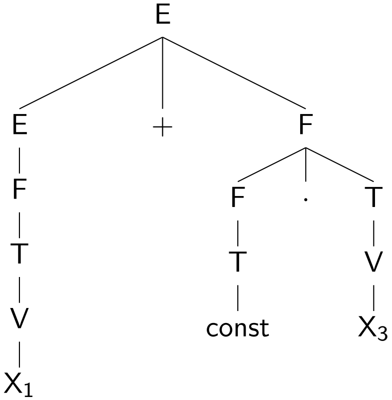
Primer delovanja ρ_O

Začetno drevo



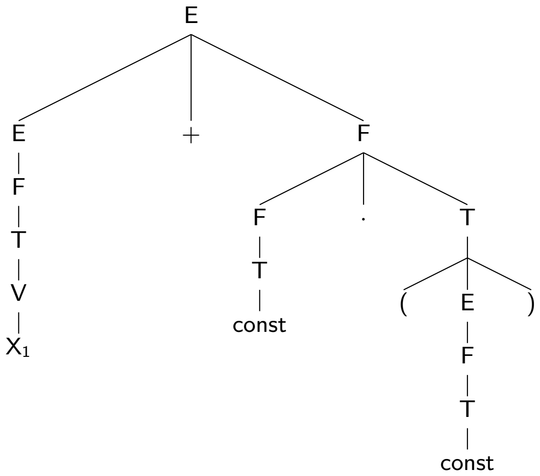
Izostritev drevesa, če je $p > 2$

Najbolj desno vozlišče V, produkcijo $V \to X_2$ zamenjamo z $V \to X_3$



Izostritev drevesa, če je $p = 2$

Najbolj desno vozlišče T, produkcijo $T \rightarrow V$ zamenjamo s $T \rightarrow (E)$

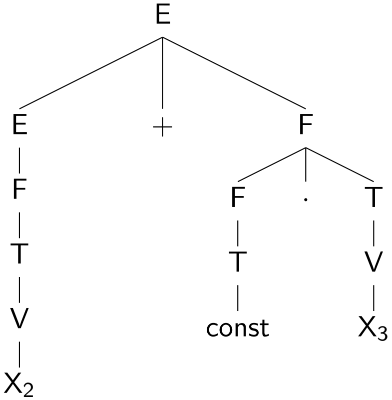


Idealen operator izostritve

1. Za vsako nekončno vozlišče v drevesu izpeljav, N, $N \rightarrow W$
   - Poišči naslednico $N \rightarrow W'$ produkcije $N \rightarrow W$
   - Če take produkcije ni, nadaljuj z naslednjo iteracijo
   - Zamenjaj produkcijo $N \rightarrow W$ z naslednico $N \rightarrow W'$
   - Vozlišča, ki so nastala zaradi W' rekurzivno zaključi z najbolj enostavnimi (plitvimi) produkcijami brez predhodnic
   - Dobljeno drevo PT' dodaj v množico izostritev R
2. Vrni množico izostritev, $\rho_I(PT) = R$

Primer izostritve za ρ_l in drevo s prosojnice 33

Najbolj levo vozlišče V, produkcijo $V \to X_1$ zamenjamo z $V \to X_2$



Algoritem Lagramge za odkrivanje enačb

Uporablja operatorje izostritve za naštevanje možnih enačb

- ρ_O v kombinaciji z izčrpnim iskanjem *G2SExhaustive*
- ρ_I v kombinaciji h hevrističnim iskanjem *G2SBeam*
- V primeru rekurzivnih gramatik in izčrpnega iskanja nastavimo maksimalno globino dreves izpeljave d_{max}
- Pri hevrističnem iskanju lahko tudi $h_{max} = \infty$

Kaj pa funkcija vrednotenja hipoteze $f(E(X), S)$?

Lagramge in vrednotenje hipotez

$$\min_{const} f(E(X), S) : f(E(X), S) = \frac{1}{|S|} \sum_{(\boldsymbol{x}, y) \in S} (y - \hat{y})^2$$

Izračun $\hat{y}$ odvisen od tipa enačb

- Navadne algebraične enačbe: izračun $\hat{y} = E(\boldsymbol{x})$
- Diferencialne enačbe: simulacija $dy/dt = E(\boldsymbol{x})$ s podatki iz S

Rešitev problema minimizacije

- Iščemo optimalne vrednosti konstant *const* v $E(X)$
- Poljuben algoritem za numerično optimizacijo (nadomestki!)

Primeri različnih gramatik

Polinomi

$$\begin{aligned} P &\rightarrow P + const \cdot T \mid const \\ T &\rightarrow T \cdot V \mid V \end{aligned}$$

Populacijska dinamika, na osnovi predznanja

$$\begin{aligned} PD &\rightarrow PDPrey, PDPredator \\ \\ PDPrey &\rightarrow Growth - Interaction \\ PDPredator &\rightarrow const \cdot \underline{Interaction} - Decay \\ \\ Growth &\rightarrow const \cdot V_{prey} \mid const \cdot (1 - V_{prey} / const) \\ &\vdots \end{aligned}$$

Učinki predznanja na primeru populacijske dinamike

d	$N(PD, d)$	$N(E, d)$
1	1	0
2	1	0
3	1	7
4	121	36
5	**1831**	7300
6	27481	14674005
7	412231	$2.3607 \cdot 10^{12}$
8	6183481	$5.5481 \cdot 10^{21}$
9	92752231	$\mathbf{3.8267 \cdot 10^{38}}$
10	$1.3913 \cdot 10^{9}$	$1.2462 \cdot 10^{68}$

Uporaba algoritmov za odkrivanje enačb

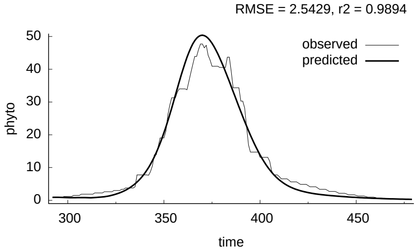
Populacijska dinamika

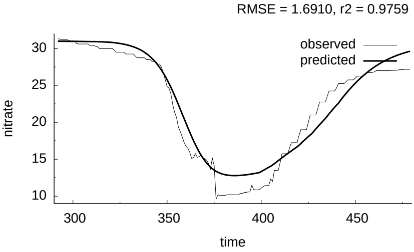
- Napovedovanje rasti planktona v Blejskem jezeru
- Snovanje čistilnih strategij v Rossovem morju, Antarktika

Dinamika biokemičnih reakcij

- Analiza interakcije proteinov pri imunski reakciji
- Modeliranje regulacijskih omrežij genov

Rezultati za Rossovo morje, Antarktika





Kako in zakaj?

Kako?

- Pretvorba relacijskega problema v navaden problem strojnega učenja
- S pomočjo vpeljave novih napovednih spremenljivk
- Kako vpeljemo nove spremenljivke?

Zakaj?

Ker lahko problem rešujemo z običajnimi algoritmi strojnega učenja.

Kako vpeljemo nove spremenljivke?

Spomnimo se kaj so notranja vozlišča v relacijskih drevesih

Poizvedbe.

Poizvedbe lahko definirajo nove spremenljivke

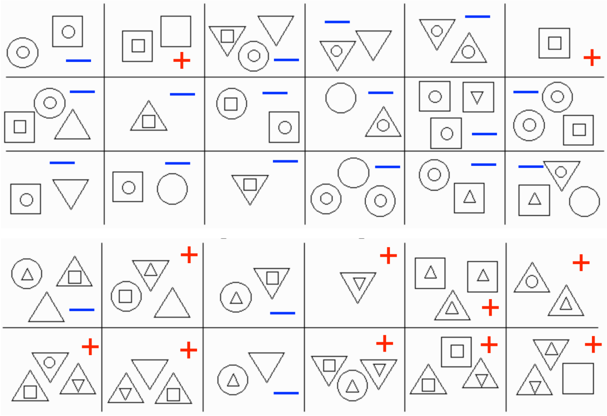
- Naredimo tabelarično učno množico iz Herbrandovih interpretacij
- Primeri so isti kot v relacijski množici
- Spremenljivke ustrezajo uspešnosti poizvedbe za vsak primer posebej

Tip novih spremenljivk

- Boolove spremenljivke: ali je bila poizvedba uspešna?
- Numerične spremenljivke: koliko krat je bila poizvedba uspešna?

BONGARD: 30 primerov za dvojiško razvrščanje

Primere oštevilčimo od leve proti desni, nato navzdol



BONGARD: Herbrandove interpretacije primerov

$e_1, \ominus$	krog(k1), krog(k2), krog(k3), kvadrat(v1), vsebuje(k1, k2), vsebuje(v1, k3)
$e_2, \oplus$	kvadrat(v1), kvadrat(v2), kvadrat(v3), vsebuje(v1, v2)
$e_{20}, \oplus$	krog(k1), kvadrat(v1), trikotnik(t1), trikotnik(t2), trikotnik(t3), vsebuje(k1, v1), vsebuje(t1, t2)

Poizvedbe, kot jih tvori algoritem *G2S*

1. $\leftarrow trikotnik(O_1)$
2. $\leftarrow krog(O_1)$
3. $\leftarrow kvadrat(O_1)$
4. $\leftarrow vsebuje(O_1, O_2)$
5. $\leftarrow trikotnik(O_1), vsebuje(O_1, O_2)$
6. $\leftarrow krog(O_1), vsebuje(O_1, O_2)$
7. $\leftarrow kvadrat(O_1), vsebuje(O_1, O_2)$
8. $\leftarrow trikotnik(O_1), vsebuje(O_1, O_2), trikotnik(O_2)$
9. $\leftarrow trikotnik(O_1), vsebuje(O_1, O_2), krog(O_2)$

$\vdots$

Od poizvedb do Boolovih spremenljivk

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	$\ldots$	Y
e_1	$\bot$	$\top$	$\top$	$\top$	$\bot$	$\top$	$\top$	$\bot$	$\bot$		$\ominus$
e_2	$\bot$	$\bot$	$\top$	$\top$	$\bot$	$\bot$	$\top$	$\bot$	$\bot$		$\oplus$
$\vdots$											$\vdots$
e_{20}	$\top$	$\top$	$\top$	$\top$	$\top$	$\top$	$\bot$	$\top$	$\bot$		$\oplus$
$\vdots$											$\vdots$

Učna množica za poljuben algoritem strojnega učenja.

Od poizvedb do numeričnih spremenljivk

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	$\dots$	Y
e_1	0	3	1	2	0	1	1	0	0		$\ominus$
e_2	0	0	3	1	0	0	1	0	0		$\oplus$
$\vdots$											$\vdots$
e_{20}	3	1	1	2	1	1	0	1	0		$\oplus$
$\vdots$											$\vdots$

Učna množica za poljuben algoritem strojnega učenja.

Odkrivanje enačb

Vpeljava novih spremenljivk z naborom transformacij

- Z množenjem do določene, omejene stopnje: X_1^2, X_1X_2, X_1X_3, …, X_1X_p, X_2^2, X_2X_3, …, X_2X_p, …X_p^2, …X_p^5
- Z apliciranjem funkcij
- S kombinacijami enih in drugih transformacij

Redka linearna regresija nad razširjenim naborom spremenljivk

$$\min_{\boldsymbol{\beta}} \sum_{(\boldsymbol{x}_T, y) \in S} (y - \boldsymbol{\beta}^T \boldsymbol{x}_T)^2 + \lambda \|\boldsymbol{\beta}\|_1$$

Z regularizacijo poskrbimo za to, da je malo parametrov $\beta \neq 0$.

Problem statičnega pristopa

Prekletstvo večdimenzionalnosti.

Osnovna ideja

Generator novih spremenljivk vgradimo v učni algoritem

- Namesto, da bi vse nove spremenljivke vpeljali vnaprej
- Algoritem tvori nove spremenljivke sproti in po potrebi

Relacijski naključni gozdovi

Naivni izbor testa

- Uporabimo algoritem *G2SExhaustive* in ρ_O za tvorjenje vseh testov
- Naključno izberemo vzorec *mtry* testov
- *mtry* je parameter naključnih gozdov, ki določa velikost vzorca naključno izbranih testov

Dejanska implementacija

- Naključno tvorjenje možnih testov
- Moramo paziti, da zagotovimo uniformno distribucijo verjetnosti izbire
- Pripravimo stohastični operator izostritve

Viri

Operatorji izostritve

- Učbenik (De Raedt 2008): Logical and Relational Learning, poglavje 3
- Magistrska naloga (Todorovski 1998): Lagramge
- (Todorovski 2010, 2017): Encyclopedia of Machine Learning

Propozicionalizacija

- Učbenik (De Raedt 2008): poglavja 4.12 in 4.13
- (Van Assche in ost 2006): relacijski naključni gozdovi