

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Inlärnning av Regler

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

- 1 Regler
 - Predikat
 - Horn-satser
- 2 Inlärnning av Predikat
 - Beslutsträd
 - Sekvensiell täckning
- 3 Induktiv Logikprogrammering
 - Top-down
 - Bottom-up

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

- 1 Regler
 - Predikat
 - Horn-satser
- 2 Inlärnning av Predikat
 - Beslutsträd
 - Sekvensiell täckning
- 3 Induktiv Logikprogrammering
 - Top-down
 - Bottom-up

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Predikat

Predikat — Logiska uttryck

Beroende av värdet av ett antal attribut

Regnigt \wedge Kallt

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Horn-satser

Horn-satser — Första ordningens logik

Kan innehålla **variabler**

$\text{syskon}(x, y) : \text{förälder}(z, x) \wedge \text{förälder}(z, y)$

Betydligt mer uttrycks kraft än predikatlogik

Grunden för t.ex. *Prolog*

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Horn-satser

Varför vill man lära regler?

- Kan översättas till "vanligt språk"
- Förklarande

Nackdelar?

- Svårt att hantera brus och osäkerhet

Regler 000 Inlärnin g av Predikat 0000 Induktiv Logikprogrammering 0000

- 1 Regler
 - Predikat
 - Horn-satser
- 2 Inlärnin g av Predikat
 - Beslutsträd
 - Sekvensiell täckning
- 3 Induktiv Logikprogrammering
 - Top-down
 - Bottom-up

Regler 000 Inlärnin g av Predikat 0000 Induktiv Logikprogrammering 0000

Beslutsträd

Disjunktion av konjunktioner

Regler 000 Inlärnin g av Predikat 0000 Induktiv Logikprogrammering 0000

Sekvensiell täckning

(Sequential Covering)

Idén bakom *Sekvensiell täckning*

Försök inte förklara allting på en gång!

- 1 Konstruera **en** regel som
 - Matchar många positiva exempel
 - Inte matchar några negativa exempel
- 2 Tag bort de positiva exempel som matchades
- 3 Upprepa tills alla positiva exempel matchats

Regler 000 Inlärnin g av Predikat 0000 Induktiv Logikprogrammering 0000

Sekvensiell täckning

Hur konstruerar man en bra regel?

Girig sökning — Välj succesivt attribut som matchar så många positiva exempel som möjligt

Regler 000 Inlärnin g av Predikat 0000 Induktiv Logikprogrammering 0000

Beam search

Alternativ till girig sökning
Flera alternativa sökvägar hålls aktuella

Regler 000 Inlärnin g av Predikat 0000 Induktiv Logikprogrammering 0000

Sekvensiell täckning

Resultaten från sökningen: flera regler

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

- 1 Regler
 - Predikat
 - Horn-satser
- 2 Inlärnning av Predikat
 - Beslutsträd
 - Sekvensiell täckning
- 3 Induktiv Logikprogrammering
 - Top-down
 - Bottom-up

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Induktiv Logikprogrammering

ILP — Inductive Logic Programming
 Automatisk konstruktion av Horn-satser från tränings exempel

Horn-satser

- Uttrycksfullare än predikat
- Variabler i beskrivningarna

Exempel

$$\begin{aligned} \text{farfar}(x, y) : & \text{ förälder}(x, z) \\ & \wedge \text{ man}(x) \\ & \wedge \text{ förälder}(z, y) \\ & \wedge \text{ man}(z) \end{aligned}$$

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Induktiv Logikprogrammering

Hur kan ett program automatiskt hitta regeln

$$\text{farfar}(x, y) : \text{ förälder}(x, z) \wedge \text{ man}(x) \wedge \text{ förälder}(z, y) \wedge \text{ man}(z)$$

från exempel av typen

$$\begin{aligned} \text{farfar}(\text{Sven}, \text{Pär}) & \mapsto \mathbf{False} \\ \text{farfar}(\text{Pär}, \text{Sven}) & \mapsto \mathbf{True} \\ \text{farfar}(\text{Lisa}, \text{Sven}) & \mapsto \mathbf{False} \end{aligned}$$

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

- **Top-down**
 Utnyttja sekvensiell täckning
 Starta från en generell regel och specialisera
- **Bottom-up**
 Starta från ett exempel och konstruera förklarande regler

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Top-down

Top-down teknik för att hitta **en regel**

- Stora generellt

$$\text{farfar}(x, y) : \mathbf{True}$$

- Generera möjliga specialiseringar

$$\begin{aligned} \text{farfar}(x, y) : & \text{ förälder}(x, y) \\ \text{farfar}(x, y) : & \text{ förälder}(x, z) \\ \text{farfar}(x, y) : & \text{ kvinna}(x) \\ \text{farfar}(x, y) : & \text{ man}(x) \end{aligned}$$

- Behåll den som bäst beskriver data
- Om regeln matchar negativa exempel **specialiseras den**

Regler 000 Inlärnning av Predikat 00000 Induktiv Logikprogrammering 0000

Top-down

Hur specialiserar man en regel?

- Lägg på en konjunktionsdel

$$\begin{aligned} \text{farfar}(x, y) : & \text{ förälder}(x, z) \wedge \text{ man}(x) \\ \text{farfar}(x, y) : & \text{ förälder}(x, z) \wedge \text{ man}(z) \end{aligned}$$

- Välj den bästa

Detta upprepas tills inga negativa exempel matchas

Hur konstruerar man de nya faktorerna i regeln?

- Prova **alla kända regler** med "gamla" variabler
- Sätt in **nya variabler**, utom på en plats
- Använd **likhet** och olikhet
- Använd **typinformation** för beskärning

farfar(x, y) : förälder(x, y)
 förälder(y, x)
 man(x)
 man(y)

farfar(x, y) : förälder(x, z)
 förälder(z, x)
 förälder(y, z)
 förälder(z, y)

farfar(x, y) : $x = y$
 $x \neq y$
 $x < y$

Bottom-up teknik

Inverse Resolution

Använd metoder för automatisk härledning "baklänges"

- Starta från **ett exempel**
- Bygg en förklarande regel
- Tag bort förklarade data
- Upprepa