

Induktiivne loogiline programmeerimine

Igor Kuzmitšov
Arvutiteaduse instituut, Tartu Ülikool
kuzmiigo@ut.ee

Andmekaevandamise uurimisseminar MTAT.03.169.
Arvutiteaduse instituut, Tartu Ülikool
Detsember 2003, lk. 34–49

Kokkuvõte

Induktiivne loogiline programmeerimine (ILP) on tehisintellekti (masinõppe) ala, mis tegeleb seoste tuvastamise ja hüpoteespredikaadi tuletamisega etteantud näidete ja taustteadmiste alusel. Selle meetodi üheks oluliseks omapäraks on näidete, taustteadmiste ja tuletatava predikaadi esitus loogiliste programmide abil, mis annab võimaluse kasutada struktureeritud andmeid ja saada kergesti tõlgendatava tulemuse. ILP on näidanud oma jõudu andmeseoste õppimisel sellistes valdkondades nagu molekulaarbioloogia (ravimite omadused, valkude kolmemõõtmeline struktuur), keeletehnoloogia (inimkeelte morfoloogia, lausete andmebaasipäringuteks teisendamine), meditsiinidiagnostika, mehhaanika. Käesoleva artikli ülesandeks on anda ülevaade induktiivsest loogilisest programmeerimisest, selle põhimeetoditest ja mõnedest rakendustest.

1. Sissejuhatus

Induktiivne loogiline programmeerimine on induktiivse masinõppe ja loogilise programmeerimise ühendus, mille eesmärkideks on arendada vahendid seoste tuvastamiseks ja hüpoteeside tuletamiseks etteantud näidete alusel ning kogemustest uute teadmiste loomiseks. Püstitatud eesmärkide saavutamiseks kasutab ILP loogilist programmeerimist, mille abil võib esitada olemasolevad valdkonna teadmised (ingl. *domain knowledge*) ja õppenäited ning saadavad tulemused (tuletatud predikaadid) loogiliste programmidenä. Nii on

võimalik kasutada struktureeritud andmeid ja saada kergesti tõlgendatavaid, inimkeelele lähedasi tulemusi.

Traditsioonilised arvutiloogika rakendused kasutavad deduktsiooni, s.t. tuletavad antud valemid olemasolevatest faktidest ja üldistest aksioomidest. Induktiivne lähenemisviis toimib aga vastassuunas: tuletab üldised reeglid (programmid) antud spetsiifilistest näidetest ja faktidest selliselt, et põhjendada teadaolevaid ja ennustada võimalikult täpselt seni nägemata (ingl. *unseen*) fakte.

Erinevalt paljudest meetoditest (nt. neurovõrkudest) on ILP-e väljund kergesti arusaadav inimestele.

Käesolev artikkel on ülevaade ILP-e ideedest, terminoloogiast, meetoditest ja rakendustest.

Sissejuhatuse loogikasse, resolutsioonimeetodisse ja loogilisesse programmeerimisesse leiab raamatust (Tamme, Tammet, & Prank 1997). Induktiivset loogilist programmeerimist käsitletakse põhjalikult raamatus (Lavrač & Džeroski 1994), mis peale sügava teoreetilise osa sisaldab ka erinevate realiseerimise detailset kirjeldust ja rakenduste ülevaadet. See raamat ja artiklid (de Mantaras & Armengol 1998; Muggleton & Raedt 1994) on ülevaate põhiallikateks.

Käesolev artikkel on üles ehitatud järgmiselt: sissejuhatuse induktiivsesse loogilisesse programmeerimisse (põhidefinitioonid ja -ideed) on antud osas 2. Osa 3 räägib ILP-e põhimeetoditest (üldistamisest ja spetsialiseerimisest) ning ILP-süsteemide tüüpidest ja realiseerimistest. Osa 4 tutvustab ILP-e rakendusi erinevates valdkondades. Osa 5 on pühendatud kokkuvõtlikele märkustele.

2. Induktiivne loogiline programmeerimine

ILP-e põhiülesandeks on mõistete õppimine (ingl. *concept learning*): kasutades mõiste etteantud positiivseid ja negatiivseid näiteid (võimalikult ka taustteadmisi) tuletada mõiste üldise kirjelduse (hüpoteesi). Saadud kirjeldusega peavad sobima kõik positiivsed näited ja mitte ükski negatiivne.

Taustteadmisteks (ingl. *background knowledge*) võivad olla vastava valdkonna teadmised, näidete ja tuletatava kirjelduse eeldused ja kitsendused.

Mõiste õppimine on otsing hüpoteeside ruumis, mille eesmärgiks on leida kirjeldus, mis kataks antud näited ja oleks võimalikult lihtne ja lühike. Valitud keel mõjub otsimisruumi suurusele ja sellele, mida ja kuidas on kirjeldatav antud ülesandes.

Mõistete induktiivse õppimise meetodeid saab liigitada erineval moel:

- juhitud/juhtimata (ingl. *supervised/unsupervised*) — kas programm õppimise käigus küsib nõu/kontrolli või ei.
- ühe/mitme mõiste õppimine (ingl. *single/multiple concept learning*). Esimesel juhul on tähtis ainult kas antud näited on ainult positiivsed või positiivsed ja negatiivsed. Teisel juhul on vaja ka lisaks eristada, kas näide demonstreerib ainult ühte mõistet või on tegemist (osaliselt) kattuvate kirjeldustega.
- lause-/relatsioonilised õppijad (ingl. *propositional/relational learners*). Lauseõppijad kasutavad lausearvutuse formalismi objektide kirjeldamiseks: objektid on fikseeritud atribuutide kogu, kusjuures iga atribuut omandab väärtusi vastavast etteantud hulgast. Seda tüüpi õppijate ülesandeks on selliste reeglite loomine, mille abil saab ennustada näidete kuuluvust määratud klassidesse atribuutide väärtuste põhjal. Lauseõppijatel on piiratud väljendusrikkus ja taustteadmiste kasutamise võimalus.

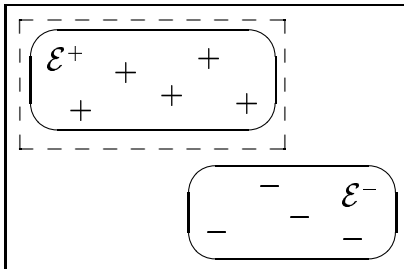
Relatsioonilised õppijad kirjeldavad objekte kui komponentidest ja nendevahelistest seostest koosnevaid struktuure. Seda tüüpi õppimistulemused on avastatud seoste (s.t. predikaatide) kirjeldused. Taustteadmiste, näidete ja hüpoteeside keeled on tavaliselt esimest järku loogika alamhulgad. See võimaldab vältida lauseõppijate piiranguid.

Meid huvitavad edaspidi relatsioonilised õppijad, mis tuletavad hüpoteese loogiliste programmide vormis. Selliseid õppijaid nimetatakse ILP-süsteemideks. Nendel on kaks induktiivselt õppimiselt päritud eesmärki: arendada vahendid relatsioonide/seoste tuvastamiseks ja hüpoteeside tuletamiseks etteantud näidete alusel ning luua uusi teadmisi olemasoleva kogemuse (taustteadmiste) baasil. ILP-t saab kasutada ka ülesannetes, kus andmete klassifitseerimine ei ole oluline ja õppimisel ei ole negatiivseid näiteid, on oluline aga avastada suuremahulistes andmetes mustreid, seoseid, omadusi. See on rakendatav näiteks andmebaaside analüüsis ja bioloogias (geneetikas).

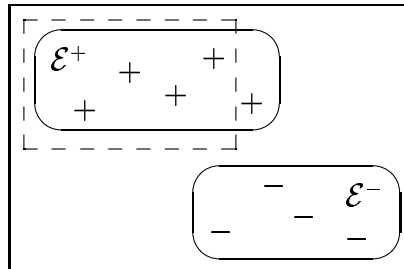
Esitame ILP-e ülesande formaalselt. Olgu antud positiivsete ja negatiivsete näidete hulk $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$, taustteadmiste hulk \mathcal{B} ja kirjeldamiskeel \mathcal{L} . Ütleme, et hüpotees \mathcal{H} katab näite $e \in \mathcal{E}$ kui leiab aset loogiline järeldumine $\mathcal{B} \cup \mathcal{H} \models e$. Toome sisse funktsiooni $\text{covers}(\mathcal{B}, \mathcal{H}, e)$, mis tagastab väärtuse *true* või *false* sõltuvalt sellest kas $\mathcal{B} \cup \mathcal{H}$ katab e või ei. Kasutame ka funktsiooni $\text{covers}(\mathcal{B}, \mathcal{H}, e) = \{e \mid e \in \mathcal{E}, \text{covers}(\mathcal{B}, \mathcal{H}, e) = \text{true}\}$.

Hüpoteesi \mathcal{H} nimetatakse *täielikuks* taustteadmiste \mathcal{B} ja näidete \mathcal{E} suhtes kui kõik positiivsed näited on kaetud: $\text{covers}(\mathcal{B}, \mathcal{H}, \mathcal{E}^+) = \mathcal{E}^+$. Hüpoteesi \mathcal{H} nimetatakse *mittevasturääkivaks* taustteadmiste \mathcal{B} ja näidete \mathcal{E} suhtes ükski negatiivne näide ei ole kaetud: $\text{covers}(\mathcal{B}, \mathcal{H}, \mathcal{E}^-) = \emptyset$. Vt. joonis 1.

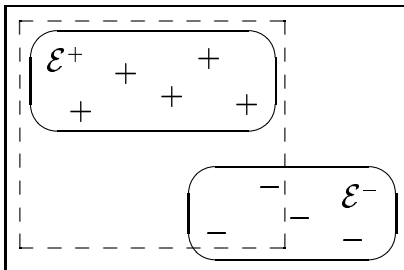
\mathcal{H} on täielik, mittevasturääkiv



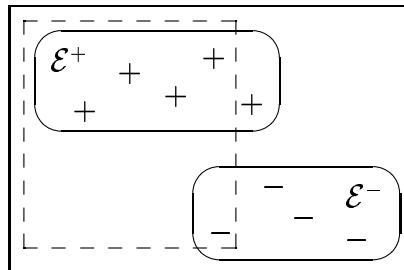
\mathcal{H} on mittetäielik, mittevasturääkiv



\mathcal{H} on täielik, vasturääkiv



\mathcal{H} on mittetäielik, vasturääkiv



Joonis 1: Hüpoteesi täielikkus ja mittevasturääkivus ($covers(\mathcal{H}, \mathcal{E})$) on näidatud punktiiriga). Allikas: (Lavrač & Džeroski 1994).

Sõnastame induktiivse loogilise programmeerimise ülesande järgmiselt: on antud taustteadmised \mathcal{B} ja näidete hulk $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$ (positiivsete ja negatiivsete näidete ühend); leida hüpoteesi \mathcal{H} mingis mõistete kirjeldamiskeeles \mathcal{L} , et \mathcal{H} on täielik ja mittevasturääkiv \mathcal{B} ja \mathcal{E} suhtes.

Termiks nimetatakse muutujat või funktsioonsümbolit koos järgneva n -kohalise termide jadaga sulgudes. Näiteks, $f(g(X), h)$ on term, kuna X on muutuja, f ja g on vastavalt 2- ja 1-kohalised funktsioonsümbolid, h on 0-kohaline ehk konstant. *Aatom* (*atomaarne valem*) on predikaat koos järgneva n -kohalise termide jadaga, näiteks $P, R(f(X), a)$. Aatomi või selle eituse nimetatakse *literaaliks* (vastavalt positiivseks või negatiivseks). *Disjunkt* (*lause*) on literaalide disjunktsioon: $\forall X_1 \dots \forall X_s (L_1 \vee \dots \vee L_m)$, kus L_i on literaalid ja X_j on nendes literaalides esinevad muutujad (kõik muutujad on vaikimisi seotud üldisuskvantoriga).

Disjunkt on võimalik esitada ka literaalide hulganäitena: disjunktile $(L_1 \vee L_2 \vee \dots \vee \overline{L_i} \vee \overline{L_{i+1}} \vee \dots)$ vastab hulk $\{L_1, L_2, \dots, \overline{L_i}, \overline{L_{i+1}}, \dots\}$. Disjunktsiooni $(L_1 \vee L_2 \vee \dots \vee \overline{L_i} \vee \overline{L_{i+1}} \vee \dots)$ saab ekvivalentsealt kirjutada ka järgmisel viisil: $(L_1 \vee L_2 \vee \dots \leftarrow L_i \wedge L_{i+1} \wedge \dots)$, kuna $\overline{X} \vee \overline{Y} \equiv \overline{X \wedge Y}$ ja $X \vee \overline{Y} \equiv X \leftarrow Y$. Loogilises programmeerimises kasutatakse viimase variandi lihtsustatud vormi: $(L_1, L_2, \dots, \leftarrow L_i, L_{i+1}, \dots)$, kus komad enne ja pärast implikatsiooni märki tähistavad vastavalt disjunktsioone ja konjunktsioone.

Horni disjunkt on disjunkt, mis sisaldab ülimalt ühte positiivset literaali. *Määratud programmidisjunkt* sisaldab täpselt ühte positiivset literaali: $T \leftarrow L_1, \dots, L_m$, kus T, L_i on atomaarsed valemid. Positiivse literaali T nimetatakse disjunktiks *peaks* ja negatiivsete literaalide konjunktsiooni nimetatakse disjunktiks *kehaks*. *Määratud programm* on määratud disjunktide kogum. *Programmidisjunkt* on disjunkt $T \leftarrow L_1, \dots, L_m$, kus T on aatom ja iga L_i on kas positiivne või negatiivne literaal. *Normaalne programm* on programmidisjunktide kogum. *Predikaadi definitsioon* on programmidisjunktide kogum, mis sisaldavad sama n -kohalise predikaatsümboli nende peades.

Näide. $tütar(X, Y) \leftarrow naine(X), vanem(Y, X)$ on määratud programmidisjunkt (programmilause). $tütar(X, Y) \leftarrow mees(X), vanem(Y, X)$ on programmilause. Esimene lause vastab valemile $\forall X \forall Y : \overline{tütar(X, Y)} \vee \overline{naine(X)} \vee \overline{vanem(Y, X)}$ või hulgale $\{tütar(X, Y), naine(X), vanem(Y, X)\}$. \triangleleft

ILP-t on kaks liiki: *empiiriline* ja *interaktiivne*. Empiiriline ILP õpib ühte relatsiooni näidete suure hulga põhjal. Interaktiivne ILP õpib korruga mitmeid (võimalikult seotud) mõisteid. Näidiste hulk on väike, aga õppimise käigus interaktiivsed süsteemid genereerivad oma näiteid ja küsivad nende klassifitseerimist (kas positiivne või negatiivne).

3. ILP meetodid

Hüpotees G on *üldisem* kui hüpotees S (ja S on spetsiifilisem kui G) siis ja ainult siis, kui $G \models S$.

Deduktiivne tuletusreegel on *spetsialiseerimisreegel* kui ta seab vastavusse lausete konjunktsioonile G lausete konjunktsiooni S nii et $G \models S$.

Induktiivne tuletusreegel on *üldistamisreegel* kui ta seab vastavusse lausete konjunktsioonile S lausete konjunktsiooni G nii et $G \models S$.

Need reeglid on kasulikud otsimisruumi lõikamiseks (ingl. *search space pruning*): kui $\mathcal{B} \cup \mathcal{H} \neq e^+$, siis ühtegi \mathcal{H} spetsialisatsioon ei rahulda e^+ ; kui $\mathcal{B} \cup \mathcal{H} \cup \{e^-\} \models false$, siis iga \mathcal{H} üldistus on mittevasturääkiv $\mathcal{B} \cup \{e^-\}$ suhtes.

Otsimisruumi struktureerimiseks toome sisse lausete (ingl. *clause*) osalise järjestuse. Selleks kasutatakse θ -liigitust. Lause c θ -liigitab c' kui leidub selline substituutsioon $\theta = \{X_1/t_1, \dots, X_k/t_k\}$, et $c\theta \subseteq c'$. Substituutsiooni θ rakendamine tähendab muutuja X_i kõigi esinemiste vahetamist sama t_i vastu. Kaks lauset c ja c' on *ekvivalentsed* θ -liigituse mõttes kui c θ -liigitab c' ja c' θ -liigitab c . Lause c on *taandatud* kui ta ei ole ekvivalentne θ -liigituse mõttes ühegi oma pärisalamhulgaga. Lause c on *vähemalt sama üldine kui* c' ($c \leq c'$) kui c θ -liigitab c' . Lause c on *rohkem üldine kui* c' kui $c \leq c'$ ja $c' \not\leq c$.

Näide. $c = \text{tütar}(X, Y) \leftarrow \text{vanem}(Y, X)$. Rakendades substituutsiooni $\theta = \{X/marja, Y/anna\}$ saame

$$c\theta = \text{tütar}(marja, anna) \leftarrow \text{vanem}(anna, marja).$$

Lause c θ -liigitab lauset $c' = \text{tütar}(X, Y) \leftarrow \text{naine}(X), \text{vanem}(Y, X)$ kasutades tühja substituutsiooni $\theta = \emptyset$, kuna

$$\{\text{tütar}(X, Y), \overline{\text{vanem}(Y, X)}\} \subset \{\text{tütar}(X, Y), \overline{\text{naine}(X)}, \overline{\text{vanem}(Y, X)}\}.$$

Lause c θ -liigitab lauset

$$\begin{aligned} \text{tütar}(marja, anna) \leftarrow \text{naine}(marja), \text{vanem}(anna, marja), \\ \text{vanem}(anna, tiit) \end{aligned}$$

kasutades substituutsiooni $\theta = \{X/marja, Y/anna\}$. ◁

Sisseviidud osalise järjestuse kasulikud omadused. Kui c θ -liigitab c' siis $c \models c'$. θ -liigitus defineerib võre üle taandatud lausete. Siit järeldub, et on olemas üksainus vähim ülemraja (ingl. *least upper bound, LUB*) ja üksainus suurim alamraja (ingl. *greatest lower bound, GLB*). Kahe taandatud lause c

ja c' vähim üldine üldistus (ingl. *least general generalisation*) $lgg(c, c')$ on c ja c' vähim ülemraja θ -liigituse võres.

Peale otsimisruumi struktureerimise ja otsimise löikamise on θ -liigitus ka hüpoteesruumi kahe uurimisstrateegia aluseks.

3.1. Üldistamine (generalization)

Üldistamise meetodid sooritavad otsingu hüpoteesruumist alt-üles, alustades kõige spetsiifilise disjunktiga (näidete alusel) ja üldistades seda kuni on võimalik säilitada mittevasturääkivust.

Disjunkti c üldistust c' saab leida rakendades θ -liigitusel põhineva *üldistusoperaatori* ρ , mis seab vastavusse disjunktile tema üldistuste hulka: $\rho(c) = \{c' \mid c' \in \mathcal{L}, c' < c\}$. Üldistusoperaatori põhioperatsioonid on pöördsubstitutsiooni rakendamine ja disjunkti kehast literaali eemaldamine.

3.1.1. Relatiivne vähim üldine üldistus

Kahe disjunkti c ja c' vähim üldine üldistus $lgg(c, c')$ on disjunktide vähim ülemraja θ -liigituse võres. lgg arvutamine toimib järgmiselt.

Termide $lgg(t_1, t_2)$

1. $lgg(t, t) = t$
2. $lgg(f(s_1, \dots, s_n), f(t_1, \dots, t_n)) = f(lgg(s_1, t_1), \dots, lgg(s_n, t_n))$
3. $lgg(f(s_1, \dots, s_n), g(t_1, \dots, t_n)) = V$, kus $f \neq g$ ja V on muutuja, mis esindab $lgg(f(s_1, \dots, s_n), g(t_1, \dots, t_n))$
4. $lgg(s, t) = V$, kus $s \neq t$, ja vähemalt üks neist on muutuja, V on muutuja, mis esindab $lgg(s, t)$

Aatomite $lgg(A_1, A_2)$

1. $lgg(p(s_1, \dots, s_n), p(t_1, \dots, t_n)) = p(lgg(s_1, t_1), \dots, lgg(s_n, t_n))$
2. $lgg(p(s_1, \dots, s_n), q(t_1, \dots, t_n))$ on defineerimata, kui $p \neq q$

Literaalide $lgg(L_1, L_2)$

1. $lgg(L_1, L_2) = lgg(A_1, A_2)$ kui L_1 ja L_2 mõlemad on positiivsed aatomid: $L_1 = A_1, L_2 = A_2$
2. $lgg(L_1, L_2) = lgg(\overline{A_1}, \overline{A_2}) = \overline{lgg(A_1, A_2)}$ kui L_1 ja L_2 mõlemad on negatiivsed aatomid: $L_1 = \overline{A_1}, L_2 = \overline{A_2}$

3. $lgg(L_1, L_2)$ on defineerimata, kui üks literaalidest on positiivne, teine aga on negatiivne

Disjunktide $lgg(c_1, c_2)$

Olgu $c_1 = \{M_1, \dots, M_m\}$ ja $c_2 = \{N_1, \dots, N_n\}$. Siis $lgg(c_1, c_2) = \{L_{ij} = lgg(M_i, N_j) \mid M_i \in c_1, N_j \in c_2, lgg(M_i, N_j) \text{ on defineeritud}\}$

Näited.

- $lgg([a, b, c], [a, c, d]) = [a, X, Y]$
- $lgg(f(a, a), f(b, b)) = f(lgg(a, b), lgg(a, b)) = f(V, V)$
- $lgg(\text{vanem}(\text{anna}, \text{marja}), \text{vanem}(\text{anna}, \text{tiit})) = \text{vanem}(\text{anna}, X)$
- $lgg(\text{vanem}(\text{anna}, \text{marja}), \overline{\text{vanem}(\text{anna}, \text{tiit})})$ on määramata
- $lgg(\text{vanem}(\text{anna}, X), \text{tütar}(\text{marja}, \text{anna}))$ on määramata
- $c_1 = \text{tütar}(\text{marja}, \text{anna}) \leftarrow \text{naine}(\text{marja}), \text{vanem}(\text{anna}, \text{marja}),$
 $c_2 = \text{tütar}(\text{ester}, \text{tiit}) \leftarrow \text{naine}(\text{ester}), \text{vanem}(\text{tiit}, \text{ester}).$
 Siis $lgg(c_1, c_2) = \text{tütar}(X, Y) \leftarrow \text{naine}(X), \text{vanem}(Y, X)$

Relatiivne vähim üldine üldistus (ingl. *relative least general generalization*, $rlgg$) on vähim üldine üldistus taustteadmiste suhtes. Olgu taustteadmised koosnevad faktidest, ja olgu K kõikide faktide konjunktsioon. Siis kahe aatomi (positiivse näite) *relatiivne vähim üldine üldistus* taustteadmiste suhtes on defineeritud järgmiselt: $rlgg(A_1, A_2) = lgg((A_1 \leftarrow K), (A_2 \leftarrow K))$.

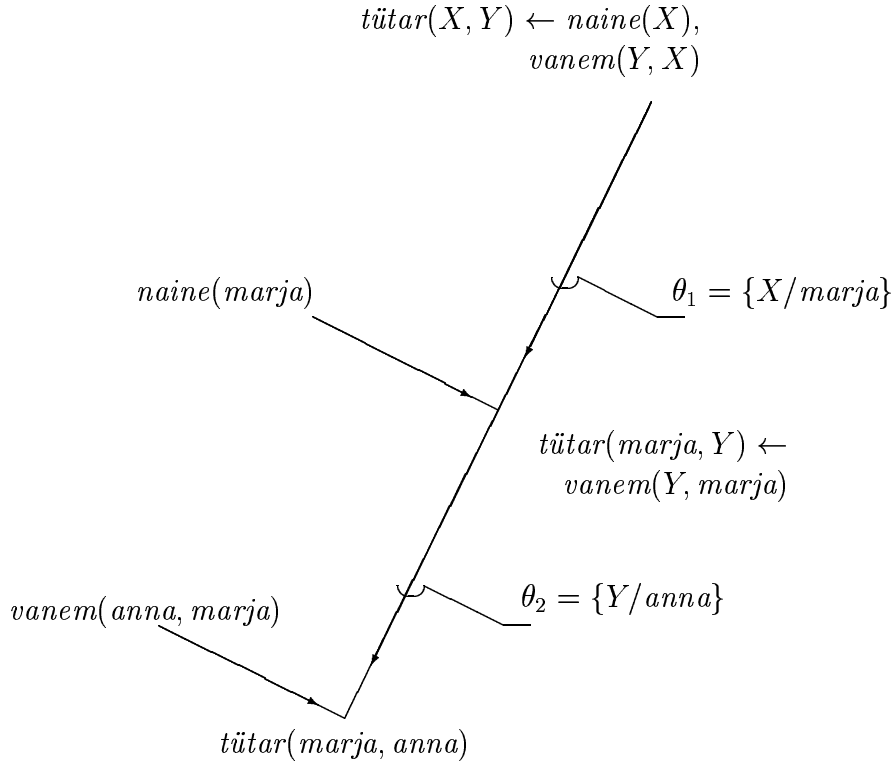
Näide. Olgu $K = \text{vanem}(\text{anna}, \text{marja}) \wedge \text{vanem}(\text{anna}, \text{tiit}) \wedge \text{vanem}(\text{tiit}, \text{ester}) \wedge \text{vanem}(\text{tiit}, \text{ivo}) \wedge \text{naine}(\text{anna}) \wedge \text{naine}(\text{marja}) \wedge \text{naine}(\text{ester}), e_1 = \text{tütar}(\text{marja}, \text{anna}), e_2 = \text{tütar}(\text{ester}, \text{tiit})$.

Siis $rlgg(e_1, e_2) = lgg((e_1 \leftarrow K), (e_2 \leftarrow K))$ annab (muutujate indeksid on vastavate esindatud termide algustähed):

$$\begin{aligned} \underline{\text{tütar}(V_{m,e}, V_{a,t})} &\leftarrow \text{vanem}(\text{anna}, \text{marja}), \text{vanem}(\text{anna}, \text{tiit}), \\ &\text{vanem}(\text{tiit}, \text{ester}), \text{vanem}(\text{tiit}, \text{ivo}), \text{naine}(\text{anna}), \\ &\text{naine}(\text{marja}), \text{naine}(\text{ester}), \text{vanem}(\text{anna}, V_{m,t}), \\ &\underline{\text{vanem}(V_{a,t}, V_{m,e})}, \text{vanem}(V_{a,t}, V_{m,i}), \text{vanem}(V_{a,t}, V_{t,e}), \\ &\text{vanem}(V_{a,t}, V_{t,i}), \text{vanem}(\text{tiit}, V_{e,i}), \text{naine}(V_{a,m}), \\ &\text{naine}(V_{a,e}), \underline{\text{naine}(V_{m,e})} \end{aligned}$$

$$\begin{aligned} \text{tütar}(V_{m,e}, V_{a,t}) &\leftarrow \text{vanem}(V_{a,t}, V_{m,e}), \text{naine}(V_{m,e}) \\ \text{tütar}(X, Y) &\leftarrow \text{naine}(X), \text{vanem}(Y, X). \end{aligned}$$

◁



Joonis 2: Resolutsiooni puu. Allikas: (Lavrač & Džeroski 1994).

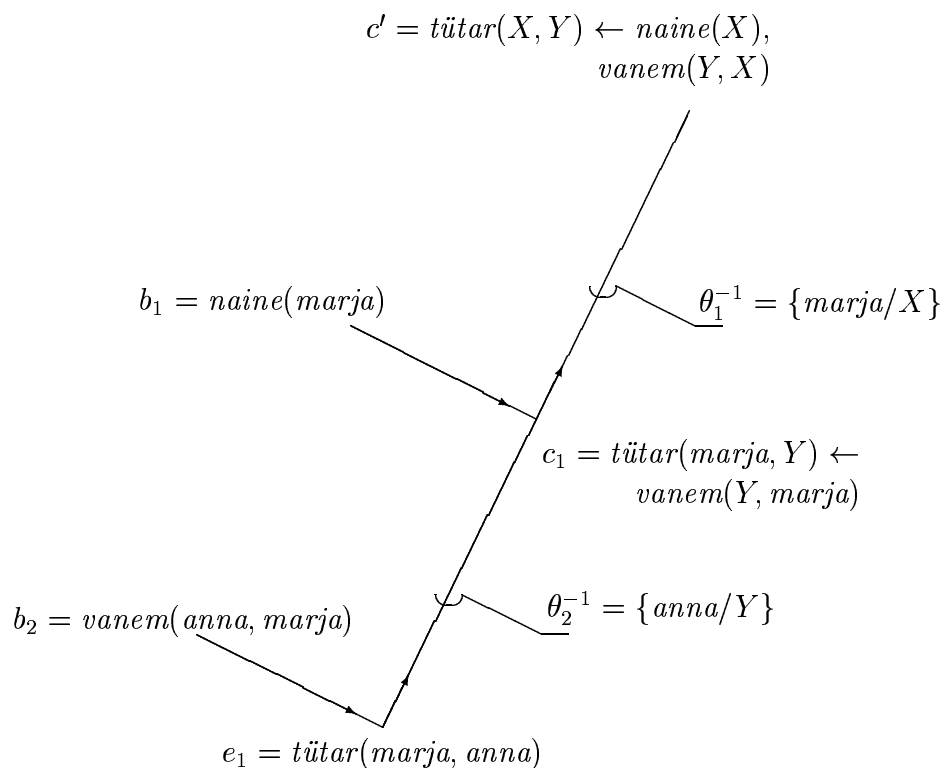
3.1.2. Pöördresolutsioon

Pöördresolutsioon pöörab resolutsiooni kasutades pöördsubstitutsioonil põhinevat üldistusoperaatorit. Substitutsiooni θ pöördsubstitutsioon θ^{-1} on substitutsioon, mis vahetab termid tagasi muutujate vastu.

Pöördresolutsioon toimib järgmiselt. Olgu antud taustteadmised $b_1 = naine(marja)$ ja $b_2 = vanem(anna, marja)$, jooksev hüpotees $\mathcal{H} = \emptyset$, positiivne näide $e_1 = tütar(marja, anna)$.

Kõigepealt, pöördresolutsioon proovib leida disjunkt c_1 nii et $\{b_2\} \cup \{c_1\} \models e_1$ ja saab lisada c_1 hüpoteesi \mathcal{H} asemele. Kasutades pöördsubstitutsiooni $\theta_2^{-1} = \{anna/Y\}$ saame $c_1 = ires(b_2, e_1) = tütar(marja, Y) \leftarrow vanem(Y, marja)$ (ingl. *ires* — *inverse resolution*). c_1 saab jooksvaks hüpoteesiks \mathcal{H} nii et $\{b_2\} \cup \mathcal{H} \models e_1$.

Teiseks, kasutame pöördsubstitutsiooni $\theta_1^{-1} = \{marja/X\}$ et leida $c' = ires(b_1, c_1) = tütar(X, Y) \leftarrow naine(X), vanem(Y, X)$. Jooksvas hüpoteesis \mathcal{H} saab nüüd vahetada c_1 üldisema c' vastu nii et $\mathcal{B} \cup \mathcal{H} \models e_1$. Saame, et $\mathcal{H} = \{tütar(X, Y) \leftarrow naine(X), vanem(Y, X)\}$.



Joonis 3: Pöördresolutsiooni puu. Allikas: (Lavrač & Džeroski 1994).

Resolutsiooni- ja pöördresolutsiooni puud on kujutatud joonistel 2 ja 3.

3.2. Spetsialiseerimine (specialization)

See meetod uurib hüpoteesruumi ülalt-alla, spetsialiseerides (konkretiseerides) üldiseid hüpoteese. Selleks kasutatakse θ -liigitusel põhinevat täpsustusoperaatorit (*spetsialiseerimisoperaatorit*), mis seab vastavusse disjunktile tema täpsustuste hulka: $\rho(c) = \{c' \mid c' \in \mathcal{L}, c < c'\}$.

ILP-e peaspetsialiseerimismeetodiks on *ülalt-alla otsing täpsustusgraafides*. Graaf on suunatud ja ilma tsükliteta, sõlmedeks on programmidisjunktid ja kaarteks on täpsustuse põhioperatsioonid: substituutsiooni rakendamine (muutuja vahetus termi vastu) ja literaali lisamine disjunktli kehasse. Õppijad alustavad kõige üldisema disjunktiga ja täpsustavad seda kuni see ei kata enam ühtegi negatiivset näidet.

Näiteks on MIS algoritm:

Initsialiseerida hüpoteesi \mathcal{H} (tühja) disjunktide hulgaga keeles \mathcal{L} .

kuni on näiteid

Lugeda järgmine näide

korrata

kui on kaetud negatiivne näide **siis**

kustutada valesid disjunkte \mathcal{H} -st

kui on positiivne näide e , mis ei ole kaetud **siis**

täpsustusgraafi laiuti-otsingu abil tuletada disjunkt c ,

mis kataks e ja lisada c hulka \mathcal{H}

kuni \mathcal{H} on täielik ja mittevasturääkiv

väljastada hüpoteesi \mathcal{H}

lõpp

Näide. Täpsustuspuus otsing on illustreeritud joonisega 4. Olgu antud järgmised taustteadmised:

vanem(anna, marja). vanem(anna, tiit). vanem(tiit, ester).

vanem(tiit, ivo). naine(anna). naine(marja). naine(ester).

Õppiija saab positiivse näite $e_1 = \textit{tütar}(\textit{marja}, \textit{anna})$. Otsing algab predikaadi $\textit{tütar}$ kõige üldisema definitsiooniga: $c = \textit{tütar}(X, Y) \leftarrow$. Kuna c katab e_1 , jooksvaks hüpoteesiks on $\mathcal{H} = \{c\}$.

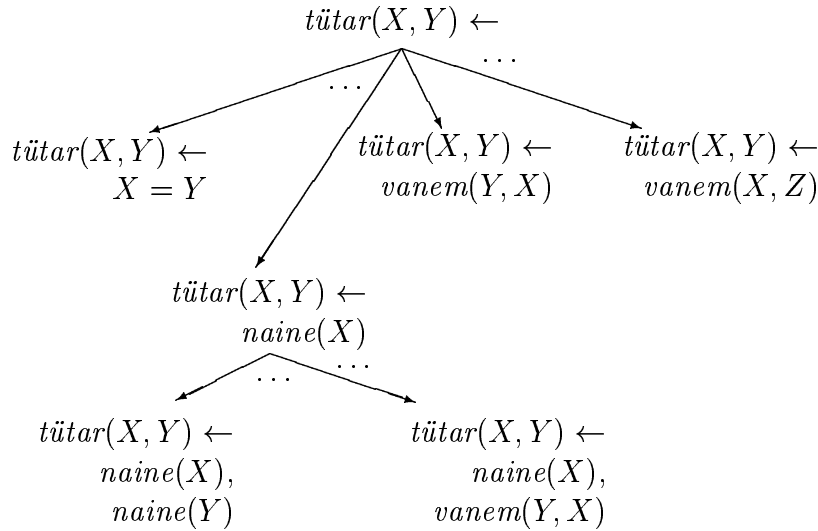
Järgmiseks positiivseks näiteks on $e_2 = \textit{tütar}(\textit{ester}, \textit{tiit})$. See on kaetud olemasoleva hüpoteesiga, sellega muudatusi pole vaja.

Olgu järgmine näide negatiivne: $e_3 = \textit{tütar}(\textit{tiit}, \textit{anna})$. Hüpoteesist kustutatakse lause c , sest see katab negatiivse näite, ja genereeritakse c täpsustuste hulka $\rho(c) = \{\textit{tütar}(X, Y) \leftarrow L\}$, kus L on:

- literaal, mille argumentideks on lause pea muutujad: $X = Y$, $\textit{naine}(X)$, $\textit{naine}(Y)$, $\textit{vanem}(X, X)$, $\textit{vanem}(X, Y)$, $\textit{vanem}(Y, X)$, $\textit{vanem}(Y, Y)$;
- või literaal, mis toob lause keha sisse uue muutuja Z ($Z \neq X$ ja $Z \neq Y$): $\textit{vanem}(X, Z)$, $\textit{vanem}(Z, X)$, $\textit{vanem}(Y, Z)$, $\textit{vanem}(Z, Y)$.

Saadud täpsustusi kontrollitakse ühe kaupa. Esimeseks täpsustuseks võetakse $\textit{tütar}(X, Y) \leftarrow X = Y$. See ei sobi, kuna ei kata positiivset näidet e_1 . Järgmiseks võetud $c' = \textit{tütar}(X, Y) \leftarrow \textit{naine}(X)$ sobib positiivsete ja negatiivsete näidete eristamiseks, see ongi jooksvaks hüpoteesiks: $\mathcal{H} = \{c\}$.

Kui on saadud järgmine negatiivne näide $e_4 = \textit{tütar}(\textit{ester}, \textit{anna})$, hüpotees saab vasturääkivaks, kuna e_4 on kaetud. Algoritm teeb hüpoteesi tühjaks ja proovib genereerida uut lauset, mis kataks e_1 . Kõigepealt kontrollitakse ülejäänud c täpsustusi. $\textit{tütar}(X, Y) \leftarrow \textit{naine}(Y)$ ja $\textit{tütar}(X, Y) \leftarrow \textit{vanem}(Y, X)$ ka ei sobi vasturääkivuse tõttu.



Joonis 4: Täpsustusgraafi osa. Allikas: (Lavrač & Džeroski 1994).

Edasi kontrollitakse $tütar(X, Y) \leftarrow X = Y$ täpsustusi. Need ei sobi, kuna ükski ei kata e_1 . Järgmiseks proovitakse c' täpsustusi, mille hulgast $tütar(X, Y) \leftarrow naine(X), vanem(Y, X)$ osutub sobivaks hüpoteesiks, mis on täielik ja mittevasturääkiv. \triangleleft

4. Rakendusi

Tänu relatsioonilisele loogikaformalismile ILP-süsteemid said edukalt rakendatud erinevates valdkondades, mille ülesanded ei ole väljendatavad atribuut-väärtus keeltes (nt. valkude ruumilise struktuuri ennustamine, ravimite loomine jt.). Samuti kasutatav loogilise programmeerimise keel tagas paljudele arusaadava tulemuste.

Valdkonnad nagu bioloogia, meditsiin ja geneetika (Bryant *et al.* 2001) said hea tulemuse rakendades ILP-t lähteandmete muustrite avastamisel. On ka teisi rakendusi, näiteks helistamiskeskuse log-failide analüüs teeninduskvaliteedi täiustamiseks (Zelezny *et al.* 2000), mehhaanika, keeletehnoloogia jpt. (Lavrač & Džeroski 1994; Muggleton 1999; Bratko & Muggleton 1995; Mooney & Califf 1996)

Järgnev osa tutvustab olemasolevaid ILP-süsteeme ja kahte liiki rakendusi: teadusavastamine ja teadmiste omandamine (ingl. *scientific discovery and knowledge acquisition*) ning programmeerimisassistendid (ingl. *programming assistants*).

4.1. ILP-süsteeme

On olemas erinevaid ILP realisatsioone, näiteks FOIL, GOLEM, MOBAL, LINUS, mille kirjeldusi saab raamatust (Lavrač & Džeroski 1994). Teiste süsteemide loetelusid ja lühitutvustusi saab artiklitest (Muggleton & Raedt 1994; de Mantaras & Armengol 1998).

4.2. Teadusavastamine ja teadmiste omandamine

Valkude ruumilise struktuuri ennustamine (ingl. *Protein 3D shape prediction*). See on üks raskemaid molekulaarbioloogia probleeme: antud aminohapete jääkide jada, ennustada proteiini struktuuri kolmemõõtmelises ruumis. Siiaamaani rakendatud meetodite (statistika, neurovõrgud, otsustuspuud) täpsus ei ületanud 60%. ILP-e omadused (struktuursed seosed, taustteadmiste kasutamine) on võimaldanud saavutada mõnedes testides 80% taseme. (Muggleton 1999; Turcotte, Muggleton, & Sternberg 2001)

Näide. Saadud reegel kirjeldab omadust „Rossmann fold“ (Turcotte, Muggleton, & Sternberg 2001):

```
fold('NAD(P)-binding Rossmann-fold', X) :-  
    adjacent(X, A, B, 1, e, h),  
    adjacent(X, C, D, 6, e, h),  
    coil(A, B, 1).
```

Tähendus: Ahelale *A* esimesel positsioonil järgneb heeliks *B*. Ahelale *C* kuuendal positsioonil järgneb heeliks *D*. *A* ja *B* vaheline pööre on ühe amiinohappe pikkune. ◁

Ravimite projekteerimine (ingl. *drug design*). Suurim osa farmatseutilisi uurimisi on olemasolevate ravimite täiustatud variantide leidmine. See protsess koosneb komponentide suure arvu võimalike kombinatsioonide testimisel ja nõuab palju ajalisi ja rahalisi ressursse (ühe ravimi väljatöötamise keskmine hind on 300 miljonit USD). ILP-e kasutamine andis paremat tulemust võrreldes traditsiooniliste statistiliste meetoditega. Ühendite keemiliste omaduste ja meditsiinilise mõju automaatne avastamine ja testimine tähendab märkimisväärset kasumit selle ala uuringutes.

Keeletehnoloogia (ingl. *Natural Language Processing*). Keeletehnoloogia on alati kasutanud masinõppe ja loogilise programmeerimise meetodeid. Ka selles valdkonnas ILP ületab olemasolevaid lähenemisi, nt. inimkeele küsimuste andmebaaside päringuteks teisendamine, keelte morfoloogia. (Muggleton 1999)

Näide inglise keele mineviku õppimisest (Mooney & Califf 1996). On antud predikaat, mis jagab listi kaheks mittetühjaks listiks:

```
split([X, Y | Z], [X], [Y | Z]).
split([X | Y], [X | W], Z) :- split(Y, W, Z).
```

250 näidete põhjal ILP-süsteem FOIDL andis järgmise tulemuse:

```
past(A, B) :- split(A, C, [e,p] ), split(B, C, [p,t] ), !.
past(A, B) :- split(A, C, [y] ), split(B, C, [i,e,d]),
               split(A, D, [r,y] ), !.
past(A, B) :- split(A, C, [y] ), split(B, C, [i,e,d]),
               split(A, D, [l,y] ), !.
past(A, B) :- split(B, A, [m,e,d]), split(A, C, [m] ),
               split(A, [s], D ), !.
past(A, B) :- split(B, A, [r,e,d]), split(A, C, [u,r] ), !.
past(A, B) :- split(B, A, [d] ), split(A, C, [e] ), !.
past(A, B) :- split(B, A, [e,d] ), !.
```

Lõplike elementide võrgud (ingl. *Finite element meshes*). ILP osutub kasulikuks ka selle mehhaanikameetodi rakendamisel projekteerimissüsteemides (CAD) materjalide ja detailide deformatsiooni ja koormuse uurimiseks. (Bratko & Muggleton 1995)

4.3. Programmeerimisassistendid

Induktiivne meetod võib anda vahendeid loogiliste programmide loomiseks ja testimiseks. ILP-e põhirakenduseks on programmide süntees näidete ja osaliste spetsifikatsioonide alusel. Uuritakse ka programmide analüüsi, testimise ja verifitseerimise, programmide ja andmebaaside omaduste tuletuse, *reverse engineering*'u võimalusi. (Muggleton & Raedt 1994)

Loogiliste programmide loomine. ILP-t saab kasutada loogiliste programmide genereerimiseks kasutades ainult tulevase programmi spetsifikatsiooni (süntees) või programmi ebaefektiivset varianti (teisendamine/transformatsioon).

Andmebaaside ja programmide omaduste tuletamine. Andmebaaside reeglipärasuste leidmist saab kasutada andmekaevandamiseks ja baasi terviklikkuse kontrollimiseks (ingl. *integrity constraints*). Programmide regulaarsuse saab tõlgendada spetsifikatsioonina ja kontrollida vastavalt programmi korrektsust.

Programmide testimine. Tavaline silumine tegeleb teadaolevate vigadega. Testimine ja verifitseerimine katsuvad leida potentsiaalseid vigu, genereerides testülesannete komplekte.

5. Kokkuvõte

Huvi induktiivse loogilise programmeerimise vastu on põhjendatud arenatud teoreetilise baasi, rakendamise efektiivsuse ja lausega. See meetod pärib masin- ja mõistete õppelt oma eesmärgi: relatsioonide tuvastamine ja uute teadmiste loomine näidete ja taustteadmiste baasil. Loogilise programmeerimise kasutamine võimaldab vältida teiste meetodite probleeme ja piiranguid, ja samas esitada tulemusi inimestel kergelt arusaadavas vormis.

Käesolev ülevaade tutvustas mõistete õppimise põhiideid, lause- ja relatsioonilisi õppijaid, ILP-e aluseid, hüpoteesruumi otsimise strateegiaid ja rakendusi suures valdkondade hulgas.

Peale mainitud on ka olulisteks arengusuundadeks teoreemide tõestus ja arvuliste andmete töötlus. Tähtsaks uurimisvaldkonnaks on ka mittetäiuslike andmete käsitlemine, näiteks juhuslikud vead näidetes ja taustteadmistes (nn. müra), mitteküllaldased näited, mis ei võimalda tuletada täiuslikke ja mittevasturääkivaid reeglipärasusi.

Induktiivne loogiline programmeerimine areneb nii teoreetilisel kui ka rakenduslikul küljel, tal on palju olemasolevaid ja potentsiaalseid rakendusi, võimalusi saada laialt levinud kasulikuks ja võimsaks vahendiks.

Viited

- Bratko, I., and Muggleton, S. H. 1995. Applications of inductive logic programming. *Communications of the ACM* 38(11):65–70.
- Bryant, C. H.; Muggleton, S. H.; Oliver, S. G.; Kell, D. B.; Reiser, P.; and King, R. D. 2001. Combining inductive logic programming, active learning and robotics to discover the function of genes. *Linköping Electronic Articles in Computer and Information Science* 6(12).
- de Mantaras, R. L., and Armengol, E. 1998. Machine learning from examples: Inductive and lazy methods. *Data & Knowledge Engineering* 25:99–123.
- Lavrač, N., and Džeroski, S. 1994. *Inductive Logic Programming: Techniques and Applications*. New York: Ellis Horwood.

- Mooney, R. J., and Califf, M. E. 1996. Learning the past tense of english verbs using inductive logic programming. In Wermter, S.; Riloff, E.; and Scheler, G., eds., *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*. Berlin: Springer. 370–384.
- Muggleton, S., and Raedt, L. D. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming* 19(20):629–679.
- Muggleton, S. 1999. Inductive logic programming: issues, results and the challenge of learning language in logic. *Artificial Intelligence* 114(1–2):283–296.
- Tamme, T.; Tammet, T.; and Prank, R. 1997. *Loogika: mõtlemisest tõestamiseni*. Tartu Ülikooli kirjastus.
- Turcotte, M.; Muggleton, S. H.; and Sternberg, M. J. E. 2001. The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Machine learning* 1,2:81–96.
- Zelezny, F.; Miksovsky, P.; Stepankova, O.; and Zidek, J. 2000. Ilp for automated telephony. In Cussens, J., and Frisch, A., eds., *Work-in-progress Track of the 10th International Conference on Inductive Logic Programming*, volume 1, 276–286. Imperial College, London.