UPORABA PREDZNANJA V INDUKTIVNEM UČENJU

INFORMATICA 4/91

Keywords: machine learning, inductive learning, background knowledge, inductive logic programmnig

Nada Lavrač
Istitut Jožef Stefan

Večina uveljavljenih metod za avtomatsko zajemanje znanja temelji na avtomatskem učenju iz primerov in uporablja atributni jezik za predstavitev primerov in naučenih konceptov. Atributni jezik je zelo omejen, saj ne omogoča opisovanja kompleksnih strukturiranih objektov ter relacij med objekti. V zadnjem času gre vse več pozornosti sistemom za avtomatsko učenje, ki uporabljajo izraznejše jezike prvega reda za predstavitev naučenih konceptov. Ti jeziki omogočajo uporabo relacij, ki izražajo predznanje o učnih primerih in domeni sami. Induktivno avtomatsko učenje relacij imenujemo tudi induktivno logično programiranje. V članku opišemo različne načine uporabe predznanja v atributnem učenju in v učenju relacij. Opišemo tudi uporabo predznanja v sistemu induktivnega logičnega programiranja LINUS. Na kratko predstavimo dve aplikaciji tega sistema.

THE USE OF BACKGROUND KNOWLEDGE IN INDUCTIVE CONCEPT LEARNING

Inductive concept learning is most frequently used for automatic knowledge acquisition. Most of the successful inductive learning methods use a propositional attribute-value language for the representation of training examples and concepts. This language is limited and does not allow for representing complex structured objects and relations among objects. The goal of a new research area named Inductive Logic Programming (ILP) is to develop systems which can learn relational descriptions of concepts in a first-order language of logic programs. A first-order language allows for describing relations which express experts' background knowledge about training examples and the domain as a whole. The paper describes the ways of using background knowledge in inductive concept learning, both in learning attribute and relational descriptions. Furthermore, the ways of using background knowledge in the system LINUS is described. LINUS is an integrated inductive learning environment which can be used for learning attribute descriptions using background knowledge, as well as for inductive logic programming. Two applications of LINUS are briefly described: learning rules for early diagnosis of rheumatic diseases and learning illegal chess endgame positions.

1. Uvod

V svojem razvoju je umetna inteligenca prišla do stopnje, ko so njene metode, tehnike in orodja postale splošno uporabne v raznovrstnih računalniških aplikacijah. Med njimi so najbolj znani, najuspešnejši in zato tudi komercialno najbolj zanimivi *ekspertni sistemi* (Jackson 1990), ki v izvrševanju nalog na ozkem problemskem področju dosegajo raven vrhunskih strokovnjakov - ekspertov. 'Inteligenca' ekspertnega sistema temelji na bazi znanja, specifični za konkretno problemsko področje. Proces izgradnje baze znanja je najtežavnejša faza v razvoju ekspertnega sistema in je znana pod imenom 'Feigenbaumovo ozko grlo'. Dolgotrajnost in težavnost konstrukcije baz znanja sta bila povod za številne

raziskave, katerih cilj je olajšati in pospešiti proces zajemanja znanja za ekspertne sisteme.

Večina uveljavljenih metod za avtomatsko zajemanje znanja temelji na *avtomatskem učenju iz primerov.* V tem pristopu iz danih primerov ekspertnih odločitev z induktivnim sklepanjem izpeljemo splošna pravila. Tudi sam ekspert včasih najlaže posreduje svoje znanje s pomočjo dobro izbranih primerov.

Večina praktično uporabnih sistemov za avtomatsko učenje, kot na primer ASISTENT (Cestnik, Kononenko in Bratko 1987), uporablja *atributni jezik* za predstavitev primerov in naučenih konceptov (opisov razredov). Pri teh sistemih so primeri predstavljeni z n-tericami vrednosti atributov ter ustreznim razredom, medtem ko so naučeni koncepti opisani z odločitvenimi drevesi ali z if-then pravili. Vse informacije o domeni so vsebovane v učnih primerih. Atributni jezik je zelo omejen, saj ne omogoča opisovanja kompleksnih strukturiranih objektov ter relacij med objekti. Zato obstaja vrsta nalog, ki jih z atributnim učenjem iz primerov ni mogoče rešiti.

S problemom omejenosti opisnega jezika se lahko spopademo na več načinov:

- Metode za *konstruktivno učenje* omogočajo, da sistem avtomatsko konstruira nove, sestavljene izraze. V interakciji s sistemom ekspert izbere in poimenuje tiste od predlaganih avtomatsko konstruiranih izrazov, ki bi jih bilo smiselno uporabiti v učenju opisov konceptov.

- Nove izraze lahko na osnovi svojega znanja o problemski domeni predlaga ekspert. Ekspert lahko torej poda svoje *predznanje* (angl. background knowledge) kot funkcije vrednosti obstoječih atributov lahko pa poda tudi smiselne relacije med objekti problemskega prostora.

- Moč atributnega jezika lahko povečamo tako, da izberemo izraznejši logični jezik prvega reda za predstavitev naučenih konceptov.

V zadnjem času gre vse več pozornosti sistemom za avtomatsko učenje, ki uporabljajo izraznejše jezike prvega reda za predstavitev naučenih konceptov. Ti jeziki omogočajo uporabo *relacij*, ki izražajo *predznanje* o učnih primerih in domeni sami. Medtem ko je v atributnih jezikih naloga učenje opisa razredov iz učnih primerov, je v relacijskih jezikih naloga učenje *logičnih definicij relacij* iz primerov ter predznanja. V relacijskem učenju so učni primeri opredeljena dejstva oz. n-terice vrednosti argumentov relacije, katere definicije se hočemo naučiti. Po analogiji z razredom je vsak učni primer označen z $\oplus$, če pripada, oz. z $\ominus$, če ne pripada dani relaciji. Predznanje je tudi izraženo v obliki relacij, ki so podane s pripadajočimi n-tericami, ali pa z logičnimi definicijami (programi) v programskem jeziku *prolog.* Ker so naučene definicije relacij tudi izražene kot *logični programi*, induktivno avtomatsko učenje relacij imenujemo tudi *induktivno logično programiranje* (Muggleton 1991).

V članku opišemo različne načine uporabe predznanja v atributnem učenju in v učenju relacij. V drugem razdelku definiramo problem induktivnega učenja konceptov. S primeri predstavimo naloge atributnega učenja, atributnega učenja z upoštevanjem predznanja ter učenja relacij v induktivnem logičnem programiranju. V tretjem razdelku obravnavamo uporabo predznanja v sistemu induktivnega logičnega programiranja LINUS. Na kratko opišemo algoritem in dve aplikaciji sistema LINUS: učenje medicinskih diagnostičnih pravil v revmatologiji ter učenje relacij v šahovski končnici.

2. Induktivno učenje konceptov

Koncept C lahko definiramo kot podmnožico prostora objektov. Induktivno učenje konceptov pomeni metodo učenja iz primerov, ki omogoča razpoznavanje objektov v C.

Učni primer za učenje koncepta C je par:

$$\langle Objekt, Razred \rangle$$

kjer je *Objekt* izbrani opis objekta, *Razred* pa $\oplus$ ali $\ominus$, kar označuje, da *Objekt* pripada ali ne pripada konceptu C. Učni primer je *pozitiven*, če *Objekt* pripada konceptu C ($Razred = \oplus$) in je *negativen* primer ali *protiprimer*, če ne pripada C ($Razred = \ominus$). Namesto da bi dani objekt klasificirali v enega od dveh razredov $\oplus$ and $\ominus$, ga lahko klasificiramo v enega od več medsebojno izključujočih se razredov C_c, $c = 1, \ldots, N$.

Problem induktivnega učenja konceptov lahko formalno definiramo takole (de Raedt 1991):

	A_1	A_2	A_3	...	*Razred*
$primer_1$	v_{11}	v_{12}	v_{13}	...	C_1
$primer_2$	v_{21}	v_{22}	v_{23}	...	C_2
$primer_3$	...	...	...	...	...

Tabela 1: Množica primerov, zapisana v obliki tabele.

Podani so:

- jezik za opis primerov L_E
- množica pozitivnih primerov P
- množica negativnih primerov N
- jezik za opis konceptov L_C
- relacija pokritosti med L_C in L_E
- kriterij kvalitete definiran na nizih iz L_C

Poišči:

- opis koncepta C iz L_C, ki zadošča kriteriju kvalitete

Kriterij kvalitete lahko zahteva, da je zgrajeni opis koncepta konsistenten in popoln glede na dano množico primerov. Opis koncepta je *konsistenten*, če ne pokriva nobenega negativnega primera. Opis je *popoln*, če pokriva vse pozitivne primere. Zahtevo po konsistenstnosti in popolnosti moramo omiliti, ko se učimo iz šumnih in nepopolnih podatkov. Tako lahko preprečimo, da bi sistem zgradil preveč specifične opise konceptov.

Glede na izbor jezika za opis konceptov in primerov ločujemo dve glavni področji v induktivnem učenju: *atributno učenje* in *učenje relacij.*

2.1 Induktivno učenje atributnih opisov konceptov

V *atributnem učenju* so primeri predstavljeni z n-tericami vrednosti atributov ter z ustreznim razredom. Objekt lahko pripada enemu od N medsebojno izključujočih se razredov C_c. Množico primerov $\langle Objekt, Razred \rangle$ lahko predstavimo s tabelo 1.

Naloga atributnega učenja je poiskati klasifikacijsko *pravilo (hipotezo, opis koncepta)* kot funkcijo vrednosti atributov, ki ga lahko uporabimo za napovedovanje razreda neznanega objekta. Naučeni opis koncepta ima lahko obliko if-then pravila:

A1	*A2*	*A3*	*A4*	*A5*	*Razred*
da	*balon*	*da*	*kvadrat*	*kvadrat*	*P*
da	*zastava*	*da*	*osmerokotnik*	*osmerokotnik*	*P*
da	*meč*	*da*	*krog*	*osmerokotnik*	*N*
da	*meč*	*ne*	*kvadrat*	*osmerokotnik*	*N*
ne	*meč*	*ne*	*osmerokotnik*	*krog*	*N*
ne	*zastava*	*ne*	*krog*	*osmerokotnik*	*N*

Tabela 2: Primeri prijaznih (P) in neprijaznih (N) robotov.

$$\begin{array}{ll} if & (A_i = v_i) \wedge (A_j = v_j) \wedge \ldots \\ then & Razred = C_c \end{array}$$

ki je konsistentno in popolno glede na dano množico primerov.

Naučena if-then pravila opisujejo posamezne razrede. Sistemi pa se lahko naučijo tudi opisa vseh razredov v obliki *odločitvenega drevesa*, katerega listi označujejo posamezne razrede.

Za ilustracijo učenja odločitvenih dreves izberimo primer iz sveta prijaznih in neprijaznih robotov (Wnek, Sarma, Wahab in Michalski 1990, Sutlič 1991). Opis robota je podan z naslednjimi atributi in njihovimi vrednostmi:

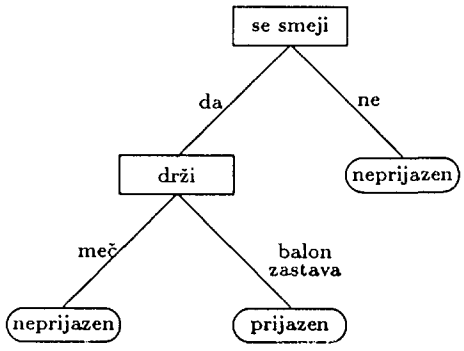
A1 - se smeji:	*da, ne*
A2 - drži:	*balon, zastava, meč*
A3 - ima kravato:	*da, ne*
A4 - oblika glave:	*krog, kvadrat, osmerokotnik*
A5 - oblika telesa:	*krog, kvadrat, osmerokotnik*

V tabeli 2 je podanih šest učnih primerov. Za vsako 5-terico vrednosti atributov je podan razred, ki določa, ali je robot prijazen ali ne.

Naloga učenja je poiskati pravilo, ki opredeli, ali je robot prijazen ali ne. Za učenje odločitvenih dreves lahko uporabimo sistem ASISTENT (Cestnik, Kononenko in Bratko 1987). Naučeno odločitveno drevo je prikazano na sliki 1.

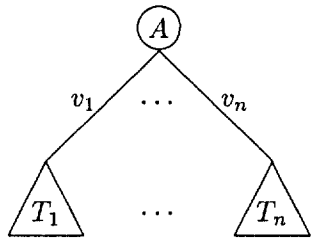
Opišimo Quinlanov algoritem ID3 (Quinlan 1986), ki je osnova algoritma ASISTENT. Naj bo S množica učnih primerov, $C_1, \ldots C_N$ pa odločitveni razredi.

- *če* vsi primeri iz S spadajo v isti razred C_c
- *potem* označi list drevesa s C_c
- *sicer*



Slika 1: Odločitveno drevo, zgrajeno s sistemom ASISTENT iz šestih primerov robotov.

- izberi 'najinformativnejši' atribut A z vrednostmi $v_1, \ldots, v_n$
- razdeli učno množico S v $S_1, \ldots, S_n$ glede na vrednosti $v_1, \ldots, v_n$
- rekurzivno zgradi poddrevesa $T_1, \ldots, T_n$ za $S_1, \ldots, S_n$
- zgradi odločitveno drevo T, kot je prikazano na sliki 2.



Slika 2: Odločitveno drevo, ki ga zgradi ID3.

Preiskovalna hevristika algoritma je določena z *informativnostjo* atributa, ki se izračuna na osnovi *entropije* učne množice $E(S)$, ki izraža količino informacije, potrebne za klasifikacijo enega objekta:

$$E(S) = -\sum_{c=1}^{N} p_c \cdot log_2 p_c$$

kjer je p_c apriorna verjetnost razreda C_c (izračunana z relativno frekvenco razreda C_c v S).

Entropija učne množice po razbitju po vrednostih $v_1, \ldots, v_n$ atributa A je:

$$E(S/A) = -\sum_{v=1}^{n} p_v \cdot \sum_{c=1}^{N} \frac{p_{vc}}{p_v} \cdot log_2 \frac{p_{vc}}{p_v}$$

kjer je p_v apriorna verjetnost, da ima A vrednost v_v, p_{vc} pa je apriorna verjetnost, da ima primer, ki spada v razred C_c, vrednost v_v atributa A.

Informativnost atributa I(A) je mera, ki minimizira število testov, potrebnih za klasifikacijo novega objekta: $I(A) = E(S) - E(S/A)$. *Najinformativnejši atribut* je tisti, pri katerem doseže informativnost maksimum $max\ I(A)$, kar je ekvivalentno minimizaciji entropije $min\ E(S/A)$.

V sistemu ASISTENT je bil algoritem ID3 izboljšan na več načinov. Omogoča obravnavo manjkajočih podatkov, obravnavo numeričnih (zveznih) vrednosti atributov, obravnavo NULL listov, v katere ne spada noben učni primer, obravnavo SEARCH listov, v katere spadajo primeri iz več različnih razredov, binarizacijo atributov, ter obravnavo netočnih podatkov, tako da vpelje rezanje dreves; rezanje omogoča izgradnjo manjših drevesa, ki so lažje razumljiva in imajo večjo klasifikacijsko točnost.

Drugi primeri sistemov, ki se učijo opisov konceptov v obliki odločitvenih dreves so npr. C4 (Quinlan et al. 1986) in CART (Breiman et al. 1984). Primeri sistemov, ki se učijo if-then pravil pa so AQ15 (Michalski et al. 1986), njegov predhodnik NEWGEM (Mozetič 1985), CN2 (Clark in Niblett 1989) ter GINESYS (Gams 1989).

2.2 Uporaba predznanja v atributnem učenju

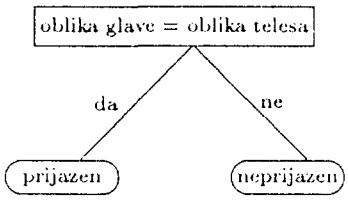
Kot smo omenili v uvodu, lahko sistem v naučenih pravilih upošteva nove izraze, ki mu jih na osnovi svojega znanja o problemski domeni predlaga ekspert.

Pri odločanju in pri razlagi odločitev eksperti namreč uporabljajo svoje znanje o problemu. To znanje je mogoče izraziti kot funkcije vrednosti atributov (s katerimi so opisani učni primeri) ali pa z relacijami med atributi ali objekti problemskega področja. To ekspertovo znanje, imenovano *predznanje* (angl. background knowledge), lahko sistem koristno uporabi pri učenju kompleksnejših relacij. Predznanje lahko sistem uporabi kot pomožno znanje v atributnem učenju. V učenju relacij pa postane relacijsko predznanje bistvenega pomena za učenje.

V atributnem učenju sistem upošteva predznanje tako, da za vsak nov (sestavljen) izraz, podan v predznanju, vpelje nov atribut. Če poda ekspert svoje znanje v funkcijski obliki, se za posamezni

A1	*A2*	*A3*	*A4*	*A5*	*A6*	*Razred*
da	*balon*	*da*	*kvadrat*	*kvadrat*	*da*	*P*
da	*zastava*	*da*	*osmerokotnik*	*osmerokotnik*	*da*	*P*
da	*meč*	*da*	*krog*	*osmerokotnik*	*ne*	*N*
da	*meč*	*ne*	*kvadrat*	*osmerokotnik*	*ne*	*N*
ne	*meč*	*ne*	*osmerokotnik*	*krog*	*ne*	*N*
ne	*zastava*	*ne*	*krog*	*osmerokotnik*	*ne*	*N*

Tabela 3: Primeri prijaznih (P) in neprijaznih (N) robotov. V stolpcu $A6$ so podane vrednosti novega atributa *oblika glave = oblika telesa.*



Slika 3: Odločitveno drevo, zgrajeno s sistemom ASISTENT iz šestih primerov robotov z uporabo predznanja.

učni primer vrednost novega atributa izračuna kot funkcija vrednosti osnovnih atributov. Če pa je predznanje podano v obliki relacij, ima novo vpeljani atribut vrednost *true*, v kolikor vrednosti ustreznih atributov primera ustrezajo relaciji, sicer ima vrednost *false*.

V primeru iz sveta robotov vpeljimo en sestavljen izraz, s katerim testiramo relacijo enakosti: *oblika glave = oblika telesa.* S tem smo vpeljali nov atribut $A6$ z naborom vrednosti *da* (*true/resnično*) in *ne* (*false/ni resnično*). Množico učnih primerov, dopolnjeno z vrednostmi novega atributa $A6$, prikažemo v tabeli 3.

V gradnji odločitvenega drevesa iz primerov podanih v tabeli 3 se novi atribut $A6$ izkaže kot najbolj informativen, zato se pojavi v korenu drevesa zgrajenega s sistemom ASISTENT. Ker vrednosti atributa $A6$ v celoti ločita med prijaznimi in neprijaznimi roboti, je to tudi edini atribut v zgrajenem drevesu. Drevo je podano na sliki 3.

Iz primera je razvidno, da je lahko atribut, ki je zgrajen na osnovi predznanja, bolj informativen kot so osnovni atributi. Z uporabo predznanja lahko zgradimo kompaktnejše opise konceptov, ki imajo lahko večjo klasifikacijsko točnost.

2.3 Induktivno logično programiranje

V *učenju relacij* učni primeri izražajo relacije med elementi problemskega prostora (objekti), podano pa je tudi dodatno problemsko znanje (predznanje) o učnih primerih in domeni sami. Medtem ko je v atributnih jezikih naloga učenje opisov konceptov (razredov) iz učnih primerov, je v relacijskih jezikih naloga učenje logičnih definicij relacij iz primerov in predznanja.

Nalogo učenja relacij v jeziku prvega reda lahko formuliramo takole:

Podani so:

- jezik za opis primerov L_E
- množica pozitivnih in negativnih primerov (dejstev $\oplus$ in $\ominus$) neznane relacije p
- jezik za opis konceptov L_C, ki določa sintaktične omejitve za definicijo predikata p
- jezik za opis predznanja L_B
- predznanje - množica definicij pomožnih predikatov in funkcij q_i, ki jih lahko uporabimo v definiciji predikata p
- relacija pokritosti med L_C in L_E glede na L_B
- kriterij kvalitete definiran na definicijah predikatov iz L_C

Poišči:

- definicijo ciljnega predikata p kot množico stavkov iz L_C, ki zadošča kriteriju kvalitete

Množica pozitivnih ($\oplus$) in negativnih ($\ominus$) primerov je podana z opredeljenimi dejstvi (angl. ground facts), kar pomeni, da so podane vrednosti argumentov relacije, t.j., da v argumentih relacije ne nastopajo spremenljivke. Predznanje je podano z funkcijami in relacijami, ki vsebujejo informacijo o argumentih ciljne relacije p. Naloga učenja je poiskati definicijo relacije p v odvisnosti od relacij iz predznanja, ki bo ustrezala kriteriju kvalitete, npr. ki bo popolna in konsistentna glede na dana dejstva. Definicija je popolna, če razlaga (pokriva) vsa pozitivna dejstva, konsistentna pa, če ne pokriva nobenih negativnih dejstev.

Naučena definicija predikata p sestoji iz množice stavkov oblike

$$p(X_1, X_2, \ldots, X_k) \leftarrow L_1, \ldots, L_n$$

kjer je telo stavka konjunkcija literalov $L_i = q_i(Y_1, Y_2, \ldots, Y_m)$.

Ilustrirajmo zgornjo definicijo na preprostem problemu učenja družinskih relacij (Džeroski 1991). Naj bo ciljna relacija *daughter* in naj bo predznanje sestavljeno iz relacij *female* in *parent*. Za relacijo *daughter* so podani štirje učni primeri: dva pozitivna in dva negativna.

$$\begin{array}{ll} daughter(maja, ana). & \oplus \\ daughter(eva, tom). & \oplus \\ daughter(tom, ana). & \ominus \\ daughter(eva, ana). & \ominus \end{array}$$

Podano je naslednje predznanje:

$$\begin{array}{ll} parent(ana, maja). & female(ana). \\ parent(ana, tom). & female(maja). \\ parent(tom, eva). & female(eva). \\ parent(tom, ivo). & \end{array}$$

Iz učnih primerov in predznanja se je mogoče naučiti definicije predikata *daughter*:

$$daughter(X, Y) \leftarrow female(X), parent(Y, X).$$

Naučene definicije relacij so podane s programskimi stavki v sintaksi programskega jezika prolog. Definicijo lahko v kompleksnejših primerih sestavlja več programskih stavkov (angl. program clauses). Množico stavkov z istim predikatom v glavi stavka imenujemo *definicija predikata* (angl. predicate definition). Definicija predikata tvori prologovo proceduro oziroma prologov logični program.

Sistemi, ki se učijo logičnih programov, spadajo med sisteme *induktivnega logičnega programiranja (ILP)*. Induktivno logično programiranje je mlado raziskovalno področje. V svetu so bile prve raziskave s tega področja opravljene leta 1981, ko je bil razvit sistem MIS (Model Inference System, Shapiro 1981), ki se zna učiti logičnih programov, temu pa so sledili sistemi MARVIN (Sammut in Banerji 1986), CIGOL (Muggleton in Buntine 1988), GOLEM (Muggleton in Feng 1990) in FOIL (Quinlan 1990). Področje je dobilo svoje ime šele leta 1990, prva delavnica (First International Workshop on Inductive Logic Programming) pa je bila marca 1991.

Področje ILP je razvito tudi v Sloveniji. Razvita sta bila ILP sistema LINUS (Lavrač 1990) in mFOIL (Džeroski 1991), ki sta bila uporabljena v več domenah, znanih iz literature o avtomatskem učenju (Lavrač, Džeroski in Grobelnik 1991), za učenje medicinskih diagnostičnih pravil na področju revmatologije (Lavrač, Džeroski, Pirnat in Križman 1991), za učenje načrtovanja mrež končnih elementov (Džeroski in Dolšak 1991) ter za učenje iz šumnih podatkov (Lavrač in Džeroski 1991, Džeroski in Lavrač 1991a). Teoretično primerjavo sistemov LINUS in FOIL smo podali z grafi specializacije (Džeroski in Lavrač 1991b). ILP pristop je bil uporabljen tudi za učenje kvalitativnih modelov (Bratko, Muggleton in Varšek 1991).

3. Uporaba predznanja v integriranem okolju za avtomatsko učenje LINUS

Sistem induktivnega logičnega programiranja LINUS (Lavrač 1990, Lavrač, Džeroski in Grobelnik 1991) temelji na ideji, da je možno transformirati problem učenja relacij v problem atributnega učenja z uporabo predznanja. Na ta način lahko uporabimo obstoječe atributne učne algoritme, npr. sistem ASISTENT, ki so že dosegli veliko stopnjo uporabnosti in znajo obravnavati nepopolne in šumne podatke.

Sistem LINUS omogoča uporabo dodatnega problemskega znanja v učenju opisov konceptov (opisov posameznih razredov objektov) ali učenju logičnih definicij relacij med objekti problemskega prostora. V sistemu LINUS opišemo učne primere in naučena pravila v jeziku *deduktivnih hierarhičnih baz podatkov* (DHBP), t.j. v logičnem jeziku prega reda, ki je izraznejši od atributnih jezikov 0-tega reda, ki jih uporablja večina uveljavljenih sistemov za induktivno učenje. Izbor tega preprostega jezika prvega reda omogoča sistemu LINUS, da z uporabo atributnih učnih algoritmov učinkovito zgradi logične opise relacij.

Stavki DHBP so omejena oblika programskih stavkov; prepovedani so rekurzivni stavki, spremenljivkam pa so pridružene ustrezne končne zaloge vrednosti (tipi). Dodatna omejitev LINUSu onemogoča uvajanje novih spremenljivk v telo stavka. Predznanje je pri LINUSu izraženo v formalizmu *deduktivnih baz podatkov* (DBP), ki dovoljuje rekurzivne definicije predikatov.

Sistem LINUS vključuje tri atributne učne algoritme ASISTENT, NEWGEM in CN2. ASISTENT je predstavnik TDIDT (Top-Down Induc-

tion of Decision Trees) družine učnih algoritmov (Quinlan 1986). NEWGEM je član družine AQ algoritmov (npr. Michalski et al. 1986). CN2 združuje najboljše lastnosti ID3 in AQ algoritmov s tem, da dovoli uporabo statističnih metod, podobnih rezanju drevesa, in da generira if-then pravila.

3.1 Opis algoritma LINUS

Program, ki omogoča vključitev sistema LINUS v okolje logičnega programiranja imenujemo DHBP vmesnik. Program je implementiran v prologu. Vmesnik pretvori pozitivna dejstva in negativna dejstva (negativna dejstva so podana ali pa jih generira DHBP vmesnik) iz DHBP oblike v obliko n-teric vrednosti atributov, atributni učni algoritem se nauči opisa koncepta v atributnem jeziku, DHBP vmesnik pa pretvori te opise v obliko DHBP stavkov. Najpomembnejši del DHBP vmesnika je generiranje novih atributov za učenje. Z upoštevanjem zaloge vrednosti tipov argumentov ciljne relacije namreč preveri, katere so možne aplikacije pomožnih predikatov in funkcij iz predznanja ter le-te upošteva kot dodatne atribute, ki jih bo uporabil v učenju. Vsak učni primer dopolni z vrednostmi teh dodatnih atributov, to je z vrednostmi *da* (*true*) ali *ne* (*false*) za pomožne predikate, oziroma z vrednostmi izhodnih argumentov za pomožne funkcije. DHBP vmesnik torej implementira osnovno idejo uporabe predznanja v atributnem učenju, ki smo jo razložili v razdelku 2.2.

Učenje z LINUSom poteka v naslednjih korakih:

- pripravi množico pozitivnih in negativnih dejstev,
- pretvori dejstva iz DHBP oblike v n-terice vrednosti atributov,
- nauči se opisa koncepta z uporabo atributnega učnega algoritma,
- pretvori naučena pravila iz oblike if-then pravil v obliko DHBP stavkov.

Algoritem opišemo s preprostim primerom. Denimo, da je relacija $r(a,b,c)$ podana z naslednjimi tremi dejstvi, označenimi z $\oplus$:

$$r(a1,b1,b1).$$
$$r(a1,b2,b2).$$
$$r(a2,b1,b2).$$

Podana je tudi informacija o *imenih* in *tipih* spremenljivk. Prvi argument relacije r ima ime a, njegov tip je *tip_a*, drugi in tretji argument b in c pa sta oba istega tipa *tip_b*. Vsakemu tipu je pridružena informacija o zalogi vrednosti spremenljivk: $\{a1,a2\}$ za *tip_a* in $\{b1,b2\}$ za *tip_b*.

Dodatno znanje o problemu je podano v obliki definicij pomožnih predikatov in funkcij. Uporabimo npr. binarni pomožni predikat p, definiran z naslednjimi dejstvi:

$$p(a1,b1).$$
$$p(a1,b2).$$
$$p(a2,b2).$$

V splošnem so lahko pomožni predikati in funkcije definirani z množico rekurzivnih tipiziranih programskih stavkov (v formalizmu DBP). Argumenta predikata p sta tipa *tip_a* in *tip_b*. LINUS uporablja tudi vgrajeni pomožni predikat *enakost* (=/2).

Množica *negativnih dejstev* je lahko vnaprej podana, lahko pa jo sistem generira iz pozitivnih dejstev. V načinu, ki temelji na *predpostavki zaprtega sveta (cwa)*, LINUS generira vse možne kombinacije vrednosti argumentov ciljne relacije. Vsa generirana dejstva, razen podanih pozitivnih dejstev, LINUS uporabi kot negativna dejstva in jih označi z $\ominus$:

$$r(a1,b1,b2).$$
$$r(a1,b2,b1).$$
$$r(a2,b1,b1).$$
$$r(a2,b2,b1).$$
$$r(a2,b2,b2).$$

V drugem koraku algoritem preveri vse možne aplikacije pomožnih predikatov in funkcij glede na tipe spremenljivk. V našem primeru so to $p(a,b)$ in $p(a,c)$, enakost pa lahko uporabi nad argumentoma istega tipa, torej b=c. Iz treh pozitivnih primerov relacije r LINUS generira naslednje šesterice oblike:

$$\langle a, \quad b, \quad c, \quad (b=c), \quad p(a,b), \quad p(a,c) \rangle$$

$$\langle a1, \quad b1, \quad b1, \quad true, \quad true, \quad true \rangle$$
$$\langle a1, \quad b2, \quad b2, \quad true, \quad true, \quad true \rangle$$
$$\langle a2, \quad b1, \quad b2, \quad false, \quad false, \quad true \rangle$$

Na enak način LINUS generira šesterice tudi iz negativnih dejstev.

V tretjem koraku generirane n-terice pozitivnih in negativnih primerov uporabi za učenje s sistemi ASISTENT, NEWGEM ali CN2. Generirani opis koncepta v obliki odločitvenega drevesa, VL1 pravila ali if-then pravil, se prepiše v prologovo obliko if-then pravil.

V četrtem koraku algoritem pretvori if-then pravila iz prologove oblike v obliko DHBP stavkov. V našem preprostem primeru dobimo naslednji opis:

$$r(A,B,C) \leftarrow A = a1, B = C.$$
$$r(A,B,C) \leftarrow C = b2,\ not\ p(A,B).$$

V generiranih pravilih nastopata poleg literalov *A=a1* in *C=b2*, ki jih je mogoče dobiti tudi z atributnim učenjem, tudi literala *B=C* in *not p(A,B)*, ki sta dobljena z uporabo relacijskega predznanja.

Z izborom atributov in predznanja, ki naj se uporabi za učenje, lahko torej sistem LINUS uporabimo na tri načine:

- kot atributni učni algoritem, če sistemu povemo, kateri od argumentov ciljne relacije naj upošteva kot odločitveni razred ter če v celoti izključimo predznanje,
- kot algoritem za atributno učenje, ki zna uporabljati predznanje, če sistemu povemo, kateri od argumentov ciljne relacije naj upošteva kot odločitveni razred ter če upoštevamo nove atribute, dobljene z uporabo predznanja,
- kot algoritem za relacijsko učenje, če obravnavamo samo pozitivne in negativne primere relacije $\oplus$ in $\ominus$ ter se učimo samo na osnovi predznanja.

3.2 Uporaba predznanja v atributnem učenju opisov posameznih razredov v revmatologiji

LINUS predstavlja razširitev atributnih algoritmov z možnostjo uporabe predznanja in ga lahko uporabimo za atributno učenje. Takemu načinu učenja pravimo *učenje razredov* (class learning mode, Lavrač, Džeroski in Grobelnik 1991). Sistem smo v tem načinu uporabili za učenje diagnostičnih pravil v revmatologiji. Eksperimenti so podrobno opisani v (Lavrač, Džeroski, Pirnat in Križman 1991). Na voljo smo imeli podatke o 462 pacientih Univerzitetnega kliničnega centra v Ljubljani. Pacienti so bili razvrščeni v 200 diagnoz, ki so bile grupirane v 8 razredov diagnoz: *A1* degenerativne bolezni hrbtenice (158 primerov), *A2* degenerativne bolezni sklepov (128), *B1* vnetne bolezni hrbtenice (16), *B234* druge vnetne bolezni (29), *C* zunajsklepni revmatizem (21), *D* metabolni revmatizem (24), *E* neznačilne revmatske težave (32) ter *F* nerevmatske bolezni (54). V eksperimentu smo upoštevali le anamnestične podatke, opisane z vrednostmi naslednjih 16 atributov: spol, starost, družinska anamneza, sedanje težave, sedanja bolezen, bolečine v sklepih, število bolečih sklepov, število oteklih sklepov, bolečine v hrbtenici, ostale bolečine, jutranja okorelost, koža, sluznice, oči, druge težave in terapija.

LINUS smo uporabili za učenje opisov posameznih razredov diagnoz. 70 % podatkov smo uporabili za učenje, 30 % pa za testiranje klasifikacijske točnosti naučenih pravil. Eksperiment smo ponovili desetkrat, z naključno porazdelitvijo v učne in testne primere. V učenju smo upoštevali predznanje o značilnih skupinah simptomov. Identificirali smo 6 značilnih skupin simptomov, npr. karakteristične kombinacije simptomov prve skupine so naslednje:

Bolečine v sklepih	Jutranja okorelost
bp	$\leq$ 1 ura
artrotična	$\leq$ 1 ura
artritična	$>$ 1 ura

Karakteristične kombinacije v šesti skupini simptomov pa so podane v spodnji tabeli:

Število oteklih sklepov	Število bolečih sklepov
0	0
0	$1 \leq$ št. sklepov ≤ 30
$1 \leq$ št. sklepov ≤ 10	$0 \leq$ št. sklepov ≤ 30

Kot primer navedimo diagnostično pravilo za metabolni revmatizem:

```
Razred = metabolni_revmatizem ←
  skupina6( St_otekl_sklepov, St_bol_sklepov,
            '1=<os=<10&0=<bs=<30' ),
  Starost > 30, Starost =< 40,
  Spol = moski.

Razred = metabolni_revmatizem ←
  Bol_v_hrbtenici = bp,
```

Bol_v_sklepih = artriticne,
Spol = moski.

Razred = metabolni_revmatizem ←
Bol_v_hrbtenici = bp,
Starost =< 40,
Sedanje_tezave > 11,
Spol = moski,
skupina5(Bol_v_sklepih, Bol_v_hrbtenici,
St_bol_sklepov, Vrednost),
member(Vrednost, ['artriticne&bp&1=<bs=<30',
neznacilno]).

3.3 Uporaba predznanja v učenju relacij v šahovski končnici

Kot primer uporabe predznanja v učenju logičnih definicij relacij izberimo učenje nelegalnih pozicij v šahovski končnici beli kralj in bela trdnjava proti črnemu kralju. Različni eksperimenti v tej domeni so podrobno opisani v (Lavrač, Džeroski in Grobelnik 1991, Lavrač in Džeroski 1991, Džeroski in Lavrač 1991a).

Ciljna relacija je podana s predikatom *nelegalen(A,B,C,D,E,F)*. Argumenti označujejo pozicijo belega kralja (A, B), bele trdnjave (C, D) in črnega kralja (E, F).

Predznanje je podano z relacijami *sosedna_vrsta(X,Y)*, *soseden_stolpec(X,Y)*, *manjsa_vrsta(X,Y)*, *manjsi_stolpec(X,Y)* in *enakost* $X = Y$. Spremenljivke so tipizirane. Vpeljemo dva tipa: *vrsta* z vrednostmi $\{1,2,3,4,5,6,7,8\}$ in *stolpec* z vrednostmi $\{a,b,c,d,e,f,g,h\}$.

Za eksperimente smo imeli na voljo :

- 5 učnih množic po 100 primerov, testno množico 5000 primerov
- 5 učnih množic po 1000 primerov, 5 testnih množic po 4500 primerov

Iz množice 100 učnih primerov se je LINUS naučil naslednje definicije relacije *nelegalen*:

nelegalen(A,B,C,D,E,F) ← *C=E.*
nelegalen(A,B,C,D,E,F) ← *D=F.*
nelegalen(A,B,C,D,E,F) ← *B=F,*
soseden_stolpec(A,E),
nelegalen(A,B,C,D,E,F) ←
soseden_stolpec(A,E),
sosedna_vrsta(B,F).
nelegalen(A,B,C,D,E,F) ← *A=E,*
sosedna_vrsta(B,F).
nelegalen(A,B,C,D,E,F) ← *A=E, B=F.*

Rezultate učenja s sistemom LINUS smo primerjali z nekaterimi drugimi ILP sistemi. V petih ponovitvah eksperimenta na množicah s po 100 učnimi primeri je bila dosežena naslednja povprečna klasifikacijska točnost naučenih pravil: 98.1% LINUS z uporabo ASISTENTa, 88.4% LINUS z uporabo NEWGEMa. Na istih učnih in testnih množicah je CIGOL dosegel 77.2% točnost, FOIL pa 90.8% točnost naučenih pravil.

4. Zaključek

Članek poda pregled uporabe predznanja v induktivnem učenju konceptov. Predstavi metodo, s katero je mogoče sisteme za atributno učenje nadgraditi tako, da se z uporabo predznanja naučijo tudi relacijskih opisov konceptov. Z uporabo te metode smo zgradili integrirano okolje za avtomatsko učenje LINUS, ki uporablja obstoječe atributne učne algoritme za atributno učenje in za induktivno logično programiranje. V prispevku smo se osredotočili na možnosti uporabe predznanja za izgradnjo kompaktnejših atributnih opisov ter za učenje relacijskih opisov ter le na kratko opisali sam učni algoritem ter dve aplikaciji sistema LINUS.

Reference

Bratko, I., Muggleton, S. in Varšek, A. (1991) Learning qualitative models of dynamic systems. Eighth International Workshop on Machine Learning. Evanston, IL: Morgan Kaufmann.

Breiman, L., Friedman, J.H., Olshen, R.A. in Stone, C.J. (1984) Classification and regression trees. Belmont: Wadsworth.

Cestnik, B., Kononenko, I. in Bratko, I. (1987) ASSISTANT 86: A knowledge elicitation tool for sophisticated users. V: Bratko, I. in Lavrač, N. (ur.) Progress in machine learning. Wilmslow: Sigma Press.

Clark, P. in Niblett, T. (1989) The CN2 induction algorithm. Machine Learning 3 (4), 261-284. Kluwer Academic Publishers.

Džeroski, S. in Dolšak, B. (1991) Primerjava algoritmov za avtomatsko učenje relacij na problemu načrtovanja mrež končnih elementov. XXVI Jugoslovanska konferenca ETAN, Ohrid, Makedonija.

Džeroski, S. in Lavrač, N. (1991a) Learning relations from noisy examples: An empirical comparison of LINUS and FOIL. Eighth International Workshop on Machine Learning. Evanston, IL: Morgan Kaufmann.

Džeroski, S. in Lavrač, N. (1991b) Refinement graphs for FOIL and LINUS. V: Muggleton, S.H. (ur.) Inductive logic programming. Glasgow, UK: Academic Press (v tisku).

de Raedt, L. (1991) Interactive concept-learning. Doktorska disertacija, Department of Computer Science, Katholieke Universiteit Leuven, Belgium.

Džeroski, S. (1991) Handling noise in inductive logic programming. Magisterska naloga, Fakulteta za elektrotehniko in računalništvo, Univerza v Ljubljani.

Gams, M. (1989) New measurements highlight the importance of redundant knowledge. Fourth European Working Session on Learning, EWSL 89, 71-79. Montpellier, France: Pitman.

Jackson, P.J. (1990) Introduction to expert systems (Druga izdaja). Addison-Wesley.

Lavrač, N. (1990) Principles of knowledge acquisition in expert systems. Doktorska disertacija, Tehniška fakulteta, Univerza v Mariboru.

Lavrač, N., Džeroski, S. in Grobelnik, M. (1991) Learning nonrecursive definitions of relations with LINUS. Fifth European Working Session on Learning, EWSL 91. Porto, Portugal: Springer-Verlag.

Lavrač, N. in Džeroski, S. (1991) Inductive learning of relational descriptions from noisy examples. First Inductive Logic Programming Workshop, Viana de Castelo, Portugal.

Lavrač, N., Džeroski, S., Pirnat, V. in Križman, V. (1991) Learning rules for early diagnosis of rheumatic diseases. Third Scandinavian Conference on Artificial Intelligence, SCAI'91. Roskilde, Denmark: IOS Press.

Michalski, R.S., Mozetič, I., Hong, J. in Lavrač, N. (1986) The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. Fifth National Conference on Artificial Intelligence, AAAI-86, 1041-1045. Philadelphia, PA: Morgan Kaufmann.

Mozetič, I. (1985) NEWGEM: Program for learning from examples - Technical documentation and user's guide. Technical Report, University of Illinois at Urbana-Champaign, Department of Computer Science. Tudi: Delovno poročilo IJS-DP-4390, Institut Jožef Stefan, Ljubljana.

Muggleton, S.H. (1991) Inductive logic programming. New Generation Computing 8 (4), 295-318.

Muggleton, S.H. in Buntine, W. (1988) Machine invention of first-order predicates by inverting resolution. Fifth Int. Conference on Machine Learning, 339-352. Ann Arbor, MI: Morgan Kaufmann.

Muggleton, S.H. in Feng, C. (1990) Efficient induction of logic programs. First Conference on Algorithmic Learning Theory. Tokyo: Ohmsha.

Quinlan, J.R. (1986) Induction of decision trees. Machine Learning 1 (1), 81-106. Kluwer Academic Publishers.

Quinlan, J.R. (1990) Learning logical definitions from relations. Machine Learning 5 (3), 239-266. Kluwer Academic Publishers.

Quinlan, J.R., Compton, P.J., Horn, K.A. in Lazarus, L (1986) Inductive knowledge acquisition: A case study. Second Australian Conference on Application of Expert Systems. New South Wales Institute of Technology, Sydney, Australia.

Sammut, C. in Banerji, R.B. (1986) Learning concepts by asking questions. V: Michalski, R.S., Carbonell, J.G. in Mitchell, T.M. (ur.) Machine learning: An artificial intelligence approach (Volume II). Los Altos: Morgan Kaufmann.

Shapiro, E.Y. (1981) An algorithm that infers theories from facts. Seventh Int. Joint Conference on Artificial Intelligence, 446-451. Vancouver, BC: Morgan Kaufmann.

Sutlič, I. (1991) Integrirano okolje za avtomatsko učenje LINUS. Diplomska naloga, Fakulteta za elektrotehniko in računalništvo, Univerza v Ljubljani.

Wnek, J., Sarma, J., Wahab, A.A. in Michalski, R.S. (1990) Comparing learning paradigms via diagrammatic visualization: A case study in single concept learning using symbolic, neural net and genetic algorithm methods. Fifth Int. Symposium on Methodologies for Intelligent Systems. Knoxville, TN.