Съдържание

Глава 1

Въведение

Индуктивното машинно самообучение е област от изкуствения интелект, чиито цели са създаването на системи, които да намират компактно описание на класове на базата на примери, принадлежащи на тях. Обикновенно тази задача се решава чрез търсене в пространство от хипотези, като най-често този процес се управлява от два типа евристични функции. Едният от тях използва принадлежността на примерите към класовете, за определянето на различни видове точност (грешка) на хипотезите. Другите функции, като ентропия, компресия и информационна печалба, използват кодирането на примери и хипотези.

В тази дипломна работа ще разгледаме въпросът за използването на разстояние между примери и хипотези в системите за индуктивно самообучение. Болшинството от изследванията досега върху метрика разглеждат примери представени на езика атрибут-стойност (който е еквивалентен на съждителното смятане). Обектите, с които ние ще работим, са термове, атоми и клаузи, които широко се използват в машинното самообучение и по-специално в индуктивното логическо програмиране. Причината за това е тяхната по-голяма изразителност от тази на езика атрибут-стойност.

В тази област има доста малко изследвания, като може би първите са тези на Жил Бисон [3] и после на Емде и Ветшерек [6]. Функциите, които те дефинират, изчисляват близостта между два атома, използвайки основни знания. Най-близка по идея е работата на Алан Хътчинсън [8], който използва синтактичните различия между термове и между атоми за дефинирането на псевдометрика.

Тъй като обектите, които ще разглеждаме се оказва, че образуват полурешетки, първо ще разгледаме принципно някои подходи за дефинирането на псевдометрика в такива структури. Ще използваме псевдометрика, а не метрика, тъй като според една теорема всяка метрична решетка е модуларна, а както ще видим по-нататък това условие е доста силно и не е изпълнено от нито една от структурите, които ще разглеждаме.

За намирането на разстояние между два елемента ще използваме техния инфимум. В машинното самообучение намирането на точна долна граница съответства на обобщаване на множество от елементи. Малко са системите (напр. GOLEM [13]), които използват този подход за решаване на индуктивната задача, тъй като основен проблем при тях е получаването на преобобщаване при избирането на неподходящо множество от примери. Както ще видим в предложената от нас система за индуктивно самообучение използването на разстояние между хипотези и примери ни дава възможост да избягваме този проблем, като всеки път избираме по-възможност най-специфичното обобщение.

Друг принос на настоящата работа е използването на унифициран подход за решаването на различните типове задачи на индуктивното машинно самообучение.

Организация на дипломната работа

В глава 2 са разгледани няколко възможни начина за дефиниране на псевдометрика в полурешетки. В глава 3 се изследват подробно свойствата на алгебричните структури, определени от термове, атоми и клаузи, като се разглежда приложимостта на различните видове псевдометрики върху тях. В глава 4 е дадено кратко формално изложение на областта на индуктивното машинно самообучение, а в глава 5 е представен един алгоритъм за машинно самообучение, който се базира на дефинираната по-рано структура и псевдометрика между атоми. В глава 6 са изложени резултатите от приложението на предложения алгоритъм за машинно самообучение в различни области и задачи. В приложения А и Б се дава кратко изложение на понятия от областта на дискретната математика и логическото програмиране и са включени за пълнота на изложението.

Глава 2

Псевдометрика в полурешетка

Идеята за псевдометриката в полурешетка е следната. Предполага се, че ако два елемента не са сравними спрямо някаква частична наредба, то вероятно не бихме могли да намерим и разстоянието между тях. За това ще въведем функция с определени свойства, която измерва разликата само между сравними елементи. Тогава за всеки два елемента, които не са сравними, ще сумираме разликите между тях и точната им долна граница. Поради тези изисквания е необходимо частично нареденото множество да бъде полурешетка с инфимум.

Дефиниция 2.1 (Разлика между елементи) Нека частично нареденото множество $(L, \preceq)$ е полурешетка с инфимум. Да предположим, че за всеки два елемента $a, b \in L$, за които $a \preceq b$, е дефинирана функция $s(a, b)$ със следните свойства:

S1 $s(a, b) \geq 0$;

S2 $s(a, a) = 0$.

S3 За всеки три елемента $a, b, c \in L$, ако $a \preceq b, a \preceq c$ и $c \preceq b$, тогава

$$s(a, b) \leq s(a, c) + s(c, b).$$

Както ще видим по-надолу това изискване е необходимо, за да се удовлетвори неравенството на триъгълника за дефинираната от нас псевдометрика;

$$\begin{tikzcd}
& b \arrow[dl] \arrow[dd] \\
c \arrow[dr] & \\
& a
\end{tikzcd}$$

S4 При същите предположения, като при S3

$$s(c, b) \leq s(a, b).$$

Това условие следва от нашето предположение, че сумата от разликите до инфимумът на два елемента е по-малка от сумата от разликите до произволна тяхна долна граница;

S5 Ако съществува супремум d на елементите a и b и $c = \inf\{a, b\}$, тогава

$$s(c, a) + s(c, b) \leq s(a, d) + s(b, d).$$

Това свойство ни дава правото да използваме сумата от разликите до инфимума на два елемента, вместо до супремума.

$$\begin{tikzcd}
 & d \arrow[dl] \arrow[dr] & \\
a \arrow[dr] & & b \arrow[dl] \\
 & c &
\end{tikzcd}$$

При тези условия ще казваме, че функцията s е *разлика* в полурешетката $(L, \preceq)$. Обърнете внимание, че ако навсякъде в горната дефиниция заменим използването на супремум с инфимум, получаваме аналогично твърдение за разлика в полурешетка със супремум.

В случая, когато неравенството от точка S3 е изпълнено като равенство, от S1 и S3 автоматично следва условие S4. Тогава функцията s ще наричаме *точна разлика*.

Дефиниция 2.2 (Псевдометрика в полурешетка) Нека е дадена полурешетка с инфимум $(L, \preceq)$ и разлика между нейните елементи s. Ще дефинираме псевдометрика между елементите от L спрямо s по следния начин:

$$d(a, b) = s(\inf\{a, b\}, a) + s(\inf\{a, b\}, b).$$

Доказателство:

1. $d \geq 0$ от S1;

2. $d(a, a) = 0$ за всяко a:

$$\begin{aligned} d(a, a) &= s(\inf\{a, a\}, a) + s(\inf\{a, a\}, a) = \\ &= s(a, a) + s(a, a) = 0; \end{aligned}$$

3. d е симетрично поради симетричността на събирането;

4. Ще покажем, че за всеки три елемента $a_1, a_2, a_3 \in L$ е изпълнено неравенството $d(a_1, a_2) \leq d(a_1, a_3) + d(a_3, a_2)$. Нека $b_1 = \inf\{a_1, a_3\}$, $b_2 = \inf\{a_2, a_3\}$, $c = \inf\{a_1, a_2\}$ и $d = \inf\{b_1, b_2\}$.

$$\begin{tikzcd}
a_1 \arrow[d] \arrow[dr] & a_3 \arrow[dl] \arrow[dr] & a_2 \arrow[dl] \arrow[d] \\
b_1 \arrow[dr] & c & b_2 \arrow[dl] \\
 & d &
\end{tikzcd}$$

От транзитивността на наредбата следва, че $d \preceq a_1$ и $d \preceq a_2$. Тъй като c е инфимум на a_1 и a_2, следва че $d \preceq c$.

$$\begin{aligned} s(c, a_1) &\leq s(d, a_1) \\ &\leq s(d, b_1) + s(b_1, a_1). \end{aligned} \tag{1}$$

По аналогичен начин

$$s(c, a_2) \leq s(d, b_2) + s(b_2, a_2). \tag{2}$$

$$s(d, b_1) + s(d, b_2) \leq s(b_1, a_3) + s(b_2, a_3). \tag{3}$$

Като съберем (1), (2) и (3) получаваме:

$$\begin{aligned} & s(c,a_1)+s(c,a_2)+s(d,b_1)+s(d,b_2) \leq \\ & \quad \leq \quad s(d,b_1)+s(b_1,a_1)+s(d,b_2)+s(b_2,a_2)+s(b_1,a_3)+s(b_2,a_3) \end{aligned}$$

$$\begin{aligned} s(c,a_1)+s(c,a_2) & \leq s(b_1,a_1)+s(b_1,a_3)+s(b_2,a_2)+s(b_2,a_3) \\ d(a_1,a_2) & \leq d(a_1,a_3)+d(a_2,a_3). \end{aligned}$$

2.1 Разлика в полурешетка с модуларни интервали

В случая, когато всички интервали в дадена полурешетка са модуларни решетки с крайна дължина, съвсем естествено е да дефинираме разликата между два елемента, като дължината на коя да е свързана верига между тях, тъй като според твърдение A.24 всички свързани вериги в модуларна решетка имат равни дължини.

Твърдение 2.3 Нека е дадена полурешетка с инфимум $(L,\preceq)$, в която всеки интервал $[a,b]$ е модуларен с крайна дължина. Тогава дължината на произволна свързана верига между елементите a и b е разлика. Такъв вид разлика ще бележим със символа s_m.
Доказателство: Първите две условия за разлика следват директно от дефиницията за дължина на верига. Лесно се вижда, че условие S3 от дефиницията на разлика е изпълнено като равенство, следователно тази функция е точна разлика. Нека $a,b,c,d \in L$, a и b да не са сравними и $c=\inf\{a,b\}$ и $d=\sup\{a,b\}$. Тогава съществуват свързани вериги $c=x_0,\ldots,x_k=a,\ldots,x_n=d$ и $c=y_0,\ldots,y_l=b,\ldots,y_n=d$. Да допуснем, че неравенството S5 не е изпълнено, т.е.

$$\begin{aligned} s_m(c,a)+s_m(c,b) & > s_m(a,d)+s_m(b,d) \\ k+l & > n-k+n-l \\ 2k+2l & > 2n \\ k+l & > n \geq l+1 \\ k & > 1, \end{aligned}$$

тъй като $k \leq n-1$ и $l \leq n-1$. Тогава множеството $\{a=x_k, x_{k-1}, b, c, d\} \subset L$ е подрешетка на L и според твърдение A.22 решетката L не е модуларна, следователно допускането ни е грешно.

По аналогичен начин можем да докажем и обратното твърдение, следователно условието $s(a,d)+s(b,d) \geq s(c,a)+s(c,b)$ се изпълнява като равенство.

2.2 Разлика чрез височина на елемент

В този раздел ще разгледаме един друг начин за дефиниране на разлика, като използваме функция, която оценява самите елементи на дадената полурешетка.

Дефиниция 2.4 (Височина на елемент) Ще казваме, че функцията $h(x)$ е височина на елементите от дадена полурешетка с инфимум $(L,\preceq)$, ако притежава следните свойства:

H1 ако $a \preceq b$, то $h(a) \leq h(b)$ (монотонност);

H2 ако съществува супремумът d на елементите a и b и $c=\inf\{a,b\}$, тогава $h(a)+h(b) \leq h(c)+h(d)$.

Твърдение 2.5 Функцията $s_h(a,b) = h(b) - h(a)$ е разлика.
Доказателство:

S1) $s_h(a,b) = h(b) - h(a) \geq 0$ (от условие H1).

S2) $s_h(a,a) = h(a) - h(a) = 0$.

S3) Нека $a, b, c \in L$ и $a \preceq b, a \preceq c$ и $c \preceq b$, тогава

$$\begin{aligned} s_h(a,b) &= h(b) - h(a) = \\ &= h(b) - h(c) + h(c) - h(a) = \\ &= s_h(c,b) + s_h(a,c). \end{aligned}$$

Следователно функцията s_h е точна разлика и условие S4 е изпълнено автоматично.

S5) Нека съществува супремумът d на елементите a и b и $c = \inf\{a,b\}$, тогава

$$\begin{aligned} s_h(c,a) + s_h(c,b) &= h(a) - h(c) + h(b) - h(c) = \\ &= h(a) + h(b) - 2h(c) = \\ &= 2(h(a) + h(b)) - h(a) - h(b) - 2h(c) \leq \\ &\leq 2(h(c) + h(d)) - h(a) - h(b) - 2h(c) = \\ &= h(d) - h(a) + h(d) - h(b) = \\ &= s_h(a,d) + s_h(b,d). \end{aligned}$$

Тогава за разликата между два елемента a и b се получава

$$\begin{aligned} d(a,b) &= s_h(\inf\{a,b\}, a) + s_h(\inf\{a,b\}, b) = \\ &= h(a) - h(\inf\{a,b\}) + h(b) - h(\inf\{a,b\}) = \\ &= h(a) + h(b) - 2h(\inf\{a,b\}) \end{aligned}$$

Глава 3

Алгебрични структури определени от термове, атоми и клаузи

3.1 Алгебрична структура определена от термове

В този раздел ще разгледаме свойствата на алгебричната структура, определена от термове, като ще се интересуваме от съществуването на условията, разгледани в предишната глава, за приложимост на различните видове псевдометрики.

Дефиниция 3.1 Нека A е азбука на предикатно смятане от първи ред. Множеството от всички термове над A ще бележим с $\mathcal{T}$.

Дефиниция 3.2 (Обобщение на термове, $\preceq$) Ще казваме, че термът t_1 е по-общ от t_2, означено с $t_1 \preceq t_2$, ако съществува субституция θ, такава че $t_1\theta = t_2$. Термът t_2 се нарича *специализация* на t_1, а t_1 е *обобщение* на t_2.

Твърдение 3.3 Релацията обобщение на термове върху множеството от всички термове $\mathcal{T}$ е полунаредба:

1. От $t\varepsilon = t$ следва, че $t \preceq t$ за всяко $t \in \mathcal{T}$ (рефлексивност);

2. Нека $t_1 \preceq t_2$ и $t_2 \preceq t_3$. Тогава съществуват субституции θ и σ, такива че $t_1\theta = t_2$ и $t_2\sigma = t_3$. Нека $\rho = \theta\sigma$. Тогава $t_1\rho = t_3$, т.е. $t_1 \preceq t_3$ (транзитивност).

По-нататък ще разглеждаме само частичната наредба над класовете на еквивалентност, индуцирана от релацията обобщение на термове, която ще отбелязваме със същия символ. За практически приложения от всеки клас на еквивалентност ще избираме произволен представител, с който ще работим.

Ясно е, че произволна променлива v е универсална долна граница O на множеството $\mathcal{T}$, тъй като за произволен терм t субституцията $\theta = \{v/t\}$ ни дава $v\theta = t$.

От теоремата за унификация следва, че ако за някое крайно множество от термове съществува горна граница, то за него съществува и точна горна граница. Освен това лесно се вижда, че съществуват термове (например всички основни термове), за които не съществува горна граница. Следователно частично нареденото множество от термове *не е полурешетка със супремум.*

Първото сериозно изследване върху алгебричната структура, определена от термове и атоми е на Рейнолдс [17]. Сега ще изложим накратко и без доказателства по-важните негови резултати. Трябва да се отбележи, че той разглежда множеството от всички атоми, разширено с универсална долна и горна граница, но все пак, поради голямата синтактична близост между термове и атоми, неговите резултати са приложими директно и към термове.

Твърдение 3.4 (Точна долна граница) За произволно множество от термове (включително и безкрайно) съществува точна долна граница спрямо дефинираната наредба между термове.

Процеса на намирането на инфимум на термове се нарича *анти-унификация*. Сега ще представим предложения от Рейнолдс алгоритъм за намиране на точна долна граница на два терма a и b. Същият алгоритъм е независимо открит и от Плоткин [16].

Алгоритъм 3.5 (Анти-унификация) Като вход на алгоритъма се подават два терма a и b, а резултатът е терм c, който е техният инфимум. Нека $u_1, u_2, \ldots$ е последователност от променливи, неучастващи нито в a, нито в b.

1. Нека $m = 0, a' = a, b' = b, \zeta = \varepsilon$ и $\eta = \varepsilon$.

2. Ако $a' = b'$ тогава изход, като резултатът е термът $c = a' = b'$.

3. Нека k е най-лявата позицията, в която a' и b' се различават и нека s и t са термовете, които започват от тази позиция съответно в a' и b'.

4. Ако за някое i, такова че $1 \leq i \leq m, u_i\zeta = s$ и $u_i\eta = t$, тогава замени терма s на позиция k в a' с променливата u_i и замени терма t на позиция k в b' със същата променлива u_i и отиди на стъпка 2.

5. В противен случай увеличи m с единица, замени терма s на позиция k в a' с променливата u_m и замени терма t на позиция k в b' със същата променлива u_m, замени субституцията ζ с $\zeta \cup \{u_m/s\}$ и субституцията η с $\eta \cup \{u_m/t\}$ и отиди на стъпка 2.

След приключването на алгоритъма субституциите ζ и η са такива, че $c\zeta = a$ и $c\eta = b$.

Следователно частично нареденото множеството $(\mathcal{T}, \preceq)$ е *пълна полурешетка с инфимум.*

Дефиниция 3.6 (Размер на терм) Нека е даден терм $t \in \mathcal{T}$. Тогава функцията $size(t)$, ще дефинираме като броят на всички символи в t, минус броя на различните променливи в t.

Рейнолдс е доказал, че функцията $size$ е строго монотонна, т.е. ако $t_1 \prec t_2$, то $size(t_1) < size(t_2)$. Използвайки това свойство, лесно се проверява следното твърдение, че *всички свързани вериги между два терма имат крайна дължина.*

Последното твърдение от Рейнолдс, което ще представим, ни дава възможност да намираме всички горни покрития на даден терм t, чрез прилагането на определен вид субституции.

Твърдение 3.7 Термът t_2 покрива t_1, ако съществува субституция θ такава, че $t_1\theta$ е вариант на t_2 и θ има следния вид:

1. $\theta = \{v/u\}$, където v и u са променливи, принадлежащи на t_1;

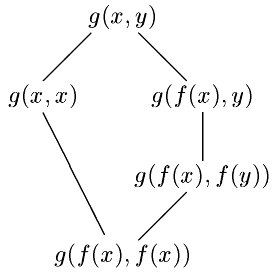
2. $\theta = \{v/c\}$, където v е променлива принадлежаща на t_1, а c е произволна константа;

3. $\theta = \{v/f(u_1, \ldots, u_n)\}$, където v е променлива, принадлежаща на t_1, f е функционален символ с n аргумента и $u_1, \ldots, u_n$ са различни променливи, непринадлежащи на t_1.

Твърдение 3.8 Нека a и b са термове, принадлежащи на $\mathcal{T}$, такива че $a \preceq b$. Тогава интервалът $[a, b] = \{t \in \mathcal{T} | a \preceq t \preceq b\}$ е решетка.
Доказателство: Нека термовете $t_1, t_2 \in [a, b]$. Тъй като за всеки два терма съществува инфимум c, който по дефиниция е по-голям от всяка друга долна граница на t_1 и t_2 и в частност $a \preceq c$, следователно $c \in [a, b]$. От $t_1, t_2 \in [a, b]$, следва че $t_1 \preceq b$ и $t_2 \preceq b$, което означава, че b е горна граница за t_1 и t_2. Следователно съществува супремум d на двата терма, който също принадлежи на интервала $[a, b]$.

Твърдение 3.9 В множеството $\mathcal{T}$ съществуват интервали, които не са модуларни.
Доказателство: Нека азбуката, над която работим, съдържа функционалния символ f с един аргумент и функционален символ g с два аргумента. На фигура 3.1 е показан интервалът $[g(x,y), g(f(x), f(x))]$. Тогава според твърдение A.22 той не е модуларен.

Освен това от фигурата се вижда, че този интервал не е дори и полумодуларен, нито от горе, нито от долу.



Фигура 3.1: Немодуларен интервал[1]

Твърдение 3.10 Нека a и b са термове, които имат инфимум c и нека съществува техният супремум d, т.е. те са унифицируеми. Тогава функцията $f(t)$, която е равна на броя на функционалните символи в терма t, е височина на елемент.
Доказателство: Нека с $f(\theta)$ да означим броя на функционалните символи в субституцията θ.

H1) Нека $a \preceq b$. Тогава съществува субституция θ такава, че $a\theta = b$ и следователно $f(b) = f(a) + f(\theta)$. Тогава $f(b) \geq f(a)$.

H2) Нека субституциите θ_1 и θ_2 са такива, че $c\theta_1 = a$ и $c\theta_2 = b$. Тогава $f(a) = f(c) + f(\theta_1)$ и $f(b) = f(c) + f(\theta_2)$. Тъй като в d се съдържат всички функционални символи от терма c и субституциите θ_1 и θ_2, тогава

$$\begin{array}{rcl} f(d) & \geq & f(c) + f(\theta_1) + f(\theta_2) \\ f(c) + f(d) & \geq & f(c) + f(\theta_1) + f(c) + f(\theta_2) \\ f(c) + f(d) & \geq & f(a) + f(b). \end{array}$$

3.2 Алгебрични структури определени от атоми

Ако сравним синтактично термовете и атомите, то единствената разлика между тях е, че атомите могат да започват само с предикатен символ, докато термовете могат да започват с функционален символ или да бъдат променливи или константи. Това ни налага и ограничение върху множеството от атоми, като ще разглеждаме само атоми, които имат еднакъв брой аргументи и започват с един и същ предикатен символ. В противен случай инфимумът на два атома с различен брой аргументи или предикатни символи би бил променлива, което не е атом. Лесно се проверява, че всички свойства на структурата, определена от термове, разгледани в предишния раздел са валидни и в случая, когато разглеждаме множество от атоми, които се състоят от един и същи предикатен символ и еднакъв брой аргументи.

Сега ще разгледаме структурата, определена от атоми без функционални символи.

[1]На всички чертежи на термове и клаузи ще изобразяваме по-общите (по-малките) елементи по-нагоре.

Твърдение 3.11 Произволен интервал от атоми без функционални символи е полумодуларен отгоре.

Доказателство: Нека е даден интервалът $[t_1, t_2]$ и нека атомите a и b да принадлежат на него и да не са сравними. Тогава съществува инфимум c и супремум d на a и b, които също принадлежат на този интервал. Това означава, че съществуват субституции $\theta_1, \theta_2, \varphi_1$ и φ_2 такива, че $c\theta_1 = a, c\theta_2 = b, a\varphi_1 = d$ и $b\varphi_2 = d$. С $I(t)$ ще означаваме множеството от всички позиции в атома a, на които се среща термът t.

(ξ) Когато атомите a и b покриват c, тогава субституциите θ_1 и θ_2 ще имат вида $\theta_1 = \{v_1/t_1\}$ и $\theta_2 = \{v_2/t_2\}$, където променливите v_1 и v_2 принадлежат на c и поне един от термовете t_1 и t_2 е променлива u, принадлежаща на c (в противен случай няма да съществува супремума на a и b).

1. Нека $v_1 = v_2 = v$. Тогава субституциите имат вида $\theta_1 = \{v/u\}$ и $\theta_2 = \{v/t\}$, където u е променлива, принадлежаща на c, а t е или константа, или променлива, принадлежаща на c. Очевидно след прилагането на двете субституции единствените позиции, на които a и b се различават са $I^c(v)$, като в атома a на тези позиции се намира променливата u, а в b - термът t. Тогава за да унифицираме a и b трябва да приложим субституцията $\varphi_1 = \{u/t\}$ към a. След прилагането на субституцията, на позиции $I^c(u)$ в атома a ще се съдържа термът t, а в b - променливата u, като това ще бъдат всички места, на които тя се намира. Тогава ако към b приложим субституцията $\varphi_2 = \{u/t\}$, тогава $d = a\varphi_1 = b\varphi_2$, ще бъде точната горна граница на a и b и тя ще покрива атомите a и b, тъй като субституциите φ_1 и φ_2 имат вида от твърдение 3.7.

2. Нека $v_1 \neq v_2$. Тогава субституциите имат вида $\theta_1 = \{v_1/u\}$ и $\theta_2 = \{v_2/t\}$, където u е променлива, принадлежаща на c, а t е или константа, или променлива, принадлежаща на c. Да приемем, че $t \neq v_1$. След прилагането на двете субституции, на позиции $I^c(v_2)$ в a ще се съдържа променливата v_2, а в b - термът t. Тогава, за да ги унифицираме, към a трябва да приложим субституцията $\varphi_1 = \{v_2/t\}$. Другите позиции, на които $a_1 = a\varphi_1$ и b се различават, са $I^c(v_1)$. В атома b на тези места се намира променливата v_1 и това са всички места, на които тя се намира, защото $t \neq v_1$. Ако $u = v_2$, в атома a_1 на тези позиции се съдържа t. Тогава към b трябва да приложим субституцията $\varphi_2 = \{v_1/t\}$, за да получим $d = a\varphi_1 = b\varphi_2$. В противен случай, към b трябва да приложим субституцията $\varphi_2 = \{v_1/u\}$, за да получим $d = a\varphi_1 = b\varphi_2$. И в двата случая според твърдение 3.7 d покрива както a, така и b.

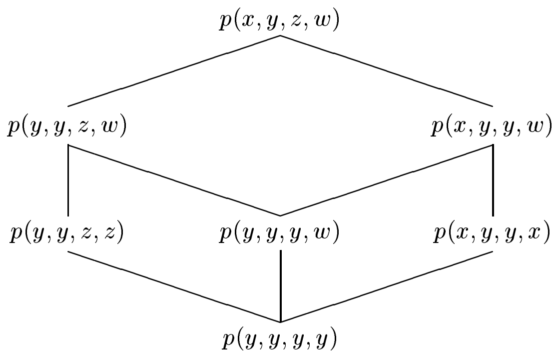
За да покажем, че съществуват интервали, които не са полумодуларни отдолу, ще дадем един контрапример. На фигура 3.2 е показана част от полурешетката на атомите с предикатен символ p и четири аргумента. Вижда се, че атомът $p(y, y, y, y)$ покрива атомите $a = p(y, y, z, z)$ и $b = p(x, y, y, x)$ със субституциите $\{z/y\}$ и $\{x/y\}$ (w, x, y и z са променливи). Техният инфимум е атомът $p(x, y, z, w)$, който очевидно не се покрива от a и b.

От фигура 3.2 се вижда, че функцията *size не е височина на елемент*, тъй като $size(p(y, y, z, z)) + size(p(x, y, y, x)) = (5-2)+(5-2) = 6 \nleq size(p(y, y, y, y)) + size(p(x, y, z, w)) = (5-4)+(5-1) = 5$. Все пак тя може да се използва, като подходяща евристична функция.

3.3 Алгебрични структури определени от клаузи

В този раздел ще разгледаме свойствата на алгебричните структури определени от различни наредби между клаузи.

Дефиниция 3.12 Нека A е азбука на предикатното смятане от първи ред. Множеството от всички клаузи над A ще бележим с $\mathcal{C}$, а множеството от всички клаузи на Хорн - с $\mathcal{H}$.



Фигура 3.2:

3.3.1 Наредба спрямо логическо следване

Най-естествената релация между клаузи, за която се сещаме да използваме, е логическото следване $\models$.

Дефиниция 3.13 (Логическо следване, $\preceq_I$)
Нека са дадени клаузите $C, D \in \mathcal{C}$. Ще казваме, че клаузата C *е по-обща спрямо логическо следване* от D, означено с $C \preceq_I D$, ако $\{C\} \models D$.

Основен проблем при използването на тази наредба в практически приложения е нейната неразрешимост, както за общи клаузи, така и за клаузи на Хорн.

Освен това в [15] е показано, че само за общи клаузи съществува супремум и инфимум спрямо логическо следване, като за наличието на точна долна граница е необходимо поне една от клаузите да не е тавтология и да не съдържа фунционални символи.

3.3.2 Наредба спрямо θ-поглъщане

Наредбата θ-поглъщане между клаузи е въведена от Плоткин [16].

Дефиниция 3.14 (θ-поглъщане, $\preceq_\theta$) Нека са дадени клаузите $C, D \in \mathcal{C}$. Ще казваме, че C *θ-поглъща* D, означено с $C \preceq_\theta D$, ако съществува субституция θ, такава че $C\theta \subseteq D$.

Твърдение 3.15 Релацията θ-поглъщане между клаузи, е полунаредба:

1. От $C \subseteq C$, следва че $C \preceq_\theta C$ (рефлексивност);

2. Нека са дадени клаузите C, D и E, за които $C \preceq_\theta D$ и $D \preceq_\theta E$. Тогава съществуват субституции θ и σ, такива че $C\theta \subseteq D$ и $D\sigma \subseteq E$. Следователно $C\theta\sigma \subseteq E$, което означава че $C \preceq_\theta E$ (транзитивност);

Дефиниция 3.16 (Еквивалентни клаузи спрямо θ-поглъщане, $\sim_\theta$)
Ще казваме, че клаузите C и D са *еквивалентни спрямо θ-поглъщане*, означено с $C \sim_\theta D$, ако C θ-поглъща D и D θ-поглъща C. За всяка клауза C, множеството от всички клаузи, които са еквивалентни на нея, ще наричаме клас на еквивалентност.

Обърнете внимание обаче, че класовете на еквивалентност при θ-поглъщането са безкрайни и елементите в тях не са варианти, което създава трудности при практически приложения на тази релация. За да се справим с този проблем, трябва от всеки клас на еквивалентност да се избере само по един представител. За тази цел ще разгледаме специален вид клаузи - редуцирани.

Дефиниция 3.17 (Редуцирана клауза) Клаузата C се нарича *редуцирана* тогава и само тогава, когато не съществува литерал $L \in C$, такъв че клаузите C и $C \setminus \{L\}$ да са еквивалентни.

Плоткин е доказал [16], че ако две клаузи са еквивалентни, то техните редукции са варианти. Това твърдение ни дава възможност еднозначно да изберем по един представител от всеки клас на еквивалентност. Множеството от всички редуцирани клаузи ще бележим с $\mathcal{C}_r$.

Едно важно свойство на θ-поглъщането е, че ако $C \preceq_\theta D$, тогава $C \preceq_I D$. Обратното твърдение обаче не винаги е вярно. Да разгледаме клаузите: $C = list([V|W]) \leftarrow list(W)$ и $D = list([X,Y|Z]) \leftarrow list(Z)$. Ако разгледаме само първата клауза, тя покрива списъци с произволна дължина, докато само втората покрива всички списъци с четна дължина. Като резолвираме клаузата C със себе си се вижда, че $C \models D$, но твърдението C θ-поглъща D не е вярно. Това наблюдение показва, че θ-поглъщането е по слаба наредба отколкото логическото следване.

Плоткин е доказъл, че за всеки две клаузи съществува точна долна граница, която той нарича *най-малко общо обобщение спрямо θ-поглъщане* (lgg_θ). Следователно частично нареденото множество $(\mathcal{C}_r, \preceq_\theta)$ е полурешетка с инфимум.

Едно интересно свойство на θ-поглъщането е наличието на безкрайно намаляващи и безкрайно растящи вериги.

Пример 3.18 (Безкрайно растяща верига) Да разгледаме клаузите

$$D = q \leftarrow p(X_1, X_1)$$
$$D_n = q \leftarrow \{p(X_i, X_j) | 1 \leq, i, j \leq n, i \neq j\}, n \geq 2.$$

Доказано е, че всички те са редуцирани и $D_2 \prec D_3 \prec \ldots \prec C$ е безкрайно растяща верига.

Следователно съществуват клаузи C_1 и C_2, такива че $C_1 \preceq_\theta C_2$ (например $C_1 = D_2, C_2 = D$) и интервалът $[C_1, C_2]$ не е с крайна дължина. Това означава, че не можем да използваме резултата от раздел 2.1 за разлика в случая, когато два елемента образуват интервал с крайна дължина.

3.4 Други изследвания върху разстояние между термове и клаузи

Най-близка по идея е работата на Хътчинсън [8]. В нея за определянето на псевдометрика между термове е използвана разлика между тях, която е дефинирана като броят на различните функционални символи, които се срещат в субституцията между двата елемента. Това определение не е много подходящо, тъй като чрез него не могат да се уловят разлики между термове, които представляват рекурсивни структури. Например термът $t_1 = s(X)$ е по-общ от термът $t_2 = s(s^n(0)), n \geq 0$, със субституцията $\theta = \{X/s^n(0)\}$. Според [8] разликата между t_1 и t_2 ще бъде 2, докато $s(t_1, t_2) = f(\theta) = n$.

По отношение на разстоянието между клаузи, Хътчинсън използва хаусдорфово разстояние, разглеждайки клаузите като множество от термове.

Глава 4

Въведение в индуктивното машинно самообучение

В тази глава ще направим кратко изложение на индуктивното машинно самообучение (ИМС).

Машинното самообучение е област от изкуствения интелект, чиито цели са да разработи методи, техники и средства за създаване на системи, които могат да се променят с цел да изпълняват „по-добре“ дадена задача. Под „по-добре“ може да се разбира по-ефективно, с по-голяма точност или възможността да решават по-широк клас от проблеми. В машинното самообучение съществуват различни подходи за решаване на задачата, като индуктивно самообучение, обучение със запомняне на примери, обучение чрез невронни мрежи, генетични алгоритми и др.

4.1 Формална дефиниция на индуктивната задача

Основната цел на ИМС е да се изведат описания на класове на базата на примери. Нека $\mathcal{L}_E$ е език за описание на примерите. Тогава формално под клас $\mathcal{C}$ ще разбираме подмножество на $\mathcal{L}_E$: $\mathcal{C} \subseteq \mathcal{L}_E$. Нашата цел е да построим вътрешно представяне на $\mathcal{C}$, което да се описва на език на хипотезите $\mathcal{L}_H$. Връзката между хипотезите и примерите се дава чрез релация за изводимост $\rightarrow$, която е подмножество на $\mathcal{L}_H \times \mathcal{L}_E$. Тогава класът $\mathcal{C}(h)$, който се определя от хипотезата $h \in \mathcal{L}_H$ се дефинира по следния начин: $\mathcal{C}(h) = \{a \in \mathcal{L}_E | h \rightarrow a\}$. Релацията $\rightarrow$ може естествено да се разшири и до изводимост между хипотези, като $h_1 \rightarrow h_2, h_1, h_2 \in \mathcal{L}_H$, ако $\mathcal{C}(h_1) \supseteq \mathcal{C}(h_2)$, т.е. хипотезата h_1 е по-обща от хипотезата h_2.

Като пример ще разгледаме една формална система, която се използва от голяма част от системите за индуктивно самообучение. Примерите в нея се описват на езика атрибут-стойност, в който обектите представляват множество от двойки $A_i = V_i, i = 1, \ldots, n$, където A_i е име на атрибут, а V_i е неговата стойност, която принадлежи на дадена област D_i. Ако се фиксира последователността на атрибутите за даден език, то примерите могат да се представят и като вектор от вида $(V_1, \ldots, V_n)$.

Езика за описание на хипотезите се състои от правила, записани като конюнкция от двойки атрибут-стойност: $A_{i_1} = V_1 \wedge \ldots \wedge A_{i_k} = V_k, k \leq n$. Релацията за изводимост $\rightarrow$ се дефинира по следния начин: ще казваме, че дадена хипотеза извежда даден пример, ако стойностите на всички атрибути от хипотезата съвпадат със стойностите на същите атрибути в примера.

Като друг пример за формална система за машинно самообучение можем да разгледаме тази, която се използва в индуктивното логическо програмиране (ИЛП) и която се базира на предикатното смятане от първи ред. Основната цел в тази област е извеждането на логически програми, използвайки примери за тяхното коректно поведение. Езикът на примерите в тази система представлява множеството от всички основни атоми с даден предикатен сивол и брой аргументи. Хипотезите се състоят от множество от (ограничени) клаузи на Хорн, чиято глава има същата сигнатура като тази на примерите. Като релацията за изводимост $\rightarrow$ се използва правилото за резолютивен извод.

Съществен е въпросът, дали всеки клас $\mathcal{C}$ може да се опише от хипотеза, принадлежаща на $\mathcal{L}_H$. Тъй като това не винаги е възможно, то в много системи езикът на хипотезите $\mathcal{L}_H$ се разширява до дизюнктивен език $\mathcal{L}_H^{\vee}$, в който елементите представляват дизюнкции от вида $h_1 \vee \ldots \vee h_n$ от обекти, описани на езика $\mathcal{L}_H$. Тогава класът, който се определя от дадена дизюнктивна хипотеза $H = h_1 \vee \ldots \vee h_n$ се дефинира като $\mathcal{C}(H) = \{a \in \mathcal{L}_E | \exists i (i = 1, \ldots, n), h_i \rightarrow a\}$.

Най-често даден клас е възможно да се описва от няколко различни хипотези. В този случай се прилага някакъв критерий за избор на хипотеза, като много често се предпочитат хипотези с по-просто представяне. Когато използваме дизюнктивен език за представяне на хипотезите е естествено да предпочитаме тези, които имат по-малък брой дизюнкти.

Най-общо задачата за ИМС може да се дефинира по следния начин. Дадено е подмножество $\mathcal{E} \subset \mathcal{C}$ от представители на някакъв клас $\mathcal{C}$, принадлежащи на даден език на примери $\mathcal{L}_E$. Да се намери хипотеза $h \in \mathcal{L}_H$, такава че тя описва целия клас $\mathcal{C}$, т.е. $\mathcal{C}(h) = \mathcal{C}$. С други думи *по дадено частично описание на някакъв клас да изведем представяне на целия.*

4.2 Видове системи за ИМС

В различните системи за ИМС примерите, с които те работят, могат да принадлежат на един или на няколко класа, като това може да бъде или да не бъде явно указано. В зависимост от това различаваме няколко вида системи за машинно самообучение:

- Всички примери принадлежат само на един клас. В този случай задачата се нарича *самообучение от положителни примери.* Веднага се вижда, че хипотезата, покриваща всички възможни примери, и хипотезата, покриваща само дадените ни положителни примери, са две възможни решения на задачата, но те не са интересни, тъй като не ни дават никаква допълнителна информация за решавания проблем. Следователно най-съществената част от такава система е използването на подходящ критерий за избор на хипотези.

- Примерите принадлежат на няколко класа, но броят на класовете, както и принадлежността на примерите към тях не са известни. В този случай разглеждаме задачата за *концептуална клъстеризация*, или *самообучение без учител*, чиято цел е да се групират примерите в *организирано множество* от класове. Под организирано множество от класове най-често се разбира те да бъдат подредени в йерархия, но съществуват и системи, които използват като представяне ациклични графи.

- За част от примерите е указана принадлежността към даден клас, докато за останалата е указано, че те не принадлежат към търсения клас. Следователно множеството $\mathcal{E}$ се разбива на две непресичащи се множества $\mathcal{E}^+$ и $\mathcal{E}^-$. Тази задача се нарича *обучение от положителни и отрицателни* примери, като търсената хипотеза h трябва да отговаря на следните две условия:

  - $\mathcal{E}^+ \subseteq \mathcal{C}(h)$ (достатъчност);
  - $\mathcal{E}^- \cap \mathcal{C}(h) = \emptyset$ (непротиворечивост).

- Търсим описание на няколко класа, като за всеки от примерите е указана принадлежността му към някой от тях. В този случай множеството от примерите $\mathcal{E}$ се разбива на няколко непресичащи се множества $\mathcal{E}^i, i = 1, \ldots, n$, където n е броят на класовете, като всяко от тях съдържа примерите от съответния клас. Най-често тази задача се свежда до обучение от положителни и отрицателни примери, като за всеки клас примерите, принадлежащи на него, се разглеждат като положителни, а всички останали като отрицателни.

Обикновено задачата за машинно самообучение се решава чрез евристично търсене в множеството от всички хипотези, до намирането на такава, която отговаря на условията на задачата, а в случая на концептуална клъстеризация в пространството от дървета или графи от хипотези.

4.3 Класификационна точност на хипотеза

Много често условието за непротиворечивост на индуктивната задача не може да бъде изпълнено, поради недостатъчно изразителен език на хипотезите, невъзможност на алгоритъма да намери подходяща хипотеза или грешки в примерите. За това е необходимо да използваме някакъв друг критерий за избор на *добра* хипотеза. Една такава оценка за качество на хипотеза е нейната класификационна точност.

Дефиниция 4.1 (Класификационна точност) *Класификационната точност* на дадена хипотеза h при даден клас $\mathcal{C}$ се дефинира по следния начин:

$$A(\mathcal{C}|h) = \frac{|\mathcal{C}(h) \cap \mathcal{C}|}{|\mathcal{C}|}, \tag{1}$$

т.е. вероятността даден обект, описан от хипотезата, да принадлежи на съответния клас.

Тогава класът, който хипотезата h описва, ще бъде този, за когото тя е най-точна.

Тъй като на практика разполагаме с ограничен брой примери, описващи търсения клас, трябва да използваме някакви методи за приближено пресмятане на класификационната точност на разглежданата от нас хипотеза.

Да означим с $n(h)$ броя на всички примери, които се извеждат от дадена хипотеза h: $n(h) = |\{e \in \mathcal{E} | h \to e\}|$, а с $n^i(h), i = 1 \ldots, k$ - броя на примерите, принадлежащи на i-тия клас, които тя извежда: $n^i(h) = |\{e \in \mathcal{E}^i | h \to e\}|$.

Най-простата формула за приближаване на класификационната точност на дадена хипотеза h при даден клас $\mathcal{C}^i$ е

$$A_{rf}(\mathcal{C}^i|h) = \frac{n^i(h)}{n(h)}. \tag{2}$$

При малък брой примери по-добри резултати дава следната формулата за Лапласово приближение [4]

$$A_{Lapl}(\mathcal{C}^i|h) = \frac{n^i(h) + 1}{n(h) + k}, \tag{3}$$

където k е броят на класовете. В същност горната формула е специален случай на формулата за m-приближение:

$$A_{mest}(\mathcal{C}^i|h) = \frac{n^i(h) + p(\mathcal{C}^i)m}{n(h) + m}, \tag{4}$$

където $p(\mathcal{C}^i)$, е вероятността на класа $\mathcal{C}^i$, а m е параметър, който се настройва.

Глава 5

Система за машинно самообучение

В тази глава ще представим една система за машинно самообучение, която се базира на структурата и релацията между атоми, които разгледахме в глава 3. Първо ще дадем дефиниции на основните понятия, свързани с предложената от нас система за ИМС. След това ще представим общ алгоритъм за построяване на обобщаваща полурешетка на атоми и после ще покажем начините за решаване на отделните типове задачи от ИМС, които описахме в предишната глава.

5.1 Постановка на задачата

Като език на хипотезите $\mathcal{L}_H$ ще разглеждаме множеството от всички атоми с фиксиран предикатен символ p и брой на аргументите n, които могат да се построят над дадена азбука за предикатно смятане A. Езикът на примерите $\mathcal{L}_E \subset \mathcal{L}_H$ ще представлява множеството от всички основни атоми от езика $\mathcal{L}_H$. Като релация за изводимост $\rightarrow$ ще използваме частичната наредба между атоми, дефинирана чрез прилагането на субституция, като $p_1 \rightarrow p_2$, ако $p_1 \preceq p_2$.

Така дефинирания от нас език на примерите съвпада с този, който се използва от системите за индуктивно логическо програмиране, а хипотезите в нашата система съответстват на единични клаузи. Те са малко по-изразителни от езика за представяне на правила, тъй като възможността за наличието на една и съща променлива в няколко различни аргумента в съответния атом представлява неявно използване на равенство.

Все пак избрания от нас език на хипотезите има и някои сериозни ограничения. Първото от тях е липсата на отрицание, както на ниво аргументи, така и за цялата хипотеза. Другото ограничение е невъзможността за използване на обекти, които имат различна наредба от поглъщане между термове, като например естествени или реални числа.

Както видяхме в предишната глава, множеството от всички атоми с фиксиран предикатен символ и брой аргументи $\mathcal{L}_H$ образува полурешетка с инфимум спрямо релацията $\preceq$. Тогава в термините на дискретната математика дадено множество от примери $\mathcal{E} \subset \mathcal{L}_E$ представлява антиверига в полурешетката $\mathcal{L}_H$.

Дефиниция 5.1 (Обобщаваща полурешетка на антиверига)
Нека частично нареденото множество $(\mathcal{L}, \preceq)$ е полурешетка с инфимум и множеството $\mathcal{E} \subset \mathcal{L}$ е антиверига в него. Тогава множеството $G \subseteq \mathcal{L}$, ще наричаме обобщаваща полурешетка на $\mathcal{E}$, ако G е полурешетка с инфимум и множеството $\mathcal{E}$ се състои от всички максимални елементи на G.

Обърнете внимание, че G не е подполурешетка на $\mathcal{L}$, тъй като не сме поставили изискването инфимумът на всеки два елемента, принадлежащи на $\mathcal{L}$, също да принадлежи на

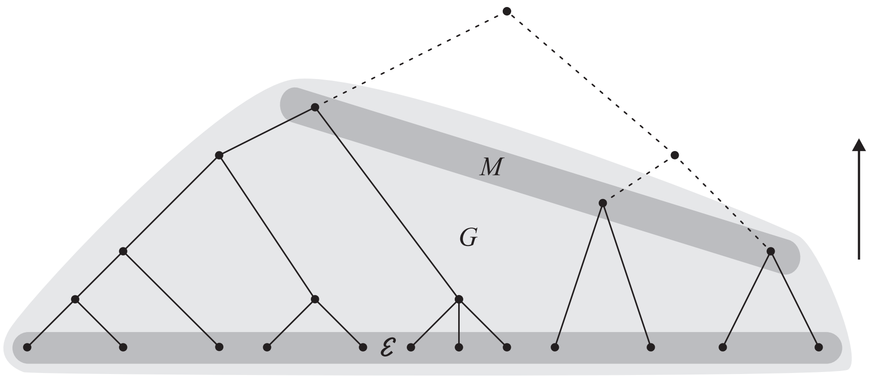
$\mathcal{L}$. Освен това обобщаващата полурешетка не е и дърво, тъй като не сме поставили ограничението супремумът на два елемента, които не принадлежат на $\mathcal{E}$, да не принадлежи на полурешетката.

5.2 Основен алгоритъм

Сега ще разгледаме един основен алгоритъм за построяване на обобщаваща полурешетка, който ще използваме с малки изменения за решаването на различните задачи от ИМС.

Алгоритъм 5.2 (Основен алгоритъм)
Вход: Антиверига $\mathcal{E}$.
Изход: Обобщаваща полурешетка G на E.

1. $G = \mathcal{E}$, $M = G$.
2. Ако в множеството M има един елемент, тогава изход.
3. $n = |M|$.
4. Построяваме списъка $T = \{g_{ij} | i = 2, \ldots, n, j = 1, \ldots, i-1, g_{ij} = \inf\{g_i, g_j\}, g_i, g_j \in M\}$.
5. От списъка T избираме елемента g_{kl} с най-малка оценка $m(g_{kl})$.
6. $UC = \{g_i | g_{kl} \prec g_i, g_i \in M, i = 1, \ldots, n\}$.
7. $M = M \setminus UC$.
8. $M = M \cup \{g_{kl}\}, G = G \cup \{g_{kl}\}$.
9. Отиди на стъпка 2.



Фигура 5.1:

В основния цикъл на алгоритъма 5.2 се използват две множества G и M. Първото от тях представлява частично наредено множество, което е подмножество на крайната обобщаваща полурешетка, а второто съдържа всички негови минимални елементи (фигура 5.1). В началото на алгоритъма тези множества се инициализират с всички елементи от $\mathcal{E}$. На всяка стъпка в списъка T се генерират инфимумите на всички двойки елементи от M, като се избира елементът с най-малка оценка m. После от M се премахват всички по-големи от него елементи и на края избраният елемент се добавя към двете множества G и M. Когато M остане само с един елемент, алгоритъмът завършва и множеството G представлява обобщаваща полурешетка на $\mathcal{E}$. Ясно е, че така предложеният алгоритъм генерира възможно най-специфичните хипотези, извеждащи определено множество от примери.

5.2.1 Оценка на хипотеза

Основната идея на предложения алгоритъм е на всяка стъпка да се правят по възможност най-малки обобщения. Тъй като работим с частично наредено множество, не можем да използваме наредбата между елементите за избиране на най-специфичния от тях, защото не всички са сравними. За това ще използваме някаква функция $m : \mathcal{L} \rightarrow R$ за оценка на общността на хипотезите и която ще се базира на въведената по-рано от нас разлика в полурешетка.

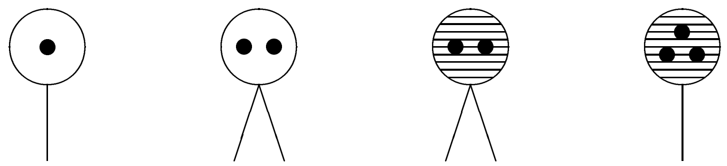
Ако функцията m е строго монотонно намаляваща, можем да направим една оптимизация в алгоритъма. След построяването на всички инфимуми на елементите от M на стъпка 4, можем да премахнем от списъка T всички елементи, които са по-малки от други, тъй като тяхната оценка ще бъде по-голяма от тази на съответния по-голям елемент.

Дефиниция 5.3 (Сложност на хипотеза) С функцията $L(a)$ ще означаваме сложността на елемента $a \in \mathcal{L}$, която се дефинира по следния начин:

$$L(a) = \frac{\sum\limits_{a \preceq x, x \in max(\mathcal{L})} s(a,x)}{n},$$

където с $max(\mathcal{L})$ сме означили множеството от всички максимални елементи на $\mathcal{L}$, а с n техния брой, т.е $n = |max(\mathcal{L})|$.

Тази функция има смисъла на средно аритметично от разликите между дадения елемент и всички максимални елементи, които са по-големи от него. В нашия случай като максимални елементи на множеството $\mathcal{L}$ ще разглеждаме примерите от $\mathcal{E}$.



Фигура 5.2: Едноклетъчни организми

Ще разгледаме един пример за работата на алгоритъма. На фигура 5.2 са дадени четири едноклетъчни животни, които се различават по цвета на тялото (светъл/тъмен), по броя на ядрата (1,2 или 3) и по броя на опашките (1 или 2). За формалното описание на примерите ще използваме атом с предикатен символ *cell* и три аргумента, като ще използваме език без функционални символи. Тогава разликата между два елемента ще бъде дължината на произволна свързана верига между тях. Множеството от примери $\mathcal{E}$ ще съдържа следните атоми: $\mathcal{E} = \{cell(b,1,1), cell(b,2,2), cell(d,2,2), cell(d,3,1)\}$.

1. Инициализация - $G = M = \mathcal{E}$.

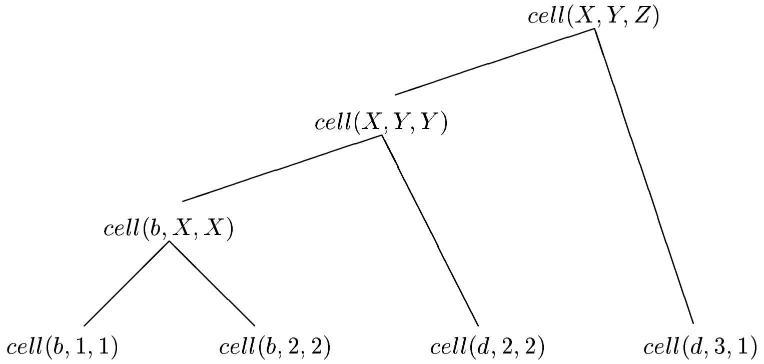
2. Построяваме множеството $T = \{cell(b,X,X) : 1,\ cell(X,Y,Z) : 3,\ cell(X,Y,1) : 2,\ cell(X,2,2) : 1,\ cell(d,X,Y) : 2\}$ от инфимумите на всички елементи от M (след двоеточието сме означили средното разстояние от хипотезата, до всички примери, които тя извежда).

3. Избираме елемента $g = cell(b,X,X)$ с най-малка оценка 1.

4. Построяваме множеството $UC = \{cell(b,1,1), cell(b,2,2)\}$.

5. $M = (M \setminus UC) \cup \{g\} = \{cell(b, X, X), cell(d, 2, 2), cell(d, 3, 1)\}$.
6. $G = G \cup \{g\} = \{cell(b, 1, 1), cell(b, 2, 2), cell(d, 2, 2), cell(d, 3, 1), cell(b, X, X)\}$.
7. Построяваме $T = \{cell(X, Y, Y) : 2, cell(X, Y, Z) : 3\}$.
8. Избираме елемента $g = cell(X, Y, Y)$ с оценка 2.
9. $UC = \{cell(b, X, X), cell(d, 2, 2)\}$.
10. $M = \{cell(X, Y, Y), cell(d, 3, 1)\}$.
11. $G = \{cell(b, 1, 1), cell(b, 2, 2), cell(d, 2, 2), cell(d, 3, 1), cell(b, X, X), cell(X, Y, Y)\}$.
12. Построяваме $T = \{cell(X, Y, Z) : 3\}$.
13. От T избираме единствения елемент $g = cell(X, Y, Z)$.
14. $UC = \{cell(X, Y, Y), cell(d, 3, 1)\}$.
15. $M = \{cell(X, Y, Z)\}$.
16. $G = \{cell(b, 1, 1), cell(b, 2, 2), cell(d, 2, 2), cell(d, 3, 1), cell(b, X, X), cell(X, Y, Y), cell(X, Y, Z)\}$.
17. В множеството M е останал само един елемент и следователно прекратяваме изпълнението на алгоритъма.

На фигура 5.3 е показано множеството G след приключване на работата на алгоритъма, което представлява обобщаваща полурешетка на примерите от множеството $\mathcal{E}$.



Фигура 5.3: Обобщаваща полурешетка на едноклетъчни организми.

5.2.2 Уникалност на хипотеза

По време на работата на алгоритъма след добавянето на новия елемент g_{kl} към множеството G, на стъпка 8 е възможно да се получи така, че даден елемент от M да има повече от едно долно покритие. Това означава, че този елемент, както и всички други, които са по-големи от него се извеждат от повече от една хипотеза.

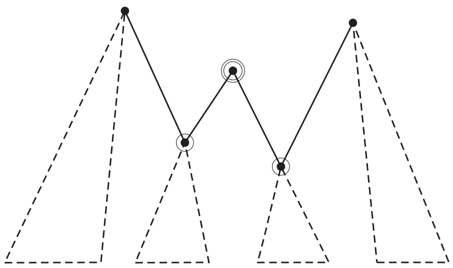
Освен това в случая, когато всички горни покрития на дадена хипотеза имат повече от едно долно покритие, то самата тя не представлява интерес, защото целия клас, който се описва от нея, може да се изведе от други хипотези.

За да решим този проблем, ще въведем понятието за неуникалност на елемент по следния начин:

Дефиниция 5.4 (Неуникалност на елемент)

1. ако даден елемент има повече от едно долно покритие то той, както и всички елементи по-големи от него, са неуникални;

2. ако всички горни покрития на даден елемент са неуникални, то и той е неуникален.



Фигура 5.4: Неуникални хипотези

На фигура 5.4 са показани двата случая за неуникалност на хипотеза, като с една окръжност са оградени хипотезите с повече от едно долно покритие, а с две - тези, за които всички горни покрития са неуникални.

Сега ще представим един рекурсивен алгоритъм за маркиране на елементите на дадено частично наредено множество като неуникални.

Алгоритъм 5.5 (Маркиране за неуникалност)
Вход: Частично наредено множество L.
Изход: Всички неуникални елементи от L са маркирани като такива.

1. Ако множеството L е празно, тогава край.

2. Избираме произволен елемент $a \in L$.

3. Ако a не е маркиран и има повече от едно долно покритие, маркирай a и всички елементи по-големи от него.

4. Ако a не е маркиран и всички негови горни покрития са маркирани, маркирай a.

5. $L' = L \setminus \{a\}$.

6. Изпълни рекурсивно алгоритъма за множеството L'.

Ще променим основния алгоритъм 5.2 по следния начин. В края на цикъла ще добавим следните стъпки:

– Маркират се за неуникалност елементите от множеството G.

– $NU = \{a \in M | a$ е маркиран като неуникален в $G\}$.

– $M = M \setminus NU, G = G \setminus NU$.

По-долу е представен разширен алгоритъм, който включва оптимизация и маркиране за неуникалност.

Алгоритъм 5.6 (Разширен алгоритъм)
Вход: Антиверига $\mathcal{E}$.
Изход: Обобщаваща полурешетка G на E.

1. $G = \mathcal{E}$, $M = G$.

2. Ако в множеството M има един елемент, тогава изход.
3. $n = |M|$.
4. Построяваме списъка $T = \{g_{ij} | i = 2, \dots, n, j = 1, \dots, i-1, g_{ij} = \inf\{g_i, g_j\}, g_i, g_j \in M\}$.
5. От T премахваме всички немаксимални елементи *(незадължително)*.
6. От списъка T избираме елемента g_{kl}, с най-малка оценка $m(g_{kl})$.
7. $UC = \{g_i | g_{kl} \prec g_i, g_i \in M, i = 1, \dots, n\}$.
8. $M = M \setminus UC$.
9. $M = M \cup \{g_{kl}\}, G = G \cup \{g_{kl}\}$.
10. Маркират се за неуникалност елементите от множеството G.
11. $NU = \{a \in M | a$ е маркиран, като неуникален в $G\}$.
12. $M = M \setminus NU, G = G \setminus NU$.
13. Отиди на стъпка 2.

5.3 Приложение на алгоритъма

В този раздел, ще покажем, как предложения алгоритъм за построяване на обобщаващи полурешетки може да се използва за решаване на различните типове задачи от ИМС.

5.3.1 Концептуална клъстеризация

Така предложения в предишния раздел разширен алгоритъм отговаря директно на условията на задачата за концептуална клъстеризация, т.е. построява организирано множество от класове. В нашия случай подредбата между класовете представлява полурешетка с инфимум.

5.3.2 Научаване на фиксирано множество от класове

За този тип задача, резултатът от алгоритъма трябва да бъде множество от хипотези (ще използваме дизюнктивен език), които да извеждат отделните класове.

Когато класификацията на примерите е предварително известна, тогава съществено по време на обобщаващия процес е избираните хипотези да извеждат примери само от един клас. Подходяща функция, която отчита хомогенността на дадена хипотеза h е ентропията, която ни дава броя на битовете, необходими за кодирането на принадлежността на примерите извеждани от дадена хипотеза h към отделните класове:

$$E(h) = -\sum_{i=1}^{n} \frac{n^i(h)}{n(h)} \log_2 \frac{n^i(h)}{n(h)},$$

където n е броя на класовете, които се извеждат от дадената хипотеза. Освен това е необходимо да променим и функцията за сложност L, така че резултатът да бъде средния брой битове необходими за кодирането на разстоянията от хипотезата до примерите, които тя извежда, а не самите разстояния:

$$L(h) = \frac{\sum\limits_{a \preceq x, x \in max(\mathcal{L})} \log_2 s(a, x)}{n}.$$

Тогава оценката m ще бъде равна на

$$m(h) = L(h) + E(h).$$

Като критерий за спиране на обобщаването на дадено множество от хипотези, ще използваме оценката за класификационна точност, въведена в предишната глава. По този начин, ако точността на избраната от нас хипотеза стане по-малка от максималната точност на хипотезите, които я покриват (множеството UC), ще спираме обобщаването на тези елементи и ще ги използваме, като окончателни хипотези.

Алгоритъм 5.7 (Намиране на описание на множество от класове)
Вход: Множество от примери $\mathcal{E}$, с дефинирана принадлежност към клас $\mathcal{C}^i, i = 1, \ldots, k$.
Изход: Множество от хипотези H.

1. $G = \mathcal{E}, M = \mathcal{E}, H = \emptyset$.
2. Ако в множеството M има по-малко от два елемента, тогава $H = H \cup M$ и изход.
3. $n = |M|$.
4. Построяваме списъка $T = \{g_{ij} | i = 2, \ldots, n, j = 1, \ldots, i-1, g_{ij} = \inf\{g_i, g_j\}, g_i, g_j \in M\}$.
5. От T премахваме всички неминимални елементи *(незадължително)*.
6. От списъка T избираме елемента g_{kl}, с най-малка оценка $m(g_{kl})$.
7. $UC = \{g_i | g_{kl} \prec g_i, g_i \in M, i = 1, \ldots, n\}$.
8. Ако $A(g_{kl}) < \max_{g \in UC} A(g)$, тогава $H = H \cup UC$ и отиди на стъпка 2.
9. $M = M \setminus UC$.
10. $M = M \cup \{g_{kl}\}, G = G \cup \{g_{kl}\}$.
11. Маркират се за неуникалност елементите от множеството G.
12. $NU = \{a \in M | a$ е маркиран, като неуникален в $G\}$.
13. $M = M \setminus NU, G = G \setminus NU$.
14. Отиди на стъпка 2.

5.3.3 Обучение от положителни и отрицателни примери

Задачата за обучение от положителни и отрицателни примери може директно да се сведе до задачата за научаване на фиксирано множество от класове, като положителните и отрицателните примери се разгледат като два отделни класа.

Ако описанието на класа на отрицателните примери е по-просто от това на положителните е възможно след приключването на алгоритъма да проверим кой от двата класа има по-просто описание (например съдържа по-малко дизюнкти). Ако това е класа на отрицателните примери, можем да използваме непринадлежността на даден елемент към него, за да го окачествим като положителен. За тази цел е необходимо и да разширим езика на хипотезите, като позволим използването на отрицание на хипотези в него.

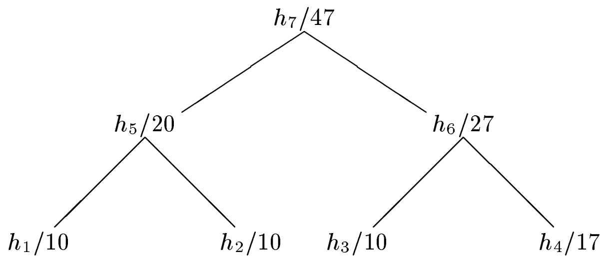
Глава 6

Експерименти и резултати

В тази глава ще разгледаме резултатите от използването на предложената система за машинно самообучение. Във всичките задачи, които ще разглеждаме, примерите и хипотезите ще се описват на език без функционални символи. Част от базите данни са взети от [11].

6.1 Концептуална клъстеризация

За този експеримент ще изпозлваме база данни, която се състои от описание на заболяванията по соевите разстения. Първата публикация, в която е използвана тази област е на Михалски [12]. В действителност ще използваме само част от оригиналната база данни, тъй като в нея много от атрибутите имат липсващи стойности и съществуват няколко класа, за които има по шест или по-малко примера. Самите примери се състоят от 21 номинални атрибута, описващи състоянието на растението, като листа, корени, семена, растеж и др. Броя на различните заболявания е четири с разпределение на примерите 10, 10, 10 и 17 (тогава общия борй на примерите е 47). За всеки атрибут множеството от константи, които ще използваме за описване на различните му стойности, ще бъде различно от множествата от константи описващи останалите атрибути, тъй като те описват различни характеристики на растенията.



Фигура 6.1:

Ще използваме разширения алгоритъм 5.6, като за примерите няма да бъде указана принадлежността им към съответния вид заболяване. Тогава резултатът от алгоритъма ще бъде обобщаваща полурешетка, като ще очакваме примерите от един клас да се групират заедно. На фигура 6.1 са показани най-общите хипотези от множеството G (след чертата е даден броя на примерите, които се извеждат от съответната хипотеза). Оказва се, че всяка от хипотезите h_1, h_2, h_3 и h_4 описва коректно всичките примери от съответния клас и нито един пример от другите.

6.2 Обучение на множество от класове

За този тип задача ще използваме една база данни създадена от Ричард Форсайт [11], която се състои от описания на 100 животни, които принадлежат на седем различни класа: бозайници, птици, влечуги, риби, земноводни, насекоми и прешленести. Всяко животно се описва с 12 двоични атрибута: козина, пера, яйца, мляко, лети, водно, хищник, зъби, гръбначно, диша, перки, опашка и един числов - брой на краката. От оригиналните атрибути са махнати три - отровно, домашно и размер, които според мен не са подходящи за класификация на животните.

След изпълнението на алгоритъма се получават следните хипотези, за описване на съответните класове:

1. *Клас бозайници:*
   $h_1/13$ = животно(A, няма_пера, B, има_мляко, C, D, E, F, гръбначно, диша, G, H, I).

2. *Клас птици:*
   $h_2/7$ = животно(няма_козина, има_пера, има_яйца, няма_мляко, A, B, C, няма_зъби, гръбначно, диша, няма_перки, 2_крака, има_опашка).

3. *Клас влечуги:*
   $h_3/3$ = животно(няма_козина, няма_пера, има_яйца, няма_мляко, не_лети, водно, A, B, гръбначно, диша, няма_перки, C, има_опашка),
   $h_4/1$ = животно(няма_козина, няма_пера, няма_яйца, няма_мляко, не_лети, водно, хищник, има_зъби, гръбначно, диша, няма_перки, 0_крака, има_опашка).

4. *Клас риби:*
   $h_5/2$ = животно(няма_козина, няма_пера, има_яйца, няма_мляко, не_лети, водно, A, има_зъби, гръбначно, не_диша, има_перки, 0_крака, има_опашка).

5. *Клас земноводни:*
   $h_6/3$ = животно(няма_козина, няма_пера, има_яйца, няма_мляко, не_лети, водно, A, има_зъби, гръбначно, диша, няма_перки, 4_крака, B).

6. *Клас насекоми:*
   $h_7/4$ = животно(A, няма_пера, има_яйца, няма_мляко, B, не_водно, C, няма_зъби, безгръбначно, диша, няма_перки, 6_крака, няма_опашка).

7. *Клас прешленести:*
   $h_8/7$ = животно(няма_козина, няма_пера, A, няма_мляко, не_лети, B, хищник, няма_зъби, безгръбначно, C, няма_перки, D, E),
   $h_9/2$ = животно(няма_козина, няма_пера, има_яйца, няма_мляко, не_лети, не_водно, A, няма_зъби, безгръбначно, B, няма_перки, 0_крака, няма_опашка).

При класа на прешленестите се окзава, че хипотезата h_9 описва мекотелите, а h_8 останалите животни от този клас.

6.3 Обучение от положителни и отрицателни примери

За този тип самообучение ще използваме задачите за монасите. Това е една изкуствена област създадена от Себастиан Трън [18], като целта е била да се сравнят качествата на съществуващите алгоритми за машинно самообучение, като тестовете на различните системи най-често са правени от техните автори.

Обектите от областта представляват монаси, които се описват от следните атрибути: форма на главата (квадрат, овал, осмоъгълник), форма на тялото (същите стойности, като форма на главата), усмихва се (да, не), държи (сабя, знаме, балон), цвят на дрехата (червен, син, зелен, жълт) и има ли връзка (да, не). Тогава общия брой обекти е 432. В тази област са дефинирани три различни задачи, които разделят по различен начин множеството от обекти на положителни и отрицателни, като за всяка задача са дадени две множества от обучаваща (124 примера) и тестова (308 примера) извадки.

Правилото, което описва първата задача е: *формата на главата е равна на формата на тялото или цвета на дрехата е червен.* Нашата система успява да изведе същите две правила, които имат следния вид:

$h_1 =$ монах(A, A, B, C, D, E),

$h_2 =$ монах(A, B, C, D, червен, E).

Обърнете внимание, че в хипотезата h_1, използването на равенство за формата на тялото и главата е представено неявно, чрез една и съща променлива A на мястото на първите два атрибута.

Другите две задачи за монасите не могат да се решат от нашата система, поради използването на отрицание на ниво атрибут при описанието на класовете.

Глава 7

Заключение

В тази глава ще посочим някои недостатъци на предложената от нас система за машинно самообучение, както и насоки за бъдещи изследвания.

В глава 3 разгледахме няколко начина за дефиниране на псевдометрика между различни структури, определени от термове и атоми, а в раздел 3.3 показахме някои от свойствата на алгебричната структура, определена от клаузи. Интересен е разбира се въпросът за дефинирането на псевдометрика между клаузи спрямо релацията θ-поглъщане или спрямо друга, по-ограничена релация. В случая с релацията $\preceq_\theta$ това трябва да стане чрез определянето на функция *височниа* на клауза, тъй като показахме съществуването на интервали с безкрайна дължина (пример 3.18).

Дефинирането на псевдометрика между клаузи ще ни позволи да разширим използвания от нас език на хипотези (който в момента представлява единични клаузи) и който въпреки че е по-изразителен от езика атрибут-стойност, както видяхме е доста ограничен.

Друга възможна област за изследване е намирането и сравняването на различни функции за сложност на хипотеза от тази, която се използва в момента от системата (дефиниция 5.3).

За прилагането на системата за ИМС за решаване на задачата за самообучение от положителни примери е необходимо намирането на някакъв критерий за спиране на обобщаването, но без да се използва принадлежността на примерите към различни класове, тъй като в този случай всички те принадлежат само на един.

Много често в системите за машинно самообучение се разглежда допълнителен компонент - *основни знания*. В настоящата работа не сме разгледали тази тема, тъй като съществува стандартен подход за добавяне на основни знания към пропозиционални системи за самообучение [9], посредством разширяване на примерите с допълнителни атрибути. Освен това използването на клаузи, като език език за хипотези, също би решил този въпрос.

Приложение А

Основни понятия от дискретната математика и алгебрата

В това приложение ще представим някои основни дефиниции и твърдения от областта на дискретната математика и алгебрата.

Дефиниция А.1 (Псевдометрика, Метрика) Функцията $d : A \times A \to R$, ще наричаме *псевдометрика* в множеството A, ако са изпълнени следните условия:

1. $d(a, b) \geq 0$, за всеки два елемента $a, b \in A$;
2. $d(a, a) = 0$, за всеки елемент $a \in A$;
3. $d(a, b) = d(b, a)$, за всеки два елемента $a, b \in A$ (симетричност);
4. $d(a, b) + d(b, c) \geq d(a, c)$, за всеки три елемента $a, b, c \in A$ (неравенство на триъгълника).

Ако е изпълнено и условието

$$d(a, b) = 0 \Rightarrow a = b,$$

тогава функцията d се нарича метрика.

А.1 Релации

Дефиниция А.2 (Релация) *Двуместна релация* или просто *релация* над дадено множество A, ще наричаме всяко подмножество $R \subseteq A \times A$ на декартовото произведение $A \times A$. За по-кратко вместо $(a, b) \in R$ че записваме aRb.

Дефиниция А.3 (Типове релации) Нека $R \subseteq A \times A$. Казваме, че релацията R е:

- *рефлексивна*, ако $\forall a \in A(aRa)$;
- *симетрична*, ако $\forall a, b \in A(aRb, bRa)$;
- *транзитивна*, ако $\forall a, b, c \in A(aRb, bRc \Rightarrow aRc)$;
- *антирефлексивна*, ако $\neg\exists a \in A(aRa)$;
- *антисиметрична*, ако $\forall a, b \in A, a \neq b\ (aRb \Rightarrow \neg bRa)$ или $\forall a, b \in A\ (aRb, bRa \Rightarrow a = b)$.

Ако R е рефлексивна и транзитивна, казваме че тя е *полунаредба* над множеството A. Ако R е рефлексивна, транзитивна и антисиметрична, казваме че тя е *частична наредба* над множеството A. Ако R е рефлексивна, транзитивна и симетрична, казваме че тя е *релация на еквивалентност* над множеството A.

Дефиниция А.4 (Клас на еквивалентност) Нека R е релация на еквивалентност над A. *Класа на еквивалентност* за всеки елемент $a \in A$ ще отбелязваме с $[a]$ и е равен на множеството от всички елементи от A еквивалентни на a спрямо R, т.е. $[a] = \{b | b \in A, aRb\}$.

Твърдение А.5 Класовете на еквивалентност $[]$ над дадено множество A, са непразни, непресичащи се и покриват цялото множество.

Една полунаредба R над множеството A индуцира релация на еквивалентност $\sim$ над A по следния начин: $a \sim b$ за $a, b \in A$, ако aRb и bRa. Използвайки тази релация на еквивалентност, една полунаредба R над A индуцира частична наредба R' над класовете на еквивалентност $[]$ в A по следния начин: $[a]R'[b]$, ако aRb. Използвайки това свойство, когато ни е необходима частична наредба, а разполагаме само с полунаредба R, ще считаме, че използваме частичната наредба индуцирана над класовете на еквивалентност $[]$ индуцирани от R.

От тук нататък релацията частична наредба ще бележим със символа $\preceq$, като $a \preceq b$, ако aRb. Ако $a \preceq b$ и $a \neq b$, тогава ще пишем $a \prec b$. Освен това релацията $a \preceq b$ може да се записва и като $b \succeq a$, а $a \prec b$, като $b \succ a$.

Дефиниция А.6 (Частично наредено множество) Под *частично наредено множество* ще разбираме множество A с дефинирана в него частична наредба $\preceq$ и ще означаваме с $(A, \preceq)$.

Дефиниция А.7 (Покритие) Под *a покрива b* в частично наредено множество $(A, \preceq)$ се разбира, че $a \succ b$ и не съществува $c \in A$, за което $a \succ c \succ b$. Ако a покрива b, тогава a ще наричаме *горно покритие* на b, а b ще наричаме *долно покритие* на a.

Дефиниция А.8 (Верига, Свързана верига, Антиверига) Под *верига* с дължина n в частично наредено множество $(A, \preceq)$, ще разбираме редица от елементи $x_0, x_1, \ldots, x_n$, в която $x_{i-1} \prec x_i$, за $i = 1, \ldots, n$. Под *свързана верига* с дължина n, ще разбираме редица от елементи $x_0, x_1, \ldots, x_n$, в която всяко x_i покрива x_{i-1}, за $i = 1, \ldots, n$. Множеството $S \subseteq A$ се нарича *антиверига*, ако нито една двойка елементи от него не са сравними.

Дължината $l(A)$ на едно частично наредено множество е точната горна граница на дължините на веригите в него. Ако тя е крайна, тогава ще казваме, че A е *множество с крайна дължина*.

Дефиниция А.9 (Минимален и максимален елемент)
Нека е дадено частично нареденo множеството $(A, \preceq)$. Елементът $a \in A$ ще наричаме *минимален*, ако не съществува $b \in A, b \neq a$, такъв че $b \preceq a$. Елементът $c \in A$ ще наричаме *максимален*, ако не съществува $b \in A, b \neq c$, такъв че $c \preceq b$.

Дефиниция А.10 (Долна и горна граница) Нека е дадено частично наредено множеството $(A, \preceq)$ и $B \subseteq A$. Елементът $a \in A$ ще наричаме *долна граница* на множеството B, ако за всяко $b \in B, a \preceq b$. Елементът $c \in A$ ще наричаме *горна граница* на множеството B, ако за всяко $b \in B, b \preceq c$.

Дефиниция А.11 (Инфимум и супремум) Нека е дадено частично наредено множеството $(A, \preceq)$ и $B \subseteq A$. Долната граница a' на B наричаме *инфимум* и бележим с $\inf B$, ако $a \preceq a'$ за всяка долна граница a на B. Горната граница c' на B наричаме *супремум* и бележим с $\sup B$, ако $c \preceq c'$ за всяка горна граница c на B. Очевидно е, че ако B има инфимум или супремум, то те са единствени. Ако цялото множество A има инфимум или супремум, то те се бележат съответно с O и I и се наричат универсални граници.

A.2 Решетки

По-голямата част от дефинициите и твърденията в този раздел са взети от Бирхоф [2].

Дефиниция A.12 (Операция) Нека A е произволно множество. Функцията $\varphi : A^n \to A$ наричаме *n-местна операция*. Ние ще използваме само двуместни операции от вида $\varphi_2 : A^2 \to A$, които вместо $\varphi_1(a,b)$, ще записваме, като $a\varphi_2 b$.

Дефиниция A.13 (Типове операции) Нека A е множество, а φ двуместна операция. Казваме, че φ е:

1. *асоциативна*, ако $\forall a,b,c \in A(a\varphi(b\varphi c) = (a\varphi b)\varphi c)$;

2. *комутативна*, ако $\forall a,b \in A(a\varphi b = b\varphi a)$;

3. *идемпотентна*, ако $\forall a \in A(a\varphi a)$;

4. ако ψ е двуместна операция и $\forall a,b \in A(a\varphi(a\psi b) = a$ и $a\psi(a\varphi b) =$, тогава казваме, че φ и ψ са *поглъщащи*.

Дефиниция A.14 (Решетка) Множество L с две двуместни операции $\vee$ (съединение) и $\wedge$ (сечение), които са асоциативни, комутативни, идемпотентни и поглъщащи наричаме *решетка*.

Дефиниция A.15 (Полурешетка) Множество L с една двуместна операция, която е асоциативна, комутативна и идемпотентна наричаме *полурешетка*.

Ясно е, че ако $(A,\vee,\wedge)$ е решетка, то $(A,\vee)$ е полурешетка със супремум, а $(A,\wedge)$ е полурешетка с инфимум. Следователно по-нататък навсякъде, където разглеждаме свойства на полурешетки, ще смятаме, че те са валидни и за решетки.

Следващото твърдение ни дава възможност да излпозваме алтернативни дефиниции на решетка и полурешетка.

Твърдение A.16 Нека $(L,\preceq)$ е частично наредено множество, в което всяко двуелементно подмножество има супремум и инфимум. Тогава множеството L с двуместните операции $a \vee b = \sup\{a,b\}$ и $a \wedge b = \inf\{a,b\}$ е *решетка*.

Съответно, ако всяко двуелементно подмножество на едно частично наредено множество $(L,\preceq)$ има супремум или инфимум, тогава множеството L с двуместната операция $a \vee b = \sup\{a,b\}$ наричаме полурешетка със супремум, а множеството L с двуместната операция $a \wedge b = \inf\{a,b\}$, полурешетка с инфимум.

Дефиниция A.17 (Подрешетка) Едно подмножество S на L е подрешетка на L, тогава и само тогава, когато S е затворено спрямо операциите $\vee$ и $\wedge$.

Дефиниция A.18 (Интервал) Нека е дадено частично наредено множество $(A,\preceq)$ и елементите $a,b \in A$ са такива, че $a \prec b$. *Интервал* определен от a и b се нарича подмножеството $[a,b] = \{x \in A | a \preceq x \preceq b\}$ на A. Ако множеството A е решетка, то всеки интервал в него ще бъде подрешетка.

Дефиниция A.19 (Модуларна решетка) Една решетка е *модуларна*, когато удовлетворява следното модуларно тъждество: ако $x \preceq z$, то:

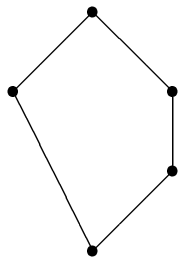
$$x \vee (y \wedge z) = (x \vee y) \wedge z.$$

Твърдение A.20 За всяка модуларна решетка са изпълнени следните две условия за покриване:

(ξ) Ако за всеки от различните елементи a и b, съществува елемент c, който се покрива от тях, то $a \vee b$ покрива както a, така и b;

(ξ') Ако за всеки от различните елементи a и b, съществува елемент c, който ги покрива, то и a и b покриват $a \wedge b$.

Дефиниция A.21 (Полумодуларна решетка) Чрез условията от горното твърдение се дефинират полумодуларни решетки по следния начин. Всяка решетка с крайна дължина, в която е изпълнено условието (ξ) се нарича полумодуларна отгоре, а решетка с крайна дължина, в която е изпълнено условието (ξ') се нарича полумодуларна отдолу.



Фигура A.1: N_5

Твърдение A.22 Една решетка е модуларна тогава и само тогава, когато не съдържа решетката N_5 от фигура A.1 като подрешетка.

Дефиниция A.23 (Полумодуларно множество) Нека $(A, \preceq)$ е частично наредено множество с крайна дължина и най-малък елемент O. Ще казваме, че A е *полумодуларно отгоре*, ако е изпълнено следното условие:

(σ) ако всеки от различните елементи a и b покрива c, то съществува елемент $d \in A$, такъв че покрива a и b.

Полумодуларно отдолу частично наредено множество се дефинира по аналогичен начин:

(σ') ако c покрива всеки от различните елементи a и b, то съществува елемент $d \in A$, такъв че a и b покриват d.

Твърдение A.24 Ако частично нареденото множество $(A, \preceq)$ е полумодуларно, то всички свързани вериги, които имат едни и същи краища имат еднаква дължина.

Твърдение A.25 Нека L да бъде решетка с крайна дължина. Тогава следните условия са еквивалентни:

1. L е модуларна решетка;
2. L е полумодуларно отгоре и отдолу частично наредено множество;
3. всички свързани вериги, които имат едни и същи краища имат еднаква дължина.

Приложение Б

Основи на логическото програмиране

В това изложение са представени основите на логическото програмиране (ЛП). За по-подробно запознаване с теорията на ЛП се обърнете към [10].

Логическото програмиране се базира на предикатното смятане от първи ред.

Б.1 Синтаксис

Дефиниция Б.1 (Азбука) Азбуката на предикатното смятане от първи ред се състои от следните множества от букви:

1. променливи (изброимо много)
2. функционални символи, всеки с местност ≥ 0
3. предикатни символи, всеки с местност ≥ 0
4. логически съюзи
5. квантори
6. символи за пунктуация

Функционалните символи с 0 аргумента се наричат константи и се отбелязват с a вместо с $a()$. Множествата 4, 5 и 6 са едни и същи за всяка азбука, докато множествата 2 и 3 могат да варират. За произволна азбука, само множеството 2 може да бъде празно. Логическите съюзи са: $\neg$ (*отрицание*), $\wedge$ (*конюнкция*), $\vee$ (*дизюнкция*) и $\rightarrow$ (*импликация*, записва се и като $\leftarrow$), кванторите - $\exists$ (*квантор за съществуване*) и $\forall$ (*квантор за всеобщност*), а символите за пунктуация - (,) и ,.

В настоящата работа, променливите се означават с малките букви u, v, w, x, y, z евентуално с индекси или с големите букви $A, B, C, \ldots$, константите със символа c, функционалните символи с f, g, а термовете с буквите a, b, c, d, s, t, евентуално с индекси.

Дефиниция Б.2 (Терм) Терм се дефинира индуктивно по следния начин:

- всяка променлива е терм;
- ако f е n-местен функционален символ ($n \geq 0$) и $t_1, \ldots, t_n$ са термове, тогава $f(t_1, \ldots, t_n)$ е терм.

Дефиниция Б.3 (Основен терм) *Основен терм* е терм, който не съдържа променливи.

Дефиниция Б.4 (Формула) Формула се дефинира индуктивно по следния начин:

- ако p е n-местен предикат и $t_1, \ldots, t_n$ са термове, тогава $p(t_1, \ldots, t_n)$ е формула (наречена *атомарна формула* или по-просто *атом*);

- ако F и G са формули, тогава $(\neg F)$, $(F \wedge G)$, $(F \vee G)$ и $(F \to G)$ са формули;

- ако F е формула и x променлива, тогава $(\exists x F)$ и $(\forall x F)$ са формули.

Дефиниция Б.5 (Затворена формула) *Затворена формула* е формула, в която няма свободни променливи.

Дефиниция Б.6 (Литерал) *Литералът* е атом или отрицанието на атом. *Положителен литерал* е просто атом, а *отрицателен литерал* е отрицанието на атом.

Дефиниция Б.7 (Клауза) *Клауза* е универсално квантифицирана дизюнкция от литерали. Клауза, която има не повече от един положителен литерал, се нарича клауза на Хорн. Клауза с един положителен литерал, се нарича *програмна клауза*, а клауза без положителни литерали - *целева клауза*. Клаузите обикновено се означават като множество от литерали.

Нека $\{A, B_1, \ldots, B_n\}$ е клауза на Хорн, където A е положителен, а $B_i, i = 1, \ldots, n$ са отрицателни литерали. Тогава литералът A се нарича *глава* на клаузата, а литералите $B_1, \ldots, B_n$-*тяло* на клаузата.

Б.2 Субституции и унификация

Дефиниция Б.8 (Израз) *Израз* е терм, литерал или крайно множество от литерали. *Прост израз* е терм или литерал.

Дефиниция Б.9 (Субституция) Субституция θ е крайно множество от вида $\{v_1/t_1, \ldots, v_n/t_n\}$, където v_i е променлива, t_i е терм различен от v_i и променливите $v_1, \ldots, v_n$ са различни. Всеки елемент v_i/t_i се нарича свързване на променливата v_i.

Дефиниция Б.10 (Празна субституция) Субституцията, определена от празно множество от свързвания, ще наричаме *празна субституция* и ще я бележим с ε. Обърнете внимание, че $E\varepsilon = E$, за всеки израз E.

Дефиниция Б.11 (Прилагане на субституция)
Нека $\theta = \{v_1/t_1, \ldots, v_n/t_n\}$ е субституция и E е израз. Резултатът от прилагането на θ върху E се означава с $E\theta$ и се нарича *екземпляр*. Той се получава, като едновременно се заместят всички срещания на променливата v_i в E с терма t_i $(i = 1, \ldots, n)$.

Ако $S = \{E_1, \ldots, E_n\}$ е крайно множество от изрази и θ е субституция, тогава $E\theta$ е множеството $\{E_1\theta, \ldots, E_n\theta\}$.

Дефиниция Б.12 (Композиция на субституции)
Нека $\theta = \{u_1/s_1, \ldots, u_m/s_m\}$ и $\sigma = \{v_1/t_1, \ldots, v_n/t_n\}$ са субституции. *Композицията* $\theta\sigma$ на θ и σ се дефинира с множеството

$$\{u_1/s_1\sigma, \ldots, u_m/s_m\sigma, v_1/t_1, \ldots, v_n/t_n\}$$

от което са премахнати всички свързвания $u_i/s_i\sigma$, за които $u_i = s_i\sigma$, и всички v_j/t_j, за които $v_j \in \{u_1, \ldots, u_m\}$.

Следните елементарни свойства на субституциите прилагам без доказателство:

Твърдение Б.13 Нека θ, σ и ρ са субституции. Тогава:

1. $\theta\varepsilon = \varepsilon\theta = \theta$.

2. $(E\theta)\sigma = E(\theta\sigma)$, за всеки израз E.

3. $(\theta\sigma)\rho = \theta(\sigma\rho)$.

Дефиниция Б.14 (Преименуваща субституция) Нека E е израз и V е множеството от всички променливи, които се срещат в E. *Преименуваща субституция* за E е субституция $\{x_1/y_1, \ldots, x_n/y_n\}$, където $\{x_1, \ldots, x_n\} \subseteq V$, y_i са различни променливи и $(V \setminus \{x_1, \ldots, x_n\}) \cap \{y_1, \ldots, y_n\} = \emptyset$.

Дефиниция Б.15 (Варианти) Нека E и F са изрази. Казваме, че E и F са *варианти*, ако съществуват субституции θ и σ такива, че $E\theta = F$ и $F\sigma = E$. Отношението *варианти* между изрази е релация на еквивалентност, тъй като тя е рефлексивна, симетрична и транзитивна.

Лесно се доказва следното твърдение. Нека E и F са изрази, които са варианти. Тогава съществуват субституции θ и σ, такива че $E\theta = F$ и $F\sigma = E$, където θ е преименуваща субситуция за E, а σ е преименуваща субституция за F.

Дефиниция Б.16 (Унификатор) Нека S е крайно множество от изрази. Субституцията θ се нарича *унификатор*, ако $S\theta$ се състои само от един елемент. Унификатора θ се нарича *най-общ унификатор* (НОУ) за множеството S, ако за всеки унификатор σ за S съществува субституция ρ, такава че $\sigma = \theta\rho$.

От дефиницията следва директно, че ако θ и σ са най-общи унификатори на $\{E_1, \ldots, E_n\}$ тогава $E_1\theta$ и $E_1\sigma$ са варианти.

Алгоритъм Б.17 (Намиране на НОУ) Като вход на алгоритъма се подават два терма a и b. Ако двата терма не са унифицируеми, алгоритъмът завършва и съобщава това. В противен случай, резултатът е субституция θ, която е НОУ за термовете a и b и терм c, такъв че $c = a\theta = b\theta$.

1. Нека $a' = a$ и $b' = b$ и $\theta = \varepsilon$.

2. Ако $a' = b'$ тогава изход, като резултатът е термът $c = a' = b'$ и субституцията θ.

3. Нека k е най-лявата позицията, в която a' и b' се различават и нека s и t са термовете, които започват от тази позиция съответно в a' и b'.

4. Ако нито един от термовете a' и b' не е променлива, или единият от тях е променлива, която се съдържа в другия, тогава термовете не са унифицируеми и изход.

5. Нека термът s е променлива. Тогава $\varphi = \{s/t\}, a' = a'\varphi, b' = b'\varphi, \theta = \theta\varphi$.

6. Отиди на стъпка 2.

Твърдение Б.18 (Теорема за унификацията) Ако два терма a и b са унифицируеми, то алгоритъмът Б.17 винаги завършва на стъпка 2 и субституцията θ ще бъде НОУ за термовете a и b.

Б.3 Семантика

Дефиниция Б.19 (Интерпретация) *Интерпретация* на език за предикатно смятане от първи ред L се състои от:

1. Непразно множество D, което се нарича област на интерпретацията.

2. Смислова функция, която:

   – на всяка константа от L съпоставя елемент от D;

– на всяка n-местна функция от L съпоставя изображение от D^n в D;
– за всеки n-местен предикат от L, съпоставя изображение от D^n в $\{true, false\}$ (или релация в D^n).

Дефиниция Б.20 (Свързване на променливи) Нека I е интерпретация на език за предикатно смятане от първи ред L. *Свързване на променливите (спрямо I)* е съпоставяне на всяка променлива от L елемент от областта на I.

Дефиниция Б.21 (Стойност на термове) Нека I е интерпретация с област D и нека A е свързване на променливите. *Стойност на термовете (спрямо I и A)* се дефинира по следния начин:

1. На всяка променлива се дава свързването и́ спрямо A.

2. На всяка константа се дава свързването и́ спрямо I.

3. Ако $t'_1, \ldots, t'_n$ са съпоставянията на термовете $t_1, \ldots, t_n$ и f' е съпоставянето на f, тогава $f'(t'_1, \ldots, t'_n) \in D$ е съпоставянето на терма $f(t_1, \ldots, t_n)$.

Дефиниция Б.22 (Стойност на формула) Нека I е интерпретация с област D и нека A е свързване на променливите. Тогава на всяка формула от L може да се даде стойност лъжа или истина (спрямо I и A) по следния начин:

1. Ако формулата е атом от вида $p(t_1, \ldots, t_n)$, тогава стойността и се получава, като се изчисли стойността на $p'(t'_1, \ldots, t'_n)$, където p' е изображението съпоставено на p от I и $t'_1, \ldots, t'_n$ са стойностите на термовете $t_1, \ldots, t_n$, спрямо I и A.

2. Ако формулата има вида $\neg F, F \wedge G, F \vee G, F \rightarrow G$, тогава стойността на формулата се дава от следната таблица:

F	G	$\neg F$	$F \wedge G$	$F \vee G$	$F \rightarrow G$
лъжа	лъжа	истина	лъжа	лъжа	истина
лъжа	истина	истина	лъжа	истина	истина
истина	льжа	лъжа	лъжа	истина	лъжа
истина	истина	льжа	истина	истина	истина

3. Ако формулата има вида $\exists x F$, тогава стойността и́ е истина, ако съществува елемент $d \in D$, такъв че F има стойност истина спрямо I и $A(x/d)$, където с $A(x/d)$ е означено свързване на променливите A, в което x има стойност d. В противен случай формулата има стойност лъжа.

4. Ако формулата има вида $\forall x F$, тогава стойността и́ е истина, ако за всеки елемент $d \in D$, F има стойност истина спрямо I и $A(x/d)$, В противен случай формулата има стойност лъжа.

От горната дефиниция е ясно, че стойността на затворена формула не зависи от свързването на променливите в нея. Следователно можем еднозначно да говорим за стойност на затворена формула спрямо интерпретация.

Дефиниция Б.23 (Модел) Нека I е интерпретация и F е затворена формула. Ще казваме, че I е *модел* на F, ако стойността на F спрямо I е истина.

Нека S е множество от затворени формули. Ще казваме, че I е модел на S, ако I е модел на всяка формула от S.

Дефиниция Б.24 (Логическо следване) Нека S е множество от затворени формули и F е затворена формула. Ще казваме, че F е *логическо следствие* на S и ще записваме $S \models F$, ако всеки модел на S е модел и на F.

Лесно се проверява, че логическото следване е полунаредба между формули.

Библиография

[1] Здравко Марков. *Индуктивни методи за машинно самообучение.* СОФТЕХ, 1996.

[2] G. Birkhoff. *Lattice Theory.* American Mathematical Society, 1967.

[3] G. Bisson. Conceptual clustering in a first order logic representation. In B. Neumann, editor, *ECAI-92*, pages 458–462. John Wiley & Sons, 1992.

[4] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In Y. Kodratoff, editor, *Machine Learning - Proceedings of the Fifth European Conference*, pages 151–163. Springer-Verlag, 1991.

[5] P. Clark and T. Niblett. Induction in noisy domains. In I. Bratko and N. Lavrač, editors, *Progress in Machine Learning (Proceedings of the 2nd European Working Session on Learning*, pages 11–30. Sigma, 1987.

[6] W. Emde and D. Wettschereck. Relational instance-based learning. In L. Saitta, editor, *ICML-96*, pages 122–130. Morgan Kaufmann, 1996.

[7] J. H. Gennari, P. Langley, and D. Fisher. Model of incremental concept formation. In J. G. Carbonell, editor, *Machine Learinng: paradigms and methods.* MIT Press, 1990.

[8] A. Hutchinson. Metrics on terms and clauses. In M. van Someren and G. Widmer, editors, *Machine Learning: ECML-97*, volume 1224 of *Lecture Notes in Artificial Intelligence*, pages 138–145. Springer-Verlag, 1997.

[9] N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications.* Artificial Intelligence. Ellis Horwood, 1994.

[10] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1984.

[11] C. J. Merz and P. M. Murphy. UCI Repository of machine learning databases. [http://www.ics.uci.edu/˜mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[12] R. Michalski and R. Chilausky. Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2), 1980.

[13] S. Muggleton and C. Feng. Efficient induction of logic programs. In S. Muggleton, editor, *Inductive Logic Programming*, The APIC Series. Academic Press, 1992.

[14] S. Muggleton, A. Srinivasan, and M. Bain. Compression, significance and accuracy. In D. Sleeman and P. Edwards, editors, *Proceedings of the Ninth International Conference on Machine Learning*, pages 338–347. Morgan Kaufmann, 1992.

[15] S.-H. Nienhuys-Cheng and R. de Wolf. Least generalizations and great specializations of sets of clauses. *Journal of Artificial Intelligence Research*, 4:341–363, 1996.

[16] G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

[17] J. C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence*, 5:135–153, 1970.

[18] S. B. Thrun et al. The MONK's problems - a performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, Dec. 1991.

[19] P. R. J. van der Laag. *An Analysis of Refinement Operators in Inductive Logic Programming.* PhD thesis, Tinbergen Institute Research, 1995.