4. Strojové učení

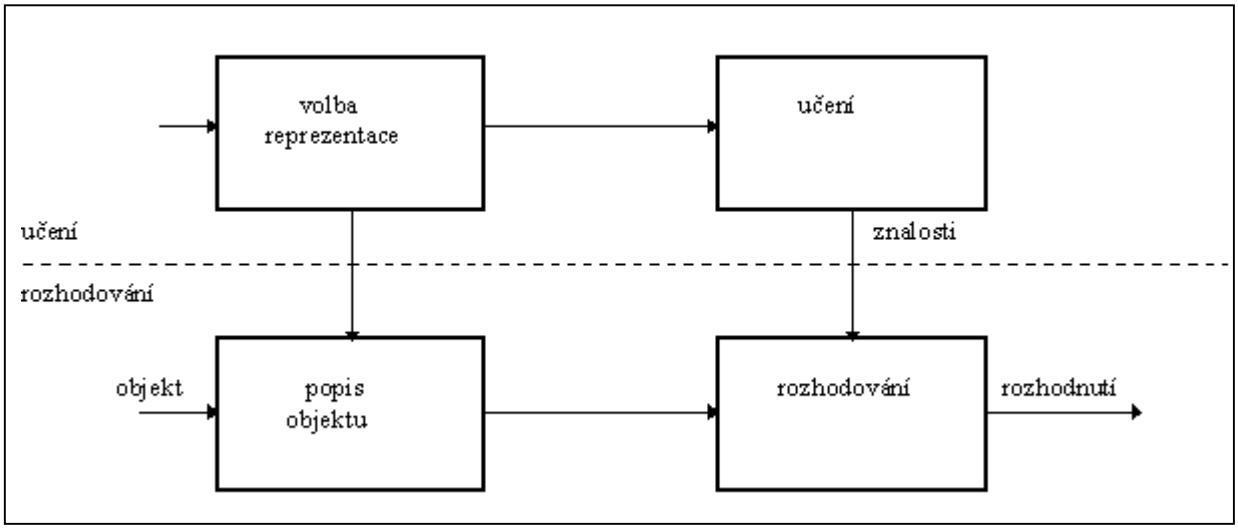
4.1 Základní pojmy

Důležitou vlastností živých organismů je schopnost přizpůsobovat se měnícím se podmínkám (adaptovat se), eventuálně se učit na základě vlastních zkušeností. Schopnost učit se bývá někdy dokonce považována za definici inteligence. Je proto přirozené, že vybavit touto vlastností i systémy technické je jedním z cílů umělé inteligence. Navíc v řadě praktických případů, kdy není dostatek apriorních znalostí o řešeném problému, ani jinak postupovat nelze.

Prvky učení můžeme pod různými názvy nalézt v řadě vědních disciplin; ve statistice se objevují explorační analýza dat (exploratory data analysis) nebo inteligentní analýza dat (intelligent data analysis), v umělé inteligenci se hovoří o metodách rozpoznávání obrazů (pattern recognition), strojového učení (machine learning) nebo automatizovaného získávání znalostí (automated knowledge acquisition), v (kybernetické) teorii řízení najdeme adaptivní a učící se systémy, v souvislosti se získáváním znalostí z databází (knowledge discovery in databases) se používá termín dolování z dat (data mining). V různých disciplínách se k problematice učení přistupuje z různých pohledů, používá se rozdílná terminologie, různé metody reprezentace znalostí i různé algoritmy pro získávání znalostí či jejich využívání. Přesto je možné nalézt jakési společné jádro, které se pokusíme popsat v této části. Uvádíme zde tedy jen jakýsi souhrnný pohled, jednotlivé metody budou podrobněji popsány v následujících podkapitolách.

V zásadě lze rozlišit dva typy učení: *učení se znalostem* (*knowledge acquisition*) a *učení se dovednostem* (*skill refinement*). První typ hledá koncepty, obecné zákonitosti apod. (např. jak rozpoznat defraudanta) u druhého typu jde o to zdokonalit své schopnosti na základě procvičování nějaké činnosti (např. jak nalézt cestu v bludišti).

U učících se systémů je většinou časově oddělena fáze učení od fáze používání znalostí v další činnosti systému (viz Obr. 1). Během učení si systém vytvoří obecnou reprezentaci jednotlivých typů chování resp. tříd (např. obecný popis spolehlivých a nespolehlivých klientů banky). Pokud chceme nalezené znalosti používat „ručně“, můžeme tímto krokem skončit. Při automatizovaném používání těchto znalostí se naučenému systému předkládají nové případy a systém se sám rozhoduje (např. klasifikuje nové klienty banky jako spolehlivé nebo nespolehlivé).



Obr. 1 Obecné schéma učícího se systému

Podíváme-li se na různé metody učení z hlediska úsilí, které je třeba vynaložit na získání nových znalostí resp. dovedností, lze rozlišovat mezi [Michalski a kol., 1983]:

1. *učení zapamatováním* (*rote learning* neboli biflování) - systém pouze zaznamenává data nebo dílčí znalosti dodané externím zdrojem; neprovádí se žádná transformace,
2. *učení se z instrukcí* (*learning from instruction*, *learning by being told*) - systém získává znalosti z externího zdroje a integruje je se znalostmi již získanými; provádí se transformace znalostí ze vstupního jazyka do vnitřní reprezentace,
3. *učení se z analogie* (*learning by analogy, instance-based learning, lazy learning*) - získávání znalostí je založeno na zapamatování si případů resp. situací podobných těm, které bude třeba v budoucnu řešit,
4. *učení na základě vysvětlení* (*explanation-based learning*) - při učení se využívá několik málo příkladů a rozsáhlé znalosti z dané oblasti (*background knowledge*),
5. *učení se z příkladů* (*learning from examples*) - zde se využívá velké množství příkladů (a protipříkladů) konceptu, který se má systém naučit, role background knowledge naopak ustupuje do pozadí; používanou metodou je *indukce*,
6. *učení se z pozorování a objevováním* (*learning from observation and discovery*) - opět se pracuje s velkým množstvím dat (tedy za využití indukce); systém si ale často sám musí vytvářet koncepty které se pak pokouší popisovat, navíc data získaná pozorováním nemusí být tak „hezká“ jako přiklady poskytnuté učitelem.

Principy používané v systémech pro získávání znalostí (strojové učení) byly převzaty z řady disciplin:

- *statistické metody* - pro získávání znalostí se používají regresní metody, diskriminační analýza, shluková analýza, nebo bayesovské metody. Tyto metody hledají popisy konceptů v podobě matematických funkcí, vektorů nebo podmíněných pravděpodobností,
- *symbolické metody umělé inteligence* - indukce rozhodovacích stromů a pravidel nebo principy případového usuzování (Case-Based Reasoning, CBR) umožňuje získat znalosti v podobě srozumitelné pro uživatele. Symbolické metody mohou pomoci uživateli při vyhledávání zajímavých vztahů v datech (databázích) a při odhalování jejich struktury. Podstatné je, že se tyto metody orientují spíše na vztahy logického typu než na matematické formule a tím poskytují (na rozdíl od klasických metod statistické analýzy dat) konceptuální, lidem bližší závěry. Znalosti získané symbolickými metodami lze také použít v tzv. „tradiční“ umělé inteligenci (např. v expertních systémech),
- *subsymbolické metody umělé inteligence* - pro získávání znalostí se používají neuronové sítě, bayesovské sítě nebo genetické algoritmy. Reprezentace nalezených znalostí opět není (podobně jako u statistických metod) pro uživatele příliš srozumitelná (např. váhy vazeb mezi neurony v neuronové síti).

Jednou z klíčových otázek strojového učení je, jakou informaci o tom, že se učí správně, má systém k dispozici. Tato informace může mít podobu

1. příkladů zařazených do tříd (konceptů), které se má systém naučit - v této situaci mluvíme o *učení s učitelem* (*supervised learning*); učitel poskytuje systému explicitní informaci o požadovaném chování,
2. odměn za správné chování a trestů za chování nesprávné - tento způsob se používá, pokud cílem systému je naučit se nějakou činnost nebo chování (např. pohyb robota v bludišti); mluvíme o *reinforcement learning,*

3. nepřímých náznaků - systém pozoruje učitele a z jeho chování usuzuje, co je příklad a co protipříklad hledaného konceptu (např. inteligentní vyhledávací systém v prostředí Internetu z toho, které nalezené odkazy uživatel aktivoval dedukuje, které WWW stránky jsou relevantní a představují tedy příklady konceptu popsaného uživatelovým dotazem). Tomuto způsobu učení můžeme říkat *učení se napodobováním* resp. *zaučováním* (v orginále *apprenticeship learning,* apprentice znamená učeň) - systém pozoruje učitele a z jeho akcí získává *implicitní informaci* o požadovaném chování,
4. systém nemá k dispozici žádnou doplňkovou informaci, pracuje pouze s příklady a „zajímavé" koncepty si vytváří sám - tento způsob se nazývá *učení bez učitele* (*unsupervised learning*) a je typický pro učení se objevováním.

Další rozlišení metod získávání znalostí může být podle:

- způsobu reprezentace příkladů použitých v procesu učení
    1. *atributy* - vlastnosti objektů reprezentovaných řádky v datové tabulce, např.

barva_vlasu	vyska	vousy	vzdelani
:	:	:	:
cerna	180	ano	VS

atributy mohou být v zásadě dvou typů: *kategoriální* (*diskrétní*) a *numerické* (*spojité*). Toto členění je postačující pro většinu algoritmů strojového učení, různě se totiž zpracovávají kategoriální a numerická data. Kategoriální atributy lze dále rozdělit na *binární* (nabývající pouze hodnot *ano* nebo *ne* - viz atribut *vousy*), *nominální* (nabývající jedné z konečného počtu hodnot, které nejsou navzájem uspořádány - viz atribut *barva_vlasu*) a *ordinální* (nabývající jedné z konečného počtu navzájem uspořádaných hodnot - viz atribut *vzdelani*).

    2. *relace* - řada navzájem provázaných relací mezi objekty a atributy, např.

otec(jan_lucembursky, karel_IV)

Většina systémů používá atributy, ty ale neposkytují tak silné prostředky pro reprezentaci znalostí jako relace (použití atributů je analogické reprezentaci znalostí za použití výrokové logiky, použití relací je analogické predikátové logice).

- způsobu zpracování příkladů
    1. *dávkové* - při dávkovém zpracování se pracuje se všemi příklady najednou, znalosti se tedy vytvářejí „od nuly",
    2. *inkrementální* - při inkrementálním zpracování se příklady zpracovávají postupně; dílčí znalosti získané na základě dříve předložených příkladů se modifikují na základě příkladů dalších.

Většina systémů pracuje v dávkovém režimu, v případě potřeby „doučit systém" na základě nových příkladů nebo „přeučit systém" (pokud hledaný koncept je proměnlivý v čase) je ale vhodnější použít inkrementální způsob učení.

- formy učení
    1. *empirické učení* - z velkého množství příkladů a z malého (často žádného) množství znalostí se metodami induktivní inference získá obecný popis daného konceptu; používá se při učení se z příkladů, z pozorování a objevováním,
    2. *analytické učení* - zobecnění se provádí na základě jediného (nebo taky žádného) příkladu a rozsáhlého množství znalostí z dané oblasti (např. to že se nemá sahat na rozpálená kamna se každé dítě naučí na základě nejvýše jednoho pokusu); používá se při učení na základě vysvětlování.

Na tomto místě je třeba zdůraznit, že „umělé“ metody učení nedosahují možností metod „přirozených“:

- formalizmus použitý pro popis situací nebo konceptů, které se má systém naučit je poměrně jednoduchý,
- koncepty, které se systém učí často odpovídají pouze jedné úrovni abstrakce zatímco člověk je schopen své koncepty uspořádávat do hierarchií,
- většina metod spoléhá na učitele, který dohlíží na celý výukový proces,
- většina metod předpokládá, vše všechna potřebná data jsou k dispozici před začátkem učení; člověk je schopen na základě další zkušenosti průběžně aktualizovat své znalosti.

Přesto se „umělé“ metody učení studují (a úspěšně používají) řadu let. V současné době procházejí zvýšeným zájmem především v souvislost se získáváním znalostí z databází.

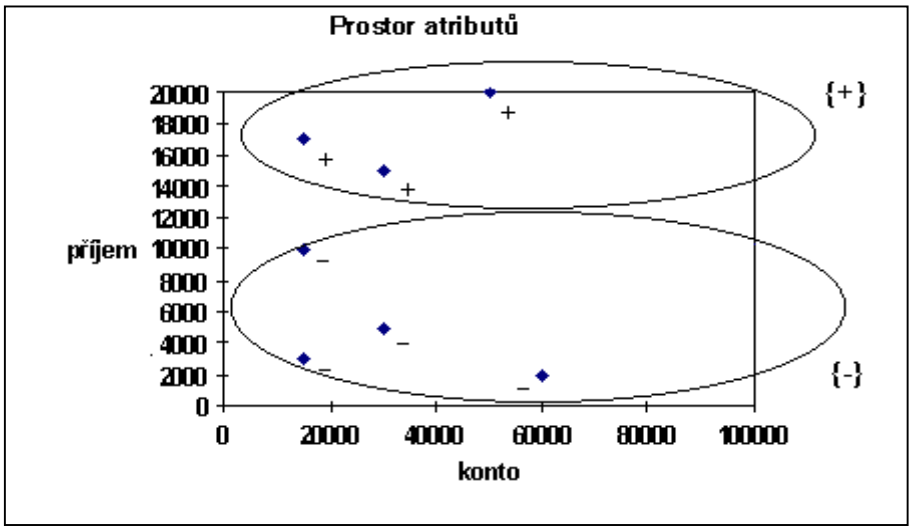
V centru naší pozornosti budou empirické metody učení se konceptům na základě příkladů rozhodnutí resp. na základě pozorování a objevování. Použitým přístupem bude *induktivní inference* kdy na základě konečného počtu příkladů budeme hledat obecný popis konceptu (ať už daného učitelem nebo vytvořeného systémem).

Empirické metody učení vycházejí z předpokladu, že jednotlivé objekty (příklady, pozorování) lze popsat pomocí charakteristik takových, že objekty patřící k témuž konceptu mají podobné charakteristiky (tyto metody bývají někdy nazývány *učení na základě podobnosti* - *similarity-based learning*). Pokud jsou objekty popsány hodnotami atributů, lze je reprezentovat jako body v mnohorozměrném prostoru, jehož dimenze je dána počtem těchto atributů[1]. Učení na základě podobnosti pak vychází z představy, že objekty představující příklady téhož konceptu vytvářejí *shluky* v tomto prostoru. Cílem učení je tedy nalézt vhodný popis těchto shluků.

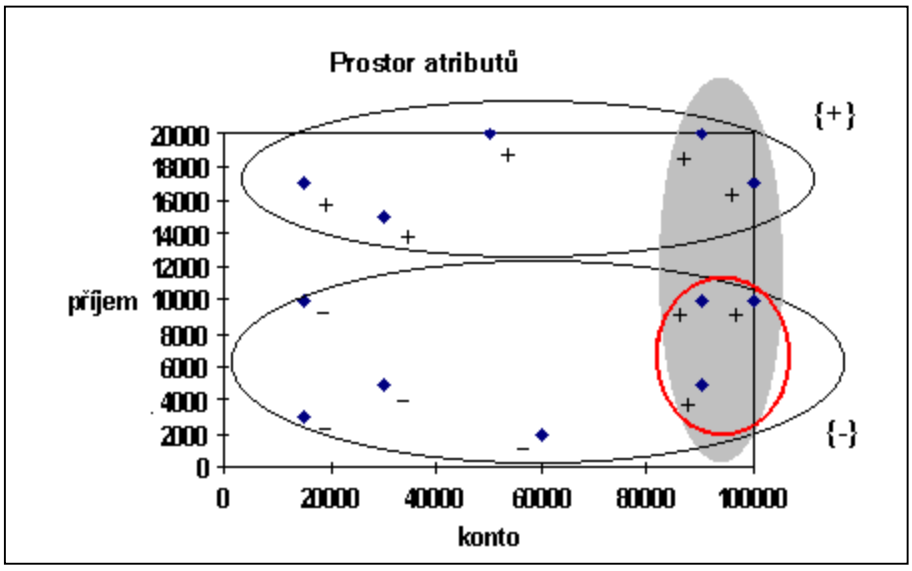
Hlavním problémem při použití výše uvedeného přístupu je nalezení oněch vhodných charakteristik. Z hlediska procesu dobývání znalostí z databází je toto úkolem kroků předzpracování dat. Ovšem ani ve chvíli, kdy máme nalezeny vhodné charakteristiky, není ještě vyhráno. Otázkou zůstává dostatečné množství dostatečně reprezentativních dat. Tento problém je ilustrován na obrázcích Obr. 2 a Obr. 3. V obou případech se snažíme na základě výše příjmu a výše konta v bance nalézt popis klientů, kterým banka půjčí (klienti +) a kterým nepůjčí (klienti -). Na základě několika příkladů klientů se zdá, že shluky odpovídající klientům obou skupin jsou zachyceny na Obr. 2. Další příklady spolehlivých klientů nás ale přesvědčí o našem omylu (příklady v šedém oválu na Obr. 3).

Popis konceptu, který byl nalezen na základě použitých příkladů tedy nemusí odpovídat jiným (dosud nezpracovaným) příkladům téhož konceptu. Z tohoto důvodu se obvykle data použitá při induktivním získávání znalostí rozdělují na část *trénovací* a část *testovací*. Trénovací data se použijí ve fázi učení, testovací data pak představují příklady, které slouží k prověření získaných znalostí. V některých případech se používají dokonce tři soubory dat: data *trénovací*, data *validační* (používaná pro eventuelní modifikaci znalostí získaných na základě trénovacích dat) a data *testovací*.

[1] Atributům se někdy říká příznaky (features), odtud anglický název tohoto prostoru – feature space.



Obr. 2 Málo dat



Obr. 3 Více dat

Pokusme se výše uvedené úvahy formalizovat. Inspirací nám bude formalizace úlohy učení s učitelem uvedená v [Kotek a kol., 1980].

Analyzovaná data jsou uložena v tabulce *D*, tvořené *n* řádky a *m* sloupci.

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots\cdots & x_{1m} \\ x_{21} & x_{22} & \cdots\cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots\cdots & x_{nm} \end{bmatrix}$$

Řádky tabulky reprezentují sledované *objekty*. Někdy se místo termínu objekt používají termíny záznam (v databázi), příklad, případ, pozorování apod. *i*-tý objekt je tedy řádek $\boldsymbol{x}_i$.

$$\mathbf{x}_i = [x_{i1}\ x_{i2}\ \ldots\ x_{im}]$$

Sloupce datové tabulky odpovídají *atributům*. Podobně jako v případě objektů, i zde se používají další termíny – veličina, proměnná, znak. *j*-tý atribut (*j*-tý sloupec) označíme symbolem A_j.

$$A_j : \begin{bmatrix} x_{1j} \\ x_{2j} \\ : \\ x_{nj} \end{bmatrix}$$

V tuto chvíli nebudeme rozlišovat, zda se jedná o atributy *kategoriální* (symbolické, diskrétní) nebo atributy *numerické* (spojité). Tato informace bude důležitá až pro jednotlivé typy metod.

U klasifikačních úloh předpokládáme, že existuje atribut, který obsahuje informaci o zařazení objektů do tříd (v případě klasifikace v užším smyslu) nebo který obsahuje predikovanou hodnotu (v případě predikce). Říkejme tomuto atributu *cílový* a označme ho symbolem C.

$$C : \begin{bmatrix} y_1 \\ y_2 \\ : \\ y_n \end{bmatrix}$$

Ostatním, necílovým atributům A_j budeme říkat *vstupní* atributy. Opět můžeme v literatuře nalézt řadu dalších názvů: cílovému atributu se někdy říká závislá proměnná, závislá veličina nebo vysvětlovaná veličina, vstupním atributům se někdy říká nezávislé proměnné, nezávislé veličiny nebo vysvětlující veličiny.

Přidáme-li cílový atribut do datové tabulky, získáme data vhodná pro použití některé metody učení s učitelem. Cílem těchto metod je na základě dat tvořených hodnotami vstupních atributů i cílového atributu odvodit znalosti použitelné pro klasifikaci nových objektů. Datům používaným k tomuto účelu se obvykle říká *trénovací data* (trénovací příklady). Příslušnou datovou tabulku budeme značit $\boldsymbol{D}_{TR}$

$$\mathbf{D}_{TR} = \begin{bmatrix} x_{11} & x_{12} & \ldots\ldots & x_{1m} & y_1 \\ x_{21} & x_{22} & \ldots\ldots & x_{2m} & y_2 \\ : & : & & : & : \\ x_{n1} & x_{n2} & \ldots\ldots & x_{nm} & y_n \end{bmatrix}$$

Objekt (trénovací příklad) z této tabulky budeme značit

$$\mathbf{o}_i = [\mathbf{x}_i, y_i]$$

Předpokládejme, že pro každý objekt $\boldsymbol{o}_i$ známe všechny hodnoty $\boldsymbol{x}_i$ i hodnotu y_i.

V drtivé většině situací předpokládáme, že nezáleží na pořadí objektů v datové tabulce[2] . Budeme tedy data považovat za množinu objektů

$$D_{TR} = \{\mathbf{o_i}, i = 1,..,n\}$$

Klasifikační úlohu můžeme chápat jako úlohu nalézt takové *znalosti* (reprezentované rozhodovací funkcí *f*), které by umožňovaly k hodnotám vstupních atributů nějakého objektu přiřadit vhodnou hodnotu atributu cílového

$$f: \mathbf{x} \rightarrow y.$$

Rozhodovací funkci *f* přitom chápeme v dosti širokém významu. Je-li klasifikace založena na algoritmu používajícím rozhodovací stromy, je tato funkce (a tedy i hledané znalosti) reprezentována jedním konkrétním rozhodovacím stromem. Je-li klasifikace založena na neuronové síti, je tato funkce (a tedy i hledané znalosti) reprezentována topologií konkrétní sítě a váhami vazeb mezi neurony.

V průběhu klasifikace se tedy pro hodnoty vstupních atributů ***x*** nějakého objektu ***o*** odvodí hodnota cílového atributu. Označme tuto odvozenou hodnotu $\hat{y}$.

$$\hat{y} = f(\mathbf{x}).$$

Odvozená hodnota $\hat{y}$ se pro objekty z trénovacích dat může lišit od skutečné hodnoty y. Můžeme tedy pro každý objekt $\mathbf{o}_i \in D_{TR}$ vyčíslit *chybu klasifikace* $Q_f(\mathbf{o}_i, \hat{y}_i)$. V případě numerického atributu C může být touto chybou například čtverec rozdílu skutečné a odvozené hodnoty cílového atributu

$$Q_f(\mathbf{o_i}, \hat{y}_i) = (y_i - \hat{y}_i)^2$$

v případě kategoriálního atributu C může být touto chybou informace o tom že se odvozená a skutečná hodnota vzájemně liší,

$$Q_f(\mathbf{o_i}, \hat{y}_i) = \begin{cases} 1 & \text{pro } y_i \neq \hat{y}_i \\ 0 & \text{pro } y_i = \hat{y}_i \end{cases}$$

Pro celou trénovací množinu D_{TR} pak můžeme vyčíslit souhrnnou chybu $Err(f,D_{TR})$, například jako střední chybu

$$Err(f, D_{TR}) = \frac{1}{n}\sum_{i=1}^{n} Q_f(\mathbf{o_i}, \hat{y}_i).$$

Cílem učení je nalézt takové znalosti f^*, které by minimalizovaly tuto chybu

$$Err(f^*, D_{TR}) = \min_f Err(f, D_{TR}).$$

V příadě, že trénovací data neobsahují kontradikce, tedy že platí

$$\forall\, \mathbf{o}_1, \mathbf{o}_2 \in D_{TR}: \quad \mathbf{x}_1 = \mathbf{x}_2 \Rightarrow y_1 = y_2$$

lze teoreticky nalézt takovou reprezentaci konceptů f^*, že $Err(f^*,D_{TR}) = 0$. Můžeme tedy nalézt znalosti bezchybně klasifikující příklady v trénovací množině. Naším cílem je ale samozřejmě nalézt znalosti obecnější, použitelné i pro klasifikaci objektů nových. Ne vždy je nulová chyba $Err(f^*,D_{TR})$ dosažená na trénovacích datech zárukou kvality nalezených znalostí. Přílišná orientace na trénovací

[2] Výjimku tvoří prostorová data (např. data z geografických informačních systémů), nebo časová data (např. vývoj cen akcií), kdy uspořádání mezi objekty vyplývá z povahy těchto dat.

data může vést k „přeučení systému“ (*overfitting*); získané znalosti pak nereflektují obecnější zákonitosti, ale pouze kopírují strukturu použitých příkladů (viz Obr. 2 a Obr. 3). Důraz tedy klademe na to, že v průběhu učení zobecňujeme použité příklady na celou aplikační oblast. Schopnost nalezených znalostí *generalizovat* se obvykle ověřuje experimentálně na tzv. *testovacích datech* D_{TST}; tedy pomocí chyby $Err(f^*,D_{TST})$. Testovací data mají stejnou strukturu atributů, jako data trénovací, obsahují tedy i cílový atribut. Jedná se ale o objekty, které nabyly použity v průběhu učení.

4.2 Učení jako prohledávání

Předpokládejme, že jak vstupní atributy tak i cílový atribut jsou kategoriální. Obor hodnot j-tého atributu označme V_j a počet hodnot j-tého atribut označme K_j. Zápisem $A_j(v_k)$ kde $v_k \in V_j$ budeme značit k-tou hodnotu j-tého atributu. Hodnotě atributu budeme říkat *kategorie*. Kategorii můžeme chápat dvojím způsobem:

1. Z pohledu predikátové logiky se jedná o atomickou formuli vyjadřující *vlastnost objektu* (např. kategorie *pohlaví(muž)* vyjadřuje vlastnost býti mužem). O každém objektu můžeme rozhodnout, zda má nebo nemá uvedenou vlastnost (splňuje nebo nesplňuje uvedenou formuli):

$$\forall \mathbf{o}_i : \; A_j(v_k)(\mathbf{o}_i) = \begin{cases} 1 & \text{pro } x_{ij} = v_k \\ 0 & \text{pro } x_{ij} \neq v_k \end{cases}$$

2. Z množinového pohledu definuje kategorie *množinu objektů* majících danou vlastnost:

$$\{A_j(v_k)\} = \{\mathbf{o_i} : x_{ij} = v_k\}$$

Spojováním kategorií logickou spojkou $\wedge$ budeme vytvářet *kombinace*. Označme si

$$\mathrm{Comb} \;=\; [A_{j_1}(v_{k_1}), A_{j_2}(v_{k_2}),..., A_{j_l}(v_{k_l})] \;=\; A_{j_1}(v_{k_1}) \wedge A_{j_2}(v_{k_2}) \wedge ... \wedge A_{j_l}(v_{k_l}).$$

Kombinaci tedy můžeme chápat jako seznam (množinu) kategorií nebo jako konjunkci kategorií. Počet kategorií v kombinaci *Comb* budeme nazývat *délkou* kombinace a značit *l(Comb)* [3]. Vždy budeme předpokládat, že v kombinaci se nevyskytují dvě kategorie téhož atributu.

Libovolnou kombinaci *Comb* budeme opět interpretovat buď jako logickou formuli

$$\forall \mathbf{o}_i : \; \mathrm{Comb}(\mathbf{o}_i) = \begin{cases} 1 & \text{pro } x_{ij_1} = v_{k_1} \wedge x_{ij_2} = v_{k_2} \wedge ... \wedge x_{ij_l} = v_{k_1} \\ 0 & \text{jinak} \end{cases}$$

nebo jako množinu objektů

$$\{\mathrm{Comb}\} = \{\mathbf{o_i} : x_{ij_1} = v_{k_1} \wedge x_{ij_2} = v_{k_2} \wedge ... \wedge x_{ij_l} = v_{k_1}\}.$$

Počet prvků množiny $\{Comb\}$ nazveme *četnost kombinace Comb* a označíme *n(Comb)*.
Platí-li $\mathrm{Comb}(\mathbf{o_i}) = 1$, říkáme, že kombinace *Comb pokrývá* objekt $\boldsymbol{o_i}$. Množina $\{Comb\}$ je tedy množina objektů pokrytých kombinací *Comb*.

[3] Kategorie je tedy kombinace délky 1.

Jak již bylo řečeno, kombinace se vytvářejí z kategorií. Přidáváním kategorií ke kombinaci vznikají její *nadkombinace*, odebráním kategorií z kombinace vznikají její *podkombinace*. Jsou-li *Comb1*, *Comb2* dvě různé kombinace takové, že

$$\forall A_j(v_k): A_j(v_k) \in \mathrm{Comb}_1 \Rightarrow A_j(v_k) \in \mathrm{Comb}_2,$$

je kombinace *Comb1* *podkombinací* kombinace *Comb2* a kombinace *Comb2* *nadkombinací* kombinace *Comb1*.

Pomocí pojmu podkombinace můžeme definovat částečné uspořádání mezi kombinacemi[4]. Je-li kombinace *Comb1* podkombinací kombinace *Comb2*, potom říkáme, že kombinace *Comb1* je *obecnější* než kombinace *Comb2* a že kombinace *Comb2* je *speciálnější* než kombinace *Comb1* a zapisujeme

$$\mathrm{Comb}_1 \succ \mathrm{Comb}_2$$

Je-li kombinace *Comb1* *obecnější* než kombinace *Comb2*, potom *Comb1* pokrývá alespoň všechny ty objekty, které pokrývá *Comb2*. Pro takové dvě kombinace tedy platí

$$\forall \mathbf{o}_i: \mathrm{Comb}_2(\mathbf{o}_i) = 1 \Rightarrow \mathrm{Comb}_1(\mathbf{o}_i) = 1$$

resp.

$$\{\mathrm{Comb}_2\} \subseteq \{\mathrm{Comb}_1\}$$

a tedy

$$n(\mathrm{Comb}_2) \leq n(\mathrm{Comb}_1).$$

Je-li kombinace *Comb1* *obecnější* než kombinace *Comb2*, pak je délka kombinace *Comb1* menší než délka kombinace *Comb2*

$$l(\mathrm{Comb}_1) < l(\mathrm{Comb}_2).$$

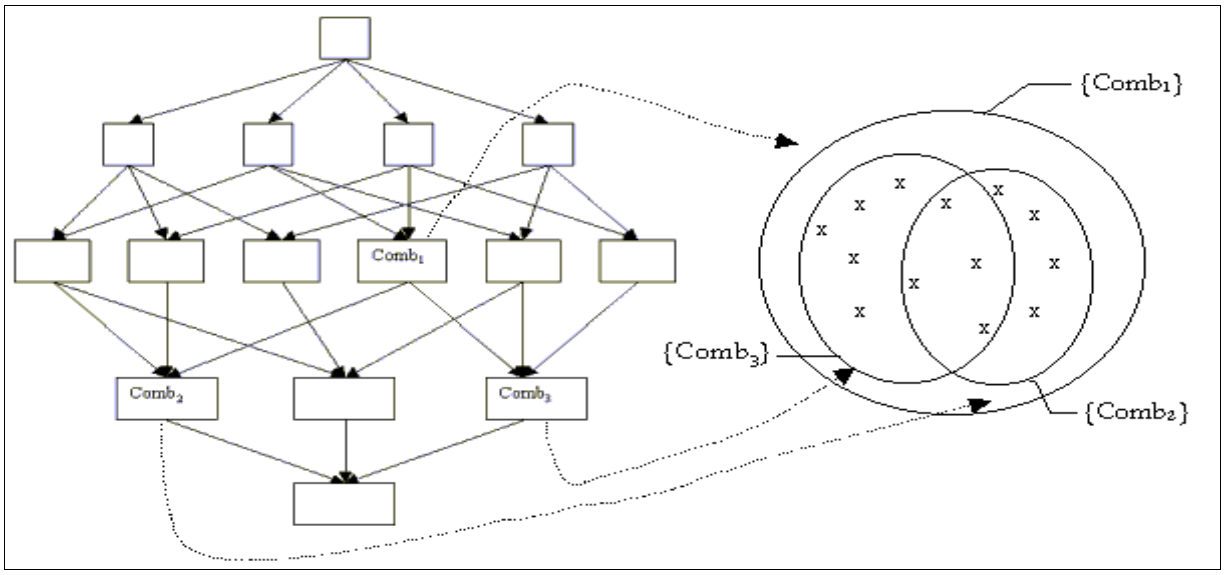
Počet možných kombinací všech možných délek závisí na počtu vstupních atributů a na počtu hodnot jednotlivých atributů. Je-li *m* počet vstupních atributů a K_j počet hodnot *j*-tého atributu, je

$$\prod_{j=1}^{m} (K_j + 1)$$

počet všech kombinací, které lze vytvořit.

Všechny kombinace lze uspořádat podle obecnosti do tzv. *prostoru kombinací*. Prostor kombinací má podobu orientovaného grafu, kde uzly jsou kombinace a každá hrana směřuje z nějaké kombinace délky *l* do její nadkombinace délky *l+1*. Hrany tedy vyjadřují relaci „býti obecnější“ (resp. „býti speciálnější“) – viz Obr. 4.

[4] Pro dvě kombinace $\mathrm{Comb}_1, \mathrm{Comb}_2$ totiž nemusí platit ani $\mathrm{Comb}_1 \succ \mathrm{Comb}_2$ ani $\mathrm{Comb}_2 \prec \mathrm{Comb}_1$.



Obr. 4 Prostor kombinací

V případě učení s učitelem se v datech vyskytuje cílový atribut *C*. Kategorie cílového atributu budeme značit $C(v_t)$. Každá kategorie bude reprezentovat příklady nějaké třídy (konceptu). Označme n_t počet příkladů třídy t (tedy $n_t=n(C(v_t))$) a T počet tříd. Pro n objektů v trénovacích datech D_{TR} tedy platí

$$n = \sum_{t=1}^{T} n_t$$

Velice často budeme řešit úlohu, kdy cílový atribut má pouze dvě přípustné hodnoty. V takovém případě budeme jednu z hodnot považovat za koncept, jehož popis se chceme naučit (budeme značit +). Kategorie (množina) pozitivních příkladů konceptu pak bude

$$\{C(+)\} = \{\mathbf{o}_i : y_i = +\} = D^{+}_{TR}$$

a kategorie (množina) negativních příkladů (resp. protipříkladů) tohoto konceptu pak bude

$$\{C(-)\} = \{\mathbf{o}_i : y_i \neq +\} = D^{-}_{TR}$$

Data D_{TR} tedy budou rozdělena do dvou tříd.

$$D_{TR} = D^{+}_{TR} \cup D^{-}_{TR}$$

V případě učení s učitelem budeme hledat znalosti použitelné pro klasifikaci objektů do tříd. Znalosti budou reprezentovány kombinacemi, které budeme chápat jako hypotézy vyjadřující vazbu mezi hodnotami vstupních atributů na jedné straně a hodnotou cílového atributu na straně druhé. Budou nás zajímat především tzv. *konzistentní* kombinace. Kombinace *Comb* je konzistentní, právě když pokrývá pouze příklady jedné třídy:

$$\exists C(v_t) \forall \mathbf{o}_i \in D_{TR} : \; Comb(\mathbf{o}_i) = 1 \;\Rightarrow\; y_i = v_t$$

Označme $a_t=n_t(Comb)$ počet příkladů třídy t pokrytých kombinací *Comb*. Potom

$$n(Comb) = \sum_{t=1}^{T} n_t(Comb) = \sum_{t=1}^{T} a_t$$

Pro konzistentní kombinace pak existuje třída $C(v_t)$ taková, že $n(Comb) = a_t$.

V případě klasifikace do dvou tříd nás budou zajímat především kombinace konzistentní s pozitivními příklady konceptu.

$$\forall \mathbf{o}_i \in D_{TR} : \ \mathrm{Comb}(\mathbf{o}_i) = 1 \ \Rightarrow \ y_i = +$$

Dalším požadavkem bude, aby nalezené znalosti pokrývaly všechny použité trénovací příklady. Nedá se samozřejmě očekávat, že tuto vlastnost bude mít jediná kombinace. Znalosti tedy budou tvořeny množinou kombinací. Označme tuto množinu *Desc*. Pokud platí, že každý objekt z trénovací množiny je pokryt nějakou kombinací z množiny *Desc*

$$\forall \mathbf{o}_i \in D_{TR} \ \exists \mathrm{Comb} \in \mathrm{Desc} : \ \mathrm{Comb}(\mathbf{o}_i) = 1$$

řekneme, že množina *Desc* je *úplná*.

Cílem učení tedy bude nalézt úplnou množinu kombinací, které jsou konzistentní s trénovacími daty. Dalším požadavkem může být, aby tato množina byla co nejmenší [5].

Kombinace budeme hledat v dříve zmíněném prostoru kombinací. Tento prostor můžeme procházet (prohledávat) v zásadě dvěma způsoby:

- od obecnější kombinace ke speciálnější (krok *specializace*),
- od speciálnější kombinace k obecnější (krok *generalizace*).

V kroku specializace přidáme ke kombinaci nějakou kategorii, v kroku generalizace nějakou kategorii z kombinace odstraníme. Při specializaci resp. generalizaci tedy dvěma různými „směry“ procházíme prostor kombinací. Postup od obecnějších kombinací ke speciálnějším se někdy nazývá postup *shora dolů* (top down), postup od speciálnějších kombinací k obecnějším se někdy nazývá postup *zdola nahoru* (bottom up).

Pro prohledávání prostoru kombinací se nabízí řada strategií dobře známých z jiných oblastí umělé inteligence [Winston, 1992].

Jedna z možností je systematicky prohledat celý prostor kombinací. Prohledávat můžeme *do šířky* (breath-first), *do hloubky* (depth-first), podle nějaké heuristiky (například podle četností kombinací), nebo náhodně. Uvedené strategie můžeme nalézt v algoritmech pro hledání asociačních pravidel, bude tedy o nich ještě zmínka v příslušné kapitole.

Jinou možností je projít jen část prostoru – opět náhodně nebo řízeně. Při řízeném prohledávání to obvykle znamená důsledně si pro každý krok vybrat jen tu nejlepší možnost (tedy například pro specializaci dané kombinace jen tu nejperspektivnější kategorii). Jedná se tedy v zásadě o gradientní strategii (podrobněji o gradientních metodách v následující podkapitole), v kontextu symbolických metod strojového učení nazývanou *best-first* nebo *hill climbing*. Tyto strategie nalezneme například při tvorbě rozhodovacích pravidel nebo rozhodovacích stromů. Někde mezi systematickými a gradientními strategiemi leží tzv. paprskové prohledávání (*beam search*); při této strategii si pro každý krok nevybereme pouze jedinou možnost, ale paralelně sledujeme určitý počet nejlepších možností – jde tedy o omezené prohledávání do šířky.

[5] Z tohoto požadavku plyne, že by nalezené kombinace měly být co nejobecnější, aby jedna kombinace pokryla co nejvíce příkladů. Úplnou množinou konzistentních kombinací totiž může být i taková množina, ve které ke každému objektu existuje kombinace délky m - tedy vlastně trénovací množina D_{TR}^{+} zapsaná jako kombinace.

Ilustrujme si předcházející úvahy na jednoduché úloze klasifikace klientů banky z hlediska rizikovosti poskytnutí úvěru. Budeme hledat znalosti, které nám na základě hodnot atributů příjem, konto, pohlaví, nezaměstnaný, auto a bydlení umožní rozhodnout, zda vyhovět žádosti o úvěr. Trénovací data budou tvořena čtyřmi objekty (Tab. 1). Jedná se modifikovaný příklad uvedený v [Mitchell, 1997].

příjem	konto	pohlaví	nezaměstnaný	auto	bydlení	úvěr
vysoký	vysoké	žena	ne	ano	vlastní	+
vysoký	vysoké	muž	ne	ano	vlastní	+
nizký	nízké	muž	ne	ano	nájemní	-
vysoký	vysoké	muž	ne	ne	nájemní	+

Tab. 1 Trénovací data

Znalosti, které hledáme, budou tvořeny kombinacemi kategorií. Ve shodě s Mitchellem zavedeme pro konzistentní kombinaci pojem *hypotéza*. V našem případě tedy hypotéza umožňuje zařadit nějaký objekt do třídy *úvěr(+)* (tedy hypotéza reprezentuje koncept *úvěr(+)*). Všechny hypotézy je možno graficky znázornit v tzv. *prostoru hypotéz (hypothesis space) H*. Vzhledem k tomu, že jsme hypotézy definovali jako konzistentní kombinace, je tento prostor pouze částí prostoru kombinací.

Část prostoru hypotéz pro data z Tab. 1 ukazuje Obr. 5. Nejobecnější hypotézou, která pokrývá všechny příklady je hypotéza délky *0*, reprezentovaná prázdnou kombinací *[ ]*. Nejspeciálnějšími hypotézami jsou hypotézy délky *m*, které reprezentují jednotlivé pozitivní příklady. Hypotézy zapisujeme dříve zavedeným způsobem, tedy např. *[Příjem(vysoký),Konto(vysoké)]* [6].

Příkladem algoritmu, který postupuje od speciálnějšího popisu k obecnějšímu je *Find-S* [Mitchell, 1997][7]. Algoritmus *Find-S* hledá nejspeciálnější hypotézu, která je konsistentní se všemi příklady hledaného konceptu. Algoritmus postupně prochází pozitivní příklady a hledá jejich společný popis. Negativní příklady nemají na učení vliv (viz Obr. 6). V našem příkladu algoritmus postupně uvažuje hypotézy

[Příjem(vysoký),Konto(vysoké),Pohlaví(žena),Nezaměstnaný(ne),Auto(ano),Bydlení(vlastní)]

[Příjem(vysoký),Konto(vysoké),Nezaměstnaný(ne),Auto(ano),Bydlení(vlastní)]

[Příjem(vysoký),Konto(vysoké),Nezaměstnaný(ne)]

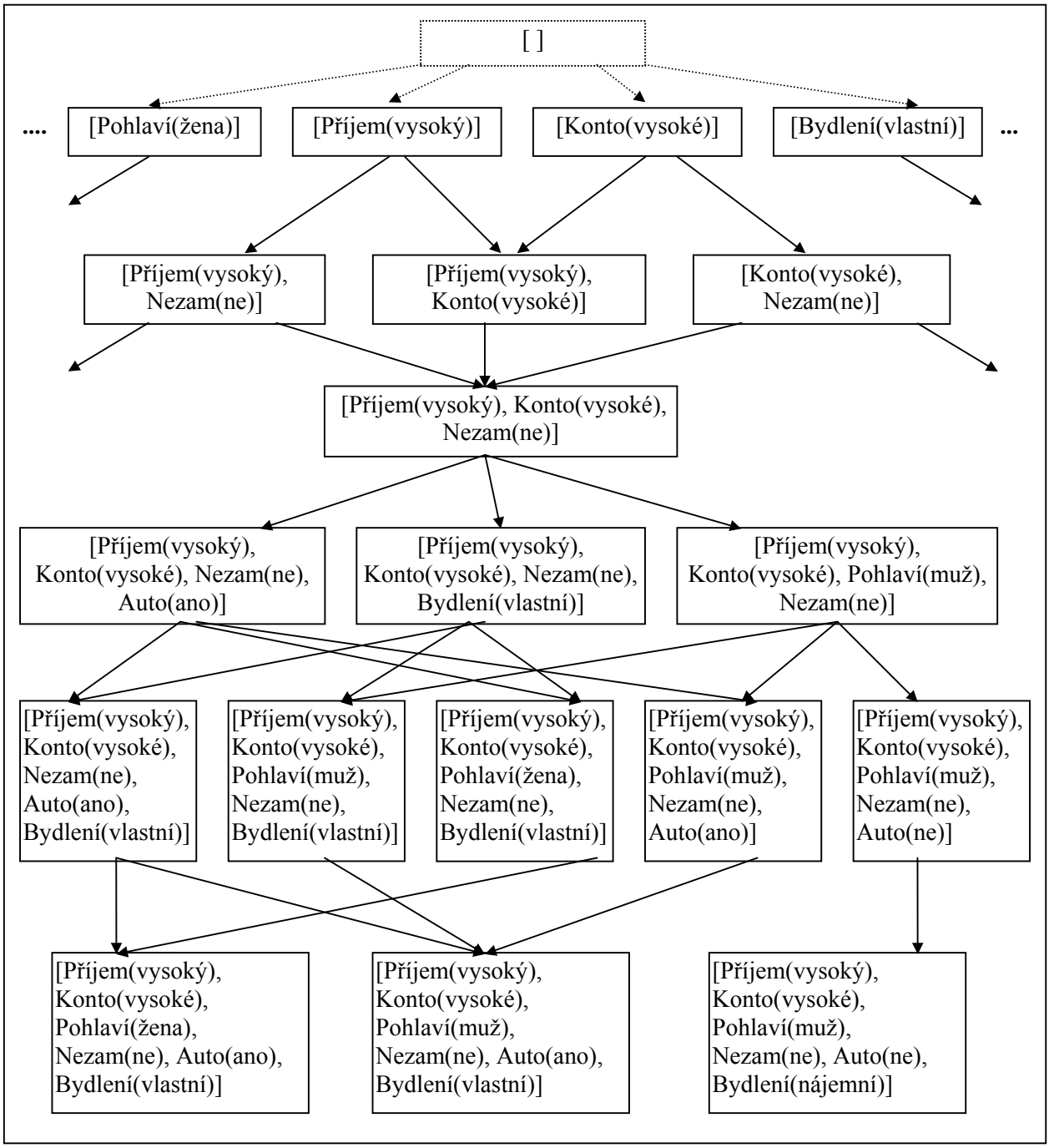
(viz.Obr. 7). Při prohledávání prostoru hypotéz tedy postupujeme zdola nahoru.

---

[6] Mitchell sám používá poněkud jiný zápis hypotéz. Obor hodnot každého vstupního atributu doplňuje o symbol ? vyjadřující, že na hodnotě atributu nezáleží (tedy že atribut je irelevantní). Kategorie *A(?)* pokrývá všechny objekty; její přidání k nějaké kombinaci *Comb* tedy nemá vliv ani na logické ani na množinové chápání kombinace *Comb*. Zavedení hodnoty *?* má čistě formální důvody. Umožňuje totiž pracovat pouze s kombinacemi tvořenými kategoriemi všech vstupních atributů (tedy s kombinacemi délky *m*). Takové kombinace Mitchell zapisuje jako seznam hodnot všech atributů. Tedy např. zápis *[vysoký, vysoké, ?, ?, ?, ?]* vyjadřuje kombinaci *[Příjem(vysoký),Konto(vysoké)]*. Pro každý atribut *A* přitom platí $A(?) \succ A(v_k)$. Nejobecnější hypotéza je tedy zapsána jako *[?, ?, ?, ?, ?, ?]*. V této reprezentaci hypotéz je krok generalizace realizován jako náhrada některé hodnoty $A(v_k)$ hodnotou *A(?)* a krok specializace realizován jako náhrada některé hodnoty *A(?)* hodnotou $A(v_k)$.

Z Mitchellova přístupu je snadno vidět, proč je počet všech kombinací dán vzorcem $\prod_i(K_i + 1)$. Vzorec počítá všechny kombinace délky *m*, kde obor hodnot každého atributu je rozšířen o hodnotu *?*.

[7] V původním Mitchellově algoritmu má krok generalizace (krok 2.1 algoritmu) podobu: „pro každý atribut A_i if kategorie $A_j(v_k)$ nepokrývá příklad ***o*** then nahraď tuto kategorii nejbližší obecnější kategorii $A_j(v_g)$ která pokrývá ***o***“

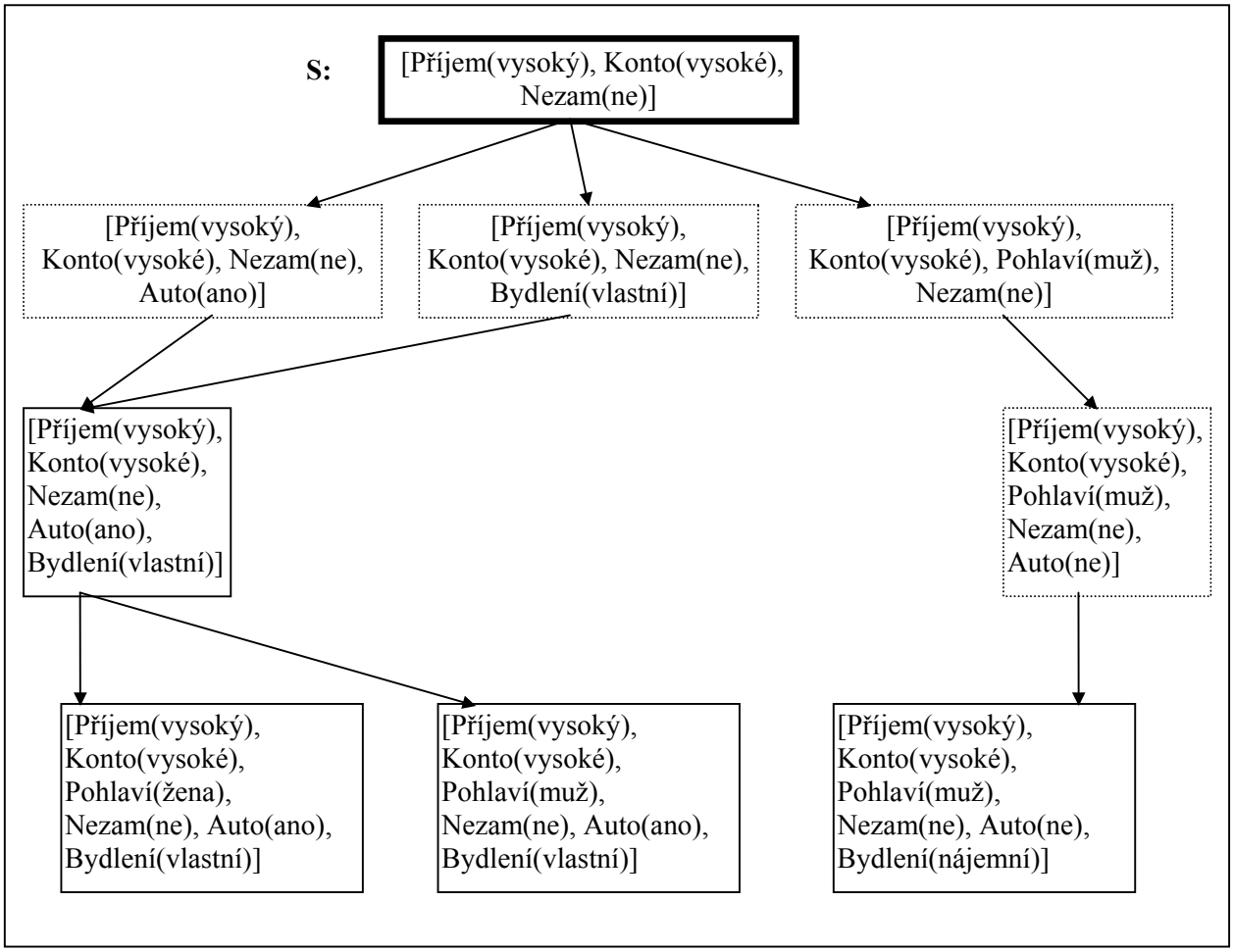


Obr. 5 Prostor hypotéz

Find-S algoritmus

1. přiřaď do h nejspeciálnější hypotézu z H
2. pro každý pozitivní příklad ***o***
   2.1. pro každou kategorii $A_j(v_k)$ z hypotézy h
      2.1.1. pokud kategorie $A_j(v_k)$ nepokrývá příklad ***o***, potom
         odstraň tuto kategorii z hypotézy h
3. vydej h

Obr. 6 Algoritmus Find-S



Obr. 7 Prostor hypotéz prohledaný algoritmem Find-S

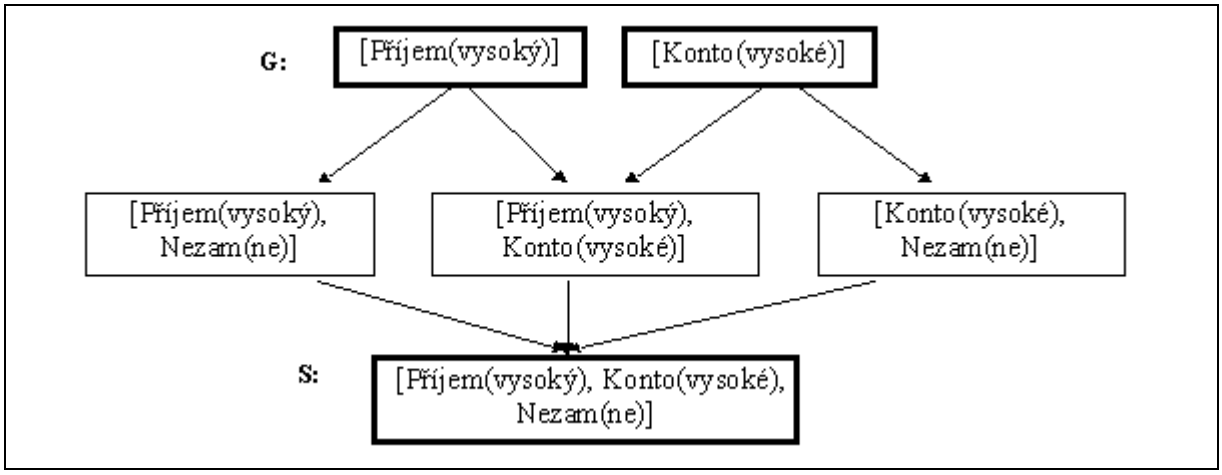
Jiným algoritmem je *Candidate-Elimination* opět popsaný v [Mitchell, 1997]. Na rozdíl od algoritmu *Find-S* budeme nyní hledat *všechny* hypotézy, které jsou konzistentní s trénovacími příklady. Tuto podmnožinu hypotéz, nazývanou *prostor řešení* (*version space),* lze reprezentovat obecnou hranicí (general boundary *G* - nejspeciálnější obecná hypotéza) a speciální hranicí (specific boundary *S* - nejobecnější speciální hypotéza). Algoritmus hledá tyto hranice z obou konců prostoru hypotéz (Obr. 8). Pro pozitivní příklad se generalizuje speciální hypotéza tak, aby tento příklad pokrývala (hledá se speciální hranice), pro negativní příklad se specializuje obecná hypotéza tak, aby tento příklad nepokrývala (hledá se obecná hranice).

Pro naše data z Tab. 1 povedou první dva (pozitivní příklady) k nalezení *S* = {*[Příjem(vysoký), Konto(vysoké), Nezaměstnaný(ne),Auto(ano),Bydlení(vlastní)]*} a *G* = {*[]*}, třetí (negativní) příklad povede ke změně *G* na {*[Příjem(vysoký)], [Konto(vysoké)], [Bydlení(vlastní)]*}. Čtvrtý, pozitivní příklad povede ke změně *S* na {*[Příjem(vysoký), Konto(vysoké), Nezaměstnaný(ne)]*} a ke změně *G* na {*[Příjem(vysoký)], [Konto(vysoké)]*}. Výsledný prostor řešení po zpracování všech příkladů je uveden na Obr. 9.

Candidate-Elimination algoritmus

1. přiřaď do *G* množinu nejobecnějších hypotéz z *H*
2. přiřaď od *S* množinu nejspeciálnějších hypotéz z *H*
3. pro každý příklad ***o***
   3.1. pokud ***o*** je pozitivní příklad, potom
      3.1.1. odstraň z *G* všechny hypotézy, které nepokrývají příklad ***o***
      3.1.2. pro každou hypotézu *s* z *S,* která nepokrývá příklad ***o***
         3.1.2.1. odstraň *s* z *S*
         3.1.2.2. přidej do *S* nejmenší generalizaci *h* hypotézy *s* takovou, že *h* pokrývá příklad ***o*** a že v *G* je hypotéza obecnější než *h*
      3.1.3. odstraň z *S* hypotézy, které jsou obecnější než jiné hypotézy v *S*
   3.2. pokud ***o*** je negativní příklad, potom
      3.2.1. odstraň z *S* všechny hypotézy, které pokrývají příklad ***o***
      3.2.2. pro každou hypotézu *g* z *G,* která pokrývá příklad ***o***
         3.2.2.1. odstraň *g* z *G*
         3.2.2.2. přidej do *G* nejmenší specializaci *h* hypotézy *g* takovou, že *h* nepokrývá příklad ***o*** a že v *S* je hypotéza speciálnější než *h*
      3.2.3. odstraň z *G* všechny hypotézy, které jsou speciálnější než jiné hypotézy v *G*
4. vydej množiny *G* a *S*

Obr. 8 Algoritmus Candidate-Elimination



Obr. 9 Prostor řešení

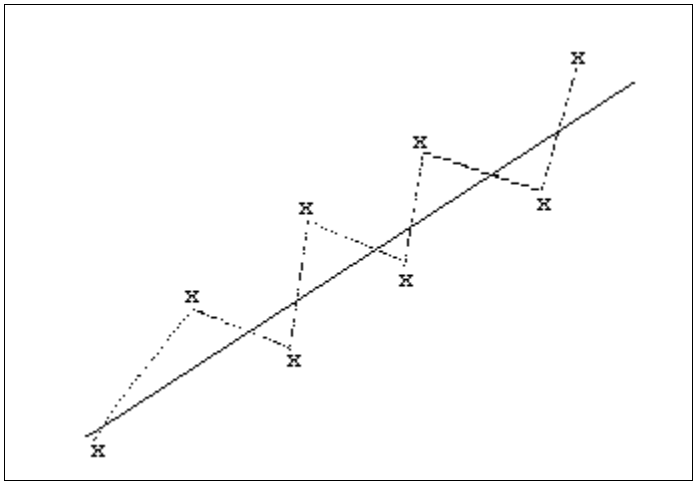
Praktické použití obou uvedených algoritmů je omezeno skutečností, že dobře fungují pouze pro data bez kontradikcí. V dalších kapitolách se budeme podrobněji zabývat *symbolickými* algoritmy, které se dokáží s tímto omezením vyrovnat.

Na závěr této podkapitoly jedna poznámka. Učení jako prohledávání zde bylo chápáno jako prohledávání prostoru kombinací. Jak uvidíme později, tento způsob odpovídá algoritmům pro tvorbu pravidel. Jako prohledávání prostoru řešení (prostoru hypotéz) lze ale chápat i jiné symbolické metody strojového učení. Prostor hypotéz by pak byl příslušně modifikován; například do podoby prostoru rozhodovacích stromů.

4.3 Učení jako aproximace funkcí

Opět vycházíme z (již dříve uvedené) představy, že objekty téže třídy tvoří shluky v prostoru atributů. Jednotlivé shluky lze od sebe oddělit hranicí kterou můžeme popsat jako funkci hodnot jednotlivých atributů. Při učení se tedy snažíme na základě příkladů objektů jednotlivých tříd nalézt tuto funkci.

Při *aproximaci* nějaké funkce se na základě hodnot této funkce v konečném počtu bodů snažíme zrekonstruovat její obecnou podobu (často vyjádřenou analyticky). Na rozdíl od *interpolace* nepožadujeme, aby nalezená funkce procházela známými body. Cílem je nalézt takovou funkci, která by co možná nejlépe vystihovala funkční hodnoty i v bodech, které nemáme k dispozici. Hledání takového popisu je založeno na vyhodnocení odchylky mezi skutečnou funkční hodnotou ve známém bodě a mezi funkční hodnotou v tomto bodě, která vychází z hledaného popisu funkce. Naším cílem je minimalizovat celkovou odchylku ve všech známých bodech (Obr. 10).



Obr. 10 Lineární interpolace vs. lineární aproximace

Při aproximování funkcí se obvykle používá *metoda nejmenších čtverců* popsaná v kapitole věnované statistickým metodám. Připomeňme zde tedy jen to, že při použití této metody minimalizujeme součet druhých mocnin odchylek mezi skutečnou hodnotou y a „vypočítanou“ hodnotou $\hat{y}$. Při výpočtu hodnoty y přitom předpokládáme určitý typ funkční závislosti $y = f(\boldsymbol{x})$ specifikovaný pouze svými parametry (označme je symbolem $\boldsymbol{q}$), a metodou nejmenších čtverců hledáme tyto parametry. Hledání minima celkové odchylky

$$\min_{f} \mathrm{Err}(f, D_{TR}) = \min_{f} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

se pak převádí na řešení rovnice

$$\frac{d}{dq} \sum_{i=1}^{n} \left(y_i - f(\mathbf{x}_i)\right)^2 = 0$$

V případě analytického vyjádření hledané funkce $f(\boldsymbol{x})$ (to je případ regresní nebo diskriminační analýzy kdy se volí typ funkce) lze z výše uvedené rovnice spočítat na základě příkladů $\mathbf{o}_i = [\mathbf{x}_i, y_i]$ její parametry $\boldsymbol{q}$.

V případě, že tvar hledané funkce neznáme, je třeba použít numerické metody. Minimum funkce se pak hledá iteračními gradientními metodami. Tyto metody jsou založeny na tom, že z daného místa (aktuální hodnoty funkce) se při hledání minima pohybujeme ve směru největšího spádu (gradientu)

$$\nabla \mathrm{Err}(\mathbf{q}) = \left[\frac{\partial \mathrm{Err}}{\partial q_0}, \frac{\partial \mathrm{Err}}{\partial q_1}, \ldots, \frac{\partial \mathrm{Err}}{\partial q_Q} \right].$$

Modifikace znalostí $\mathbf{q} = [q_0, q_1, \ldots, q_Q]$ pak probíhá podle algoritmu

$$q_j \leftarrow q_j + \Delta q_j$$

kde

$$\Delta q_j = -\eta \frac{\partial \mathrm{Err}}{\partial q_j}$$

a η je parametr vyjadřující „velikost kroku“ kterým se přibližujeme k minimu funkce *Err*.

Je-li např. chybová funkce

$$\mathrm{Err}(f, D_{TR}) = \frac{1}{2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{2} \sum_{i=1}^{n} (y_i - f(\mathbf{x_i}))^2$$

a předpokládaná funkce f lineární kombinací vstupů

$$f(\mathbf{x}) = \mathbf{q} * \mathbf{x} ,$$

můžeme odvodit gradient funkce *Err* jako

$$\frac{\partial \mathrm{Err}}{\partial q_j} = \frac{1}{2} \sum_{i=1}^{n} \frac{\partial}{\partial q_j} (y_i - \hat{y}_i)^2 = \frac{1}{2} \sum_{i=1}^{n} 2(y_i - \hat{y}_i) \frac{\partial}{\partial q_j} (y_i - \hat{y}_i) = \sum_{i=1}^{n} (y_i - \hat{y}_i) \frac{\partial}{\partial q_j} (y_i - \mathbf{qx}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)(-x_{ij})$$

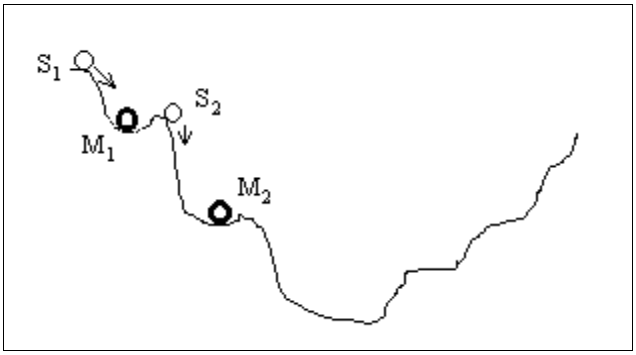
a tedy

$$\Delta q_j = \eta \sum_{i=1}^{n} (y_i - \hat{y}_i) x_{ij}$$

Gradientní metody se používají při učení neuronových sítí nebo při evolučním programování. Na rozdíl od prvního případu (analytické vyjádření hledané funkce), kdy nalezneme globální minimum celkové odchylky, gradientní metody dokonvergují do nejbližšího minima, které bývá často pouze minimem lokálním. Gradientní metody tak můžeme přirovnat k pohybu kuličky v hrbolatém terénu (Obr. 11). Výsledek (parametry hledané funkce) silně závisí na počátečním stavu, ze kterého minimum začínáme hledat. Ze stavu S_1 dokonvergujeme do minima M_1, ze stavu S_2 dokonvergujeme do minima M_2. Jednou z cest jak se s tímto problémem vypořádat je tzv. *simulované žíhání* (simulated annealing).

Myšlenka simulovaného žíhání vychází z analogie s metalurgií. Zde se žíháním nazývá opětovné zahřátí kovu během jeho chladnutí. Výsledkem tohoto procesu je pevnější materiál, atomy kovu se lépe uspořádají. Simulovaným žíháním (v souvislosti s gradientními metodami) se myslí drobná změna parametrů funkce v okamžiku, kdy algoritmus dokonvergoval do minima. Změnou parametrů se samozřejmě z minima vychýlíme. Při opakovaném použití gradientní metody je pak jistá šance, že algoritmus dokonverguje do minima hlubšího; vychýlení nám totiž může umožnit „přeskočení“ bariéry

v okolí prvního nalezeného minima. Gradientní metodou dokonvergujeme ze stavu S_1 do minima M_1, s pomocí simulovaného žíhání bychom ale mohli nalézt minimum M_2.



Obr. 11 Gradientní metody

Pozorný čtenář si jistě všiml, že výklad v této podkapitole se nápadně shoduje s popisem regresních metod z kapitoly předcházející. Tato podobnost není náhodná. Učení jako aproximace funkcí je výrazně ovlivněno statistickými metodami – to se týká především neuronových sítí. Největší rozdíly jsou tedy spíše v oblasti terminologické. Tab. 2 uvádí základní odlišnosti v používaných pojmech.

strojové učení	**statistika**
učení	odhadování parametrů
trénovací data	vzorek dat
příklad (z trénovací množiny)	pozorování
cílový atribut	závislá veličina
vstupní atribut	nezávislá veličina
chyba	residuál
váha (u neuronových sítí)	regresní koeficient

Tab. 2 Terminologické rozdíly mezi strojovým učením a statistikou

Literatura:

[Bruha, 1993] Bruha,I.: Machine learning: Empirical Methods. In Proc: Softwarový seminář SOFSEM'91, 1991

[Bruha, Berka, 2000] Bruha,I. - Berka,P.: Discretization and Fuzzification of Numerical Attributes in Attribute-Based Learning. In: Szcepaniak,P.S. - Lisboa,P.J.G. - Kacprzyk,J.: Fuzzy Systems in Medicine. Springer. 2000. ISBN 3-7908-1263-3.

[Klosgen, Zytkow, 1997] Klosgen,W. - Zytkow,J.: Knowledge Discovery and Data Mining. Tutorial Notes. PKDD'97. Trondheim.

[Kotek a kol., 1980] Kotek,Z. - Chalupa,V. - Brůha,I. - Jelínek,J.: Adaptivní a učící se systémy. SNTL, Praha, 1980.

[Michalski a kol., 1983] Michalski,R. – Carbonell,J. – Mitchell,T.: Machine Learning: An Artificial Intelligence Approach. Tioga Publ., 1983.

[Mitchell, 1997] Mitchell,T.: Machine Learning. McGraw-Hill. 1997. ISBN 0-07-042807-7

[Winston, 1992] Winston,P.H.: Artificial Intelligence. Addison Wesley. 1992. ISBN 0-20-153377-4