7 Strojové učení

7.1 Úvod – co je strojové učení?

Představme si, že jsme se stali svědky přistání létajícího talíře. Roboti, kteří z něj vystoupili, jsou nadáni schopností lidského uvažování a jsou vybaveni určitými základními povědomostmi o naší planetě, získanými pozorováním. Postrádají však mnohé znalosti našich středoškolských studentů, a právě ty se snaží v rozhovoru s námi doplnit.

Řekněme, že se jednomu z nich pokoušíme vysvětlit význam pojmu "pták". Nejlépe to provedeme na příkladu, za který zvolíme opodál sedícího kosa. Návštěvník si povšimne křídel, schopnosti létat, žlutého zobáku, zpěvu, popř. se od nás dozví, že kos snáší vajíčka. Upozorníme ho, že naopak foxteriér, který se prohání opodál, nikterak nemůže být za ptáka považován. Robot si ihned všimne, že pejsek nemá křídla. A má od této chvíle za to, že ptáci se od ostatních živočichů rozpoznají právě podle křídel.

V té chvíli mu zabzučí kolem hlavy moucha. "Hle, pták," řekne s porozuměním, my jej ovšem musíme zklamat. Pozná, že samotná křídla nestačí, když je má kdejaký hmyz. Hledá tedy jinou diskriminační proměnnou a také ji brzy najde. Mouše chybí kosův žlutý zobáček. Pták se tedy pozná podle křídel a žlutého zobáku.

O něco později proletí kolem vrána. Všechno se zdá být jasné: nemáš žlutý zobák, nejsi pták. Pes a moucha ho taky neměli. I nyní ovšem upozorníme robota na jeho omyl, ale to už nám ho začíná být trochu líto. Co to jenom je, co má vrána společného s kosem, aby to neměl pes ani moucha? Tak třeba tím, že nesnáší vajíčka, se od ptáků oddělil foxteriér. Ale čím se odlišuje moucha? Už to máme: Ta vajíčka, vážený, musejí mít skořápku! Muší vajíčka ji nemají. V tom to je.

Vysoko nad námi prozpěvuje skřivan. "Jestlipak snáší vajíčka se skořápkou?" zajímá se návštěvník. Samozřejmě že snáší! "Nu, tak to tedy musí být pták." A taky že je. Přikyvujeme a jsme rádi, že si skřivana nepovšimnul už před vránou. Určitě by za rozlišovací znak považoval zpěv. A jak by k tomu přišel třeba takový páv, viďte.

Uvedený příklad nám ilustruje situaci, v níž se běžně ocitají počítačové programy, jejich úkolem je odvozovat pojmové znalosti z předložených příkladů a protipříkladů. Této úloze, kterou nazýváme učení z příkladů, bylo v posledních letech věnováno mnoho pozornosti. Smysl je zřejmý. Program, který si dokáže z příkladů sám odvodit popisy či definice pojmů, je po svém "naučení" schopen provádět klasifikace. Perspektivní aplikace je možné hledat v počítačovém vidění, v porozumění přirozenému jazyku, v tvorbě znalostních bází pro expertní systémy, ale třeba i v hledání rychlých řešení úloh náročných na prohledávání stavového prostoru.

V následujícím odstavci si zadání úlohy strojového učení dále zpřesníme a pokusíme se nastínit obecnou filozofii této disciplíny. Teprve poté přejdeme k vysvětlení známých algoritmů, na nichž si některé základní pojmy budeme ilustrovat.

7.2 Základní pojmy strojového učení

Vraťme se k historce s mimozemšťanem. Oč v ní šlo? Byli jsme ve roli učitele, který předkládá žáku příklady ilustrující určitý pojem (koncept). Příklady jsou popsány pomocí predikátů typu ma_kridla, snasi_vejce, ma_zluty_zobak, apod. Žák se snaží nalézt takový popis pojmu "pták", který se bude hodit na všechny exampláře tohoto živočicha, a naopak se nehodí na žádný exemplář jiného živočicha. Řečeno odborně, hledá takový predikátový výraz, který je pravdivý pro všechny pozitivní příklady a nepravdivý pro všechny negativní příklady.

Podívejme se pozorněji na to, jak si žák počínal. Na počátku měl k dispozici jediný pozitivní příklad, a to kosa, popsaného konečnou množinou predikátů. Výskyt negativního příkladu mu umožnil nalézt ty predikáty, které odlišují ptáka od "ne-ptáka". Tyto predikáty (ma_kridla, ma_zluty_zobak, zpiva, snasi_vejce) nazveme **diskriminačními predikáty**. Vzhledem k tomu, že se mu však v daném případě diskriminační popis zdál příliš podrobný a zdlouhavý, vybral si žák pouze jeho část, provedl tedy jeho **generalizaci** čili zobecnění. Z několika možných zobecnění si víceméně náhodně vybral ma_kridla.

Následující negativní příklad (moucha) však ukázal, že toto zobecnění nebylo zvoleno nejšťastněji, protože neumožňuje diskriminovat mouchu od ptáka. Robot proto přikročil k **specializaci** popisu na konjunkci ma_kridla a ma_zluty_zobak. Je zřejmé, že specializace je opakem generalizace.

Příchod dalšího pozitivního příkladu však naznačil, že se žák dostal do slepé uličky. Není vidět žádná možnost generalizace nebo specializace, která by umožnila zahrnout do popisu vránu tak, aby byly diskriminovány všechny dosavadní negativní příklady. Proto tentokrát sáhne po jiném popisu hledaného pojmu a pokusí se navrhnout nový: snasi_vajicka_se_skorapkou. Tento popis již vyhovuje všem příkladům.

Seznámili jsme se se dvěma důležitými operátory strojového učení: generalizace a specializace. Podstata těchto operátorů může být buď induktivní, nebo deduktivní. **Dedukce** je takový způsob inference (odvozování), který zachovává pravdivost. **Indukce** je takový způsob inference, který uchovává nepravdivost.

Zamysleme se nad touto větou: "Jestliže pan A. pije, bude opilý". To je typický induktivní úsudek. Jestliže levá strana výroku není pravdivá, není pravdivá ani pravá strana výroku. Kdo nepije, ten se neopije. Nepravdivost levé strany má za následek nepravdivost pravé strany úsudku. Naproti tomu samotný fakt pití nemusí nutně vést k opilosti. Pravdivost levé strany nevede automaticky k pravdivosti pravé strany. Pravdivost pravé strany je tedy pouze předpokládána.

V klasických vědních disciplínách jsme ovšem zvyklí spíše na deduktivní odvozování, zachovávající pravdivost. Rozpojený spínač elektrického obvodu má vždy za následek zhasnutí žárovky. Ta ovšem může zhasnout i z jiného důvodu, například kvůli přepálenému vláknu. Pravdivost levé strany se zachovává, nicméně její nepravdivost nemusí mít na pravdivostní hodnotu pravé strany přímý vliv. (Ovšem pozor, matematická

indukce je svou povahou deduktivní odvozovací mechanismus, neboť zachovává pravdivost.)

V příkladu s mimozemšťanem byla generalizace induktivní a specializace deduktivní (nad tímto tvrzením nechť se čtenář zamyslí sám). Existují však i případy opačné: deduktivní generalizace – žije-li někdo v Praze, žije i v Evropě, a induktivní specializace – vyvozujeme-li z toho, že někdo bydlí v Evropě, ten závěr, že bydlí v Praze. Obě mohou mít své opodstatnění, i když ovšem nejsou příliš běžné.

Dále je vhodné se zmínit o rozdílu mezi abstrakcí a konkretizací. O **abstrakci** hovoříme tehdy, jestliže z původního popisu ubereme část informace, například, jesliže místo "pan A. měří 193 cm", řekneme "pan A. je vysoký". **Konkretizace** je opakem abstrakce, tedy znamená doplnění informací – například větší přesností nebo přidáním podrobností. Čtenáři doporučujeme promyslet si rozdíl mezi abstrakcí a generalizací, popřípadě mezi konkretizací a specializací. Abstrakce, popř. konkretizace, se vztahuje k množství informace obsažené v tvrzení. Naproti tomu generalizace, popř. specializace, se vztahuje k velikosti množiny objektů, pro něž dané tvrzení platí. Deduktivní generalizaci lze ovšem považovat za abstrakci.

Přemýšlivý čtenář si možná uvědomil, že problém učení z příkladů, jak jsme jej zde nastínili, je problémem **prohledávání stavového prostoru** všech možných popisů daného pojmu. Při prohledávání, které se samozřejmě může opírat o kteroukoli z heuristik uvedených v odstavci 7.3, lze uplatnit šest operátorů: induktivní generalizaci, deduktivní specializaci, deduktivní generalizaci, induktivní specializaci, abstrakci a konkretizaci.

Zadání je tedy definováno takto:

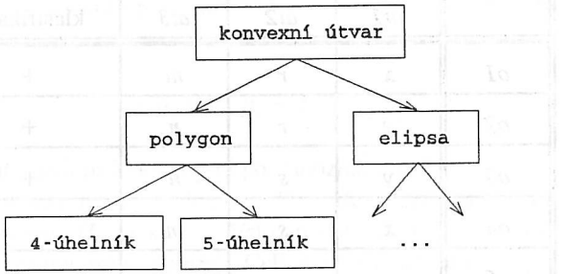
je dáno: -množina příkladů,
-popis příkladů, vyjádřený v předem určeném descripčním jazyku,
-klasifikace příkladů na pozitivní a negativní příklady určitého pojmu,
-znalosti omezující prostor prohledávání a usnadňující aplikaci operátorů,
najdi: -popis daného pojmu.

Vzhledem k tomu, že prohledávaný prostor je v mnoha praktických případech neúměrně velký, využíváme často **znalostí** *(background knowledge)*, které nám usnadňují nasazování operátorů, nebo nějakým způsobem omezují prohledávaný prostor. Znalosti mohou mít podobu produkčních pravidel reprezentujících heuristiky potřebné k prohledávání, často však mají tvar **generalizačních stromů**, které jsou pro strojové učení velice typické. Příklad generalizačního stromu vidíme na obr. 7.1. Znalosti v této podobě nejenom usnadňují nasazování uvedených operátorů, ale mohou odvozovací proces zaměřovat určitým směrem.

Dosud jsme se tvářili, že nalezení přesného popisu je vždy možné a víceméně bezproblémové. V praxi tomu tak pokaždé není. Může se stát, že deskripční jazyk není dostatečně bohatý, aby jednoznačně rozlišil všechny objekty předkládané jako pozitivní a negativní příklady, popř. mohou být v datech obsaženy chyby (například chybná klasifikace), nebo může být celý proces prohledávání tak náročný, že je nutné se spokojit s nedokonalým řešením. V takovém případě bude výsledný popis nepřesný a my se musíme rozhodnout, jaký druh nepřesnosti nám více vyhovuje. Zda chceme popis konzistentní anebo kompletní.

Konzistentní je takový popis, který jednoznačně diskriminuje pozitivní příklady od negativních, i když třeba není schopen pokrýt úplně všechny pozitivní příklady.

Kompletní je takový popis, který pokrývá všechny pozitivní příklady i za cenu, že spolu s nimi pokryje i některé negativní příklady.



Obr.7.1 Generalizační strom.

Vhodným kritériem pro posuzování a porovnávání různých přístupů k učení je **přesnost** výsledného popisu. Použijeme-li popis pro klasifikaci nových objektů (které nebyly použity při učení), udává přesnost popisu procentuální úspěšnost této klasifikace. Jestliže náš popis správně oklasifikoval patnáct z dvaceti objektů, hovoříme o přesnosti 75%. Dalším kritériem, o němž je vhodné se zmínit, je **rychlost** učení, měřená dobou či počtem příkladů nutných pro dosažení určité přesnosti. Zpravidla také usilujeme o co nejjednodušší výsledný popis, který nám umožní snadnou interpretaci.

Je třeba poznamenat, že scénka popsaná v odst.7.1 představuje takzvané inkrementální učení, kdy příklady přicházejí jeden po druhém. V této knize se budeme zabývat **neinkrementálními (dávkovými) metodami**, které předpokládají přítomnost všech příkladů od začátku učení. Dávkové algoritmy jsou základní, zatímco inkrementální algoritmy jsou zpravidla jejich odvozené varianty. Inkrementální přístupy bývají rychlejší, ale méně přesné.

7.3 Induktivní tvorba rozhodovacích stromů

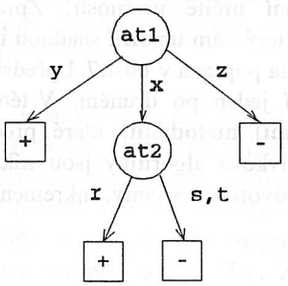
Rámcová metoda učení z příkladů, tak jak byla popsána v minulé kapitole, je velmi obecná a pro konkrétní aplikace ne vždy vhodná. Byly však vyvinuty algoritmy, které učení usnadňují. Mezi nejoblíbenější patří *TDIDT (Top-Down Induction of Decision Trees* –indukce rozhodovacích stromů shora dolů) a ID3 (Quinlan, 1986). Jak název napovídá, výsledný popis má tvar rozhodovacího stromu, podobného tomu, který je pro ilustraci uveden na obr.7.2. Vidíme, že listy stromu obsahují ohodnocení, zatímco ostatní uzly obsahují testy hodnot atributů. Při klasifikaci procházíme stromem shora dolů a provádíme

postupně testy předepsané v uzlech. Výsledky jednotlivých testů určují, kterou větví budeme pokračovat.

Tabulka 7.1. Definice a klasifikace vzorové množiny příkladů popsaných pomocí tří atributů

	at1	*at2*	*at3*	klasifikace
o1	*x*	*r*	*m*	+
o2	*y*	*r*	*n*	+
o3	*y*	*s*	*n*	+
o4	*x*	*s*	*m*	-
o5	*z*	*t*	*n*	-
o6	*z*	*r*	*n*	-

Zjednodušme si poněkud zadání. Budeme předpokládat, že příklady jsou popsány pomocí atributů, což jsou vlastně funkce s jedním argumentem (například barva, velikost, stáří). Dále předpokládejme, že nejsou k dispozici žádné předběžné znalosti, které by nám učení usnadnily. Metoda TDIDT má dvě fáze: vytváření rozhodovacího stromu a zjednodušování (prořezávání) rozhodovacího stromu. Věnujme se nejdříve první z nich.



Obr.7.2 Rozhodovací strom vytvořený z příkladů uvedených v tab.1.

Vytváření rozhodovacích stromů

Nejprve vyhledáme takový atribut, který v sobě nese největší množství informace, čili který nejlépe diskriminuje mezi kladnými a zápornými příklady. Tento atribut se stane kořenem stromu.

V dalším kroku si rozdělíme množinu příkladů na tolik podmnožin, kolik je hodnot kořenového atributu. V každé z podmnožin jsou příklady s jedinou hodnotou tohoto atributu. Poté vyhledáme v každé z těchto podmnožin další nejvýznamnější atribut a takto rekurzívně pokračujeme, dokud nevyčerpáme buď atributy, nebo příklady, anebo dokud není splněno nějaké předem dané kritérium ukončení.

Pro výběr nejvýznačnějšího atributu existují různá kritéria. Nejběžnější je měření množství informace pomocí entropie. Víme, že podle Shannonovy věty platí pro entropii j-té podmnožiny vztah

$$H_j = - p_1 \log_2 p_1 - p_2 \log_2 p_2 , \qquad (7.1)$$

kde p_1 je poměr pozitivních příkladů v j-té podmnožině k celkovému počtu prvků v této podmnožině a p_2 je poměr negativních příkladů v j-té podmnožině k celkovému počtu prvků v této podmnožině. H_j je kladné číslo, neboť p_1, p_2 jsou z intervalu $<0;1>$, a tudíž příslušné logaritmy jsou záporné. Celková velikost entropie je dána váženým součtem entropií jednotlivých podmnožin

$$H = \sum_{j=1}^{K} P_j H_j , \qquad (7.2)$$

kde K je počet podmnožin indukovaných daným atributem, H_j je entropie j-té podmnožiny a P_j je poměr velikosti j-té podmnožiny k množině všech příkladů.

Je zřejmé, že zvolíme takový atribut, pro který je hodnota $H = \sum P_j H_j$ minimální. Při konstrukci rozhodovacího stromu (obr.7.2) z dat obsažených v tabulce 7.1 byly použity tyto výpočty:

$$H(at1) = \frac{1}{3}H_x + \frac{1}{3}H_y + \frac{1}{3}H_z = \frac{1}{3} + 0 + 0 = \frac{1}{3},$$

$$H(at2) = \frac{1}{2}H_r + \frac{1}{6}H_t + \frac{1}{3}H_s \doteq 0{,}46 + 0 + 0{,}33 \doteq 0{,}8 ,$$

$$H(at3) = \frac{2}{3}H_n + \frac{1}{3}H_m = \frac{2}{3} + \frac{1}{3} = 1 ,$$

kde například

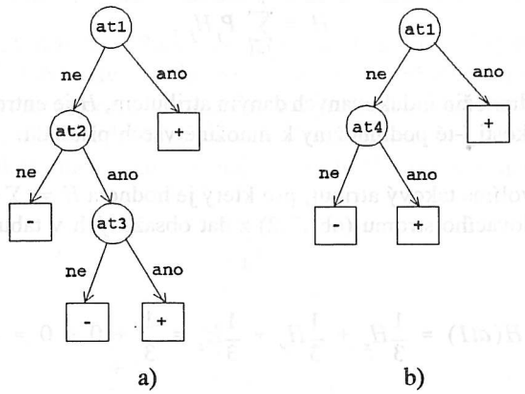
$$H_x = - [\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}] = 1 .$$

Atribut *at1* má nejmenší entropii (rovnou 1/3), a proto byl vybrán za kořen rozhodovacího stromu. Lze dále ukázat, že v podstromu definovaném hodnotou $at1 = x$ má atribut *at2* menší entropii než *at3*.

Prořezávání rozhodovacích stromů

V praxi máme zpravidla snahu výsledný strom nějakým způsobem zjednodušit, což znamená snížit počet jeho větví. Takové zjednodušení nazýváme **prořezáním.** Kromě toho, že se jednodušší stromy snáze interpretují, dosahují také často větší přesnosti. Je to způsobeno problémem zvaným *"overfitting"*, čili neúměrná přesnost stromu. Musíme si uvědomit, že v běžné praxi příklady použité pro učení obsahují šum. Učící se systém to ovšem sám od sebe a priori neví a má proto snahu odvodit zvláštní větve stromu i pro chybné příklady. Protože však předpokládáme, že chybných příkladů je daleko méně než správných, lze očekávat, že vhodným prořezáním stromu dojde k nápravě.

Předpokládejme, že z nějakého uzlu X vycházejí pouze takové větve, které jsou zakončeny listovými uzly s učitelovým ohodnocením ("+" nebo "−"). Nechť n je počet všech příkladů, které skončily v některém z těchto listů, a nechť e je počet příkladů, které skončily v některém z negativně ohodnocených listů. Nahraďme podstrom s kořenem v uzlu X pozitivně ohodnoceným listem X. Pokud použijeme takto vzniklý strom ke klasifikaci týchž příkladů, které byly použity při učení, dosáhneme přesnosti $ac = 1 - e/n$ (máme zde na mysli přesnost, kterou lze docílit na týchž datech, která byla použita pro učení).



Obr.7.3 Zjednodušení rozhodovacího stromu pomocí booleovských funkcí.

Na náhradě podstromů listem s jediným ohodnocením je založen jednoduchý způsob prořezávání stromů. Nejdříve definujeme určitou prahovou hodnotu přesnosti ac_0, například $ac_0 = 0{,}90$. Poté procházíme všechny vnitřní uzly stromu tak, že neprve projdeme uzly na nejnižší úrovni a postupujeme směrem ke kořenu. Pro každý z těchto uzlů pak spočítáme hodnotu ac na příkladech, které byly použity k učení. Konečně ty uzly, pro něž je $ac > ac_0$, nahradíme jediným listem s tím ohodnocením, jež bylo v daném podstromu nejčetnější.

Někdy však bychom rádi rozhodovací strom zjednodušili bez ztráty informace v něm obsažené, například tehdy, když jsme si jisti, že příklady neobsahovaly žádný šum a nehrozí "overfitting". Pokud jde o booleovský strom (každý z atributů může nabývat pouze jedné ze dvou hodnot), můžeme použít způsob naznačený na obr.7.3. Strom

uvedený na obr.7.3b vznikl ze stromu na obr.7.3a tak, že atributy *at2* a *at3* byly nahrazeny atributem $at4 = at2 \wedge at3$.

Metody učení pomocí rozhodovacích stromů jsou poměrně dobře rozpracovány a v současné době existuje celá řada jejich variant. Například míra informačního zisku přinášeného jednotlivými atributy nemusí být vždy měřena jen pomocí entropie, pravděpodobnosti p_i lze počítat pomocí důmyslnějších vzorců než pomocí pouhé četnosti výskytu. Existují také různé způsoby prořezávání stromů. V nedávné době byly také nalezeny metody odvozování rozhodovacích stromů za situace, kdy některé z atributů mohou nabývat spojitých numerických hodnot.

Také byla publikována řada úspěšných pokusů (Núñez, 1991) rozšířit základní algoritmus o schopnost využívat bázi znalostí, tedy neopírat se při určování pořadí významu atributů pouze o informační zisk, který v praktické aplikaci nemusí nutně hrát nejvýznamější roli. Tak třeba při medicínské diagnóze jsou některá vyšetření velice nákladná nebo bolestivá pro pacienta, a lékař se proto záměrně snaží vystačit s jednoduššími testy, i když ví, že se vystavuje riziku chybné diagnózy.

Proces tvorby rozhodovacího stromu je svou povahou induktivní. Ze stávajících příkladů usuzujeme na vlastnosti příkladů, které přijdou teprve v budoucnosti. Postupný rozvoj rozhodovacího stromu je procesem specializace, popř. konkretizace. Je možné jej také chápat jako prohledávání prostoru všech možných rozhodovacích stromů metodou prohledávání do hloubky (viz odst.2.2). Hlavní heuristika řídící toto prohledávání se opírá o předpoklad, že je lépe se nejprve zabývat těmi atributy, které nesou největší množství informace, a mají tedy nejmenší entropii (zmínili jsme se však, že toto nemusí být vždy pravidlem). Prořezávání rozhodovacího stromu je, jak již bylo řečeno, procesem generalizace.

7.4 Učení z klasifikovaných příkladů

Algoritmus TDIDT velmi jednoduchý a nenáročný, postrádá však některé typické vlastnosti programů umělé inteligence. Například využití dříve nabytých znalostí není nikterak snadné. Ukažme si nyní typičtější metodu, která už využívá celý arzenál mechanismů, které byly osvětleny v úvodní části.

Budeme se zabývat **algoritmem AQ** poprvé publikovaným v práci (Michalski, 1969), který je opět určen pro učení z příkladů popsaných pomocí hodnot atributů. Jeho výstupem jsou produkční pravidla. Lze jej shrnout do pěti bodů:

1. Rozděl množinu příkladů na dvě podmnožiny: množinu *PE* obsahující pouze pozitivní příklady a množinu *NE* obsahující pouze negativní příklady;
2. Vyber z množiny *PE* náhodně jeden příklad a označ jej *s* (budeme mu říkat jádro);
3. Nalezni všechny maximální generalizace popisu jádra *s*, přičemž limitem je množina *NE*, generalizace popisu *s* nesmí pokrýt žádný negativní příklad;
4. Podle vhodného preferenčního kritéria vyber nejlepší z těchto popisů a zařaď jej do množiny popisů;
5. Pokud množina popisů pokrývá všechny prvky *PE*, ukonči práci.

Výsledným popisem je pak disjunkce všech nalezených popisů. V opačném případě vyber nové jádro z dosud nepokrytých pozitivních příkladů.

Maximální generalizace popisu je taková generalizace, která pokrývá co největší počet příkladů. Představme si, že máme k dispozici pouze jeden pozitivní a jeden negativní příklad, jako je tomu na obr. 7.4. Nalezneme atribut, v němž se pozitivní a negativní příklad od sebe liší. Jakýkoli popis, který obsahuje jinou hodnotu tohoto atributu, než jaká byla u negativního příkladu, může být považován za přípustnou generalizaci pozitivních příkladů. Za nejobecnější popis (maximální generalizace) považujeme ten popis, o němž se domníváme, že vyhovuje co největšímu množství pozitivních příkladů, a to i těch, které nebyly předloženy k učení.

příklad	*at1*	*at2*	*at3*	klasifikace
p1	*x*	*r*	*m*	+
p2	*y*	*r*	*n*	–

První maximální generalizace: $at1 \neq y$,
Druhá maximální generalizace: $at3 \neq n$.

Obr.7.4 Maximální zobecnění.

Preferenční kritérium může být založeno na nákladové funkci tak, že přednost je dávána těm atributům, jejichž hodnoty jsou snadno dostupné. Tak třeba u medicínských dat je tělesná teplota pacienta dostupnější ("levnější") než sedimentace, kvůli které je třeba odebrat vzorky krve, a sedimentace je zase daleko levnější než řekněme tomografické vyšetření. Relativní dostupnost jednotlivých atributů může být součástí znalostí, které mají usnadňovat učení.

Ilustrujme nyní celý algoritmus na příkladu. Předpokládejme, že máme k dispozici tři pozitivní a tři negativní příklady nějakého pojmu, tak jak jsou uvedeny v tab.7.2. Předpokládejme, že náš deskripční jazyk nezná negaci. Potom analýza pojmu "+" pomocí algoritmu AQ proběhne asi takto:

Tabulka 7.2. Příklady rozdělené do množin *PE* a *NE*

příklady		*at1*	*at2*	*at3*	klasifikace
PE	*o1*	*x*	*r*	*m*	+
	o2	*y*	*r*	*n*	+
	o3	*y*	*s*	*n*	+
NE	*o4*	*x*	*s*	*m*	–
	o5	*z*	*t*	*n*	–
	o6	*z*	*r*	*n*	–

První jádro: náhodně jsme zvolili *o2*.
Nejobecnějším popisem tohoto jádra v daném deskripčním jazyku je $(at1 = y)$. Tento popis pokrývá příklady *o2* a *o3*, ale nepokrývá *o1* ani žádný z negativních příkladů.

Druhé jádro (vybrané z dosud nepokrytých pozitivních příkladů): *o1*.
Z hlediska obecnosti existují dvě rovnocené maximální generalizace

$$(at1 = x) \wedge (at2 = r),$$
$$(at2 = r) \wedge (at3 = m).$$

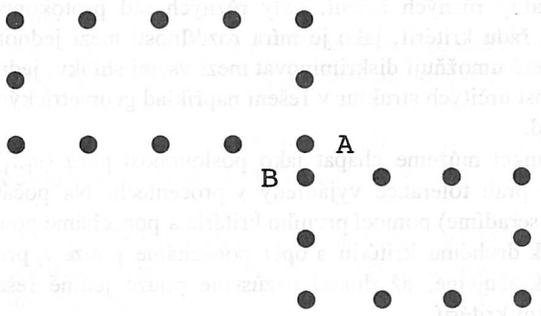
Nechť podle nějakého kritéria vyhovuje lépe první z těchto dvou popisů (například proto, že atribut *at1* se snáze měří). Můžeme proto uzavřít, že pojem "+" je nejlépe popsán takto

$$(at1 = y) \vee [\![(at1 = x) \wedge (at2 = r)]\!].$$

7.5 Učení z neklasifikovaných příkladů

Vraťme se ještě jednou k našemu příkladu s létajícím talířem. Co když budou mít návětěvníci tu smůlu, že nepotkají nikoho, kdo by jim s klasifikací poradil?

V takovém případě budou v situaci, v níž jsou často naši pozemští vědci, kteří se pokoušejí vytvořit nějakou taxonomii, čili klasifikační schéma. Nemusí vždy jít pouze o živočišnou říši, vzpomeňme si třeba na takového Mendělejeva. Před potřebou vhodně si strukturovat výsledky pozorování stojíme často i v běžném denním životě, a tato činnost je právem považována za projev inteligence. Mechanické přístupy založené například na statistice pokaždé neobstojí.



Obr.7.5 Inteligentní hledání struktury.

Podívejme se třeba na obr.7.5. Nejbližším bodem bodu *A* je *B*, a proto téměř všechny statistické přístupy ke shlukové analýze zařadí *A* do stejné kategorie s *B*. Prostým okem

však snadno zjistíme, že na obrázku jsou dva obdélníky a každý z bodů patří do jiného z nich.

Je tedy zřejmé, že pokud chceme klasifikaci provést inteligentně, nevystačíme pouze s metrikami měřícími vzdálenosti mezi objekty. Příslušnost objektu do toho kterého shluku je funkcí jeho souřadnic (například atributů), prostředí (tedy vlastností ostatních objektů), a především předpokládaných vlastností cílových konceptů. Tak například v úloze z obr.7.5 nám při inteligentním zařazování bodů *A*, *B* do různých shluků pomohl předpoklad, že shluky vytvářejí geometrické obrazce v rovině. Jinými slovy, opět potřebujeme určité znalosti.

Pro metodu, kterou zde stručně rozvedeme, připadají do úvahy dva typy znalostí: znalosti o přípustných generalizacích a specializacích a znalosti o cílech klasifikace (o přípustných či preferovaných cílových pojmech). Samotná metoda pak vychází z algoritmu, který již známe. Podstatou je hledání vhodných generalizací jader a snaha o co nejdokonalejší pokrytí celého souboru příkladů.

1. Z množiny všech příkladů náhodně vybereme k jader (číslo k zadává uživatel). Označme je $e_1,\ldots,e_k$.

2. Pro každé jádro hledáme dostatečně obecný popis, který by jej odlišil od všech ostatních jader. Každý takový popis nazveme protokonceptem. Máme tedy tolik protokonceptů, kolik je jader. Všimněte si, že v této chvíli se protokoncepty budou protínat.

3. Provedeme optimalizaci protokonceptů tak, aby byly disjunktní. Jako výsledek dostaneme množinu konceptů. Zpravidla existuje několik různých řešení (množin konceptů), z nichž pomocí vyhodnocovací funkce vybereme to nejlepší.

4. Není-li splněno zvolené kritérium ukončení, vybereme nová jádra a přejdeme zpět ke kroku 2. Jádra volíme tak, že z každého stávajícího protokonceptu vybereme jedno.

Poznámka k vyhodnocovací funkci

Vyhodnocovací funkce je součástí výchozích znalostí zadaných uživatelem a slouží k porovnání kvality různých řešení, tedy různých sad protokonceptů (shluků). Může zahrnovat celou řadu kritérií, jako je míra rozdílnosti mezi jednotlivými shluky, počet proměnných, které umožňují diskriminovat mezi všemi shluky, jednoduchost výsledného řešení, přítomnost určitých struktur v řešení například geometrických obrazců v příkladu na obr.7.5, apod.

Výslednou funkci můžeme chápat jako posloupnost párů $(c_1,t_1),(c_2,t_2),\ldots$, kde c_i je kritérium a t_i je práh tolerance vyjádřený v procentech. Na počátku jednotlivá řešení vyhodnotíme (a seřadíme) pomocí prvního kritéria a ponecháme pouze prvních t_1 procent. Poté přejdeme k druhému kritériu a opět ponecháme pouze t_2 procent nejúspěšnějších řešení a tak pokračujeme, až dokud nezůstane pouze jediné řešení, anebo dokud se nevyčerpá seznam kritérií.

Poznámka k ukončovacímu kritériu

Viděli jsme, že proces hledání nejlepšího rozdělení příkladů do shluků probíhá ve smyčce, nejdříve náhodně zvolíme k jader, kolem těchto jader vytvoříme několik konceptů, vybereme nová jádra, vytvoříme nové koncepty a tak dále, dokud není splněno

kritérium ukončení. Toto kritérium je zadáno uživatelem a jeho podstatou je sledování, zda současné řešení je lepší (podle vyhodnocovací funkce) než minulé řešení. Pakliže po několik iterací (tento počet je zadán uživatelem) nepozorujeme žádné zlepšení, proces ukončíme a vybereme to řešení, které bylo nejlépe ohodnoceno.

Tabulka 7.3. Množina příkladů, z nichž chceme utvořit dva shluky

příklad	*at1*	*at2*	*at3*
o1	*a*	2	110
o2	*a*	4	100
o3	*b*	2	9
o4	*b*	3	10
o5	*c*	5	20
o6	*c*	4	15
o7	*a*	5	200
o8	*b*	4	50

Celou proceduru si podrobněji rozebereme na příkladu. Předložme učícímu se programu neklasifikované příklady z tab.7.3. Kromě příkladů jsou k dispozici znalosti usnadňující generalizace popisů. Tyto znalosti říkají, že atribut 1 může nabývat pouze symbolických hodnot *a*,*b*,*c*, atribut 2 nabývá celočíselných hodnot od 2 do 6 a atribut 3 nabývá celočíselných hodnot od 1 do 300, přičemž hodnoty pod 30 jsou považovány za malé, hodnoty nad 150 jsou považovány za velké a všechny ostatní hodnoty jsou střední. Chceme vytvořit dva shluky.

Náhodně vybereme dvě jádra, řekněme *o2*,*o5*. Provedeme maximální generalizaci jejich popisu, tak aby popis *o2* nepokrýval *o5* a naopak. Dostaneme dva protokoncepty

$$\text{protokoncept}(o2)\text{: } (at1 \neq c) \vee (at2 \neq 5) \vee (at3 \neq 20),$$
$$\text{protokoncept}(o5)\text{: } (at1 \neq a) \vee (at2 \neq 4) \vee (at3 \neq 100).$$

Vidíme, že všechny zbývající příklady *o1*, *o3*, *o4*, *o6*, *o7* a *o8* vyhovují oběma těmto popisům, oba protokoncepty se tedy silně překrývají. Proto nyní vytvoříme z protokonceptů disjunktní koncepty.

Utvořme seznam L ze všech příkladů, které jsou pokryty více než jedním protokonceptem. Vezměme první prvek L, jímž je *o1*, vyjměme jej z tohoto seznamu a vložme jej nejprve do prvního shluku, který tedy nyní obsahuje prvky *o1*, *o2*. Zapomeňme na popis příslušného protokonceptu a vytvořme nový, co nejobecnější popis tak, aby pokrýval prvky *o1* a *o2*, ale aby se nepřekrýval s druhým protokonceptem (se žádným ze zbylých protokonceptů). Nyní máme k dispozici tyto popisy:

koncept vytvořený kolem $o1,o2$:$(at1 = a) \wedge (at2 = 2..4) \wedge (at3 =$ stredni),
koncept vytvořený kolem $o5$:$(at1 \neq a) \vee (at2 \neq 4) \vee (at3 \neq 100)$.

Podobně, vložíme-li příklad $o1$ do druhého protokonceptu místo do prvního, dostaneme následující popisy:

koncept vytvořený kolem $o2$:$(at1 \neq c) \vee (at2 \neq 5) \vee (at3 \neq 20)$,
koncept vytvořený kolem $o1,o5$:$(at1 = a \vee c) \wedge (at2 = 2..5) \wedge (at3 =$ male $\vee$ stredni).

Předpokládejme, že podle vyhodnocovacích kritérií lépe vyhovuje první dvojice popisů. Začleníme tedy $o1$ do tohoto konceptu a celou proceduru opakujeme s příkladem $o4$ a tak dále, dokud se nevyprázdní seznam L. Ve chvíli, kdy se seznam vyprázdní, máme dva koncepty, z nichž jeden pokrývá řekněme $o1$, $o2$, $o7$ a druhý $o3$, $o4$, $o5$, $o6$, $o8$. Toto řešení uložíme do paměti.

Dospěli jsme k bodu, kdy se má podle ukončovacího kritéria rozhodnout, zda se má v hledání pokračovat či nikoli. Vzhledem k tomu, že proběhla teprve první iterace, je odpověď kladná.

Vybereme nová dvě jádra, každé z jiného konceptu vytvořeného v minulé iteraci. Například $o1$, $o6$. Nyní opakujeme celou proceduru a získáme jiné řešení. Porovnáme toto řešení s předchozím pomocí vyhodnocovací funkce. Je-li toto řešení lepší než to minulé, uložíme jej do paměti místo řešení minulého a pokračujeme třetí iterací, potom čtvrtou iterací, až dokud není splněno kritérium ukončení.

7.6 Jiné metody

Na poměrně malém prostoru jsme se zde přirozeně mohli zabývat pouze některými nejznámějšími přístupy. Mnoho jiných, neméně zajímavých, jsme museli zcela opominout. Přesto si zaslouží alespoň telegrafickou zmínku.

Především se to týká přístupů, které byly inspirovány biologickými procesy. Sem řadíme umělé **neuronové sítě** (viz kapitola 9) a skupinu genetických algoritmů. Cíl **genetických algoritmů** je velice prostý, hledáme, pro jakou kombinaci hodnot atributu dosáhne nějaká ohodnocovací funkce maximum. Definičním oborem mohou být například čísla získaná převodem z binárních řetězců, kde každý bit představuje jeden booleovský atribut. Jestliže uvažujeme atributy "hezká", "pracovitá" a "mladá", potom pohledná líná dívka bude binárním řetězcem vyjádřená jako "101", což je v desítkové soustavě číslo 5. Každé přirozené číslo je tedy chápáno jako jedna z možných kombinací hodnot atributu (existuje ovšem celá řada různých variací). Ohodnocovací funkce F může třeba vycházet z informačního obsahu, jako tomu bylo u algoritmu TDIDT.

Samotný genetický algoritmus hledání maxima vychází z myšlenky existence určitého populačného vzorku, například 20 různých kombinací hodnot atributu. Pro každý exemplář určíme hodnotu F. Exempláře s nízkou hodnotou F zahynou. Ty, co přežijí, si mezi sebou v náhodně vytvořených párech vymění část své "genetické informace", například si prohodí posledních 5 bitů příslušných binárních řetězců. Navíc je nepatrné procento nově vzniklého pokolení vystaveno "mutaci", tedy náhodnému překlopení jednoho bitu z 1 na 0 či naopak. Pro novou populaci opět najdeme hodnotu F, necháme odumřít "slabé

jedince", kdežto silným umožníme vstup do reprodukčního procesu, podobně jako je tomu v přírodě. Proces zpravidla probíhá přes určitý počet pokolení.

V posledních letech byly předmětem intenzívního výzkumu metody **deduktivního učení**, jejichž cílem je vyextrahovat z rozsáhlých znalostních bází ta pravidla, která mají bezprostřední vztah k určitému konceptu v dané aplikaci. Chceme-li odlišit lokomotivu od nákladního velblouda, má pro nás pramalý význam informace o počtu pák v řídicí kabině, barvě lokomotivy a jménu strojvůdcovy babičky. Prohledávání neúměrně velkých znalostních bází se naopak nepříznivě promítá do rychlosti daného algoritmu.

Jiným způsobem omezení prohledávacího prostoru je využívání **analogií**. Tak třeba při studiu elektrického pole nám pomůže, když víme, že elektrické pole se chová analogicky magnetickému poli. Proudění kapalin se zase řídí zákony, které jsou analogické Kirchhoffovým zákonům pro elektrické obvody. Schopnost analogií zmenšit prohledávací prostor je skutečně veliká a mnoho vědců se domnívá, že právě jejich nalézání a využívání je klíčem k podstatě inteligentního chování. Přes značnou pozornost, která je tomuto problému v současné době věnována, se však zdá, že doba skutečné slávy učících se algoritmů založených na analogiích teprve přijde.

Metody, které jsme zde uvedli, vesměs předpokládají, že z určitého množství příkladů se v procesu učení odvodí znalosti ve formě pravidel nebo rozhodovacích stromů a poté se příklady zapomenou (vymažou z paměti). Někdy je však vhodné alespoň některé příklady v paměti uchovávat, například pro usnadění pozdějších aktualizací znalostí. Krajním přístupem je pak **učení založené na případech** (*Case Based Reasoning*), kdy v se paměti uchovávají pouze rozsáhlé případy (příklady) a při usuzování se hledají podobnosti mezi případy v paměti a studovanými vnějšími případy. Učení spočívá v rozpoznání obzvlášť typických případů, které se uloží.

7.7 Závěr

Strojové učení je v současné době chápáno jako disciplína umělé inteligence. Její základní technikou (ať zjevně nebo skrytě) je prohledávání stavového prostoru. K charakteristickým rysům patří využívání znalostí, práce se symbolickými či strukturovanými proměnnnými či aplikace moderních poznatků z oboru nestandardních logik. Úspěšně byly vyzkoušeny také algoritmy inspirované biologickými systémy.

Nejtypičtější aplikací strojového učení je pomoc při získávání znalostí pro expertní systémy, kde bylo dosaženo výrazných úspěchů v podobě zkrácení doby nutné pro tvorbu a ladění báze znalostí. Využití strojového učení spočívá v doplnění klasických metod získávání znalostí, kdy se expert ve spolupráci se znalostním inženýrem snaží formulovat pravidla, o možnost, že expert demonstruje některé postupy či pojmy na příkladech. Běžný je takový způsob, že systém nejprve vytvoří znalostní bázi z velkého množství příkladů a expert poté bázi upřesňuje tím, že odstraňuje pravidla vzniklá spíše náhodně, jiná pravidla přeformulovává či zjednodušuje atd. Expert může učícímu se mechanismu účinně napomoci dodáním vhodných výchozích znalostí pro učení, například generalizačních stromů.

Kromě expertních systémů se v současné době množí pokusy využít strojové učení v takových disciplínách, jako je porozumění přirozenému jazyku či počítačové vidění.

Dá se říci, že strojové učení patří mezi nejstarší disciplíny matematické informatiky. Naprosto převažující část úkonů, které dnes počítače provádějí, musela být nejprve pracně naprogramována. Právě programování tvoří v dnešní době úzký profil zabraňující ještě výraznějšímu pokroku v nasazování výpočetní techniky, a to nejenom z kapacitních důvodů, ale především pro nákladnost programátorských prací. Proto se již od padesátých let hledají způsoby, jak tvorbu softwaru automatizovat. Strojové učení založené na umělé inteligenci je jednou z metod této automatizace.

7.8 Bibliografické poznámky

Nejznámější učebnice věnovaná strojovému učení, která pokrývá celou problematiku úvodního kursu, je kniha (Kondratoff, 1988). Je ovšem třeba poznamenat, že začátečníkům se tato kniha čte velmi obtížně. Čtenářům doporučujeme "bibli" strojového učení, což je trilogie obsáhlých sborníků vybraných prací předních světových specialistů (Michalski a kol., 1983), (Michalski a kol., 1986), (Kodratoff, Michalski, 1990).

Z časopisů je třeba se zmínit o "Artificial Intelligence" a "Machine Learning".

Vážný zájemce by měl sledovat sborníky každoroční celosvětové konference o strojovém učení **"International Conference on Machine Learning"** a každoroční evropské konference **"European Working Session on Learning"**, jakož i jiné významné události umělé inteligence.

V dalším textu uvádíme některé základní články, které jsou dobrým úvodem do různých přístupů k strojovému učení.

Metoda **indukce rozhodovacích stromů** je rozebrána například v pracích (Quinlan, 1986, 1990). **Algoritmus AQ** by poprvé představen v přednášce (Michalski, 1969), některé jeho pozdější modifikace jsou vysvětleny v článku (Michalski, 1990). **Učení z neklasifikovaných příkladů** je náplní mnoha studií (Fisher, 1987, Gennari, 1989, Michalski, 1983), metodu, kterou jsme uvedli, lze nalézt v příspěvku (Michalski, 1983).

Článek (Mitchell a kol., 1986) se zabývá nejznámějším přístupem k **deduktivnímu učení**. Publikace (Aleksander, Norton, 1990), (Hinton, 1990) jsou dobrým úvodem do učení pomocí umělých **neuronových sítí** (o nich je pojednáno na jiném místě učebnice). Možnosti využití **genetických algoritmů** jsou rozebírány v práci (DeJong 90). Učení pomocí **analogií** je náplní článku (Hall, 1989). Zajímavý přístup k učení, založený na vyvozování z **knihovny případů**, je popsán v referátu (Kolonder a kol., 1985).

Práce (Kodratoff, 1991), (Michalski, 1983), (Michalski, 1991) obsahují zajímavé úvahy vztahující se k rámcové filozofii učení.

Literatura

Aleksander I., Morton H.: *An Introduction to Neural Computing.* Chapman and Hall, London, 1990.

Goldberg D.E.: *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading M.A., 1989.

Fisher D.: *Knowledge Acquisition Via Incremental Conceptual Clustering.* In:Machine Learning 2 (1987), 139-172.

Gennari J.H., Langley P., Fisher D.: *Models of Incremental Concept Formation.* In: Artificial Intelligence 40 (1989), 11-61.

Hall R.P.: *Computational Approaches to Analogical Reasoning: A Comparative Analysis.* In: Artificial Intelligence 39 (1989), 39-120.

Hinton G.E.: *Connectionist Learning Procedures.* In: Kodratoff Y., Michalski R.S. (eds.) Machine Learning: An Artificial Intelligence Approach, Volume III, Morgan-Kaufmann, 1990.

Kodratoff Y.: *Introduction to Machine Learning*, Pitman, London 1988.

Kodratoff Y., Michalski R. (eds.): *Machine Learning: An Artifical Intelligence Approach*, Volume III, Morgan-Kaufmann, San Mateo, California, 1990.

Kodratoff Y.: *Induction and Organization of Knowledge.* In: Proceedings of the first International Workshop on Multistrategy Learning, Harpers Ferry, U.S.A., November 7-9, 1991.

Kolodner J.L., Simpson R.L., Sycara-Cyransky K.: *A Process Model of Case-Based Reasoning in Problem Solving.* In: Proceedings of the IJCAI-85 Conference, Los Angeles, CA, 1985, 284-290.

Michalski R.S.: *On the Quasi-Minimal Solution of the General Covering Problem.* In: Proceedings of the 5th International Symposium on Information Processing (FCIP"69), Vol.A3, Bled, Yugoslavia, 1969, pp.125-128.

Michalski R.S., Stepp R.E.: *A Theory and Methodology of Inductive Learning.* In: Michalski R.S., Carbonnell J.G., Mitchell T.M. (eds): Machine Learning:An Artificial Intelligence Approach, Morgan-Kaufmann, 1983.

Michalski R.S.: *Learning from Observation: Conceptual Clustering.* In:Michalski R.S., Carbonnell J.G., Mitchell T.M. (eds): Machine Learning: An Artificial Intelligence Approach, Morgan-Kaufmann, 1983.

Michalski R.S.: *Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-Tiered Representation.* In: Kodratoff Y., Michalski R.S. (eds.) Machine Learning: An Artificial Intelligence Approach, Volume III, Morgan-Kaufmann, 1990.

Michalski R.S.: *Toward a Unified Theory of Learning: An Outline of Basic Ideas.* In: First World Conference on the Fundamentals of Artificial Intelligence, Paris July 1-5, 1991.

Michalski R., Carbonell J., Mitchell T.M. (eds.): *Machine Learning: An Artifical Intelligence Approach*, Volume I, Morgan-Kaufmann, San Mateo, California, 1983.

Michalski R., Carbonell J., Mitchell T.M. (eds.): *Machine Learning: An Artifical Intelligence Approach,* Volume II, Morgan-Kaufmann, San Mateo, California, 1986.

Mitchell T.M., Keller R.M., Kedar-Cabelli S.T.: *Explanation-Based Generalization: A Unifying View.* In: Machine Learning 1 (1986), pp. 47-80.

Núñez M.:*The Use of Background Knoledge in Decision Tree Induction.* In Machine Learning, 6, (1991), pp.231-250.

Quinlan J.R.: *Induction of Decision Trees.* In: Machine Learning 1, (1986), pp.81-106.

Quinlan J.R.: *Probabilistic Decision Trees.* In: Kodratoff Y., Michalski R.S.(eds.) Machine Learning: An Artificial Intelligence Approach, Volume III, Morgan-Kaufmann, 1990.