Dolování dat

Distanční studijní text

Petr Berka, Jan Górecki

Karviná 2017

SLEZSKÁ
UNIVERZITA
OBCHODNĚ PODNIKATELSKÁ
FAKULTA V KARVINÉ

Obor: Umělá inteligence

Klíčová slova: dolování dat, dobývání znalostí z databází, statistika, strojové učení, modelování, testování, předzpracování dat

Anotace: Díky výraznému pokroku v rozvoji informačních technologií ve zhruba posledních třech desetiletích zásadním způsobem narostlo množství uchovávaných dat a s tím i potřeba tato často velmi rozsáhlá data zpracovávat. Tato potřeba dala vzniknout disciplíně anglicky označované jako Data mining, často překládané jako Dolování dat, která v sobě zahrnuje metody pro předzpracování dat, jejich modelování a vyhodnocení získaných modelů.

Tento studijní text, který je věnován Dolování dat, je určen studentům magisterského stupně na vysokých školách ekonomického zaměření. Text se nejdříve věnuje samotnému pojmu *dolování dat* a s ním souvisejícím metodikám. Hlavní část textu je pak věnována základním metodám dolování dat. Jelikož tyto metody vycházejí ze statistických metod a metod strojového učení, jsou tyto metody nejdříve připomenuty ve dvou kapitolách předcházející zmíněnou hlavní část. Poslední část tohoto textu je pak věnována vyhodnocení výsledků dolování dat a metodám pro přípravu dat, které jsou nezbytnou součástí většiny aplikací metod dolování dat.

Autor: **prof. Ing. Petr Berka, CSc.**
Ing. Jan Górecki, Ph.D.

Obsah

ÚVODEM

Tento studijní text vznikl se záměrem stát se oporou při studiu předmětu Dolování dat na Obchodně-podnikatelské fakultě v Karviné, Slezská univerzita v Opavě (OPF SU) a je určen studentům magisterského stupně. Text převážně vychází z knihy Dobývání znalostí databází (Berka (2003)), jejíž obsah se již mnoho let vyučuje v obdobném předmětu na Vysoké škole ekonomické v Praze.

Při studiu textu je výhodou, pokud je čtenář již seznámen s Informačními systémy a statistickými metodami, avšak jejich neznalost není pro úspěšné absolvování předmětu překážkou.

Každá kapitola textu zahrnuje několik distančních prvků, pomocí kterých se student může v rychlosti seznámit s obsahem dané kapitoly, zjistí, co je cílem dané kapitoly, a kterým klíčovým pojmům se kapitola věnuje. V závěru každé kapitoly najde čtenář shrnutí toho nejdůležitějšího, čemu se kapitola věnovala a také sadu kontrolních otázek, pomocí nichž si čtenář může své nabyté znalosti ověřit.

RYCHLÝ NÁHLED STUDIJNÍ OPORY

V první části textu se čtenář seznámí se samotným pojmem dolování dat a s ním souvisejícími základními pojmy. V této části textu se čtenář také seznámí se třemi rozšířenými metodikami, které se používají jako určitý standard při procesu dolování dat.

Druhá kapitola tohoto studijního textu se věnuje přípravou dat, jež je nezbytnou součástí většiny procesu zahrnujících dolování dat. V této kapitole se čtenář může seznámit s metodami, které umožňují upravit hrubá data získané získaná přímo z praktických aplikací na data, která jsou již vhodná pro vybranou metodu dolování dat. Konečně je zde také představeno několik přístupu k situacím, kdy je dat je příliš mnoho anebo naopak situacím, kdy některá z dat chybí.

V další části textu se čtenář seznámí se čtyřmi statistickými metodami, které souvisí s dolováním dat. Tyto metody zahrnují kontingenční tabulky, regresní analýzu, diskriminační analýzy a také shlukovou analýzu.

Další kapitola seznámí čtenáře s přístupy k automatizaci procesu učení. Čtenář se seznámí s možnými reprezentacemi dat použitelných pro automatizované učení. Dále se čtenář seznámí s formální reprezentací znalostí a přístupem k učení (optimalizaci znalostí) na základě dostupných dat.

Následuje hlavní část studijního textu, která se věnuje základním metodám dolování dat, které často patří mezi nejčastěji používané metody dolování dat. Jsou to metody pro tvorbu rozhodovacích stromů, asociačních pravidel, metody založené na umělých neuronových sítích, dále se zde čtenář seznámí s takzvanými evolučními algoritmy, metodou klasifikace založenou na Bayesově větě a nakonec se zde čtenář také dozví něco o metodách založených na analogii.

Vzhledem k množství existujících metod dolování dat je možno pro každá data získat značné množství různých modelů těchto dat. Pro výběr toho nejvhodnějšího modelu, který bychom poté nasadili do praxe, je potřeba tyto modely nějakým způsobem ohodnotit. Závěrečná tohoto studijního textu se proto věnuje různým možnostmem hodnocení modelů získaných dolováním dat.

PŘEDMLUVA

Pro začátek malý pokus:

- Myslete si libovolné číslo mezi 1 a 10.
- Toto číslo vynásobte číslem 9.
- Sečtěte cifry v tomto čísle (např. myslíte-li si teď 25, tak 2+5).
- Vynásobte výsledek čtyřmi.
- Dělte výsledek třemi.
- Odečtěte 10.

Výsledkem je 2.

Je to, že jsme výsledek uhádli, ať už jste si na začátku mysleli jakékoli číslo mezi 1 a 10, náhoda nebo ne? Pokud si jednotlivé kroky znovu projdete, snadno sami zjistíte, že není. Podívejme se ale například na pravděpodobně nejjednodušší náhodnou událost, kterou si dokážeme představit, což je hod mincí. „Ale“, asi si řeknete, „to je přece náhodná událost, kde nikdo není schopen přesně předpovědět, která strana mince padne poté, co je mince hozena.“ To může být správná úvaha, nicméně to, že to *nikdo* není schopen předpovědět, v žádném případě neznamená, že je to v principu *nemožné*. Kdyby totiž byly přesně známy všechny faktory, které výsledek hodu mince ovlivňují, jako např. rychlost hodu a úhel hodu mince, materiálové vlastnosti mince a povrchu, kam mince dopadne, a dokonce i síla a směr větru, pak bychom byli docela schopni, s vynaložením určitého času a úsilí, výsledek hodu předvídat. Fyzikální vzorce pro to v každém případě existují.

Nyní se podívejme na jiný scénář, kde však tentokráte výsledek situace předvídat dokážeme: Sklenice se rozbije, pokud spadne z určité výšky na určitý typ povrchu. Zde, když vidíme padající sklenici, dokonce již ve zlomcích vteřiny víme: Sklenice se rozbije. Kde se v nás bere ta úžasná schopnost výsledek situace přesně předpovědět? Přece jsme padající sklenici nikdy předtím neviděli a fyzikální vzorce, které popisují rozbití sklenice, jsou pro většinu z nás zahaleny rouškou tajemství. Samozřejmě, v několika případech se může stát, že se sklenice „náhodou“ nerozbije, nicméně je to velmi nepravděpodobné. Zde je důležité si uvědomit, že případ, kdy se sklenice nerozbije, je stejně nenáhodný, jako když se rozbije, jelikož oba případy vyplývají z fyzikálních zákonů. Např. energie nárazu je v prvním případě lépe, v druhém případě hůře, absorbována povrchem, na který sklenice dopadne, apod. Otázka tedy zní: Jak to, že my lidé jsme v některých případech schopni přesně předpovědět, jak daný proces dopadne (např. padající sklenice), zatímco v jiných případech (např. hod mincí) toho schopni nejsme?

Nejčastější odpovědí na tuto otázku, kterou laikové v tomto případě používají, je popis jednoho scénáře jako "nenáhodný" a druhý jako "náhodný". Místo toho, abychom zde nad dalšími možnými odpověďmi dále a hlouběji filozofovali, předložme si následující tvrzení:

Převážná většina procesů, které můžeme pozorovat našimi smysly, ***není*** *výsledkem náhody. Důvodem naší neschopnosti procesy přesně popsat a předvídat jejich výsledky spíše spočívá ve skutečnosti, že nejsme schopni rozpoznat ani měřit potřebné ovlivňující faktory popř. tyto faktory dát do správných souvislostí.*

V případě pádu sklenice jsme rychle rozpoznali nejdůležitější faktory ovlivňující výsledek tohoto procesu, jako je materiál (sklo), výška, ze které sklenice padá, a povaha povrchu, kam sklenice padá, a v okamžiku jsme odhadli pravděpodobnost toho, že se rozbije, na základě podobných zkušeností z minula. Nicméně, to je přesně to, co u hodu mince udělat nemůžeme. Můžeme sledovat hod mincí kolikrát chceme, ale *nikdy* nedokážeme dostatečně rychle rozpoznat všechny potřebné faktory k tomu, abychom výsledek přesně předpověděli.

Vraťme se teď ale zpět k padající sklenici. Co tedy probíhalo v našich hlavách, když jsme po pádu sklenice na zem předpovídali, že se rozbije? Měřili jsme charakteristiky této události. Mohli bychom také říct, že jsme shromažďovali údaje popisující pád sklenice. Potom jsme rychle provedli úsudek pomocí *analogie*, tj. porovnali jsme aktuální pád sklenice s případy z minula, kdy jsme viděli padat brýle, poháry, porcelánové figurky apod., a výsledek jsme vyhodnotili podle „nejpodobnějšího" scénáře, tj. každý z těchto případů z minula jsme si pro sebe ohodnotili určitou *mírou podobnosti* s aktuálním scénářem. Pro takový úsudek je zapotřebí dvou věcí: 1) musíme mít data o těchto minulých případech a 2) musíme vědět, jak je *podobnost* mezi současným případem a případy minulými vůbec definována. Pak již jednoduše předpovíme výsledek tím, že se podíváme na ty případy padajících věcí z minula, které jsou aktuálnímu případu nejpodobnější, a zeptáme se: Rozbil se v těch případech padající předmět nebo ne? Jak jsme zmínili, je potřeba najít našemu případu ty nejpodobnější případy z minula, což vlastně reprezentuje určitý typ *optimalizace*. Zde je potřeba zdůraznit, že je jedno, zda optimalizujeme zmíněnou podobnost padajících předmětů nebo např. zisk nějaké společnosti – zvažovaná proměnná, v tomto případě podobnost padajících předmětů, je vždy optimalizována. Potom nám již stačí zvážit to, že např. ve většině nejpodobnějších případů se padající předmět rozbil, a zmíněným úsudkem pomocí analogie takto stanovíme i předpověď pro aktuální případ. Možná to vše zní trochu komplikovaně, ale tento typ úsudku pomocí analogie je v podstatě základem *téměř každého* lidského procesu učení se a je prováděn s ohromující rychlostí.

Zajímavé na tom všem je, že jsme se v našich hlavách právě zachovali jako nějaká *lidská verze* metody pro dolování dat, neboť tyto metody obvykle zahrnují záležitosti, jako je právě reprezentace případů pomocí dat, definici podobností případů a její optimalizaci.

Nicméně pro házení mincí je popsaný postup úsudku pomocí analogie nepoužitelný: Problém je totiž v již prvním kroku, kdy většinou nemáme dostatečně přesné údaje o ovlivňujících faktorech, jako jsou vlastnosti materiálu nebo nerovnosti podlahy, čímž je nám znemožněno pokračovat v úsudku dále. To však v žádném případě neznamená, že házení mincí je náhodná událost, ale pouze poukazuje na to, že my lidé nejsme ovlivňující faktory schopni měřit a tím proces popsat. V jiných situacích schopni měřit ovlivňující faktory být můžeme, ale pro změnu třeba nejsme schopni dát všechny tyto faktory do *správných souvislostí*, tedy že nejsme schopni správně spočítat požadované podobnosti nebo dokonce správně ani popsat probíhající proces.

Úsudek pomocí analogie taktéž v žádném případě není jediným nástrojem, jak předpovídat výsledky nových situací na základě již předem známých informací. Pokud je pozorovatel padající sklenice dotázán, jak vlastně ví, že se sklenice rozbije, odpověď bude často zahrnovat věci jako "pokaždé, když jsem viděl sklenici spadnout z výšky větší než 1,5 metru, tak se rozbila". Všimněme si zde dvou zajímavostí: 1) vztah ke zkušenostem z minula vyjádřený výrazem " pokaždé" a 2) vyvození následujícího pravidla na základě těchto zkušeností:

Pokud je padající předmět ze skla a padá z výšky větší než 1,5 metru, pak se rozbije.

Zavedení prahové hodnoty, například 1,5 metru, je fascinujícím aspektem tohoto pravidla. Ačkoliv ne vždy se předmět rozbije, pokud spadne z větší výšky, a taky ne vždy zůstane neporušen, pokud spadne z menší výšky, zavedením této prahové hodnoty se pravidlo transformuje na odhad (lidově řečeno) od oka, který nemusí platit vždy, ale ve většině případů ke správnému odhadu situace vede. Namísto úsudku pomocí analogie můžeme tedy rovnou použít tento odhad od oka, a tím rychle dospět k nejpravděpodobnějšímu výsledku po pádu předmětu na zem. Úsudek pomocí analogie a tvorba pravidel jsou dva první příklady toho, jak lidé, a také metody dolování dat, předvídají výsledek nových a neznámých situací.

Náš popis toho, co se děje v našich hlavách a také ve většině metod dolování dat, přináší ještě jeden zajímavý pohled: Při usuzování pomocí analogie není potřeba žádném kroku jakýkoli fyzikální vzorec na to, abychom řekli, zda se sklenice rozbije. To samé platí i pro výše zmíněný odhad od oka. Tedy, jak my, tak metody dolování dat, umí(me) vytvářet odhady situace či je dokonce předvídat bez znalosti úplného fyzikálního popisu daného procesu. Navíc, sběr dat popisující případ byl jen velmi hrubý a povrchní a použili jsme pouze pár ovlivňujících faktorů jako materiál padajícího předmětu (sklo) a výška pádu (zhruba 2 metry), navíc jsme je neměřili s nějak velkou přesností.

Každá situace nebo jev se řídí přírodními zákonitostmi, ať už jim rozumíme, či nikoliv. V tomu druhém případě máme často tendenci takové jevy nazývat náhodnými. Je zajímavé, že často jsme schopni tyto náhodné jevy předvídat bez úplné znalosti souvisejících přírodních zákonitosti, a to dokonce i v situacích, kdy jevy známé z minulosti nezahrnují všechny možné případy nebo jsou popsány pouze nepřesně.

V této předmluvě jsme si nastínili hlavní myšlenku toho, jakým problémů bychom se chtěli v těchto skriptech věnovat. Budeme pracovat s množstvím ovlivňujících faktorů, kde některé z nich budou popsány nedostatečně přesně nebo dokonce vůbec. Spolu s tímto se často stává, že ovlivňujících faktorů je tolik, že je velmi jednoduché se v nich ztratit. Navíc, pro modelování výsledků procesů budeme muset pracovat s případy z minulosti, jejichž počet však může snadno dosáhnout miliónů či dokonce miliard. Konečně, budeme se muset sami sebe ptát, zdali popis procesu je cíl naší práce nebo zdali úsudek pomocí analogie již je pro předpověď dostačující. Navíc, tyto úvahy budou probíhat v dynamickém prostředí s neustále se měnícími podmínkami – a nejlépe vše co nejrychleji. Pro lidi nemožné? Správně. Ne však nemožné pro metody dolování dat.

1 DOBÝVÁNÍ ZNALOSTÍ Z DATABÁZÍ A DOLOVÁNÍ DAT

RYCHLÝ NÁHLED KAPITOLY

V této kapitole se nejdříve seznámíme s pojmem *dolování dat* jako takovým a taktéž si tento pojem dáme do souvislostí s pojmem *dobývání znalostí z databází*. Dále se podíváme na proces dolování dat jak z technického, tak z manažerského hlediska. V další části této kapitoly se podíváme na základní typy úloh dolování dat a nakonec se podíváme na různé metodiky nabízející jednotný rámec pro řešení různých úloh z dolování dat.

CÍLE KAPITOLY

V této kapitole se čtenář dozví, kdy a kde se poprvé objevil termín *Dolování dat* a s ním související termín dobývání znalosti z databází. Dále si v této kapitole může čtenář udělat představu, jak vlastně probíhá proces dolování dat, a to jak z pohledu technického, tak z pohledu manažerského. Také se zde čtenář dozví, jaké základní úlohy řeší disciplína dolování dat a seznámí se se dvěmi významnými metodikami nabízející jednotný postup při dolování dat.

KLÍČOVÁ SLOVA KAPITOLY

dobývání znalostí z databází, dolování datm klasifikace, deskripce, hledání nugetů, metodiky

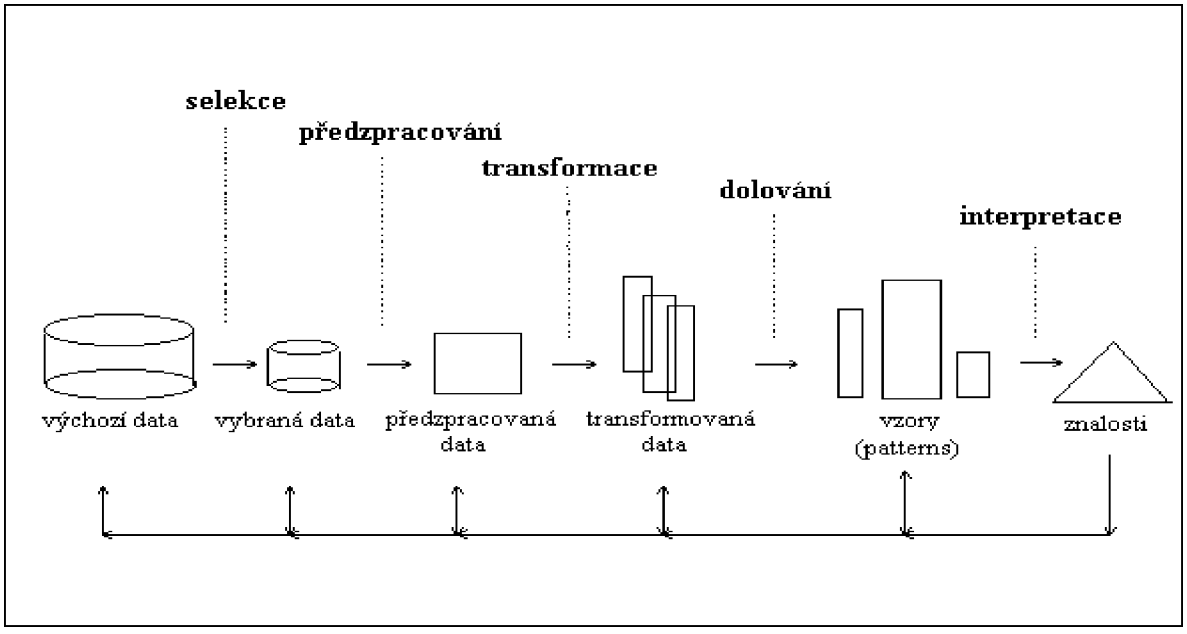
O dobývání znalostí z databází (Knowledge Discovery in Databases, KDD) se začíná ve vědeckých kruzích mluvit počátkem 90. let. První impuls přišel z Ameriky, kde se na konferencích věnovaných umělé inteligenci (mezinárodní konference o umělé inteligenci IJCAI'89 nebo konference americké asociace umělé inteligence AAAI'91 a AAA'93) pořádaly první workshopy věnované této problematice. Nebyla to ale jen umělá inteligence (přesněji řečeno metody strojového učení), které stály u zrodu dobývání znalostí z databází. Databázové technologie představují osvědčený prostředek jak uchovávat rozsáhlá data a vyhledávat v nich informace, statistika představuje osvědčený prostředek, jak modelovat a analyzovat závislosti v datech. Po léta se tyto disciplíny vyvíjely nezávisle, až přišla ta chvíle, kdy rozsah automaticky sbíraných dat začínal uživatelům přerůstat přes hlavu, a současně s tím vznikla potřeba tato data používat pro podporu (strategického) rozhodování ve firmách. Zájem finančně silných uživatelů o aplikace pak stimuloval ono propojení a dal vznik (a hlavně popularitu) dobývání znalostí z databází. Neustálý nárůst zájmu odborné komunity dokládá množství konferencí (americké konference KDD, asijské konference PAKDD, evropské konference PKDD), vznik odborných skupin (např. special

interest group for KDD - SIGKDD při americké asociaci ACM) i vznik samostatných odborných časopisů (časopis Data Mining and Knowledge Discovery vydávaný nakladetelstvím Kluwer). Tématika dobývání znalostí si postupně našla cestu i do šířeji zaměřených počítačových časopisů. Dnes již není nic neobvyklého, že na pojmy knowledge discovery, data mining, nebo business intelligence[1] narazíme i v reklamách počítačových firem.

DEFINICE Df

Dobývání znalostí z databází (KDD) lze definovat *jako netriviální extrakci implicitních, dříve neznámých a potenciálně užitečných informací z dat* [Fayyad a kol, 1996].

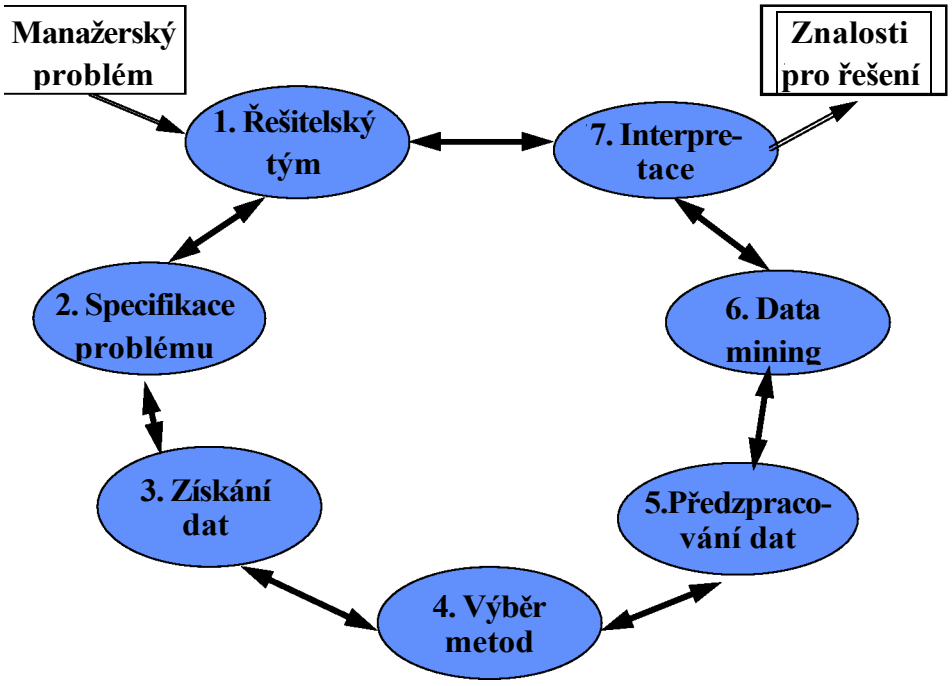
Zpočátku se pro tuto oblast razily nejrůznější názvy: information harvesting, data archeology, data destilery. Nakonec ale převládla hornická metafora; dobývání znalostí a dolování z dat (data mining). Po jistém období tápání se ustálilo i chápání KDD jako interaktivního a iterativního *procesu* tvořeného kroky selekce, předzpracování, transformace, vlastního „dolování" (data mining) a interpretace (Obrázek 1).



Obrázek 1: Proces dobývání znalostí z databází dle [Fayyad a kol, 1996].

[1] Význam pojmu business intelligence je možno (s trochou nadsázky) interpretovat touto rovnicí: business intelligence = artificial intelligence + business

Na rozdíl od "prostého" použití statistických metod a metod strojového učení se v procesu dobývání znalostí již klade důraz i na přípravu dat pro analýzu a na interpretaci výsledných znalostí. Při přípravě dat se obvykle z dat uložených ve složité struktuře např. datového skladu vytváří jedna tabulka, obsahující relevantní údaje (hodnoty atributů) o sledovaných objektech např.



o klientech banky nebo zákaznících obchodního domu). Při interpretaci se nalezené znalosti[2] hodnotí z pohledu koncového uživatele.

Zatímco schéma na Obrázek 1 popisuje „technologický“ pohled na dobývání znalostí, Anand [Anand a kol., 1996] nabízí pohled manažerský (**Obrázek 2**). Impulsem pro zahájení procesu

Obrázek 2: Manažerský pohled na proces dobývání znalostí z databází

dobývání znalostí je nějaký **reálný problém**. Cílem procesu dobývání znalostí je získání co nejvíce relevantních informací vhodných k řešení daného problému. Příkladem reálného problému je otázka nalezení skupin zákazníků obchodního domu nebo skupin klientů banky, kterým by bylo možno nabídnout speciální služby. U zákazníků obchodního domu se může jednat o zjištění, že zákazník kupuje potravinářské zboží odpovídající jisté dietě, v případě klientů banky může jít o potenciální zájemce o hypoteční úvěr. Nalezené skupiny jsou interpretovány jako takzvané segmenty trhu v dané oblasti.

Prvním krokem při řešení problému je **vytvoření řešitelského týmu**. Jeho členy musí být *expert na řešenou problematiku*, *expert na data* - jak v organizaci tak případně i na externí data

[2] Fayyad rozlišuje mezi znalostmi získanými jako výstup z kroku dolování (nazývá je vzory – patterns) a mezi znalostmi interpretovanými uživatelem. My toto rozlišení nebudeme provádět.

a *expert na metody KDD*. V případě rozsáhlejších problémů je obvyklé, že jednotliví experti mají k dispozici vlastní tým nebo alespoň využívají konzultací s dalšími experty

Prvním úkolem sestaveného týmu je **specifikace problému**, který je třeba řešit v souvislosti, z pohledu dobývání znalostí. U zákazníků obchodního domu nakupujících potravinářské zboží odpovídající jisté dietě je mimo jiné třeba specifikovat položky zboží odpovídající různým dietám. U skupin zákazníků nakupujících položku A a nenakupujících položku B je krom jiného třeba vytipovat vhodné skupiny položek, atd.

Po specifikaci problému je třeba **získat všechna dostupná data**, která mohou být použita pro řešení problému. Znamená to posoudit všechna dostupná data a zvážit, zda jsou relevantní k danému problému. Tento proces může vyvolat menší či větší přeformulováni problému. V některých případech je třeba pracovat i s daty, která jsou archivována po delší dobu ve formě datových souborů a ne v databázi, data jsou někdy dokonce uložena v několika různých systémech. Náročnost získání dat je nepřímo úměrná úrovni datové základny, která je k dispozici.

V mnohých případech je vhodné uvažovat i *externí data* popisující prostředí, ve kterém se analyzované děje odehrávají. V případě klientů banky i zákazníků obchodního domu je důležitou informací kalendářní období (např. vánoce, velikonoce, období dovolených letních a zimních, den kdy zákazníci dostávají výplatu, pondělí, úterý, ...,). Na zákazníky bude mít jistě vliv i počasí, reklama probíhající ve sdělovacích prostředcích, v některých případech i politické události.

Cílem **výběru metody** je zvolit vhodné metody analýzy dat. V rámci dobývání znalostí z databází je používána řada typů metod analýzy dat, ve většině případů je k řešení konkrétní úlohy zapotřebí kombinovat více různých metod. Mezi používané typy metod patří např. klasifikační metody, různé klasické metody explorační analýzy dat, metody pro získávání asociačních pravidel, rozhodovací stromy, genetické algoritmy, Bayesovské sítě, neuronové sítě, hrubé množiny (rough sets), velmi používané jsou i metody vizualizace. Dá se také předpokládat vývoj dalších metod.

V rámci **předzpracování dat** se data získaná k řešení specifikovaného problému připravují data do formy vyžadované pro aplikaci vybraných metod. V řadě případů se může jednat o značně náročné výpočetní operace. Do této fáze patří i odstranění odlehlých hodnot, případně doplnění chybějících hodnot.

Krok **data mining** zahrnuje aplikaci vybraných analytických metod pro vyhledávání zajímavých vztahů v datech. Obvykle jsou jednotlivé metody aplikovány vícekrát, hodnoty vstupních parametrů jednotlivých běhů závisí na výsledcích předchozích běhů. Zpravidla se nejedná o aplikace metod jenom jednoho typu, jednotlivé typy se kombinují na základě dílčích výsledků.

Cílem **interpretace** je nezbytné zpracování obvykle značného množství výsledků jednotlivých metod. Některé z těchto výsledků vyjadřují skutečnosti, které jsou z hlediska uživatele nezajímavé nebo samozřejmé. Některé výsledky je možno použít přímo, jiné je nutno vyjádřit způsobem srozumitelným pro uživatele. Jednotlivé výsledky je často vhodné uspořádat do analy-

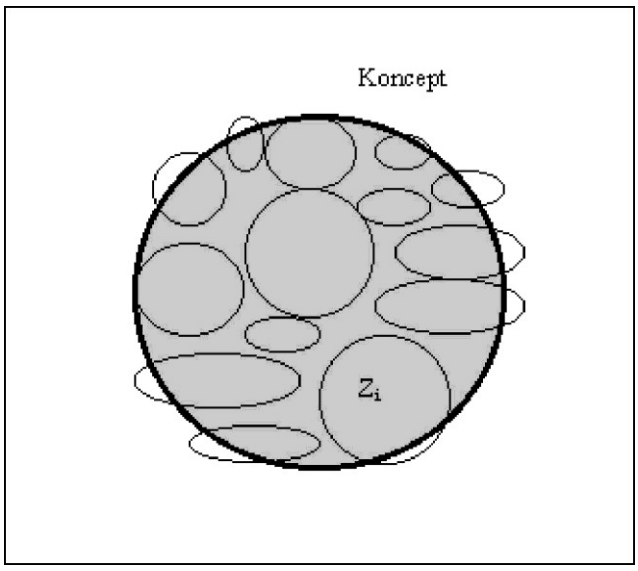
tické zprávy. Analytická zpráva však není jediným možným výstupem procesu dobývání znalostí. Výstupem může být i provedení vhodné akce jako například zapnutí monitorovacího programu.

1.1 Úlohy

V případě dobývání znalostí z databází můžeme mluvit o různých typech úloh. Jsou to především [Klosgen, Zytkow, 1997]:[3]

- klasifikace/predikce,
- deskripce,
- hledání „nugetů“.

Při klasifikaci/predikci je cílem nalézt znalosti použitelné pro klasifikaci nových případů - zde požadujeme, aby získané znalosti co nejlépe odpovídaly danému konceptu; dáváme přednost přesnosti pokrytí na úkor jednoduchosti (připouštíme větší množství méně srozumitelných dílčích znalostí tak jak je to naznačeno na Obrázek 3). Rozdíl mezi klasifikací a predikcí spočívá v tom, že u predikce hraje důležitou roli čas; ze starších hodnot nějaké veličiny se pokoušíme odhadnout její vývoj v budoucností (např. předpověď počasí nebo pohybu cen akcií).

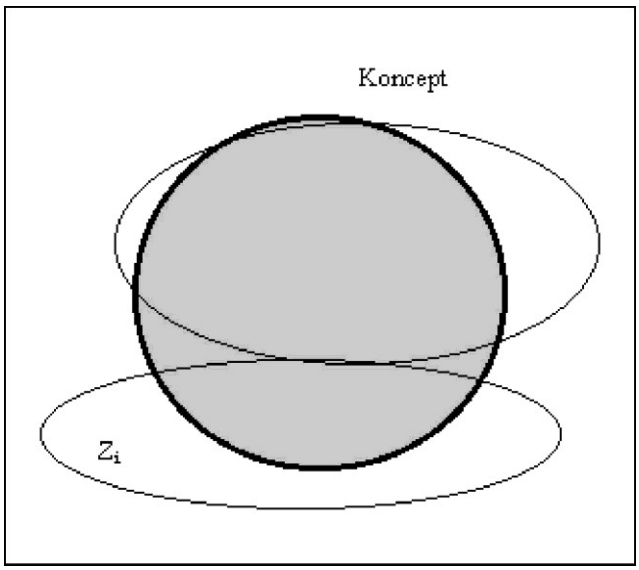


Obrázek 3: Klasifikace/Predikce

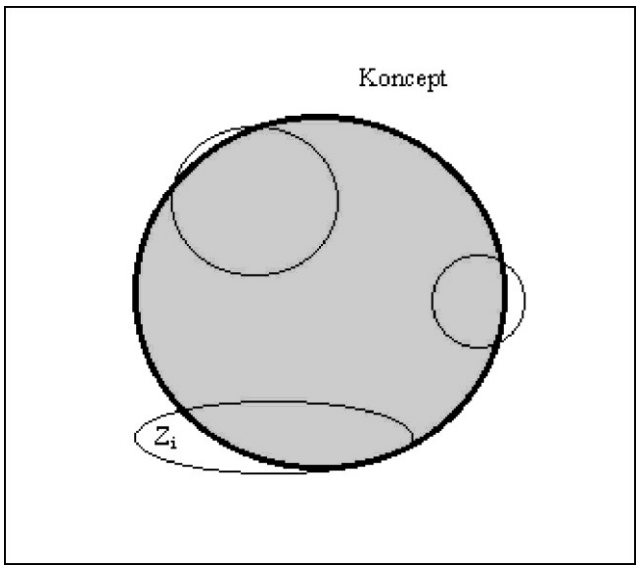
Při deskripci (popisu) je cílem nalézt dominantní strukturu nebo vazby, které jsou skryté v daných datech. Požadujeme srozumitelné znalosti pokrývající daný koncept; dáváme tedy před-

[3] Podrobnnější členění lze nalézt v [Chapman a kol, 2000]. Tvůrci metodiky CRISP-DM zde uvádějí úlohy deskripce dat a sumarizace, segmentace, deskripce konceptů, klasifikace, predikce a analýzy závislostí.

nost menšímu množství méně přesných znalostí (viz. Obrázek 4). Hledáme-li nugety, požadujeme zajímavé (nové, překvapivé) znalosti, které nemusí plně pokrývat daný koncept (Obrázek 5).



Obrázek 4: Popis (deskripce)



Obrázek 5: Nugety

Úlohy dobývání znalostí lze nalézt v celé řadě aplikačních oblastí:

- Segmentace a klasifikace klientů banky (např. rozpoznání problémových nebo naopak vysoce bonitních klientů),
- Predikce vývoje kursů akcií,
- Predikce spotřeby elektrické energie,

- Analýza příčin poruch v telekomunikačních sítích,
- Analýza důvodů změny poskytovatele nějakých služeb (internet, mobilní telefony),
- Segmentace a klasifikace klientů pojišťovny,
- Určení příčin poruch automobilů,
- Rozbor databáze pacientů v nemocnici,
- Analýza nákupního košíku (Market Basket Analysis).

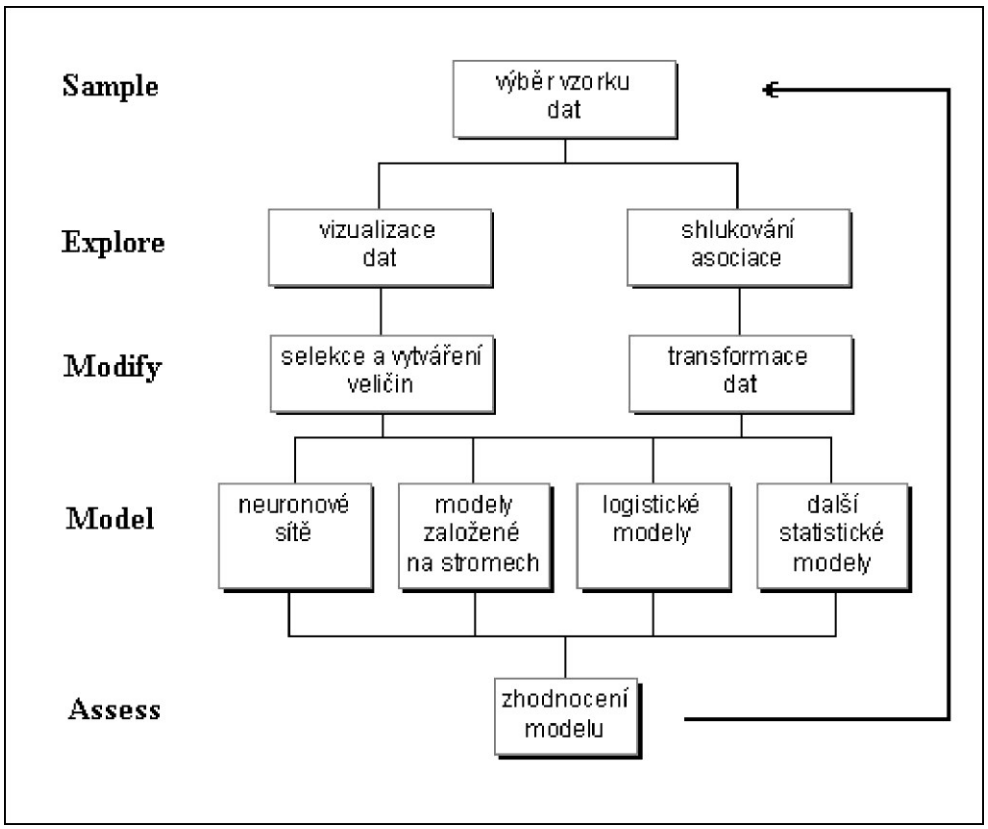
1.2 Metodiky

S postupem doby začaly vznikat metodiky, které si kladou za cíl poskytnout uživatelům jednotný rámec pro řešení různých úloh z oblasti dobývání znalostí. Tyto metodiky umožňují sdílet a přenášet zkušenosti z úspěšných projektů Za některými metodikami stojí producenti programových systémů (metodika *SEMMA* firmy SAS), jiné vznikají ve spolupráci výzkumných a komerčních institucí jako „softwarově nezávislé“ (*CRISP-DM*).

1.2.1 Metodika SEMMA

Enterprise Miner, softwarový produkt firmy SAS, vychází z vlastní metodiky pro dobývání znalostí z databází. Název *SEMMA* opět charakterizuje jednotlivé prováděné kroky:

- *Sample* (vybrání vhodných objektů),
- *Explore* (vizuální explorace a redukce dat),
- *Modify* (seskupování objektů a hodnot atributů, datové transformace),
- *Model* (analýza dat: neuronové sítě, rozhodovací stromy, statistické techniky, asociace a shlukování),
- *Assess* (porovnání modelů a interpretace).



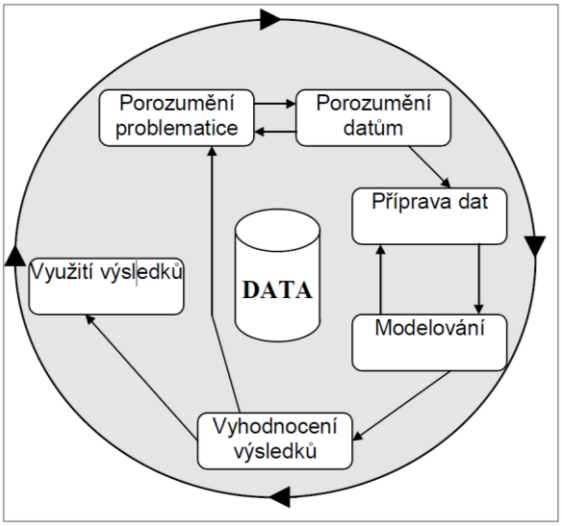
Obrázek 6: Metodika SEMMA

1.2.2 CRoss-Industry Standard Process for Data Mining

Metodika *CRISP-DM* (CRoss-Industry Standard Process for Data Mining) vznikla v rámci Evropského výzkumného projektu. Cílem projektu bylo navrhnout univerzální postup (tzv. standardní model procesu dobývání znalostí z databází), který bude použitelný v nejrůznějších komerčních aplikacích [Chapman a kol, 2000]. Vytvoření takovéto metodiky umožní řešit rozsáhlé úlohy dobývání znalostí rychleji, efektivněji, spolehlivěji a s nižšími náklady. Kromě návrhu standardního postupu má *CRISP-DM* nabízet „průvodce“ potenciálními problémy a řešeními, které se mohou vyskytnout v reálných aplikacích.

Na projektu spolupracovaly firmy NCR (přední dodavatel datových skladů), Daimler-Chrysler, ISL (tvůrce systému Clementine) a OHRA (velká holandská pojišťovna). Všechny tyto firmy mají bohaté zkušenosti s reálnými úlohami dobývání znalostí z databází.

Životní cyklus projektu dobývání znalostí je podle metodiky *CRISP-DM* tvořen šesti fázemi (Obrázek 7). Pořadí jednotlivých fází není pevně dáno. Výsledek dosažený v jedné fázi ovlivňuje volbu kroků následujících, často je třeba se k některým krokům a fázím vracet. Vnější kruh na obrázku symbolizuje cyklickou povahu procesu dobývání znalostí z databází jako takovou.



Obrázek 7: Metodika CRISP-DM

Jednotlivé fáze metodiky *CRISP-DM* bude ilustrovat poměrně realistický příklad úlohy dobývání znalostí z dat v bance XY [Berka, 1999].

ŘEŠENÁ ÚLOHA

Nejprve několik slov k pozadí úlohy:

Banka XY je zaměřena na drobné klienty kterým vede účty, poskytuje půjčky apod. Pod rostoucím tlakem konkurence chce tato banka zlepšit své služby. Management banky má jen velmi vágní představu, co je možno od metod dobývání znalostí očekávat. Doufá ale, že mu tyto nové metody umožní lépe pochopit klienty a tak například cíleněji nabízet své produkty, nebo rozlišovat mezi různými skupinami klientů (bonitní resp. problémoví).

Porozumění problematice (Business Understanding)

Tato úvodní fáze je zaměřena na pochopení cílů úlohy a požadavků na řešení formulovaných z manažerského hlediska. Manažerská formulace musí být následně převedena do zadání úlohy pro dobývání znalostí z databází.

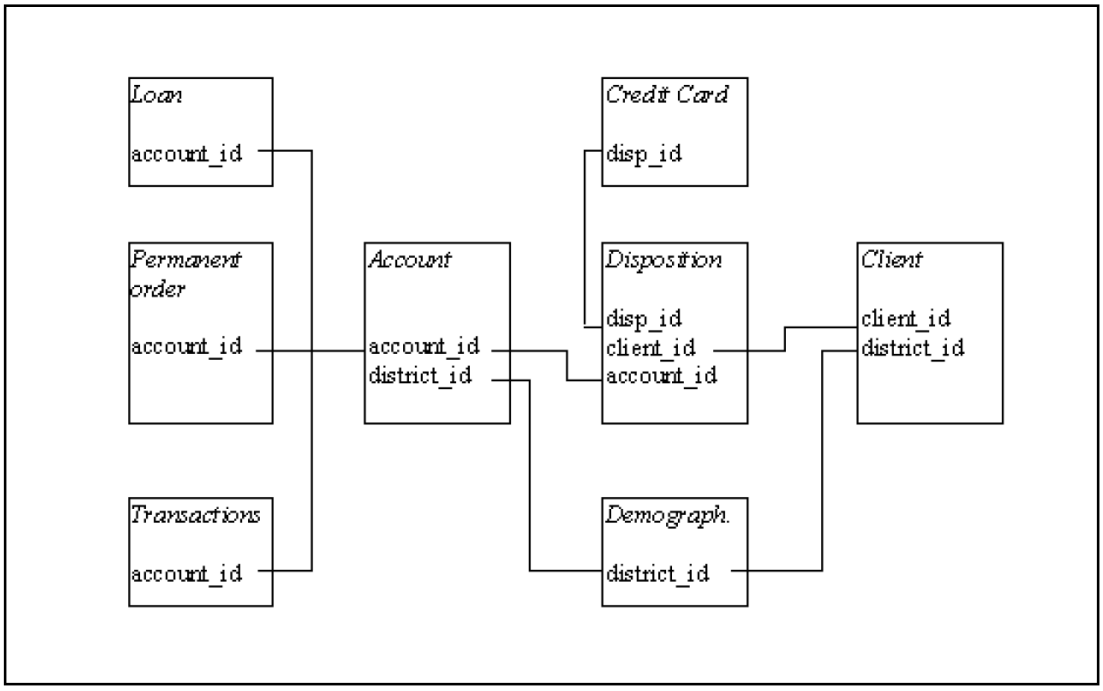
Manažerský problém, ke kterému jsou pomocí metod KDD hledány informace, může být formulován (téměř) bez vazby na informace získávané pomocí metod KDD z dostupných dat. Příkladem může být snaha nabídnout uložení části peněz na zvláštní účet s delší výpovědní lhůtou reklamou vhodně zacílenou na v tomto směru nadějnou skupinu klientů (i potenciálních) banky. Pro KDD to znamená nalézt takovou charakteristiku klientů, které zajišťují, že ve skupině klientů s touto charakteristikou bude velká část klientů mít stále dostatečně vysoký zůstatek na účtu. V tomto případě je zadání pro KDD formulováno relativně přesně, přesto je však třeba počítat s

možností přeformulování nebo upřesnění manažerského problému na základě provedených analýz. Jinou možnou úlohou je otázka včasného rozpoznání klientů, kteří představují rizikovou skupinu z hlediska splácení poskytnutého úvěru.

Porozumění datům (Data Understanding)

Fáze porozumění datům začíná prvotním sběrem dat. Následují činnosti, které umožní získat základní představu o datech, která jsou k dispozici (posouzení kvality dat, první „vhled“ do dat, vytipování zajímavých podmnožin záznamů v databázi...). Obvykle se zjišťují různé deskriptivní charakteristiky dat (četnosti hodnot různých atributů, průměrné hodnoty, minima, maxima apod.), s výhodou se využívají i různé vizualizační techniky.

*Data sledovaná bankou XY mají podobu několika navzájem propojených tabulek (*Obrázek 8*). Základní tabulkou je tabulka Account (účty). S každým účtem může disponovat nějaký klient (tabulka Client). K jednomu účtu může mít přístup více klientů, jeden klient může mít zřízeno více účtů; tato skutečnost je zachycena v tabulce Disposition, která přiřazuje klienty k účtům. Klientovi, který disponuje nějakým účtem, může být k tomuto účtu vydána kreditní karta (tabulka Credit Card). Nejdůležitější údaje o účtech jsou údaje o prováděných operacích, to je zachyceno v tabulce Transactions (transakce). Na některých účtech mohou být zřízeny trvalé platební příkazy (tabulka Permanent order), na základě některých účtů banka poskytuje úvěr (tabulka Loan). Banka poskytla pro analýzu jen určitý (relativně malý) vzorek těchto dat; první představu o podobě dat tedy bylo možno získat relativně jednoduchými nástroji (Access, Excel). Bylo konstatováno, že některé údaje v tabulce transakce (např. konstantní symbol) mají mnoho chybějících hodnot.*



Obrázek 8: Data banky XY

Příprava dat (Data Preparation)

Příprava dat zahrnuje činnosti, které vedou k vytvoření datového souboru, který bude zpracováván jednotlivými analytickými metodami. Tato data by tedy měla

- obsahovat údaje relevantní k dané úloze,
- mít podobu, která je vyžadována vlastními analytickými algoritmy.

Příprava dat tedy zahrnuje selekce dat, čištění dat, transformace dat, vytváření dat, integrování dat a formátování dat. Tato fáze je obvykle nejpracnější částí řešení celé úlohy. Jednotlivé úkony jsou obvykle prováděny opakovaně, v nejrůznějším pořadí.

Vzhledem k tomu, že data o klientech a jejich účtech jsou uložena v několika tabulkách navzájem spojených relacemi 1:n, n:1 a n:m, velkou část předzpracování představovalo vytvoření jediné tabulky obsahující údaje vybrané z více tabulek. Příslušné operace vedoucí k tomuto cíli tedy zahrnují agregování hodnot odpovídajících jednomu klientovi. Dalšími operacemi bylo např. vypočtení průměrných měsíčních zůstatků, průměrných měsíčních příjmů, převod rodného čísla klienta na jeho věk a pohlaví, apod.

Modelování (Modeling)

V této fázi jsou nasazeny analytické metody (algoritmy pro dobývání znalostí). Obvykle existuje řada různých metod pro řešení dané úlohy, je tedy třeba vybrat ty nejvhodnější (doporučuje se použít více různých metod a jejich výsledky kombinovat) a vhodně nastavit jejich parametry. Jde tedy opět o iterativní činnost (opakovaná aplikace algoritmů s různými parametry), navíc, použití analytických algoritmů může vést k potřebě modifikovat data a tedy k návratu k datovým transformacím z předcházející fáze.

Pro hledání zajímavých skupin klientů je možno použít metody shlukování nebo asociační pravidla. Pro rozpoznání rizikových klientů z hlediska půjček jsou (vzhledem k tomu, že jedna z tabulek obsahuje informace o průběhu splácení) vhodné např. algoritmy pro tvorbu rozhodovacích stromů nebo rozhodovacích pravidel. Tyto metody je vhodné kombinovat, např. shluková analýza může rozdělit klienty do skupin a rozhodovací strom pak umožní jednotlivé skupiny charakterizovat dostatečně srozumitelným způsobem.

Součástí této fáze je rovněž ověřování nalezených znalostí z pohledu metod dobývání znalostí. To může představovat např. *testování* klasifikačních znalostí na nezávislých datech.

Znalosti „deskriptivní" (charakteristiky skupiny klientů „zajímavých" z hlediska připravovaného produktu) byly předloženy expertům z banky. Znalosti klasifikační (umožňující „rozpoznat" klienty, kteří nesplácejí úvěr) byly testovány na novém vzorku dat.

Vyhodnocení výsledků (Evaluation)

V této fázi jsme se dopracovali do stavu, kdy jsme nalezli znalosti, které se zdají být v pořádku z hlediska metod dobývání znalostí. Dosažené výsledky je ale ještě třeba vyhodnotit z pohledu manažerů, zda byly splněny cíle formulované při zadání úlohy.

Některé nalezené skupiny klientů experty nepřekvapily, vědělo se o nich a banka se připravovala je oslovit dopisem. Jiné (rovněž bonitní skupiny) byly shledány zajímavými, ale budou ještě podrobeny dalšímu zkoumání. Výsledky testování klasifikačních znalostí ukázaly, že systém byl příliš „přísný", tedy správně rozpoznával klienty rizikové, ale v určitých případech (obzvláště u vyšších půjček) za rizikové označil i klienty bonitní. Bylo tedy rozhodnuto, že na všech pobočkách banky bude využíván program, který bude rozhodovat o úvěrech do určité částky.

Na závěr této fáze by mělo být přijato rozhodnutí o způsobu využití výsledků.

Využití výsledků (Deployment)

Vytvořením vhodného modelu řešení úlohy obecně nekončí. Dokonce i v případě, že řešenou úlohou byl „pouze" popis dat, získané znalosti je třeba upravit do podoby použitelné pro zákazníka (manažera - zadavatele úlohy). Podle typu úlohy tedy využití (nasazení) výsledků může na jedné straně znamenat prosté sepsání závěrečné zprávy, na straně druhé pak zavedení (hardwarové, softwarové, organizační) systému pro automatickou klasifikaci nových případů.

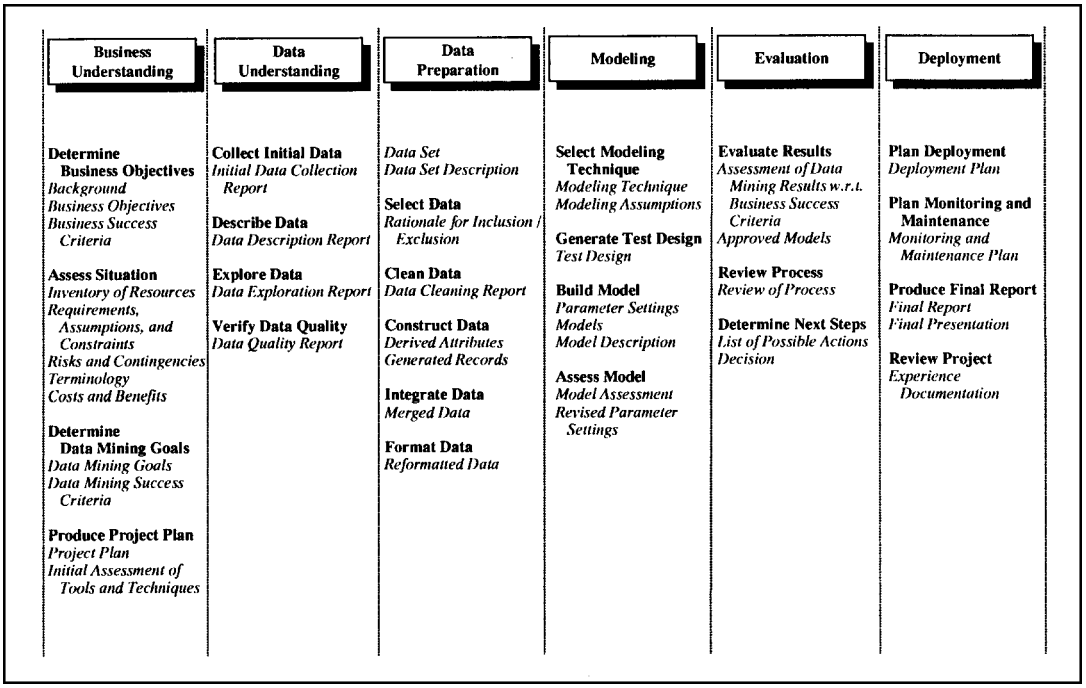
Ve většině případů je to zákazník a nikoliv analytik, kdo provádí kroky vedoucí k využívání výsledků analýzy. Proto je důležité, aby pochopil, co je nezbytné učinit pro to, aby mohly být dosažené výsledky efektivně využívány.

Systém pro rozhodování o půjčkách bude nasazen ve dvou fázích. V první fázi bude systém nasazen jen na vybraných pobočkách, po tomto poloprovozním ověření pak bude nasazen všude. Toto rozhodnutí vyžaduje:

- *implementaci klasifikačního algoritmu v uživatelsky přátelské podobě,*
- *přípravu uživatelského manuálu*
- *instalaci programu na všech pobočkách banky*
- *zaškolení uživatelů*
- *změnu metodiky poskytování úvěrů a tomu odpovídající změnu vnitřních předpisů banky*

Celkové schéma jednotlivých kroků metodiky CRISP-DM ukazuje Obrázek 9. Jednotlivé kroky procesu dobývání znalostí jsou různě časově náročné a mají i různou důležitost pro úspěšné vyřešení dané úlohy. Praktici v oboru uvádějí[4], že nejdůležitější je fáze porozumění problému (80% významu, 20% času) a časově nejnáročnější je fáze přípravy dat (80% času, 20% významu). Překvapivě málo práce zaberou vlastní analýzy (5% času, 5% významu).

[4] Následující údaje jsou převzaty z vystoupení J.U. Kietze na konferenci PKDD 2000.



Obrázek 9: Úlohy v metodologii CRISP-DM [Chapman a kol. 2002]

KONTROLNÍ OTÁZKA

1. V souvislosti s jakým vědním oborem a kdy se začíná mluvit o Dobývání znalostí z databází?
2. Jak lze definovat dobývání znalostí z databází?
3. Které jsou tři hlavní typy úloh dobývání znalostí z databází?
4. Vyjmenujte 6 fází procesu dobývání znalostí z databází dle metodiky CRISP-DM.

SHRNUTÍ KAPITOLY

V této kapitole jsme se seznámili s dolováním dat a dobýváním znalostní z databází. Potom jsme se dozvěděli něco o základních úlohách, které dolování dat. Nakonec jsme se seznámili se třemi metodikami, které nabízejí jednotných způsob postupu při dolování dat.

ODPOVĚDI

1. Počátkem 90. let 20. stolení v souvislosti s oborem Umělá inteligence.

2. Jako netriviální extrakci implicitních, dříve neznámých a potenciálně užitečných informací z dat.

3. Klasifikace/predikce, deskripce a hledání „nugetů“.

4. Porozumění problematice, porozumění datům, příprava dat, modelování, vyhodnocení výsledků a využití výsledků.

2 PŘÍPRAVA DAT

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola se věnuje přípravou dat, jež je nezbytnou součástí většiny procesu zahrnujících dolování dat. V této kapitole se čtenář může seznámit s metodami, které umožňují upravit hrubá data získaná přímo z praktických aplikací na data, která jsou již vhodná pro vybranou metodu dolování dat. Konečně je zde také představeno několik přístupu k situacím, kdy je dat je příliš mnoho anebo naopak situacím, kdy některá z dat chybí.

CÍLE KAPITOLY

Jelikož ne každá data jsou vhodná pro jakoukoli metodu dolování dat, cílem této kapitoly je seznámit čtenáře s několika praktickými přístupy k tomu, jak je možné nevyhovující data pro danou metodu dolování dat přizpůsobit. Čtenář se seznámí s postupy v případech, kdy je v datech objektů nebo atributů příliš mnoho, popřípadě chybí-li některá z hodnot. Čtenář se taktéž naučí zpracovávat numerické a kategoriální atributy takovým způsobem, aby byly vhodné pro použití ve zvolené metodě dolování dat.

KLÍČOVÁ SLOVA KAPITOLY

časová data, odvozené atributy, mnoho objektů, mnoho atributů, diskretizace, chybějící hodnoty

Příprava (předzpracování) dat je nejobtížnější a časově nejnáročnější krok celého procesu dobývání znalostí z databází. Současně je to ale krok, který má klíčový význam pro úspěch dané aplikace. Je to ta část práce, která (spolu s krokem porozumění problému) vyžaduje největší podíl spolupráce s expertem z dané aplikační oblasti. Skoro lze říci, že problémy se získáním „těch správných“ dat pro učící se systémy jsou podobné problémům se získáváním znalostí u systémů expertních.

K ZAPAMATOVÁNÍ

Cíl předzpracování je obvykle dvojí:

- vybrat (nebo vytvořit) z dostupných dat ty údaje, které jsou relevantní pro zvolenou úlohu dobývání znalostí,

- reprezentovat tyto údaje v podobě, která je vhodná pro zpracování zvoleným algoritmem.

Zatímco první cíl úzce souvisí s porozuměním problému i s porozuměním datům (a je tedy závislý na aplikační oblasti), druhý cíl (přizpůsobit data uvažovanému algoritmu) je relativně aplikačně nezávislý.

Tato kapitola se z pochopitelných důvodů zaměřuje na druhý cíl, tedy na otázku, jak uzpůsobit data pro jednotlivé analytické procedury. Při předzpracování potřebujeme znát (a tedy z hlediska knihy popsat) požadavky jednotlivých metod na podobu vstupních dat. Často může podoba dat vést k volbě metody. Přidržíme se přitom podoby, která je společná algoritmům, kterým se budeme věnovat v kapitolách 3, 4 a především 5; reprezentaci dat pomocí jedné datové tabulky, zachycující hodnoty atributů objektů. To bude tedy cílový stav, ke kterému se budeme propracovávat při řešení dále uvedených problémů. Prostředkem k dosažení tohoto cíle budou různé metody transformace, selekce a tvorby atributů a objektů.

2.1 Jiná než relační data

Kromě dat uchovávaných v relačních tabulkách, u kterých nezáleží na pořadí objektů (objekty jsou navzájem nezávislé) můžeme v řadě oblastí narazit na data s určitou strukturou. K takovým datům patří

- časová data (např. časové řady kurzů akcií)

- prostorová data (např. geografické informační systémy)

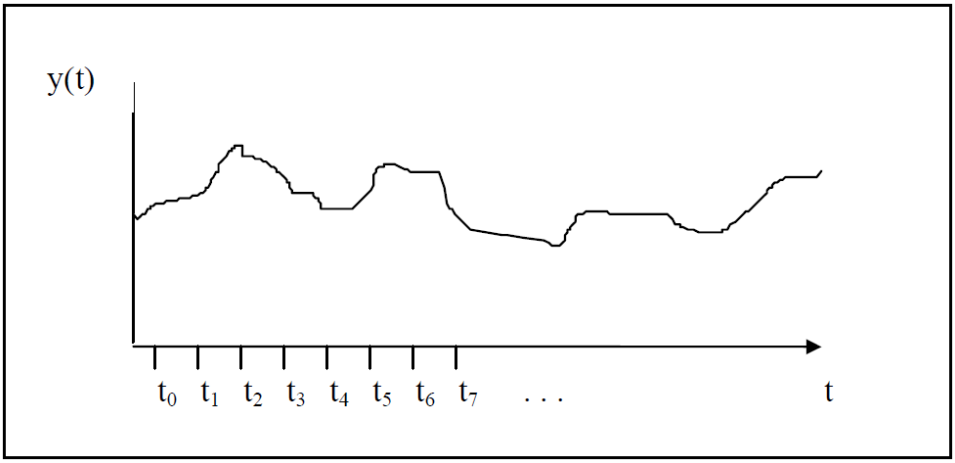
- strukturální data (např. chemické sloučeniny)

Na opačné straně stojí naopak data nestrukturovaná, například texty.

Časová data jsou obvykle tvořena hodnotami téže veličiny (nebo více veličin) zaznamenávaných v různých okamžicích. Veličinou (typicky numerickou) může být kurz ceny akcií, spotřeba elektrické energie, výsledek laboratorního testu v nemocnici, transakce na účtu v bance nebo sekvence navštívených webových stránek (tzv. clickstream). Údaje tedy mohou být zaznamenávány jak v ekvidistantních časových intervalech (denní kurzy akcií) nebo v libovolných okamžicích (návštěvy u lékaře).

Obvyklou úlohou pro časová data je predikce budoucí hodnoty veličiny (např. kurzu akcií) na základě hodnot zjištěných v minulosti. Tomu odpovídá i způsob předzpracování, který jsme již viděli v kapitole o neuronových sítích. Z časové řady (Obrázek 10) se postupně vybírá úsek dané

délky tak, že jeho „začátek“ představuje hodnoty vstupních atributů a jeho „konec“ hodnoty cílového atributu (Obrázek 11).



Obrázek 10: Původní časová řada

Jiným příkladem použití (a předzpracování) časově závislých dat je situace, kdy tato data představují jen dílčí informace popisující složitější objekt. Příkladem může být úloha diagnostikování choroby mimo jiné (ale nikoliv pouze) na základě opakovaných vyšetření. Pak se sekvence výsledků téhož laboratorního testu chápe (ve vztahu k pacientovi) jako relace *n:1* (viz dále).

vstupy				výstup
$y(t_0)$	$y(t_1)$	$y(t_2)$	$y(t_3)$	$y(t_4)$
$y(t_1)$	$y(t_2)$	$y(t_3)$	$y(t_4)$	$y(t_5)$
$y(t_2)$	$y(t_3)$	$y(t_4)$	$y(t_5)$	$y(t_6)$
	. . .			

Obrázek 11: Časová řada po transformaci

V případě prostorových dat hraje mezi jednotlivými objekty roli implicitní relace sousednosti. Obvykle předpokládáme, že hodnoty atributů u „sousedních“ objektů nebudou příliš odlišné. Většina analytických metod nedokáže tuto vazbu mezi objekty vzít do úvahy. Spíše se tedy použije při interpretaci, nebo jako dodatečné požadavky na nalezené znalosti.

2.2 Odvozené atributy

Z atributů přítomných v původních datech lze samozřejmě vytvářet odvozené atributy. Někdy tvorbu nových atributů vyžaduje způsob předzpracování (např. atributy pro agregované hodnoty při spojování relací), jindy tvorba atributů plyne z doménových znalostí (např. převod rodného čísla klienta na věk a pohlaví). Opět se tedy jedná o operaci, kdy je třeba úzce spolupracovat s expertem.

2.3 Příliš mnoho objektů

V reálných úlohách se setkáváme s databázemi, které obsahují desítky a stovky tisíc objektů, zvláštností nejsou ani ještě větší počty. V případě algoritmů pracujících v inkrementálním režimu žádný problém nevzniká. Jiné je to v případě algoritmů pracujících v dávkovém režimu, které předpokládají zpracování celých (trénovacích) dat naráz v operační paměti. Pro řešení tohoto problému se nabízí:

- použít jen určitý vzorek (sample) vybraný z celých dat,
- použít takový způsob uložení dat, který by umožnil přístup ke všem objektům, aniž by je celé ukládal do operační paměti,
- vytvořit více modelů na základě podmnožin objektů a modely poté zkombinovat.

Při výběru určitého vzorku se opět vynořuje otázka reprezentativnosti dat. Požadujeme totiž, aby vybrané objekty co nejlépe vystihovaly celá data[5]. Shodu vzorku s celými daty můžeme vyhodnocovat např. na základě shody rozdělení hodnot atributů ve vybraném vzorku i v celých datech. V praxi se často spokojíme s tím, že tuto shodu zjišťujeme pouze pro cílový atribut určující zařazení objektů do tříd. Pak je na místě dostatečně veliký náhodný výběr. Výjimku z tohoto požadavku představuje situace, kdy jsou třídy v původních datech nevyvážené. Je-li např. v celých datech podíl objektů jedné třídy 95% a objektů druhé třídy 5%, bude mít většina analytických algoritmů tendenci preferovat majoritní třídu. Nalezené znalosti by tedy měly podobu defaultu "každý objekt patří do majoritní třídy". Často nás ale zajímají znalosti týkající se té druhé třídy. V takovém případě musíme buď vzít do úvahy různé ceny za různý typ chybného rozhodnutí[6], nebo musíme upravit trénovací data do takové podoby, kdy jedna třída nebude dominovat té druhé. Ve druhém případu budeme tedy vybírat příklady různých tříd s různou pravděpodobností (resp. vážit příklady různých tříd různými vahami)[7].

Při výběru objektů do trénovací množiny se samozřejmě můžeme řídit i dalšími dodatečnými informacemi: můžeme např. vybírat typické příklady, nebo naopak atypické příklady, pouze příklady bez chybějících hodnot apod. Jistou variantou vzorkování je i křížová validace, při které se z dat opakovaně vybere jen určitá část pro trénování a jiná část pro testování (viz kapitola o evaluaci). Zde můžeme tento princip použít tak, že vytvoříme různé modely na základě různých vzorků, a pro vlastní klasifikaci použijeme model nejlepší.

2.4 Příliš mnoho atributů

V reálných úlohách se setkáváme s databázemi, které obsahují desítky a stovky atributů. Ne všechny z nich jsou samozřejmě relevantní pro zamýšlenou analýzu, takže stojíme před otázkou redukce tohoto počtu. Odpověď lze hledat u experta (který může vědět, které atributy jsou pro

[5] Pro úplné pokrytí prostoru atributů bychom pro n binárních atributů potřebovali 2^n různých objektů.
[6] Podstatně horší chybou (chybou s větší cenou) bude chybné zařazení minoritního příkladu do majoritní třídy.
[7] Extrémní variantou tohoto přístupu je použít všechny příklady minoritní třídy a doplnit je náhodně vybranými příklady majoritní třídy tak, aby podíl obou tříd ve trénovacích datech byl stejný.

danou úlohu relevantní), nebo v automatických metodách, které nabízejí následující dva způsoby:

- *transformaci*, kdy z existujících atributů vytvoříme menší počet atributů nových,
- *selekci*, kdy z existujících atributů vybereme jen ty nejdůležitější[8].

Otázka redukce dimenzionality je dobře známa z oblasti rozpoznávání obrazů[9]. Pro transformaci atributů se zde používá např. Karhounen Loevův rozvoj, faktorová analýza nebo analýza hlavních komponent. Tyto metody předpokládají použití výhradně numerických atributů. Základem všech uvedených metod je reprezentace objektů pomocí hodnot nových atributů, které vzniknou jako lineární kombinace atributů původních.

Jistou nevýhodou těchto metod tedy je skutečnost, že potřebujeme znát (měřit) hodnoty všech původních atributů. Nové, transformované atributy navíc nemusí mít srozumitelnou interpretaci, což může být v konkrétní úloze na závadu.

Při (automatizované) selekci atributů obvykle hledáme ty atributy, které nejlépe přispějí ke klasifikaci objektů do tříd. K tomuto problému můžeme přistoupit dvojím způsobem. Buď pro každý atribut spočítat nějakou charakteristiku vyjadřující vhodnost atributu pro klasifikaci (tento přístup bývá nazýván *metoda filtru* – filter approach), nebo použít nějaký algoritmus strojového učení pro vytvoření modelu z (pod)množiny atributů a vyhodnotit model (tento přístup bývá nazýván *metoda obálky* – wrapper approach).

Pro metodu filtru lze použít kritéria používaná pro volbu atributu pro větvení rozhodovacího stromu (entropii nebo informační zisk), vzájemnou informaci nebo χ^2 test. Tato kritéria vycházejí z kontingenční tabulky. Podobu tabulky pro (vstupní) atribut *A* a cílový atribut *C* ukazuje Tabulka 1, kde a_{kl} je četnost (frekvence) kombinace $(A(v_k) \wedge (C(v_l))$ a

$$r_k = \sum_{l=1}^{S} a_{kl},\ s_l = \sum_{k=1}^{L} a_{kl} \text{ a } n = \sum_{k=1}^{R} \sum_{l=1}^{S} a_{kl}.$$

Uvedené četnosti lze použít pro odhad pravděpodobností:

$$P(A(v_k) \wedge (C(v_l)) = \frac{a_{kl}}{n} \quad \text{a} \quad P(A(v_k)) = \frac{r_k}{n} \quad \text{a} \quad P((C(v_l)) = \frac{s_l}{n}$$

[8] V anglicky psané literatuře se požívá termín feature selection. V relační algebře se tento typ operace nazývá projekce.

[9] Rozpoznávání obrazů (pattern recognition) je relativně samostatná oblast umělé inteligence, řešící úlohy klasifikace tak jak jsou popsány v této knize: Na základě trénovací množiny se hledá obecný popis tříd. Každý objekt je reprezentován numerickým vektorem příznaků (feature vector), modely používané pro klasifikaci tedy reprezentují znalosti o třídách v podobě diskriminačních funkcí nebo etalonů.

Tabulka 1: Kontingenční tabulka pro atribut A a třídu C.

	$C(v_1)$	$C(v_2)$	.....	$C(v_S)$	Σ
$A(v_1)$	a_{11}	a_{12}	.....	a_{1S}	r_1
$A(v_2)$	a_{21}	a_{22}	.....	a_{2S}	r_2
:	:	:		:	:
:	:	:		:	:
$A(v_R)$	a_{R1}	a_{R2}	.....	a_{RS}	r_R
Σ	s_1	s_2	.....	s_S	n

Příslušná kritéria jsou pak

1. χ^2 (čím větší hodnota, tím lépe)

$$\chi^2 = \sum_{k=1}^{R}\sum_{l=1}^{S}\frac{(a_{kl}-e_{kl})^2}{e_{kl}} = n\times\sum_{k=1}^{R}\sum_{l=1}^{S}\frac{\left(a_{kl}-\frac{r_k\,s_l}{n}\right)^2}{r_k\,s_l}$$

2. entropie *H(A)* (čím menší hodnota, tím lépe)

$$H(A)=\sum_{k=1}^{R}\frac{r_k}{n}\,H(A(v_k))\ , \quad \text{kde} \quad H(A(v_k))=-\sum_{l=1}^{S}\frac{a_{kl}}{r_l}\log\frac{a_{kl}}{r_l}$$

3. informační míra závislosti *ID(A,C)* (čím větší hodnota, tím lépe)

$$ID(A,C)=\frac{MI(A,C)}{H(C)}=\frac{MI(A,C)}{-\sum_{l=1}^{S}\frac{s_l}{n}\log\frac{s_l}{n}}\ ,$$

kde vzájemná informace *MI(A,C)* je

$$MI(A,C)=\sum_{k=1}^{R}\sum_{l=1}^{S}P(A(v_k)\wedge C(v_l))\log\frac{P(A(v_k)\wedge C(v_l))}{P(A(v_k))\,P(C(v_l))}=\sum_{k=1}^{R}\sum_{l=1}^{S}\frac{a_{kl}}{n}\log\frac{\frac{a_{kl}}{n}}{\frac{r_k}{n}\frac{s_l}{n}}=$$

$$=\frac{1}{n}\sum_{k=1}^{R}\sum_{l=1}^{S}a_{kl}\log\frac{n\,a_{kl}}{r_k\,s_l}\ .$$

Atributy lze uspořádat podle hodnoty kritéria; pak můžeme vybrat jen určitý počet těch nejlepších. Lze postupovat „zdola nahoru“ – tedy přidáváním atributů, nebo „shora dolů“ – tedy odstraňováním atributů z původní množiny všech atributů. Nevýhodou uvedeného způsobu selekce je, že posuzuje každý atribut zvlášť a nezachycuje současný vliv více atributů na správnost klasifikace.

Jiným příkladem, jak vybírat množiny atributů, je metoda obálky, která využívá (hrubé) výpočetní síly současných počítačů [John a kol, 1994]. Z množiny všech použitelných atributů se opakovaně vybírá určitá část a ta se pak použije pro popis objektů v trénovací i testovací množině. Pro každou trénovací množinu se vytvoří jeden model, který bude otestován na testovacích datech téže struktury atributů. Z takto vytvořených modelů se pak použije ten nejlepší[10] – volba modelu zároveň určí relevantní atributy.

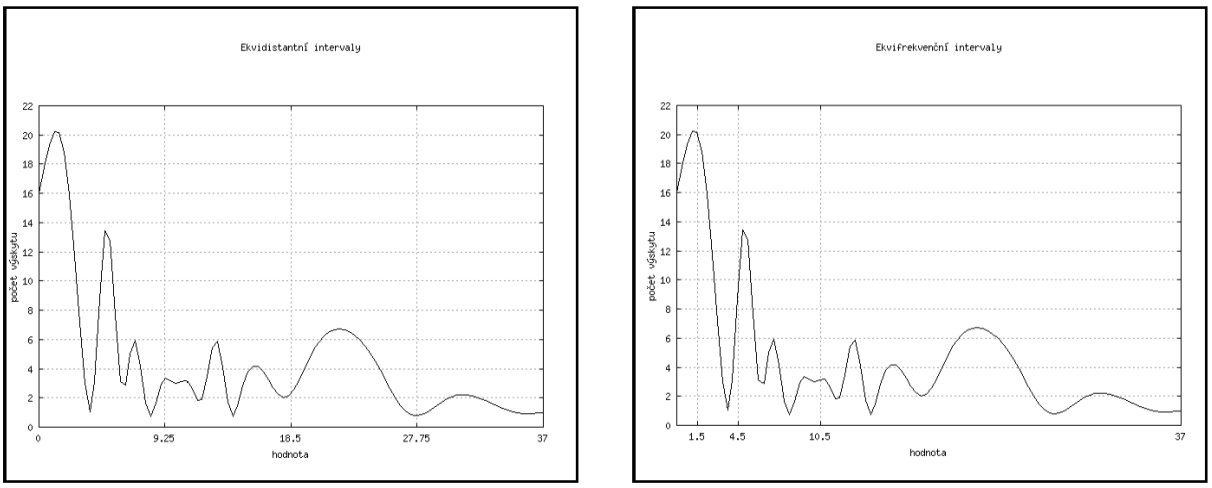
Metodu obálky lze použít „zdola nahorů“ (začít od modelů vytvořených pro jednotlivé atributy a atributy postupně přidávat), „shora dolů“ (začít od modelu vtvořeného pro původní množinu atributů a atributy postupně odstraňovat), nebo pro náhodný způsob výběru podmnožiny atributů.

2.5 Numerické atributy

Jako jisté omezení některých algoritmů dobývání znalostí se může jevit to, že pracují pouze s kategoriálními daty. Numerické atributy tedy nejprve musíme diskretizovat (rozdělit na intervaly). Někdy se diskretizace provádí v průběhu práce algoritmu, jindy se provádí ve fázi předzpracování dat.

Diskretizace často závisí na povaze řešeného problému; na znalostech, které teprve hledáme. Je-li k dispozici expert, je lépe mu tuto diskretizaci přenechat. Existují ale i metody umožňující provádět diskretizaci automaticky. K nejjednodušším metodám patří diskretizace na předem zadaný počet ekvidistantních nebo ekvifrekventních intervalů. V prvním případě se obor hodnot numerického atributu rozdělí na stejně dlouhé intervaly (Obrázek 12 vlevo); stačí tedy znát jen rozsah hodnot. Ve druhém případě se obor hodnot rozdělí na intervaly, které obsahují (zhruba) stejný počet objektů (Obrázek 12 vpravo); tady už potřebujeme znát počty výskytu jednotlivých hodnot. V obou případech se ale numerický atribut diskretizuje bez využití informací o hodnotách jiných atributů. Jiné metody využívají při diskretizaci informací o příslušnosti objektů k různým třídám, viz kapitola 7.6 v [Berka, 2003].

[10] Kritériem kvality modelu je správnost klasifikace na testovacích datech.



Obrázek 12: (vlevo) Ekvidistantní intervaly. (vpravo) Ekvifrekvenční intervaly.

2.6 Kategoriální atributy

Chceme-li použít algoritmy, které používají numerické vstupy (např. neuronové sítě), musíme hodnoty kategoriálních atributů zakódovat pomocí čísel. V případě **binárních** atributů to mohou být např. hodnoty 0, 1 nebo –1, 1, v případě **ordinálních** atributů to může být pořadové číslo hodnoty, v případě **nominálních** atributů pak v zásadě libovolné číslo.

2.7 Chybějící hodnoty

Otázkou práce s chybějícími hodnotami jsme se již zabývali. Připomeňme tedy již dříve uvedené způsoby:

1) ignorovat objekt s nějakou chybějící hodnotou,

2) nahradit chybějící hodnotu novou hodnotou „nevím“,

3) nahradit chybějící hodnotu některou z existujících hodnot atributu a sice:

   a) nejčetnější hodnotou,

   b) proporcionálním podílem všech hodnot,

   c) libovolnou hodnotou.

Zajímavou alternativou je použít pro doplnění chybějící hodnoty některý z algoritmů pro modelování; atribut s chybějícími hodnotami se považuje za cílový atribut, pro trénování a testování se použijí objekty se známou hodnotou tohoto atributu – a chybějící hodnoty se doplní na základě modelu.

KONTROLNÍ OTÁZKA

1. Jak postupovat při zpracování dat, které obsahují příliš mnoho objektů?
2. Jak postupovat při zpracování dat, které obsahují příliš mnoho atributů?
3. Jak převést nominální kategorický atribut s hodnotami např. Čech, Němec a Slovák na numerický atribut?

SHRNUTÍ KAPITOLY

V této kapitole jsme se věnovali přípravě dat, jež je nezbytnou součástí většiny procesu zahrnujících dolování dat. Seznámili jsme se s metodami, které umožňují upravit hrubá data získaná přímo z praktických aplikací na data, která jsou již vhodná pro vybranou metodu dolování dat. Konečně jsme si také představili několik přístupu k situacím, kdy je dat je příliš mnoho anebo naopak situacím, kdy některá z dat chybí.

ODPOVĚDI

1. a) Použít jen určitý vzorek vybraný z celých dat, b) použít takový způsob uložení dat, který by umožnil přístup ke všem objektům, aniž by je celé ukládal do operační paměti nebo c) vytvořit více modelů na základě podmnožin objektů a modely poté zkombinovat.
2. Odpověď na to, které atributy použít a které ne, lze hledat u experta (který může vědět, které atributy jsou pro danou úlohu relevantní), nebo v automatických metodách, které nabízejí následující a) transformaci, kdy z existujících atributů vytvoříme menší počet atributů nových, nebo b) selekci, kdy z existujících atributů vybereme jen ty nejdůležitější.
3. Pozor, narozdíl od ordinálních atributů (např. s hodnotami malý, střední a velký) nelze pouze přiřadit každé hodnotě číslo dle jejího pořadí, např. malý = 1, střední = 2 a velký = 3, kde průměr malého a velkého dává logicky hodnotu odpovídající střední. U zmíněného nominálního atributu by průměr Čecha a Slováka dával Němce, což smysl nedává. Proto je potřeba vytvořit obecně tolik nových binárních atributů (sloupců), kolik má původní nominální atribut různých hodnot, v našem příkladu tedy 3, a jejich hodnoty pak převést z původního např. *národnost* = Slovák na *národnost_Čech* = 0, *národnost_Slovák* = 1 a *národnost_Němec* = 0.

3 STATISTIKA

RYCHLÝ NÁHLED KAPITOLY

V této kapitole si čtenář osvěží čtyři významné metody pro statistickou analýzu dat, a to kontingenční tabulky, regresní analýzu, diskriminační analýza a shlukovou analýzu.

CÍLE KAPITOLY

Cílem této kapitoly je čtenáři osvěžit paměť co se týče několika významných statistických metod pro analýzu dat. Čtenář si připomene, co je to čtyřpolní kontingenční tabulka a obecná kontingenční tabulka a jak s nimi souvisí testovací statistika chí kvadrát. Dále si čtenář připomene, co obnáší regresní analýza a jak s ní souvisí metoda nejmenších čtverců. Čtenář si také připomene, co je to diskriminační analýza a osvěží si její nejjednodušší případ, což je lineární diskriminační analýza. Konečně si čtenář také zopakuje základní pojmy týkající se shlukové analýzy, jako jsou různé definice míry vzdálenosti, hierarchické aglomerativní shlukování nebo dendrogram.

KLÍČOVÁ SLOVA KAPITOLY

kontingenční tabulky, regresní analýza, diskriminační analýza, shluková analýzu, dendrogram

Statistika nabízí celou řadu teoreticky dobře prozkoumaných a zdůvodněných a léty praxe ověřených metod pro analýzu dat. Pro oblast dobývání znalostí z databází mají význam (ať už přímo jako používané metody nebo nepřímo jako zdroj inspirace)[11]:

- *kontingenční tabulky* – pro zjišťování vztahu mezi dvěma kategoriálními veličinami,
- *regresní analýza* – pro zjišťování funkční závislosti jedné numerické (spojité) veličiny na jiných numerických veličinách,
- *diskriminační analýza* – pro odlišení příkladů (pozorování) patřících do různých tříd,

[11] Vzhledem k tomu, že při dobývání znalostí máme co do činění s množstvím sledovaných veličin, jedná se převážně o mnohorozměrné statistické metody.

- *shluková analýza* – pro nalezení skupin (shluků) navzájem si podobných příkladů.

Z dalších metod zmiňme ještě *korelační analýzu* (pro posouzení, zda je mezi dvěma numerickými veličinami lineární závislost), *analýzu rozptylu* (pro posouzení rozdílu mezi průměry z různých výběrů) a *faktorovou analýzu* (pro zjišťování závislosti jedné veličiny na tzv. faktorech vytvořených jako lineární kombinace jiných veličin).

3.1 Kontingenční tabulky

V nejjednodušším případě můžeme kontingenční tabulku použít pro vyhodnocení vztahu mezi dvěma binárními veličinami. Je-li např. první veličina *příjem_klienta* s hodnotami *{vysoký, nízký}* a druhá veličina *úvěr* s hodnotami *{ano, ne},* bude mít kontingenční tabulka pro *n* pozorování podobu uvedenou v Tabulka 2. Takovéto tabulce se také někdy říká *čtyřpolní tabulka*. Jednotlivá "pole" tabulky odpovídají četnostem kombinací hodnot obou veličin v datech. Tedy:

a je počet klientů s vysokým příjmem, kteří získali úvěr,

b je počet klientů s vysokým příjmem, kteří nezískali úvěr,

c je počet klientů s nízkým příjmem, kteří získali úvěr

d je počet klientů s nízkým příjmem, kteří nezískali úvěr.

V tabulce jsou ještě uvedeny řádkové a sloupcové součty (tzv. marginální hodnoty): a+b=r, c+d=s, a+c=k, b+d=l, a+b+c+d=n

Tabulka 2: (Čtyřpolní) kontingenční tabulka

	Úvěr ano	**Úvěr ne**	**Σ**
Vysoký příjem	a	b	r
Nízký příjem	c	d	s
Σ	k	l	n

V obecném případě může každá z veličin nabývat různého počtu hodnot většího než jedna. Pak má kontingenční tabulka podobu podle Tab. 2, kde

a_{kl} je četnost (frekvence) kombinace $(X=X_k) \wedge (Y=Y_l)$,

$$r_k = \sum_{l=1}^{S} a_{kl},\ s_l = \sum_{k=1}^{R} a_{kl}, \text{ a } n = \sum_{l=1}^{S}\sum_{k=1}^{R} a_{kl}.$$

Tabulka 3: (Obecná) kontingenční tabulka

	Y_1	Y_2	…….	Y_S	$\sum$
X_1	a_{11}	a_{12}	…….	a_{1S}	r_1
X_2	a_{21}	a_{22}	…….	a_{2S}	r_2
:	:	:		:	:
:	:	:		:	:
X_R	a_{R1}	a_{R2}	…….	a_{RS}	r_R
$\sum$	s_1	s_2	…….	s_S	n

Pro zjišťování vztahu mezi oběma veličinami se obvykle používá χ^2 test. Tento test je založen na vyhodnocení rozdílu mezi pozorovanými četnostmi jednotlivých kombinací (uvedenými v tabulce) a četnostmi očekávanými při platnosti hypotézy o nezávislosti obou veličin (počítanými z marginálních hodnot). Je-li pozorována četnost

$$a_{kl},$$

je jí odpovídající očekávaná četnost (za předpokladu nezávislosti *X* a *Y*)

$$e_{kl} = \frac{r_k s_l}{n}.$$

Pro kontingenční tabulku spočítáme statistiku χ^2

$$\chi^2 = \sum_{i=1}^{R}\sum_{j=1}^{S}\frac{(a_{ij} - o_{ij})^2}{o_{ij}} = n \times \sum_{i=1}^{R}\sum_{j=1}^{S}\frac{\left(a_{ij} - \frac{r_i \cdot s_j}{n}\right)^2}{r_i \cdot s_j}.$$

Při platnosti nulové hypotézy nezávislosti veličin *X* a Y

$H_0 : P(X=X_k \wedge Y=Y_l) = P(X=X_k) \times P(Y=Y_l)$ pro všechna *k,l*

má tato statistika rozdělení χ^2 s *(R-1)(S-1)* stupni volnosti. Je-li hodnota χ^2 statistiky větší nebo rovna hodnotě χ^2 rozdělení[12] s daným počtem stupňů volnosti na zvolené hladině významnosti α

$$\chi 2 \geq \chi^2(R\text{-}1)(S\text{-}1)(\alpha)$$

zamítneme na této hladině významnosti nulovou hypotézu ve prospěch alternativní hypotézy závislosti obou veličin.

3

[12] Hodnoty rozdělení pro různé počty stupňů volnosti a různé hladiny významnosti jsou uvedeny v každé učebnici statistiky.

ŘEŠENÁ ÚLOHA

Např., vytvoříme-li na základě dat čtyřpolní tabulku Tabulka 4, spočítáme hodnotu statistiky $\chi 2 = 84.65$ (očekávané četnosti jsou uvedeny v Tabulka 5). Vzhledem k tomu, hodnota χ^2 rozdělení s jedním stupněm volnosti je (pro hladinu významnosti $\alpha=0.05$) $\chi^2(1)(0.05) = 3.84$, zamítneme nulovou hypotézu a předpokládáme závislost mezi výší příjmu a poskytnutím úvěru.

Tabulka 4: Data pro výpočet χ2 testu

	Uvěr ano	**Uvěr ne**	∑
Vysoký příjem	50	0	50
Nízký příjem	30	40	70
∑	80	40	120

Tabulka 5: Skutečné vs. očekávané počty pozorování

Kombinace	**skutečný počet**	**očekávaný počet**	**rozdíl**
Vysoký příjem, úvěr ano	50	33.3	16.7
Vysoký příjem, úvěr ne	0	16.7	-16.7
Nízký příjem, úvěr ano	30	46.7	-16.7
Nízký příjem, úvěr ne	40	13.3	26.7

χ2 test má ale jedno úskalí; lze ho použit pouze v případě, že očekávané četnosti jsou dostatečně veliké. Uvádí se tedy podmínka použitelnosti testu

$$\frac{r_k s_l}{n} \geq 5 \text{ pro všechna k, l.}$$

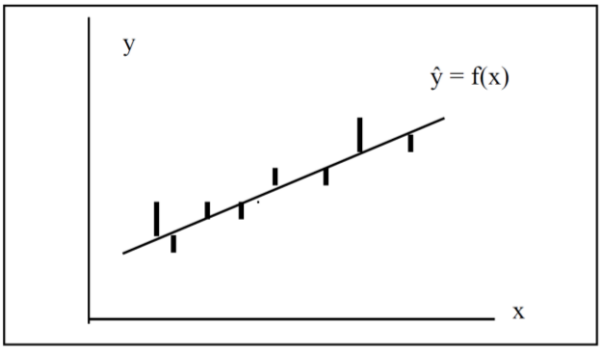
3.2 Regresní analýza

Zatímco při korelační analýze nás zajímá pouze to, zda mezi dvěma numerickými veličinami platí lineární závislost[13], v případě *lineární regrese* nás zajímají parametry této závislosti. Řešíme tedy úlohu a*proximace* pozorovaných hodnot daným typem funkce, ovšem s neznámými parametry. V nejjednodušším případě lineární regrese pro dvě veličiny hledáme hodnoty parametrů q_1 a q_0 pro rovnici

[13] Korelační koeficient $\rho \in [-1,1]$ nabývá hodnotu 1 pro přímou úměru obou veličin, hodnotu 1 pro nepřímou úměru obou veličin a hodnotu 0 v případě, že mezi veličinami není lineární závislost. To, že korelační koeficient je nulový ještě neznamená, že mezi veličinami není funkční vztah; příkladem mohou být veličiny x a y, kde y=|x|.

$$y = q_1 x + q_0 + \varepsilon.$$

Máme-li k dispozici vhodná pozorováni (dvojce hodnot $[x_i, y_i]$) můžeme parametry rovnice spočítat (přesněji řečeno odhadnout) na základě *metody nejmenších čtverců*. Tato metoda minimalizuje rozdíly mezi pozorovanou hodnotou y a očekávanou hodnotou $\hat{y} = f(x)$ spočítanou v tomto případě na základě funkce $q_1 x + q_0$ (viz ilustrační Obrázek 13 znázorňující tyto odchylky).



Obrázek 13: Metoda nejmenších čtverců

Vzhledem k tomu, že považujeme kladné rozdíly za stejně závažné jako rozdíly záporné, uvažujeme druhou mocninu[14] (kvadrát, čtverec) těchto rozdílů:

$$(y - f(x))^2$$

Úlohu pro n pozorování lze tedy formálně zapsat jako hledání

$$\min \sum_{i=1}^{n} (y_i - f(x_i))^2.$$

3.3 Diskriminační analýza

Diskriminační analýza je vlastně úloha klasifikace příkladů do předem zadaných tříd. Z pohledu statistického tedy úloha hledání závislosti jedné nominální veličiny (určující příslušnost ke třídě) na dalších numerických veličinách. Při diskriminační analýze předpokládáme, že ke každé třídě (hodnotě nominální veličiny) c_t, $t=1,...,T$ existuje *(diskriminační) funkce* f_t taková, že

$$f_t(\mathbf{x}) = \max_k f_k(\mathbf{x}),\ k=1,...,T$$

právě když příklad $\boldsymbol{x}=[x1, x2, ..., x_m]$ patří do třídy c_t.

V případě *lineární diskriminační analýzy* mají diskriminační funkce f_t podobu lineární kombinace

[14] Obecněji uvažujeme nějakou sudou funkci, tj. funkci f(x) takovou, že f(-x)=f(x). Druhé mocnině se dává přednost před absolutní hodnotou, protože druhá mocnina je hladká funkce.

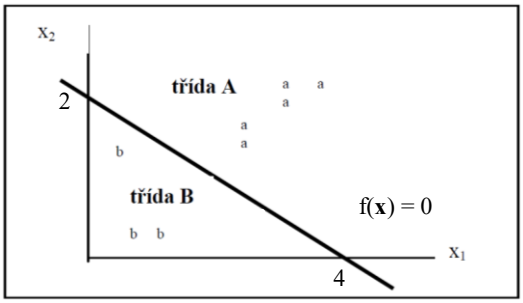
$$f_t = q_{0j} + q_{1j}\, x_1 + q_{2j}\, x_2 + ... + q_{mj}\, x_m$$

Ukažme si pouze ten nejjednodušší případ, diskriminaci do dvou tříd, kdy místo funkcí f_1 a f_2 můžeme hledat funkci

$$f(\mathbf{x}) = f_1(\mathbf{x}) - f_2(\mathbf{x}).$$

Příklady pak můžeme klasifikovat podle znaménka této funkce. Pokud si představíme příklady jako body v *m*-rozměrném prostoru veličin, bude množina řešení rovnice $f(\mathbf{x}) = 0$ představovat nadrovinu v tomto prostoru, oddělující od sebe příklady obou tříd. Pro $m=2$ a data z Tabulka 6 toto řešení odpovídá přímce, např. jak je zobrazeno Obrázek 14. Přímka daná $f(\mathbf{x}) = 0$ může mít rovnici ve tvaru

$$0 = x_1 + 2x_2 - 4.$$



Obrázek 14: Rozdělovací přímka

Tabulka 6: Data pro diskriminační analýzu při $m = 2$.

x_1	x_2	Třída (c_t)
0,5	0,5	b
1	0,5	b
0,4	1,5	b
2,5	1,7	a
2,5	2	a
3	2,3	a
3	2,5	a
4	2,5	a

3.4 Shluková analýza

Shluková analýza hledá odpověď na otázku, zda lze pozorované příklady rozdělit do skupin (shluků) vzájemně si blízkých příkladů. Vychází se tedy z toho, že umíme měřit vzdálenost mezi příklady.

Předpokládejme, že každý příklad je charakterizován *m* numerickými veličinami. Vzdálenost mezi dvěma příklady $\mathbf{x_1} = [x_{11},...,x_{1m}]$ a $\mathbf{x_2} = [x_{21},...,x_{2m}]$ lze vyjádřit různými mírami. Uveďme zde např.:

- Hammingovu vzdálenost

$$d_H(\mathbf{x_1},\mathbf{x_2}) = \sum_{j=1}^{m} |x_{1j} - x_{2j}|$$

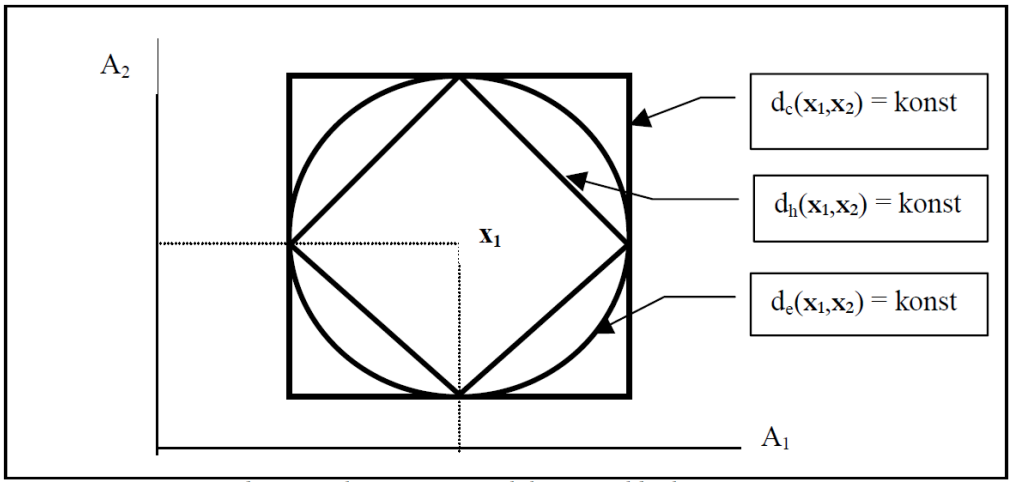
- Eukleidovskou vzdálenost

$$d_E(\mathbf{x_1},\mathbf{x_2}) = \sqrt{\sum_{j=1}^{m} (x_{1j} - x_{2j})^2}$$

- Čebyševovu vzdálenost

$$d_C(\mathbf{x_1},\mathbf{x_2}) = \max_j |x_{1j} - x_{2j}|$$

Rozdíl mezi $d_H(\boldsymbol{x_1},\boldsymbol{x_2})$, $d_E(\boldsymbol{x_1},\boldsymbol{x_2})$ a $d_C(\boldsymbol{x_1},\boldsymbol{x_2})$ ukazuje Obrázek 15. Zde jsou (pro jednotlivé míry) znázorněny body $\boldsymbol{x_2}$, které mají stejnou vzdálenost od bodu $\boldsymbol{x_1}$: kruh pro eukleidovskou vzdálenost, čtverec rovnoběžný s osami pro Čebyševovu vzdálenost a čtverec „na špičce“ pro Hammingovu vzdálenost [Hebák, Hustopecký, 1987].



Obrázek 15: Body x2 se stejnou vzdáleností od bodu x1.

Výše uvedené míry vzdálenosti závisí na měřítku veličin. Proto je třeba veličiny normovat. Konkrétní hodnota se obvykle dělí nějakou jinou hodnotou: průměrem, směrodatnou odchylkou, nebo rozpětím (max-min).

Z používaných metod shlukové analýzy zmiňme:

- hierarchické shlukování,
- metoda *K*-středů (*K*-means clustering).

Při *hierarchickém shlukování* se obvykle postupuje metodou „zdola nahoru“. Začíná se tedy v situaci, kdy každý příklad tvoří jeden samostatný shluk. Postupně se pak jednotlivé shluky spojují, až skončíme s jedním shlukem obsahujícím všechny příklady (Obrázek 16).

Algoritmus hierarchického shlukování

Inicializace

1. urči vzájemné vzdálenosti mezi všemi přiklady
2. zařaď každý přiklad do samostatného shluku

hlavní cyklus

1. dokud je vice než jeden shluk

1.1. najdi dva navzájem nejbližši shluky a spoj je 1.2. spočitej pro tento nový shluk jeho vzdálenost od ostatnich shluků

Obrázek 16: Algoritmus hierarchického shlukováni

Vzdálenost mezi shluky lze stanovit různým způsobem:

- *metoda nejbližšího souseda* - vzdálenost mezi shluky ***U*** a ***V*** je dána minimem ze vzdálenosti mezi jejich příklady

$$D(\mathbf{U},\mathbf{V}) = \min_{k,l} d(\mathbf{x}_k, \mathbf{x}_l),\ \mathbf{x}_k \in \mathbf{U},\ \mathbf{x}_l \in \mathbf{V}$$

- *metoda nejvzdálenějšího souseda* - vzdálenost mezi shluky ***U*** a ***V*** je dána maximem ze vzdálenosti mezi jejich příklady

$$D(\mathbf{U},\mathbf{V}) = \max_{k,l} d(\mathbf{x}_k, \mathbf{x}_l),\ \mathbf{x}_k \in \mathbf{U},\ \mathbf{x}_l \in \mathbf{V}$$

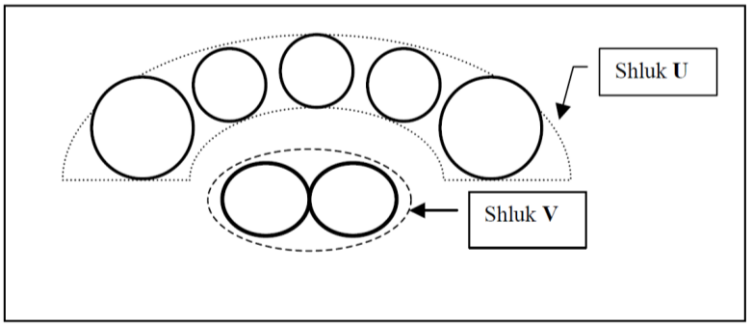
- *metoda průměrné vzdálenosti* - vzdálenost mezi shluky ***U*** a ***V*** je dána průměrem ze vzdálenosti mezi jejich příklady (n_U je počet příkladů ve shluku **U** **a** n_V je počet příkladů ve shluku **V**)

$$D(\mathbf{U},\mathbf{V}) = \frac{1}{n_U n_V} \sum_{k=1}^{n_U} \sum_{l=1}^{n_V} d(\boldsymbol{x}_k, \boldsymbol{x}_l)$$

- *centroidní metoda* - vzdálenost mezi shluky **U** a **V** je dána vzdálenosti mezi středy shluků.

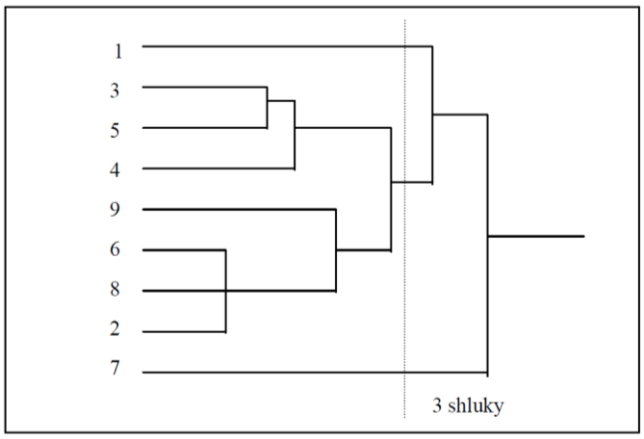
$$D(\mathbf{U},\mathbf{V}) = d(\mathbf{u}, \mathbf{v}),\ \mathbf{u} \text{ je střed shluku } \mathbf{U} \text{ a } \mathbf{v} \text{ je střed shluku } \mathbf{V}$$

Centroidy (středy shluků), zmíněné v předcházejícím výčtu, představuji jakési prototypy reprezentující jednotlivé shluky[15]. Nemusí ale platit, že ke každému shluku patři právě jeden centroid. V závislosti na tvaru shluku a zvolené míře pro výpočet vzdálenosti může být jeden shluk reprezentován vice centroidy (na Obrázek 17 znázorněnými jako tučné kružnice).[16]



Obrázek 17: Více centroidů pro jeden shluk.

Proces hierarchického shlukování bývá zachycen v podobě tzv. dendrogramu. Ten ukazuje (zleva doprava[17]) postupné spojováni shluků počínaje očíslovanými příklady. Optimální počet shluků zde není předem znám, odvodíme ho až rozborem výsledků – tak, že někde dendrogram „rozřízneme“ (Obrázek 18).



Obrázek 18: Dendrogram

Při shlukování metodou *K*-středů předpokládáme, že víme do kolika shluků je možno příklady rozdělit. Počet shluků se tedy během výpočtu nemění, mění se pouze zařazení příkladů k těmto shlukům. Proto je tato metoda méně výpočetně náročná než hierarchické shlukování (a tudíž vhodnější pro větší datové soubory). Příslušný algoritmus je uveden níže.

[15] V nejjednodušším případě můžeme centroidy chápat jako příklady, které nabývají průměrných hodnot jednotlivých veličin v rámci daného shluku. Takto chápané centroidy zaručuji optimalitu klasifikace v situaci, kdy aposteriorní pravděpodobnosti zařazeni příkladů do tříd se řídí normálním rozdělením se stejnou (jednotkovou) kovariančni matici a stejnou pravděpodobnosti jednotlivých tříd – vice viz odstavec diskriminační analýza.
[16] Všimněme si toho, že dané shluky nejsou lineárně separabilní. Lineární diskriminační analýza tedy nedokáže bezchybně od sebe odlišit příklady obou shluků.
[17] Někdy se také používá zobrazení "zespoda nahoru", viz Obrázek 19.

Metoda *K* -středů

1. náhodně zvol rozklad do K shluků

2. urči centroidy pro všechny shluky v aktuálním rozkladu

3. pro každý příklad $\boldsymbol{x}$

   3.1. urči vzdálenosti $d(\boldsymbol{x}, \boldsymbol{c}_k)$, $k = 1, \ldots, K$ kde $\boldsymbol{c}_k$ je centroid k-tého shluku

   3.2. nechť $d(\boldsymbol{x},\boldsymbol{c}_l) = \min_k d(\boldsymbol{x}, \boldsymbol{c}_k)$

   3.3. není-li $\boldsymbol{x}$ součástí shluku l (k jehož centroidu $\boldsymbol{c}_l$ má nejblíže) přesuň $\boldsymbol{x}$ do shluku l

4. došlo-li k nějakému přesunu potom jdi na 2, jinak konec

Uvedený algoritmus může mít určité varianty:

- místo počátečního rozkladu lze za centroidy prohlásit prvních K příkladů; odpadne tak krok 2 při prvním průchodu daty,
- přepočet centroidů lze provádět po každém přesunu (tedy v cyklu v kroku 3).

Výsledné shluky jsou při použití metody K-středů reprezentovány svými centroidy. Tuto reprezentaci lze snadno použít pro zařazování nových příkladů. Příklad bude (ve shodě s krokem 3.3 algoritmu) zařazen do shluku, k jehož centroidu má nejblíže.

KONTROLNÍ OTÁZKA

1. Pro co slouží χ^2 test a jakým způsobem souvisí s kontingenční tabulkou?

2. Co minimalizuje metoda nejmenších čtverců?

3. Co je základním předpokladem použití diskriminační analýzy?

4. Nechť A = [0, 0] a B = [1, 1] jsou dva dvourozměrné body. Která ze tří vzdáleností (Hammingova, Eukleidovská, Čebyševova) je pro tyto dva body největší a kolik?

5. Jak proběhne hierarchické shlukování pro 4 jednorozměrné body A = [0], B = [1], C = [3] a D = [4,5] pro *eukleidovskou* vzdálenost a metodu *nejbližšího* souseda.

6. Nechť $c_1 = [0, 0]$ a $c_2 = [5,2]$ jsou centroidy shluků 1 a 2. Ke kterému z těchto dvou shluků metoda K-středů zařadí příklad $\boldsymbol{x} = [3, 0]$, je-li používána Hammingova vzdálenost?

SHRNUTÍ KAPITOLY

V této kapitole jsme si připomněli čtyři významné metody pro statistickou analýzu dat. Např. co je to čtyřpolní kontingenční tabulka a obecná kontingenční tabulka a jak s nimi souvisí testovací statistika chí kvadrát, co obnáší regresní analýza a jak s ní souvisí metoda nejmenších čtverců, co je to lineární diskriminační analýza a zopakovali jsme si základní pojmy týkající se shlukové analýzy, jako jsou různé definice míry vzdálenosti, hierarchické aglomerativní shlukování nebo co je to dendrogram.

ODPOVĚDI

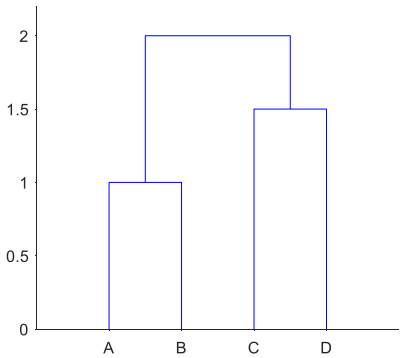
1. Pro ověření závislosti pozorovaných hodnot dvou náhodných veličin, které mohou nabývat konečný počet hodnot.

2. Rozdíly mezi pozorovanou hodnotou y a očekávanou hodnotou $\hat{y}=f(x)$.

3. To, že pro každou třídu $t = 1, \ldots, T$ v datech existuje nějaká funkce f_t a každá z těchto funkcí má vlastnost, že její hodnota pro objekt o je nejvyšší ze všech $f_1(o), \ldots, f_T(o)$, je-li objekt ze třídy odpovídající této funkci.

4. Hammingova, $d_H(A, B) = 2$.

5. Eukleidovská vzdálenost mezi jednorozměrnými body odpovídá absolutní hodnotě rozdílu hodnot bodů, tedy např. $d_E(A, C) = |0 – 3| = 3$. V první kroku tedy vznikne shluk S_{AB} z bodů A a B, které se shluknou na vzdálenosti $d_E(A, B) = 1$. V druhém kroku již hraje roli i to, že jsme zvolili metodu *nejbližšího* souseda, tedy $d_E(S_{AB}, C) = d_E(B, C) = 2$, protože $d_E(A, C) = 3$ (při volbě metody *nejvzdálenějšího* souseda by $d_E(S_{AB}, C) = d_E(A, C)$). Taky $d_E(S_{AB}, D) = d_E(B, D) = 3{,}5$. Jelikož $d_E(C, D) = 1{,}5$ je nejnižší, v druhém kroce vznikne shluk S_{CD} z bodů C a D na vzdálenosti 1,5. V posledním kroce vznikne shluk $S_{(AB)(CD)}$ na vzdálenosti $d_E(B, C) = 2$. Odpovídající dendrogram lze vidět na **Chyba! Nenalezen zdroj odkazů.**.



Obrázek 19: : Dendrogram vzniklý při shlukování v úloze 5.

6. Ke shluku 1, protože $d_H(c_1, \boldsymbol{x}) = 3 + 0 = 3$, kdežto $d_H(c_2, \boldsymbol{x}) = 2 + 2 = 4$. V případě euklei-dovské vzdálenosti by to však bylo ke shluku 2.

4 STROJOVÉ UČENÍ

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola seznámí čtenáře s přístupy k automatizaci procesu učení. Čtenář se seznámí s možnými reprezentacemi dat použitelných pro automatizované učení. Dále se čtenář seznámí s formální reprezentací znalostí a přístupem k učení (optimalizaci znalostí) na základě dostupných dat.

CÍLE KAPITOLY

Cílem této kapitoly je seznámit se s automatizovaným učením z několika pohledů. Podíváme se na různé metody učení z hlediska úsilí, které je třeba vynaložit na získání nových znalostí. Čtenář se seznámí s klíčovou otázku strojového učení, což je, jakou informaci má systém k dispozici o tom, že se učí správně. Čtenář také získá vhled do rozlišení metod získávání znalostí podle způsobu reprezentace příkladů použitých v procesu učení, způsobu zpracování příkladů nebo formy učení. Čtenář získá přehled o základních pojmech spojených s reprezentací dat ve strojové učení, což jsou atribut (numerický vs kategoriální, vstupní vs cílový), trénovací a testovací data, a dále pak význam chybové funkce.

KLÍČOVÁ SLOVA KAPITOLY

učení, typy učení, reprezentace dat, reprezentace znalostí, atribut, trénovací data, testovací data

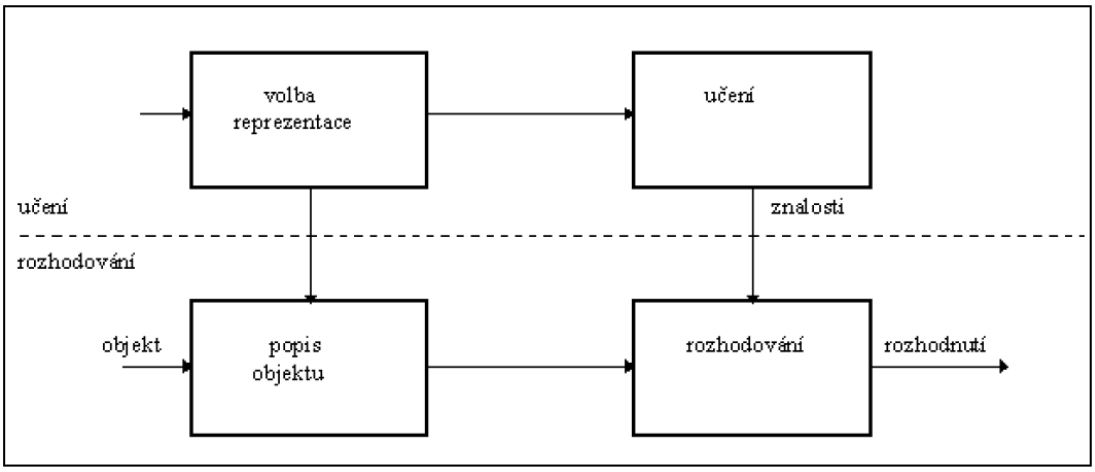
Důležitou vlastností živých organismů je schopnost přizpůsobovat se měnícím se podmínkám (adaptovat se), eventuálně se učit na základě vlastních zkušeností. Schopnost učit se bývá někdy dokonce považována za definici inteligence. Je proto přirozené, že vybavit touto vlastností i systémy technické je 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 sys-

témy, 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).



Obrázek 20: 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). Reprezentace příkladů pomocí relací překračuje rámec tohoto textu, takže se ji dále nebudeme věnovat. Zvědavý čtenář však další informace může najít např. kapitole 4 knihy [Berka, 2003].

- **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 souvislosti 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).

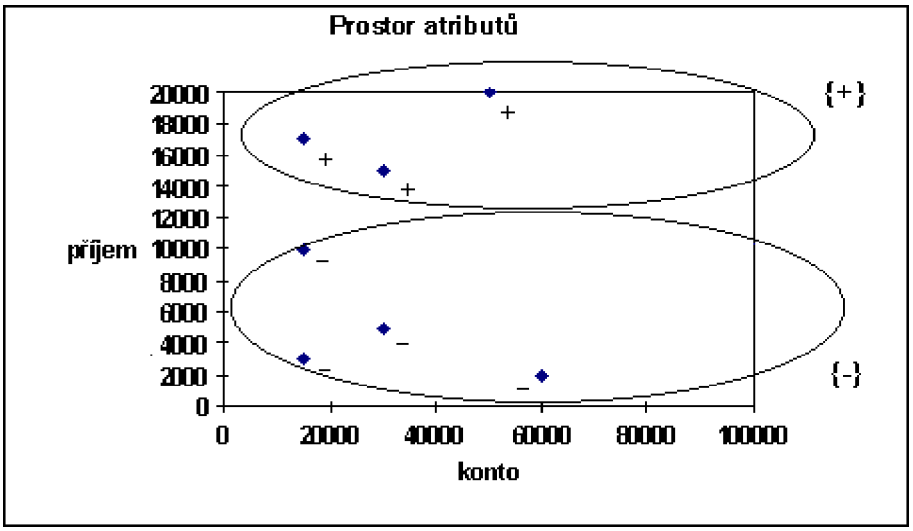
Empirické metody učení vycházejí z předpokladu, že jednotlivé objekty (příklady, pozorování) lze popsat pomocí charakteristik takových, že **objekty patřící k témuž konceptu mají podobné charakteristiky** (tyto metody bývají někdy nazývány *učení na základě podobnosti* - *similarity-based learning*). Pokud jsou objekty popsány hodnotami atributů, lze je reprezentovat jako body v mnohorozměrném prostoru, jehož dimenze je dána počtem těchto atributů[18]. 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

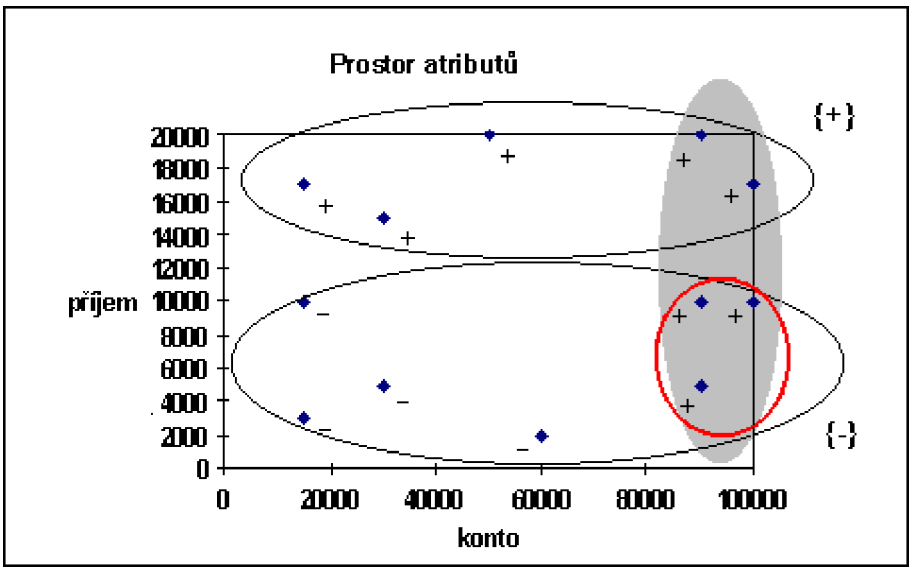
[18] Atributům se někdy říká příznaky (features), odtud anglický název tohoto prostoru – feature space.

na obrázcích Obrázek 21 a Obrázek 22. 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 Obrázek 21. Další příklady spolehlivých klientů nás ale přesvědčí o našem omylu (příklady v šedém oválu na Obrázek 22).

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*.



Obrázek 21: Málo dat



Obrázek 22: 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.

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

V tuto chvíli nebudeme rozlišovat, zda se jedná o atributy *kategoriální* (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*.

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

Ostatním, necílovým atributům *Aj* 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

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

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.[19] Budeme data tedy 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 $\boldsymbol{x}$ nějakého objektu $\boldsymbol{o}$ 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

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

[19] 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.

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

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

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

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

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 data může vést k „přeučení systému“ (*overfitting*); získané znalosti pak nereflektují obecnější zákonitosti, ale pouze kopírují strukturu použitých příkladů (viz Obrázek 21 a Obrázek 22). 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 nebyly použity v průběhu učení.

KONTROLNÍ OTÁZKA

1. Které dva typy učení lze rozlišovat?
2. Jaké druhy učení lze rozlišovat podle toho, jakou informaci má systém k dispozici o tom, že se učí správně?
3. Co je to atribut a jak souvisí s datovou tabulkou?

4. Jaký je rozdíl mezi trénovacími a testovacími daty?

5. Co je to chyba klasifikace?

SHRNUTÍ KAPITOLY

V této kapitole jsme se seznámili s přístupy k automatizaci procesu učení, možnými reprezentacemi dat použitelných pro automatizované učení, formální reprezentací znalostí a přístupem k učení (optimalizaci znalostí) na základě dostupných dat.

ODPOVĚDI

1. a) Učení se znalostem (knowledge acquisition) a b) 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).

2. Učení s učitelem, učení posilováním (odměna za správné chování a trest za chování nesprávné), učení pomocí nepřímých náznaků a učení bez učitele.

3. Atribut popisuje vlastnosti objektů reprezentovaných řádky v datové tabulce (jeden sloupec = jeden atribut objektu/řádku).

4. Trénovací data se používají pro učení systému, kdežto testovací pro hodnocení správnosti naučení systému. Toto rozdělení dat simuluje reálnou situaci, kdy systém učíme na dostupných datech a poté systém používáme pro data nová.

5. Rozdíl mezi hodnotou předpovídanou systémem a skutečnou hodnotou.

5 METODY DOLOVÁNÍ DAT

RYCHLÝ NÁHLED KAPITOLY

Tato kapitola se věnuje vybraným metodám dolování dat, které patří mezi nejčastěji používané metody dolování dat a jejichž implementace lze najít ve většině matematických a statistických softwarových systémech. Jsou to metody pro tvorbu rozhodovacích stromů, rozhodovacích pravidel, metody založené na umělých neuronových sítích, metoda SVM, dále se zde čtenář seznámí s takzvanými evolučními algoritmy, metodou klasifikace založenou na Bayesově pravidle a nakonec se zde čtenář seznámí s metodami založenými na analogii.

CÍLE KAPITOLY

Cílem kapitoly je seznámit se s pozadím vybraných metod pro dolování dat, jelikož pouze s dobrým vhledem do způsobu práce metod lze tyto metody efektivně používat i v praxi.

KLÍČOVÁ SLOVA KAPITOLY

reprezentace znalostí, rozhodovací strom, pravidlo, neuronová síť, genetický algoritmus, Naivní Bayes, metoda nejbližšího souseda

5.1 Rozhodovací stromy

5.1.1 ZÁKLADNÍ ALGORITMUS

Způsob reprezentování znalostí v podobě rozhodovacích stromů je dobře znám z řady oblastí. Vzpomeňme jen nejrůznějších „klíčů k určování“ různých živočichů nebo rostlin známých z biologie. Indukce rozhodovacích stromů patří k nejznámějším algoritmům z oblasti symbolických metod strojového učení. Při tvorbě rozhodovacího stromu se postupuje metodou „*rozděl a panuj*“ (divide and concquer). Trénovací data se postupně rozdělují na menší a menší podmnožiny (uzly stromu) tak, aby v těchto podmnožinách převládaly příklady jedné třídy. Na počátku tvoří celá trénovací data jednu množinu, na konci máme podmnožiny tvořené příklady téže třídy [Quinlan, 1979]. Tento postup bývá často nazýván „top down induction of decision trees“

(TDIDT). Postupuje se tedy metodou specializace v prostoru hypotéz (stromů) shora dolů, počínaje stromem s jedním uzlem (kořenem). Cílem je nalézt nějaký strom konzistentní s trénovacími daty, přitom se dává přednost menším, jednodušším stromům. Obecné schéma algoritmu pro tvorbu rozhodovacích stromů je na Obrázek 23.

TDIDT algoritmus

1. zvol jeden atribut jako kořen dílčího stromu,
2. rozděl data v tomto uzlu na podmnožiny podle hodnot zvoleného atributu a přidej uzel pro každou podmnožinu,
3. existuje-li uzel, pro který nepatří všechna data do téže třídy, pro tento uzel opakuj postup od bodu 1, jinak skonči.

Obrázek 23: Obecný algoritmus pro tvorbu rozhodovacích stromů

Zaměřme se nyní na klíčovou otázku celého algoritmu; jak vybrat vhodný atribut pro větvení stromu (bod 1). Cíl je zřejmý: vybrat takový atribut, který od sebe nejlépe odliší příklady různých tříd. Vodítkem pro volbu jsou charakteristiky atributu převzaté z teorie informace nebo pravděpodobnosti: entropie, informační zisk, poměrný informační zisk, χ^2, nebo Gini index.[20]

Entropie je pojem používaný v přírodních vědách (např. fyzika) pro vyjádření míry neuspořádanosti nějakého systému.

DEFINICE

V teorii informace je entropie definovaná jako funkce

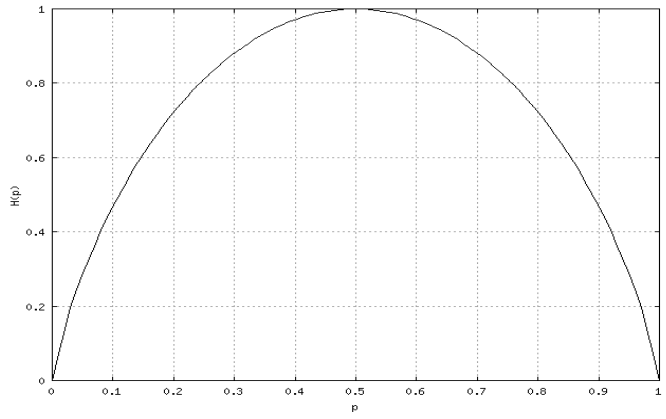
$$H = -\sum_{t=1}^{T}(p_t * log_2 p_t)$$

kde p_t je pravděpodobnost výskytu třídy t (v našem případě relativní četnost třídy t počítaná na určité množině příkladů) a T je počet tříd.

Podobu funkce H v případě dvou tříd ukazuje Obrázek 24. Graf znázorňuje průběh entropie v závislosti na pravděpodobnosti p jedné ze tříd. Je-li $p=1$ (všechny příklady patří do této třídy)

[20] Jsou i jiné možnosti. Např. Mantaras [Mantaras, 1991] používá pro výběr atributu vzdálenost mezi atributem a třídou.

nebo *p=0* (žádný příklad nepatří do této třídy), je entropie nulová. Jsou-li obě třídy zastoupeny stejným počtem příkladů (*p=0.5*), je entropie maximální.



Obrázek 24: Entropie

Výpočet entropie pro jeden atribut se provádí následujícím způsobem ($n_t(A(v))$ označuje počet příkladů (řádků v datech) ze třídy *t* s hodnotou *v* u atributu *A*, kdežto $n(A(v))$ označuje počet příkladů (ze všech tříd) s hodnotou *v* u atributu *A*):

1. Pro každou hodnotu *v*, kterou může nabýt uvažovaný atribut *A* spočítej podle výše uvedeného vzorce entropii *H(A(v))* na skupině příkladů, které jsou pokryty kategorií *A(v)*

$$\mathrm{H(A(v))} = -\sum_{t=1}^{T}\left(\frac{n_t(A(v))}{n(A(v))} * log_2 \frac{n_t(A(v))}{n(A(v))}\right)$$

2. Spočítej střední entropii *H(A)* jako vážený součet entropií *H(A(v))*, přičemž váhy v součtu jsou relativní četnosti kategorií *A(v)* v datech D_{TR}

$$\mathrm{H(A)} = \sum_{v \in Val(A)} \frac{n(A(v))}{n} H(A(v))$$

Pro větvení stromu se pak vybere atribut s nejmenší entropií *H(A)*.

Informační zisk (information gain) i *poměrný informační zisk* (information gain ratio) jsou míry odvozené z entropie. Informační zisk se spočítá jako rozdíl entropie pro celá data (pro cílový atribut) a pro uvažovaný atribut. Informační zisk tak měří redukci entropie způsobenou volbou atributu *A* (n_t označuje počet příkladů ze třídy *t*):

$$\mathrm{Zisk(A)} = \mathrm{H(C)} - \mathrm{H(A)}$$

kde

$$\mathrm{H(C)} = -\sum_{t=1}^{T} \frac{n_t}{n} * log_2 \frac{n_t}{n}$$

Zatímco v případě entropie jsme hledali atribut s minimální hodnotou, v případě informačního zisku hledáme atribut s maximální hodnotou. To je logické, uvážíme-li, že entropie počítané pro celá data nezávisí na atributu. První člen rozdílu je tedy konstantní, takže rozdíl bude maximální v případě, že druhý člen rozdílu bude minimální.

Uvedená dvě kritéria mají jednu nevýhodu; neberou do úvahy počet hodnot zvoleného atributu. Důležité je pouze to, jak dobře tento atribut od sebe odliší příklady různých tříd. Pokud bychom jako atribut pro větvení vybrali např. pořadové číslo příkladu, dosáhneme nejnižší (nulovou) entropii, protože jedné hodnotě atributu odpovídá jediný objekt. Tento atribut by nám tedy umožnil bezchybně klasifikovat trénovací data (tak, že bychom si „pamatovali“, který objekt patří do které třídy), byl by ale zcela nepoužitelný pro klasifikaci nových příkladů. Proto se někdy používá jako kritérium pro volbu atributu *poměrný informační zisk* (information gain ratio), který kromě entropie bere do úvahy i počet hodnot atributu.

$$\text{Poměrný zisk(A)} = \frac{Zisk(A)}{Větvení(A)}$$

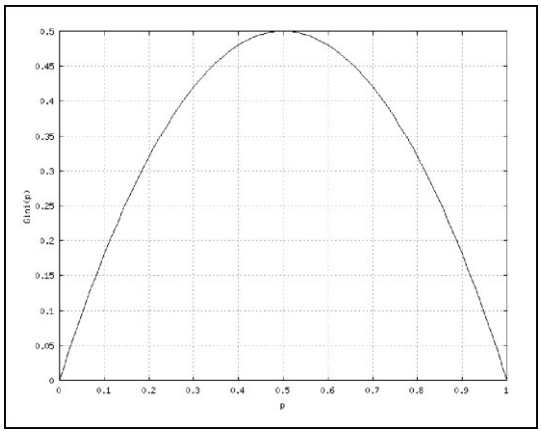
Jmenovatel ve výše uvedeném vztahu je vlastně entropie dat vzhledem k hodnotám atributu A[21]:

$$\text{Větvení(A)} = -\sum_{v \in Val(A)} \frac{n_t(A(v))}{n} * log_2 \frac{n_t(A(v))}{n}$$

Stejnou roli, jako hrála v předcházejících úvahách entropie, může hrát i tzv. *Gini index*. Tento index se spočítá jako

$$\text{Gini} = 1 - \sum_{t=1}^{T} {p_t}^2$$

kde p_t je opět relativní počet příkladů t-té třídy zjišťovaný na nějaké (pod)množině. Graf závislosti Gini indexu na pravděpodobnosti jedné ze dvou tříd ukazuje Obrázek 25. Opět je hodnota indexu minimální v případě, že příklady patří do jedné ze tříd, a maximální v případě, že příklady jsou rovnoměrně rozděleny mezi obě třídy.



Obrázek 25: Gini index

[21] V předcházejících úvahách jsme pracovali s entropií dat vzhledem k cílovým třídám. Zde je tedy entropie použita v trochu jiném smyslu.

Hodnotu Gini indexu pro jeden atribut spočítáme analogicky jako hodnotu entropie pro jeden atribut.

Na závěr této kapitoly zmiňme, že uvedený algoritmus TDIDT bude fungovat pro kategoriální data (počet podmnožin-uzlů vytvářený v kroku 2 odpovídá počtu hodnot daného atributu), která nejsou zatížena šumem (růst stromu se podle bodu 3 zastaví v okamžiku, kdy všechny příklady v daném uzlu patří do téže třídy). Detaily o tom, jak se dají tato dvě omezení překonat, se může zvídavý čtenář dočíst v kapitole 5.1 knihy [Berka, 2003].

5.2 Asociační pravidla

IF-THEN konstrukce nalezneme ve všech programovacích jazycích, používají se i v běžné mluvě (nebude-li pršet, nezmoknem). Není tedy divu, že pravidla s touto syntaxí patří společně s rozhodovacími stromy k nejčastěji používaným prostředkům pro reprezentaci znalostí, ať už získaných od expertů, nebo vytvořených automatizovaně z dat.

Termín asociační pravidla široce zpopularizoval počátkem 90. let Agrawal [Agrawal a kol, 1993] v souvislosti s analýzou nákupního košíku. Při této analýze se zjišťuje, jaké druhy zboží si současně kupují zákazníci v supermarketech (např. pivo a párek). Jde tedy o hledání vzájemných vazeb (asociací) mezi různými položkami sortimentu prodejny. Přitom není upřednostňován žádný speciální druh zboží jako závěr pravidla. V tomto smyslu budeme chápat pravidla v této kapitole.

5.2.1 Základní charakteristiky pravidel

U pravidel vytvořených z dat nás obvykle zajímá, kolik příkladů splňuje předpoklad a kolik závěr pravidla, kolik příkladů splňuje předpoklad i závěr současně, kolik příkladů splňuje předpoklad a nesplňuje závěr.... Tedy, zajímá nás, jak pro pravidlo

$$Ant \Rightarrow Suc,$$

kde *Ant* (předpoklad, levá strana pravidla, antecedent) i *Suc* (závěr, pravá strana pravidla, sukcedent) jsou kombinace kategorií[22] vypadá příslušná kontingenční tabulka. Pro n příkladů je její podoba uvedena v Tabulka 7,

[22] Poznamenejme, že numerické atributy se **vždy** musí diskretizovat před použitím algoritmu pro hledání asociačních pravidel.

Tabulka 7: Kontingenční (čtyřpolní) tabulka

	Suc	¬*Suc*	∑
Ant	*a*	*b*	*r*
¬*Ant*	*c*	*d*	*s*
∑	*k*	*l*	*n*

kde:

$n(Ant \wedge Suc) = a$ je počet příkladů pokrytých současně předpokladem i závěrem,

$n(Ant \wedge \neg Suc) = b$ je počet příkladů pokrytých předpokladem a nepokrytých závěrem,

$n(\neg Ant \wedge Suc) = c$ je počet příkladů nepokrytých předpokladem ale pokrytých závěrem,

$n(\neg Ant \wedge \neg Suc)$= d je počet příkladů nepokrytých ani předpokladem ani závěrem.

Dále platí, že:

$$n(Ant) = a+b = r,\ n(\neg Ant) = c+d = s,\ n(Suc) = a+c = k,\ n(\neg Suc) = b+d = l,\ n = a+b+c+d$$

Z těchto čísel (místo o počtu objektů pokrytých kombinací se někdy mluví o *četnosti* resp. *frekvenci* kombinace) můžeme počítat různé charakteristiky pravidel a kvantitativně tak hodnotit nalezené znalosti.

Základními charakteristikami asociačních pravidel v Agrawalově pojetí jsou *podpora* (*support*) a *spolehlivost* (*confidence*). Podpora je (absolutní, resp. relativní[23]) počet objektů, splňujících předpoklad i závěr, tedy hodnota

$$a, \text{ resp. } P(Ant \wedge Suc) = \frac{a}{a+b+c+d}.$$

Spolehlivost je vlastně podmíněná pravděpodobnost závěru pokud platí předpoklad, tedy

$$P(Suc|Ant) = \frac{P(Suc \wedge Ant)}{P(Ant)} = \frac{a}{a+b}$$

[23] Relativní četnosti kombinace *K* budeme chápat jako odhad pravděpodobnosti jejího výskytu v datech P(*K*).

5.2.2 GENEROVÁNÍ KOMBINACÍ

Základem všech algoritmů pro hledání asociačních pravidel je generování kombinací (konjunkcí) hodnot atributů. Při generování vlastně procházíme (prohledáváme) prostor všech přípustných konjunkcí, tedy konjunkcí, kde se nesmí opakovat atributy. Metod je několik:

- Do šířky
- Do hloubky
- Heuristicky

Jednotlivé metody budeme ilustrovat na příkladě vytváření konjunkcí z dat uvedených v Tab. 2. Pro zjednodušení zápisu kategorií v Obrázek 26 budeme kategorii zapisovat pořadovým číslem atributu a prvním písmenem hodnoty, tedy zápis 1v bude znamenat kategorii *příjem*(*vysoký*).

Tabulka 8: Data pro generování kombinací

příjem	konto	pohlaví	nezaměstnaný	úvěr
vysoký	vysoké	žena	ne	ano
vysoký	vysoké	muž	ne	ano
nízký	nízké	muž	ne	ne
nízký	vysoké	žena	ano	ano
nízký	vysoké	muž	ano	ano
nízký	nízké	žena	ano	ne
vysoký	nízké	muž	ne	ano
vysoký	nízké	žena	ano	ano
nízký	střední	muž	ano	ne
vysoký	střední	žena	ne	ano
nízký	střední	žena	ano	ne
nízký	střední	muž	ne	ano

Při generování *do šířky* se kombinace generují tak, že se nejprve vygenerují všechny kombinace délky jedna, pak všechny kombinace délky dvě, atd. Jde tedy o generování kombinací podle délek, kategorie jednoho atributu jsou přitom uspořádány podle abecedy (Obrázek 26 vlevo).

kombinace
1n
1v
2n
2s
2v
3m
3z
4a
4n
5a
5n
1n 2n
1n 2s
1n 2v
1n 3m
1n 3z
1n 4a
1n 4n
1n 5a
1n 5n
1v 2n
1v 2s
1v 2v
1v 3m
1v 3z
1v 2v 3z 4n 5a

kombinace
1n
1n 2n
1n 2n 3m
1n 2n 3m 4a
1n 2n 3m 4a 5a
1n 2n 3m 4a 5n
1n 2n 3m 4n
1n 2n 3m 4n 5a
1n 2n 3m 4n 5n
1n 2n 3m 5a
1n 2n 3m 5n
1n 2n 3z
1n 2n 3z 4a
1n 2n 3z 4a 5a
1n 2n 3z 4a 5n
1n 2n 3z 4n
1n 2n 3z 4n 5a
1n 2n 3z 4n 5n
1n 2n 3z 5a
1n 2n 3z 5n
1n 2n 4a
1n 2n 4a 5a
1n 2n 4a 5n
1n 2n 4n
5n
5n

Fq	kombinace
8	5a
7	1n
6	3m
6	3z
6	4a
6	4n
5	1v
5	1n 4a
5	4n 5a
5	1v 5a
4	2v
4	2s
4	2n
4	5n
4	3m 5a
4	1n 3m
4	3z 5a
4	3z 4a
4	3m 4n
4	1v 4n
4	2v 5a
4	1n 5n
4	1v 4n 5a
3	1n 5a
3	1n 3z
1	1v 2s 3z 4n 5a

Obrázek 26: Generování do šířky (vlevo), do hloubky (uprostřed) a podle četností (vpravo)

Při generování *do hloubky* se vyjde od první kombinace délky jedna a ta se pak prodlužuje (vždy o první kategorii dalšího atributu) dokud to lze[24]. Nelze-li kombinaci prodloužit, změní se kategorie „posledního“ atributu. Nelze-li provést ani to (vyčerpaly se kategorie posledního atributu), kombinace se zkrátí a současně se změní poslední kategorie. Princip generování je ilustrován na Obrázek 26 uprostřed. Kategorie jednoho atributu jsou opět uspořádány podle abecedy, kurzívou jsou vyznačeny kombinace s nulovou četností.

Oba zmíněné způsoby jsou „slepé". Provádějí se pouze na základě seznamu hodnot atributů a neberou do úvahy vlastní data. Proto můžeme vygenerovat kombinace, které se nevyskytují v datech. V seznamu kombinací na Obrázek 26 uprostřed je příklad takové kombinace zvýrazněn kursivou (šestý shora). Poslední zde uváděný způsob generování podle četností vytváří kombinace v pořadí podle jejich výskytu v datech. Jedná se o příklad heuristického prohledávání prostoru kombinací, kde heuristikou je „uvažuj kombinaci s nejvyšší četností". Při tomto způsobu generování se kombinace s nulovou četnosti objeví až na konci seznamu (Obrázek 26 vpravo).

5.2.3 ALGORITMUS APRIORI

Nejznámějším algoritmem pro hledání asociačních pravidel je *algoritmus apriori*. Tento algoritmus navrhl R. Agrawal v souvislosti s analýzou nákupního košíku [Agrawal a kol., 1996].

[24] Prodlužování skončí při dosažení maximální zadané délky konjunkce nebo při vyčerpání atributů a jejich hodnot.

Jádrem algoritmu je hledání často se opakujících množin položek (frequent itemsets). Jedná se kombinace (konjunkce) kategorií[25], které dosahují předem zadané četnosti (podpory *minsup*) v datech.

Při hledání kombinací délky *k*, které mají vysokou četnost, se využívá toho, že **již známe** kombinace délky *k*-1. Při vytváření kombinace délky *k* spojujeme kombinace délky *k*-1 . Jde tedy o generování kombinací „do šířky“. Přitom pro vytvoření jedné kombinace délky *k* požadujeme, aby všechny její podkombinace délky *k*-1 splňovaly požadavek na četnost. Tedy např. ze tříčlenných kombinací $\{A_1A_2A_3, A_1A_2A_4, A_1A_3A_4, A_1A_3A5, A_2A_3A_4\}$ dosahujících požadované četnosti vytvoříme pouze jedinou čtyřčlennou kombinaci $A_1A_2A_3A_4$. Kombinaci $A_1A_3A_4A_5$ sice lze vytvořit spojením $A_1A_3A_4$ a $A_1A_3A_5$, ale mezi tříčlennými kombinacemi chybí $A_1A_4A_5$ i $A_3A_4A_5$. Příslušný algoritmus je uveden na Obrázek 27.

Algoritmus apriori

1. do L_1 přiřaď všechny hodnoty atributů, které dosahují alespoň požadované četnosti
2. polož k=2
3. dokud L_{k-1} je neprázdná:
   a) pomocí funkce *apriori-gen* vygeneruj na základě L_{k-1} množinu kandidátů C_k
   b) do L_k zařaď ty kombinace z C_k, které dosáhly alespoň požadovanou četnost
   c) proveď $k := k + 1$

Funkce apriori-gen(L_{k-1})

1) pro všechny dvojice kombinací *p*, *q* z L_{k-1}:
   Pokud *p* a *q* se shodují v k-2 kategoriích přidej sjednocení $p \wedge q$ do C_k
2) pro každou kombinaci *c* z C_k:
   Pokud některá z jejich podkombinací délky k-1 není obsažena v L_{k-1} odstraň *c* z C_k

Obrázek 27: Algoritmus apriori

Po nalezení kombinací, které vyhovují svou četností, se vytvářejí asociační pravidla. Každá kombinace *Comb* se rozdělí na všechny možné dvojice podkombinací *Ant* a *Suc* takové, že *Suc*

[25] V případě analýzy nákupního košíku jsou kategorie typu *máslo(ano), chleba(ano)* apod., které můžeme stručněji zapisovat *máslo, chleba* a chápat jako položky zboží.

= *Comb* - *Ant*. Tedy *Ant* a *Suc* neobsahují stejnou kategorii *(Ant* ∩ *Suc =0)* a zároveň *Ant* ∧ *Suc* = *Comb.* Uvažované pravidlo *Ant* ⇒ *Suc* má pak podporu, která se rovná četnosti kombinace *Comb*. Spolehlivost pravidla se spočítá jako podíl četností kombinací *Comb* a *Ant*. Četnost kombinace *Ant* přitom již známe! Totiž vzhledem k tomu, že délka *Ant* je menší než délka *Comb*, byla kombinace *Ant* algoritmem apriori vygenerována dříve. Navíc víme, že četnost kombinace *Ant* je větší nebo rovna četnosti kombinace *Comb*[26].

Při vytváření pravidel se využívá skutečnosti, že četnost nadkombinace (kombinace s přidaným atributem nebo více atributy pomocí konjunkce) je nejvýše rovna četnosti kombinace:

$$n(Comb_1) \geq n(Comb_2) \text{ pro } Comb_1 \succ Comb_2,$$

tedy je-li $Comb_2$ nadkombinací $Comb_1$, potom počet příkladů splňujících $Comb_2$ je maximálně roven počtu příkladů splňujících $Comb_1$.

Z toho plyne zásadní idea pro zjednodušení tvorby pravidel, což je, že je-li *Ant* ≻ *Ant*', je spolehlivost pravidla *Ant*' ⇒ *Comb* - *Ant*' větší nebo rovna spolehlivosti pravidla *Ant* ⇒ *Comb* - *Ant*[27]. Tudíž nevyhovuje-li zadané minimální spolehlivosti pravidlo *Ant*' ⇒ *Comb* – *Ant*', **nevyhoví** ani žádné pravidlo *Ant* ⇒ *Comb* - *Ant*, kde *Ant* ≻ *Ant*'. Podobně, aby mohlo zadanou minimální spolehlivost splnit pravidlo *Comb* – *Suc*' ⇒ *Suc*', musí ji splnit všechna pravidla *Comb* – *Suc* ⇒ *Suc*, kde opět *Suc* ≻ *Suc*'. Např. vyhovuje-li pro kombinaci *Comb*=$A_1A_2A_3A_4$ pravidlo $A_1A_2 \Rightarrow A_3A_4$, budou vyhovovat i pravidla $A_1A_2A3 \Rightarrow A_4$ a $A_1A_2A_4 \Rightarrow A_3$.

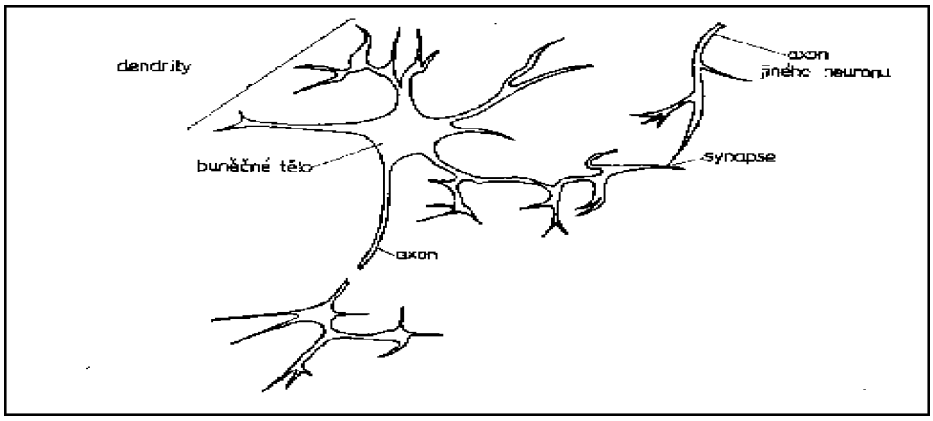
Kombinaci *Comb* začínáme tedy rozdělovat tak, že *Suc* je nejprve tvořeno jednou položkou (kombinací délky 1). U těch pravidel, která dosáhla požadovanou spolehlivost se do *Suc* přidá další položka z *Ant* atd.

5.3 Neuronové sítě

Lidský mozek je složen asi z 10^{10} nervových buněk (neuronů) které jsou mezi sebou navzájem propojeny ještě řádově vyšším počtem vazeb [Novák a kol.,1992]. Začněme tedy nejdříve jedním neuronem. Na Obrázek 28 je ukázka typického neuronu, který je tvořen tělem (soma) ze kterého vybíhá řada (desítky až stovky) kratších výběžků (dendrity) a jeden dlouhý výběžek (axon). Zatímco tělo neuronu je veliké řádově mikrometry a dendrity několik milimetrů, axon může dosahovat délky desítek centimetrů až jednoho metru. Axon každého neuronu je zakončen tzv. synapsí (typicky jedinou, ale může jich být i několik), která dosedá na jiný neuron. Přes synapse se přenášejí vzruchy mezi neurony; v důsledku chemických reakcí se v místě, kde synapse dosedá na neuron, mění propustnost buněčné membrány neuronu, tím se lokálně mění koncentrace kladných i záporných iontů vně a uvnitř buňky a tedy i membránový potenciál. Některé synaptické

[26] Kratší (méně omezující kombinaci) odpovídá alespoň tolik příkladů v datech jako kombinaci delší.
[27] Je-li *Ant'* nadkombinací kombinace *Ant,* je *Ant'* přísnější a tudíž ho splní méně příkladů. Vzhledem k tomu, že četnost *Ant'* je ve jmenovateli zlomku, přičemž čitatel zůstává stejný, je spolehlivost pravidla *Ant'* ⇒ *Comb — Ant'* alespoň taková, jako spolehlivost pravidla *Ant* ⇒ *Comb —Ant.*

vazby mají charakter excitační (zvyšují membránový potenciál), jiné mají charakter inhibiční (snižují membránový potenciál). Dílčí účinky synaptických vazeb se na neuronu kumulují a ve chvíli, kdy celkový membránový potenciál přesáhne určitý práh, neuron je aktivován a přes svoji synapsi začne působit na další neurony, se kterými je spojen.



Obrázek 28: Biologický neuron

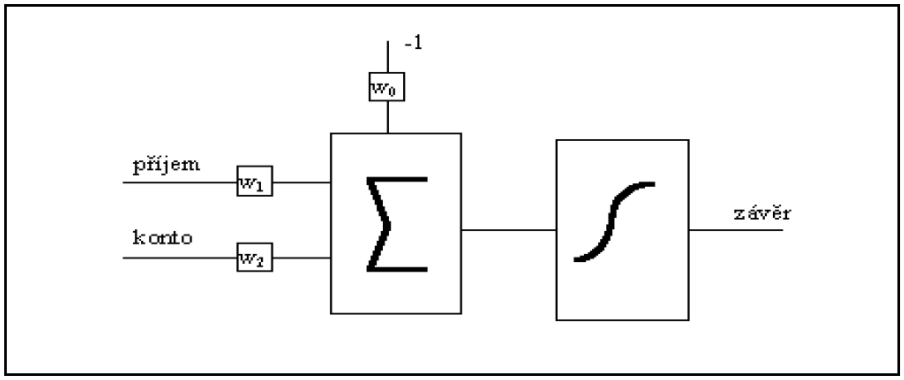
5.3.1 MODEL JEDNOHO NEURONU

Neuron tedy, zjednodušeně řečeno, přijímá kladné a záporné podněty od jiných neuronů a ve chvíli, kdy souhrn těchto podnětů překročí daný práh, sám se aktivuje. Výstupní hodnota neuronu je obvykle nějakou nelineární transformací souhrnu podnětů (viz Obr. 2). Z tohoto pohledu vycházejí matematické modely neuronu. Prvním byl tzv. „logický neuron“ McCullocha a Pittse z r. 1943, který pracoval se vstupními a výstupními hodnotami pouze 0 a 1. Dalším známým modelem byl Widrowův „adaptivní lineární neuron“ *Adaline* z roku 1960, který si popíšeme trochu podrobněji. Vstupem do *Adaline* jsou podněty (numerické hodnoty) označené $x_1,x_2,...,x_m$. Každý podnět x_i je násoben vahou w_i; tento součin vstupuje do součtového členu, kde je vytvořen vážený součet[28]

$$\text{SUM} = \mathbf{w} \cdot \mathbf{x} = \sum_{i=1}^{m} w_i x_i.$$

V případě, že tento vážený součet přesáhne práh w_0, je vystup $\hat{y}$ z *Adaline* roven 1, v opačném případě je výstup z *Adaline* roven 0 (někdy místo 0 může být hodnota -1). Na rozdíl od logického neuronu jsou tedy vstupy libovolná čísla, výstupy zůstávají dvouhodnotové:

[28] Zápis $\mathbf{w} \cdot \mathbf{x}$ vyjadřuje skalární součin.



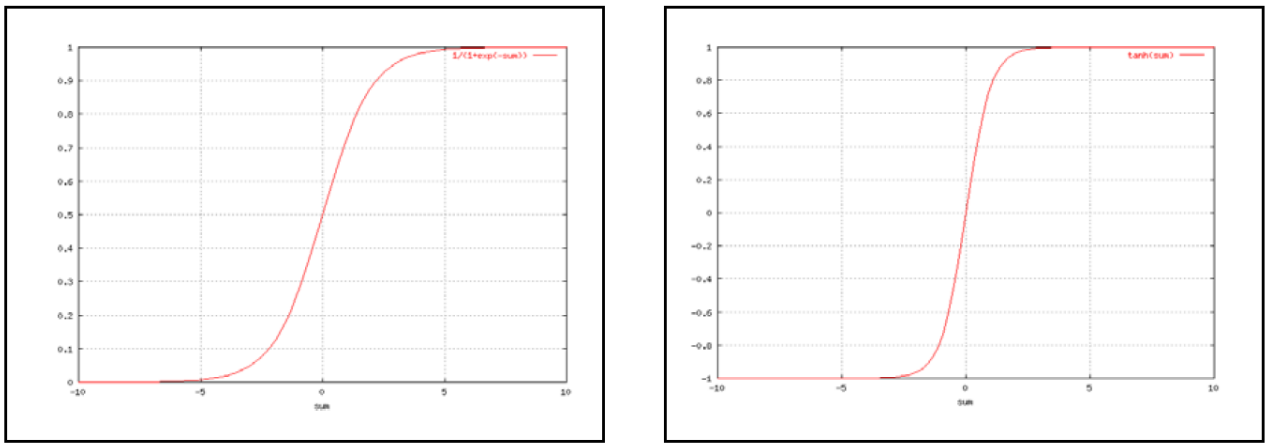
Obrázek 29: Schéma neuronu

$$\hat{y} = 1 \text{ pro } \sum_{i=1}^{m} w_i x_i \geq w_0$$

$$\hat{y} = 0 \text{ pro } \sum_{i=1}^{m} w_i x_i < w_0$$

Nelinearita v podobě skokové funkce, tak jak je použita v případě *Adaline* (dávající pouze dvě výstupní hodnoty neuronu), je ve složitějších modelech neuronu nahrazena hladkými funkcemi, které nabývají hodnot z celého intervalu[29]. Nejpoužívanějšími přenosovými (aktivačními) funkcemi jsou:

- sigmoidální funkce f(SUM) = $\frac{1}{1+e^{-SUM}}$; v tomto případě výstup neuronu $\hat{y}$ nabývá hodnot z intervalu [0, 1] (Obrázek 29 vlevo),
- hyperbolický tangens f(SUM) = tanh(SUM) ; v tomto případě výstup neuronu $\hat{y}$ nabývá hodnot z intervalu [-1, 1] (Obrázek 29 vpravo).



Obrázek 30: (vlevo) Sigmoida a (vpravo) hyperbolický tangens.

[29] Požadavek na to, aby funkce byla hladká, je důležitý pro algoritmus nastavování vah v procesu učení. Hladká funkce je totiž diferencovatelná.

Někdy naopak nelineární funkce chybí; resp. f(SUM) = SUM. V tomto případě neuron realizuje pouze vážený součet vstupů (tento typ neuronů se někdy používá ve výstupní vrstvě neuronových sítí.).

Činnost *Adaline* má jednoduchou geometrickou interpretaci. Jednotlivé vstupní podněty $x_1,x_2,...,x_m$ mohou představovat hodnoty vstupních atributů nějakého objektu (např. teplota, výška, váha ...). Každý objekt lze pak reprezentovat jako bod $\boldsymbol{x} = x_1,x_2,...,x_m$ v m-rozměrném prostoru. Bod $\boldsymbol{x}$ leží v jedné ze dvou částí prostoru oddělených od sebe rozdělující nadrovinou (pro m = 2 se pohybujeme v rovině a rozdělující nadrovina je přímka). Body ležící v jedné části prostoru můžeme považovat za obrazy objektů patřících do téže třídy. *Adaline* lze tedy považovat za *lineární klasifikátor* objektů do *dvou tříd*.

PŘÍPADOVÁ STUDIE

Pro ilustraci opět použijeme náš bankovní příklad, a sice v podobě dvou numerických atributů *konto* a *příjem* (Tabulka 9). Adaptivní lineární neuron z Obrázek 29 bude mít pro tato data váhy např.:[30]

$$w_1 = 1$$

$$w_2 = 0.2$$

$$w_0 = 16000$$

Schopnost neuronu s takto nastavenými vahami klasifikovat naše data ukazuje Obrázek 31. Rozdělovací přímka je definována rovnicí

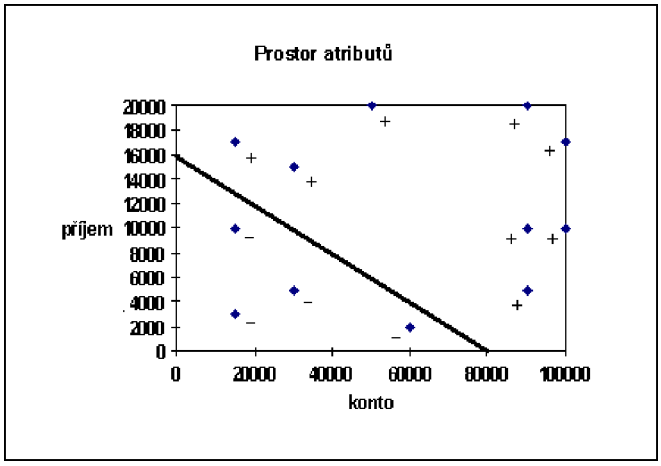
$$\text{příjem} + 0.2\ \text{konto} - 16000 = 0.$$

3

[30] Například zde znamená, že všechny váhy mohou být násobeny libovolným (stejným) číslem.

Tabulka 9: Numerická data

Klient	příjem	konto	úvěr
K101	3000	15000	ne
K102	10000	15000	ne
K103	17000	15000	ano
K104	5000	30000	ne
K105	15000	30000	ano
K106	20000	50000	ano
K107	2000	60000	ne
K108	5000	90000	ano
K109	10000	90000	ano
K110	20000	90000	ano
K111	10000	100000	ano
K112	17000	100000	ano



Obrázek 31: Adaline jako lineární klasifikátor do dvou tříd.

Často se provádí normalizace vstupních hodnot (např. na interval [-1,1]). Tato normalizace snižuje rozdíly mezi váhami. Pokud pro naše data použijeme normalizované vstupy

$$\text{norm\_příjem} = \text{příjem}/10000 - 1$$

$$\text{norm\_konto} = \text{konto}/50000 - 1$$

Získáme tyto váhy neuronu[31]:

$$w_1 = 5.8029$$

$$w_2 = 4.3746$$

$$w_0 = -2.8733$$

[31] Pro výpočet byla použita implementace neuronových sítí ze systému Weka.

Důležitou vlastností neuronů je jejich schopnost učit se. Učením se zde myslí (algoritmus) nastavení vah $\boldsymbol{w}$ na základě předložených příkladů $[\boldsymbol{x_i}, y_i]$ tak, aby systém co nejsprávněji zpracovával (např. klasifikoval) i neznámé příklady $\boldsymbol{x_k}$.[32]

Mezi první způsoby učení patří *Hebbův zákon* z roku 1949. Byl formulován jako model učení na úrovni neuronů v mozku. Vychází z představy, že se posilují ty vazby, které u daného neuronu způsobují jeho aktivaci. V (umělých) neuronech lze toto pravidlo formulovat takto:

$$\mathbf{w}_{i+1} = \mathbf{w}_i + y_i\, \mathbf{x}_i$$

kde $[\mathbf{x}_i, y_i]$ je trénovací příklad ($\boldsymbol{x_i}$ je vektor hodnot vstupních atributů a y_i je informace o zařazení příkladu $\boldsymbol{x_i}$ do třídy), $\boldsymbol{w_i}$ je váha před modifikací a $\boldsymbol{w_{i+1}}$ je váha po modifikaci. Váha $\boldsymbol{w}$ může růst nade všechny meze, což neodpovídá biologické realitě.

Adaline samotná používala jiný způsob učení, tzv. *gradientní metodu*. Zde se vycházelo z požadavku, aby chování sítě bylo co nejvíce podobno celkovému chování učitele, který provádí klasifikaci vstupních příkladů trénovací množiny. Zavádí se tedy tzv. *střední kvadratická chyba*, která má pro n příkladů z trénovací množiny D_{TR} podobu

$$\mathrm{Err}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Tuto chybu se snažíme minimalizovat (y_i je hodnota cílového atributu a $\hat{y}_i$ je výsledek zařazení sítí). Z požadavku na minimum střední kvadratické chyby lze odvodit následující pravidlo pro modifikaci vah:

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\, \mathbf{w}\ ,$$

kde

$$\Delta\, \mathbf{w} = -\eta \frac{\partial \mathrm{Err}}{\partial \mathbf{w}} = \eta \sum_{i=1}^{n}(y_i - \hat{y}_i)\, \mathbf{x}_i$$

Modifikace vah se tedy provádí až po zpracování celé trénovací množiny.

Odvození vztahu pro modifikaci vah $\boldsymbol{w}$ pro případ, že neuron realizuje pouze vážený součet vstupů, tedy že

$$\hat{y}_i = \mathbf{w} \cdot \mathbf{x}_i$$

bylo již jednou uvedeno v podkapitole věnované učení jako aproximace funkcí. Připomeňme si, že

[32] Na počátku procesu učení jsou váhy **w** nastaveny náhodně (na nějaké hodnoty blízké 0).

$$\frac{\partial Err}{\partial \mathbf{w}} = \frac{1}{2}\sum_{i=1}^{n} \frac{\partial}{\partial \mathbf{w}} (y_i - \hat{y}_i)^2 = \frac{1}{2}\sum_{i=1}^{n} 2\,(y_i - \hat{y}_i)\frac{\partial}{\partial \mathbf{w}}(y_i - \hat{y}_i) =$$

$$= \sum_{i=1}^{n} (y_i - \hat{y}_i)\frac{\partial}{\partial \mathbf{w}}(y_i - \mathbf{w}\cdot\mathbf{x}_i) = \sum_{i=1}^{n} (y_i - \hat{y}_i)(-\mathbf{x}_i)$$

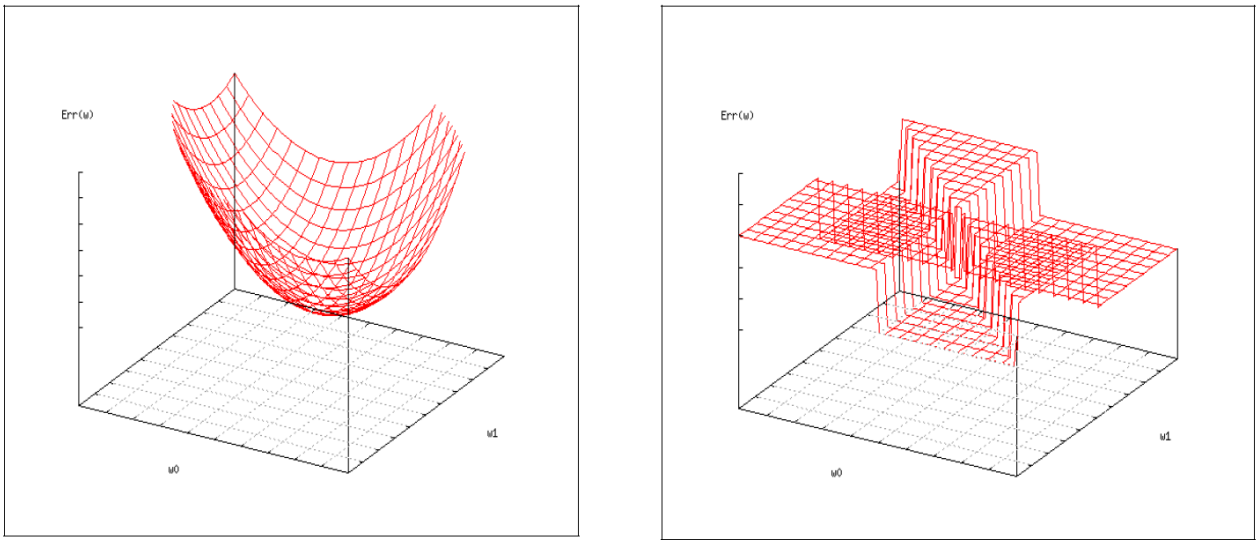
Chybová funkce *Err(w)* bude mít v tomto případě jedno globální minimum (Obrázek 32 vlevo). Uvedený postup výpočtu vah nalezne takové váhy ***w***, které budou odpovídat tomuto minimu. V případě lineárně separabilních tříd toto minimum odpovídá bezchybné klasifikaci příkladů, tedy nalezneme takové váhy ***w***, pro které

$$Err(\mathbf{w}) = 0.$$

Vzhledem k tomu, že *Adaline* používá skokovou aktivační funkci

$$\hat{y}_i = \text{sign}\,(\mathbf{w} \cdot \mathbf{x}_i),$$

je podoba chybové funkce poněkud složitější (Obrázek 32 vpravo). Přesto můžeme i pro Adaline použít výše odvozené pravidlo pro modifikaci vah. Nabývá-li totiž skutečný výstup *y* pouze hodnot ± 1, pak pro výstup neuronu $\hat{y} = \pm 1$ platí $\text{sign}(\mathbf{w}\,\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$.



Obrázek 32: (vlevo) Chybová funkce pro lineární aktivaci. (vpravo) Chybová funkce pro skokovou aktivaci.

Používanější varianta, tzv. stochastická (inkrementální) aproximace předpokládá, že se váhový vektor ***w*** modifikuje okamžitě poté, kdy při klasifikaci *i*-tého příkladu došlo k odchylce mezi požadovaným a skutečným výstupem systému

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta\mathbf{w}_i,$$
$$\Delta\mathbf{w}i = \eta\,(y_i - \hat{y}_i)\,\mathbf{x}_i\,.$$

6

Ze vzorce je vidět, že při učení chybí zpětná vazba. V případě odchylky se váhový vektor $\boldsymbol{w}$ modifikuje, aniž by se zjišťovalo, zda provedená změna měla vliv na výstup systému $\hat{y}_i$. Trénovací množinou D_{TR} se tedy prochází *opakovaně* tak dlouho, dokud není neuron „naučen“ (střední kvadratická chyba je minimální resp. dostatečně malá)[33]. V případě *Adaline* je možno dosáhnout *nulovou* chybu pouze pro *lineárně separabilní třídy*. Pouze v tomto případě *Adaline* bezchybně klasifikuje všechny prvky trénovací množiny. Pokud třídy nejsou lineárně separabilní, učení se ustálí v okamžiku nalezení minima (ne nutně globálního). Parametr η, nazývaný *learning rate*, udává „krok“, kterým se mění váhový vektor $\boldsymbol{w}$. Je-li η příliš velké, můžeme při přibližování k minimu toto minimum minout, je-li η příliš malé, iterování trvá příliš dlouho. Obvykle se volí $\eta \in [0, 10]$ (často η je 0.1 nebo 0.05, navíc s rostoucím počtem iterací se η může zmenšovat).

5.3.2 PERCEPTRON

Zatím jsme se zabývali jedním neuronem. Jak to vypadá s neuronovými sítěmi? První neuronová síť pochází z roku 1957. Rosenblattův *Perceptron* byl navržen jako model zrakové soustavy. *Perceptron* je hierarchický systém tvořený třemi úrovněmi (viz Obrázek 33). První z nich, nazývaná sítnice, slouží k přijímání informace z prostředí. Je tvořena receptory, prvky, jejichž výstup nabývá hodnoty 1 nebo 0 podle toho, zda jsou prostředím excitovány nebo ne. Výstupy receptorů jsou (přes náhodně zvolené vazby) přivedeny na asociativní elementy. Asociativní element připomíná výše popsaný adaptivní lineární neuron s tím, že všechny váhy w_i mají pevné hodnoty +1 nebo -1. Asociativní element se aktivuje (vydá hodnotu 1), pokud souhrn jeho vstupů překročí zadaný práh. Počet asociativních elementů je řádově desítky tisíc. Výstupy z asociativních elementů jsou náhodně zvolenými vazbami propojeny na reagující elementy, jejichž počet odpovídá počtu tříd, do kterých klasifikujeme. Reagující elementy realizují vážený součet

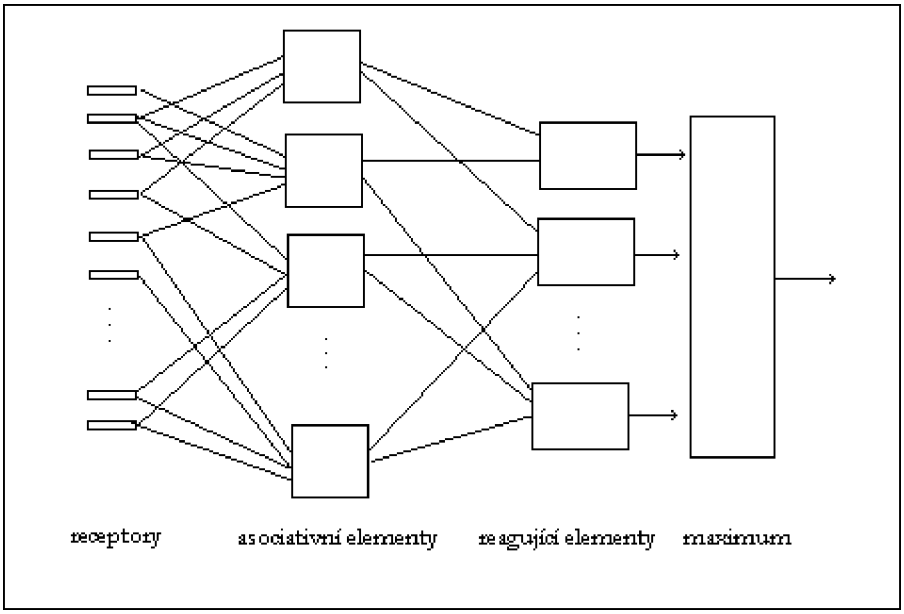
$$\sum_{i=1}^{m} w_i x_i$$

V bloku výběr maxima se vybere pro daný obraz ten reagující element, který má nejvyšší výstup, a který tedy odpovídá třídě, do které je obraz zařazen.

Učení *Perceptron*u probíhá na úrovni reagujících elementů. Pravidlo pro modifikaci vah odpovídá výše uvedené stochastické aproximaci

$$\mathbf{w}_{i+1} = \mathbf{w}_i + \Delta\mathbf{w}_i , \qquad \Delta\mathbf{w}_i = \eta\,(y_i - \hat{y}_i)\,\mathbf{x}_i .$$

[33] Počet těchto průchodů trénovací množinou je obvykle značný. Počet takzvaných iterací se pohybuje v řádech tisíců. Učení neuronů (a neuronových sítí) je tedy poněkud zdlouhavý proces.



Obrázek 33: Perceptron (podle [Kotek a kol., 1980])

Sám Rosenblatt zamýšlel perceptron jako model mozku. Myšlenka perceptronu však inspirovala mnoho techniků a rychle pronikla do aplikací jako učící se klasifikátor. Po počátečním nadšení se v roce 1969 v knize Perceptrons od M.Minského a S.Paperta objevila kritika klasifikačních schopností perceptronu. Minsky a Papert ukázali, že se perceptron jako lineární klasifikátor nedokáže vyrovnat s tak jednoduchým pojmem jako je nonekvivalence. Tato oprávněná námitka ale byla neprávem chápána jako kritika neuronových sítí jako takových. Výzkum neuronových sítí pak stagnoval až do poloviny 80. let, kdy dostal nový impuls. O renesanci neuronových sítí se zasloužili především Hopfield, Hecht-Nielsen a Kohonen. V té době se podařilo ukázat, že libovolnou spojitou funkci $f:[0,1]^n \rightarrow R^m$, $f(\mathbf{x}) = y$ lze s libovolnou přesností aproximovat pomocí třívrstvé sítě, která je tvořena n neurony ve vstupní vrstvě, $2n+1$ neurony v prostřední vrstvě a ni neurony ve výstupní vrstvě[34].

5.3.3 TOPOLOGIE SOUDOBÝCH SÍTÍ

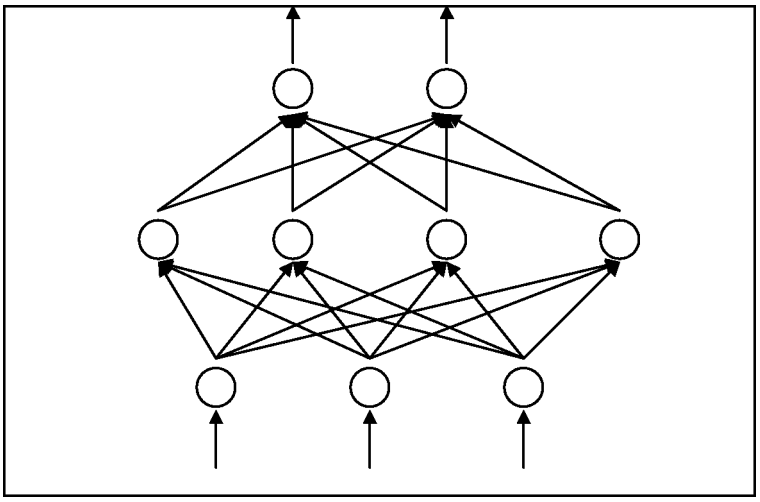
Kromě toho, že byly navrženy nové topologie neuronových sítí, podařilo se v osmdesátých letech rovněž nalézt algoritmy, které umožňovaly učení v těchto složitějších sítích[35]. Teprve tyto algoritmy umožnily návrat neuronových sítí na výsluní popularity. V současné době je známa (a používána) řada typů sítí. Podrobněji se seznámíme se jedním z nich: vícevrstvým perceptronem.

[34] Jde o modifikaci Kolmogorovova teorému z roku 1957, který ukázal, že libovolnou funkci z n-rozměrné krychle $[0,1]^n$ do množiny reálných čísel R lze vyjádřit jako funkci jedné proměnné.
[35] Viz např. práce Rumelhart, D.E. – Hinton,G.E. – Williams,R.J. Learning Internal Representations by Error Propagation z roku 1986.

DEFINICE

Vrstevná síť (vícevrstvý perceptron, multi-layer perceptron) je síť složená z vrstev neuronů, kde v rámci jedné vrstvy nejsou mezi neurony žádné vazby, ale neuron z jedné vrstvy je propojen se všemi neurony vrstvy sousední.

Nejpoužívanější topologií je síť s jednou skrytou vrstvou (Obrázek 34). Jedná se o zobecnění jednoduchého perceptronu (případně adaptivního lineárního neuronu), které má schopnost aproximovat libovolnou spojitou funkci.



Obrázek 34: Vícevrstvý perceptron s jednou skrytou vrstvou

Pro učení takovéto sítě se nejčastěji používá zobecněná gradientní metoda zvaná zpětné šíření chyby (*error backpropagation).* Na Obrázek 35 je uvedena inkrementální (stochastická) podoba tohoto algoritmu pro síť s jednou skrytou vrstvou ([Mitchell, 1997]). Opět je naším cílem minimalizovat chybu založenou na druhé mocnině rozdílu mezi skutečným a očekávaným výstupem sítě pro příklad $\boldsymbol{x}_i$.

$$\mathrm{Err}(\mathbf{w}_i) = \frac{1}{2}\sum_{v\in\text{výstupy}}^{n}(y_{i,v} - \widehat{y_{i,v}})^2$$

Změna váhy vazby vedoucí od neuronu *j* k neuronu *k* se tedy bude řídit gradientem funkce *Err*

$$\Delta\, w_{j,k} = -\eta \frac{\partial \mathrm{Err}}{\partial w_{j,k}} = -\eta \frac{\partial \mathrm{Err}}{\partial \mathrm{SUM_k}} \frac{\partial \mathrm{SUM_k}}{\partial w_{j,k}} = -\eta \frac{\partial \mathrm{Err}}{\partial \mathrm{SUM_k}}$$

kde SUM_k je vážený součet vstupů do neuronu *k*.

Základem algoritmu je výpočet chyby na výstupech jednotlivých neuronů. Nejprve se počítá chyba pro neurony ve výstupní vrstvě (vztah 2.1.2), to pak (zpětně) umožní spočítat chybu pro

neurony ve skryté vrstvě (vztah 2.1.3). Odvození těchto klíčových vztahů je uvedeno např. v kapitole 5.4 v knize [Berka, 2003].

Backpropagation algoritmus

1. inicializuj váhy sítě malými náhodnými čísly (např. z intervalu [-0.05,0.05])
2. dokud není splněno kritérium pro zastavení
   2.1. pro každý příklad [**x**,y] z trénovacích dat
      2.1.1. spočítej výstup out_u pro každý neuron *u* v síti
      2.1.2. pro každý neuron *v* ve výstupní vrstvě spočítej chybu

$$error_v = out_v (1 - out_v) (yv - out_v)$$

      2.1.3. pro každý neuron *s* ve skryté vrstvě spočítej chybu

$$error_s = out_s (1 - out_s) \sum_{v \in výstup} (ws,v\ error_v)$$

      2.1.4. pro každou vazbu vedoucí od neuronu *j* do neuronu *k* modifikuj váhu vazby

$$wj,k = wj,k + \Delta wj,k \ , \quad kde\ \Delta wj,k = \eta\ errork\ xj,k$$

Obrázek 35: Algoritmus Backpropagation[36]

Při zpětném šíření se v kroku práce sítě informace (výsledky) šíří od vstupní vrstvy k vrstvě výstupní, v kroku učení se váhy modifikují po vrstvách od výstupu k vstupu. Pro zastavení procesu učení se používají tato kritéria:

- ustálení chybové funkce (v minimu),
- dosažení předem zadaného počtu iterací,
- pokles pod předem zadanou hodnotu chybové funkce.

Vícevrstvá síť potřebuje pro učení oklasifikované příklady. Pracuje tedy v režimu učení s učitelem.

[36] Výstup z neuronu *u* je označen symbolem out_u (output), $x_{j,k}$ označuje *j*-tý vstup do *k*-tého neuronu $w_{j,k}$ označuje váhu *j*-tého vstupu do *k*-tého neuronu, neurony používají sigmoidální nelinearitu.

5.3.4 NEURONOVÉ SÍTĚ A DOBÝVÁNÍ ZNALOSTÍ Z DATABÁZÍ

Z hlediska dobývání znalostí z databází představují neuronové sítě jeden z nejpoužívanějších nástrojů pro tvorbu automatických systémů pro klasifikaci nebo predikci. Neuronové sítě jsou vhodnou alternativou k rozhodovacím stromům a pravidlům v situacích, kdy netrváme na srozumitelnosti nalezených znalostí.

Jiným dobrým důvodem pro použití neuronových sítí je převaha numerických atributů. Zatímco symbolické metody (stromy, pravidla) jsou šity na míru kategoriálním datům a numerická data vyžadují speciální zacházení (diskretizaci), nyní je situace opačná. Neuronové sítě jsou vhodnější pro data numerická a problémy působí data kategoriální. Standardním řešením tohoto problému je tzv. *binarizace*. Pro jeden kategoriální atribut vytvoříme tolik nových binárních atritbutů, kolik měl původní atribut různých hodnot. Hodnota 1 u nového atributu A'_k vyjadřuje, že původní atribut A měl pro daný objekt hodnotu v_k.; ostatní atributy A'_q , $q \neq k$ pak mají hodnotu 0. Má-li původní atribut pouze dvě možné hodnoty, binarizaci nemusíme provádět; stačí když jedné z hodnot přiřadíme novou hodnotu 1 a druhé z hodnot novou hodnotu 0. Tato binarizace se často provádí v rámci algoritmu pro učení neuronové sítě, uživatel tedy nebývá zatěžován nutností data transformovat ručně.

Vezměme k ruce opět náš příklad bankovní aplikace. Pro vytvoření klasifikačního modelu můžeme použít vícevrstvý perceptron s jednou skrytou vrstvou. Topologii sítě bude určovat zvolená úloha i podoba trénovacích dat. Počet neuronů ve vstupní vrstvě bude vycházet z počtu vstupních atributů. Vzhledem k tomu, že všechny atributy jsou kategoriální, musíme provést binarizaci. Získáme tak 6 binárních atributů *příjem_vysoký*, *konto_vysoké*, *konto_střední*, *konto_nízké*, *pohlaví_muž*, *nezaměstnaný*, *úvěr* ; ve vstupní vrstvě tedy bude 6 neuronů. Podobu dat po binarizaci ukazuje Tabulka 10. Výstupní vrstva bude tvořena jedním neuronem reprezentujícím závěr *úvěr(ano)*. Jeho výstup (v intervalu [0, 1]) bude interpretován jako doporučení zda půjčit ($\hat{y}>0.5$) nebo nepůjčit ($\hat{y}<0.5$)[37].

Problémem bývá určení počtu neuronů ve skryté vrstvě. Zdálo by se, že čím více, tím lépe. S rostoucím počtem neuronů ve skryté vrstvě (resp. s růstem počtu skrytých vrstev) se zvyšuje nelinearita chování sítě, rostou ale nároky na proces učení (potřebný počet příkladů, doba učení). Příliš rozsáhlá síť má rovněž tendenci k přeučení (overfitting); může se příliš zaměřit na nevýznamné podrobnosti vyskytující se v trénovacích datech, které ale nemají význam z hlediska řešené úlohy. Neexistuje obecný návod jak zvolit počet neuronů (jedna z používaných heuristik radí, aby neuronů ve skryté vrstvě bylo dvakrát tolik, jako je neuronů ve vstupní vrstvě). Přidržíme se této heuristiky a zvolíme 10 neuronů. V praxi je obvyklé zkusit více sítí s různými parametry (topologiemi, přechodovými funkcemi ...) a na základě jejich chování na testovacích datech vybrat tu nejlepší.

[37] Obecně platí, že ve výstupní vrstvě je tolik neuronů, kolik je tříd mezi kterými se systém rozhoduje. V případě dvou tříd ale stačí jeden neuron.

Pro výše popsanou topologii 6-10-1 získáme pomocí algoritmu zpětného šíření pro naše data takové nastavení vah, že síť bude bezchybně klasifikovat všechny trénovací příklady (úplný popis sítě je uveden v příloze).

Naše data z Tabulka 10 jsou lineárně separabilní (což jsme dopředu nemuseli vědět), stačila by tedy i jednodušší síť tvořená jedním neuronem[38]. Obecně ale platí, že vícevrstvý perceptron nalezne libovolně složitý popis tříd, tedy v prostoru atributů libovolně složitou „hranici" mezi třídami.

Tabulka 10: Binarizovaná data

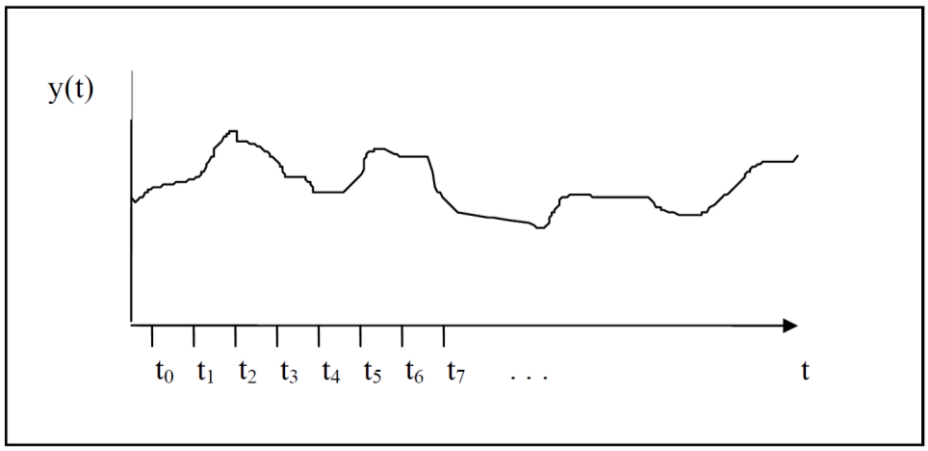
klient	příjem_ vysoký	konto_ vysoké	konto_ střední	konto_ nízké	pohlaví_ muž	nezaměst naný	úvěr
k1	1	1	0	0	0	0	1
k2	1	1	0	0	1	0	1
k3	0	0	0	1	1	0	0
k4	0	1	0	0	0	1	1
k5	0	1	0	0	1	1	1
k6	0	0	0	1	0	1	0
k7	1	0	0	1	1	0	1
k8	1	0	0	1	0	1	1
k9	0	0	1	0	1	1	0
k10	1	0	1	0	0	0	1
k11	0	0	1	0	0	1	0
k12	0	0	1	0	1	0	1

Naše bankovní data ilustrují jeden typ použití neuronových sítí, klasifikační úlohy. Typičtějším příkladem použití neuronových sítí je úloha predikce časových řad (cen akcií, směnných kurzů měn, spotřeby elektrické energie nebo meteorologické situace). Vícevrstvý perceptron může predikovat jak kvalitativní vývoj (vzrůst, pokles), tak i konkrétní hodnoty. S výhodou se v tomto typu aplikací použije skutečnost, že neuronové sítě bez problémů zpracovávají numerické atributy. Časová řada má obvykle podobu hodnot dané veličiny získané v po sobě jdoucích časových okamžicích (Obrázek 36)[39]. Z této řady je třeba získat trénovací příklady pro neuronovou síť. Řekněme, že nás bude zajímat predikce na jeden časový okamžik dopředu; výstupní vrstva bude tedy tvořena jediným neuronem. Počet neuronů ve vstupní vrstvě závisí na tom, jak dlouhý historický úsek chceme brát při predikci v úvahu. Toto silně závisí na predikované veličině; o chování veličiny můžeme mít nějaké další dodatečné informace (předpokládané periodické nebo sezónní chování apod.). Budeme-li např. chtít predikovat ze čtyř posledních hodnot, bude mít vstupní vrstva čtyři neurony. Počet neuronů ve skryté vrstvě bude opět předmětem experimentování. Takto navržené topologii odpovídá podoba trénovacích dat uvedená na Obrázek 37 (pro případ, že chceme predikovat přímo hodnotu veličiny), resp. na Obrázek 38 (pro případ, že

[38] Rovnice odpovídající rozdělující nadroviny je 6.7898 příjem_vysoký + 2.8184 konto_vysoké - 1.9507 konto_střední - 5.1480 konto_nízké - 0.5172 pohlaví_muž - 3.7801 nezaměstnaný - 2.1758 = 0.

[39] Obvykle se hodnoty veličiny zaznamenávají po uplynutí stejného časového úseku (ekvidistantní vzorkování), některé modernější přístupy pro predikci např. směnných kurzů ale používají vzorkování, které závisí na chování dané veličiny - tzv. tiková data se zaznamenávají v okamžiku, kdy změna kurzu překročí zadaný práh.

chceme predikovat jen změnu veličiny). V obou případech vytváříme trénovací příklady ze sekvencí po sobě jdoucích hodnot časové řady.



Obrázek 36: Časová řada

vstupy				výstup
$y(t_0)$	$y(t_1)$	$y(t_2)$	$y(t_3)$	$y(t_4)$
$y(t_1)$	$y(t_2)$	$y(t_3)$	$y(t_4)$	$y(t_5)$
$y(t_2)$	$y(t_3)$	$y(t_4)$	$y(t_5)$	$y(t_6)$
	. . .			

Obrázek 37: Trénovací data pro predikci hodnot

vstupy				výstup
$sign(y(t_1) - y(t_0))$	$sign(y(t_2) - y(t_1))$	$sign(y(t_3) - y(t_2))$	$sign(y(t_4) - y(t_3))$	$sign(y(t_5) - y(t_4))$
$sign(y(t_2) - y(t_1))$	$sign(y(t_3) - y(t_2))$	$sign(y(t_4) - y(t_3))$	$sign(y(t_5) - y(t_4))$	$sign(y(t_6) - y(t_5))$
	. . .			

Obrázek 38: Trénovací data pro predikci změny

Použití neuronových sítí s sebou nese problém interpretace. Znalosti získané neuronovými sítěmi jsou zcela nesrozumitelné pro uživatele (znalosti jsou dány topologií sítě a váhami vazeb mezi neurony). S tím souvisí i neschopnost sítí podávat vysvětlení. Z neuronových sítí se tedy stává černá skříňka, do které není vidět. Objevují se ale pokusy převést znalosti uložené v neuronových sítích do srozumitelnější podoby [Shavlik, 1992]. Obdobně se zkoumá možnost využít doménové znalosti při inicializaci neuronových sítí (např. systém KBANN popsaný v [Towell, Shavlik 1994]). V [Berka, Sláma, 1998] lze pak nalézt postup, kdy se nejprve (na základě znalostí) nastaví topologie sítě, ta se naučí z dat váhy vazeb a pak je zpět převedena do vhodnější reprezentace. Ve všech uvedených přístupech se vychází z toho, že vazby mezi neurony lze interpretovat jako pravidla. Tak např. pro neuron na Obrázek 29 by měla takováto pravidla podobu:

IF příjem(vyhovuje) THEN úvěr(ano) (w1)

IF konto(vyhovuje)	THEN úvěr(ano) (w2)
IF příjem(nevyhovuje)	THEN úvěr(ano) (-w1)
IF konto(nevyhovuje)	THEN úvěr(ano) (-w2)
	úvěr(ano) (-w0)

Při klasifikaci nového případu se použijí všechna aplikovatelná pravidla způsobem známým z kompozicionálních expertních systémů[40].

Na závěr této části ještě jedna poznámka. Neuronové sítě se výrazně liší od tradiční von Neumannovy koncepce počítače. Není v nich striktně oddělen procesor a paměť a informace v nich není lokalizována na nějakém pevném místě (adrese) ale je „rozprostřena" ve vahách po celé síti. To umožňuje fungování sítě i v případě částečného poškození sítě, při neúplných nebo zašuměných datech apod. Mají v tomto smyslu blíže k (lidskému) mozku než klasické počítače.

5.4 Evoluční algoritmy

Jinou skupinou metod strojového učení, které vycházejí z biologických principů, jsou evoluční algoritmy. Zdrojem inspirace se tentokrát stal mechanismus evoluce, chápaný jako Darwinův přirozený výběr: nějaký živočišný druh se během svého vývoje zdokonaluje tak, že z generace na generaci se přenáší genetická informace jen těch nejsilnějších jedinců. Mezi evoluční algoritmy patří genetické algoritmy, evoluční programování, evoluční strategie a genetické programování.

K ZAPAMATOVÁNÍ

Pro všechny tyto metody jsou společné některé základní pojmy: výběr (selekce), mutace a reprodukce (křížení) jedinců z určité populace, a princip nesystematického náhodného generování a testování.

5.4.1 ZÁKLADNÍ PODOBA GENETICKÝCH ALGORITMŮ

U zrodu genetických algoritmů stál v 60. letech J. Holland [Holland, 1962]. Jeho myšlenkou bylo použít evoluční principy, založené na metodách optimalizace funkcí a umělé inteligenci, pro hledání řešení nějaké úlohy. V případě algoritmické realizace procesu evoluce je podstatný způsob reprezentování jedinců (chromozomů). V nejjednodušším případě můžeme chromozom chápat jako řetězec složený ze symbolů 0 a 1 (*genů*), budeme rovněž předpokládat, že všechny

[40] Kompozicionální expertní systémy jsou založeny na myšlence, že závěr konzultace se odvodí ze všech aplikovatelných pravidel tak, že se zkombinují dílčí příspěvky těchto pravidel.

řetězce mají stejnou délku. Genetická výbava jedinců z jedné populace může přecházet do populace následující přímo, jen drobně modifikovaná, nebo prostřednictvím potomků. Snahou algoritmu je přitom přenést z jedné populace do druhé jen to nejlepší. Jedinci v populaci představují potenciální řešení nějaké úlohy (např. hypotézy pokrývající příklady konceptů). Kritériem hodnocení jedinců v populaci je tedy kvalita tohoto řešení vyjádřená pomocí funkce *fit* (fitness function). V případě učení se konceptům je touto funkcí např. přesnost jedince (hypotézy) při klasifikaci, v případě hledání extrému nějaké funkce je to hodnota této funkce.

Genetický algoritmus(fit,N,K,M)

Inicializace

1. přiřaď *t:= 0* (počítadlo generací)
2. náhodně vytvoř populaci *P(t)* velikosti *N*
3. urči hodnoty funkce *fit* pro každého jedince *h* v *P(t)*

Hlavní cyklus

1. dokud není splněna podmínka pro zastavení
   - 1.1. proveď selekci:
     - 1.1.1. vyber z *P(t)* jedince, kteří se přímo přenesou do *P(t+1)*
   - 1.2. proveď křížení:
     - 1.2.1. vyber z *P(t)* dvojice jedinců určených k reprodukci
     - 1.2.2. aplikuj na každou dvojici operaci křížení
     - 1.2.3. zařaď potomky do *P(t+1)*
   - 1.3. proveď mutaci:
     - 1.3.1. vyber z *P(t+1)* jedince určené k mutaci
     - 1.3.2. aplikuj na každého jedince operaci mutace
   - 1.4. přiřaď *t:= t + 1* (nová populace má opět velikost *N*)
   - 1.5. spočítej pro každé $h \in P(t)$ hodnotu *fit(h)*
2. vrať jedince *h* s nejvyšší hodnotu *fit(h)*

Obrázek 39: Základní podoba genetického algoritmu

Základní podoba genetického algoritmu tak, jak ji uvádí Mitchell [Mitchell, 1997] je na Obrázek 39. Algoritmus začíná pracovat s nějakou výchozí, náhodně zvolenou populací, kterou postupně modifikuje (zdokonaluje). Činnost algoritmu končí po splnění nějaké podmínky. Tou nejčastěji bývá dosažení předem zadaného maxima pro funkci *fit*, může to ale být i vyčerpání maximálního času (počtu generací). Operacemi, které vedou k vytvoření nové populace, jsou *výběr* (*selekce*), *mutace* a *křížení*.

Selekce jedince *h* se provádí na základě hodnoty funkce *fit(h)*. V literatuře se uvádějí různé možnosti:

- *ruletové kolo*

pravděpodobnost, že bude vybrán jedinec *h* je úměrná poměru $\frac{fit(h)}{\sum_i fit(h_i)}$

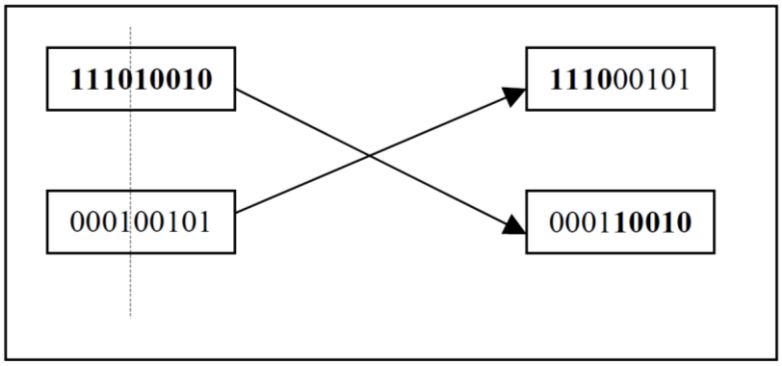
- *pořadová selekce*

  nejprve jsou jedinci v populaci uspořádáni podle hodnoty *fit*, selekce se pak provádí na základě pravděpodobnosti, která je úměrná pořadí jedince v tomto uspořádání,

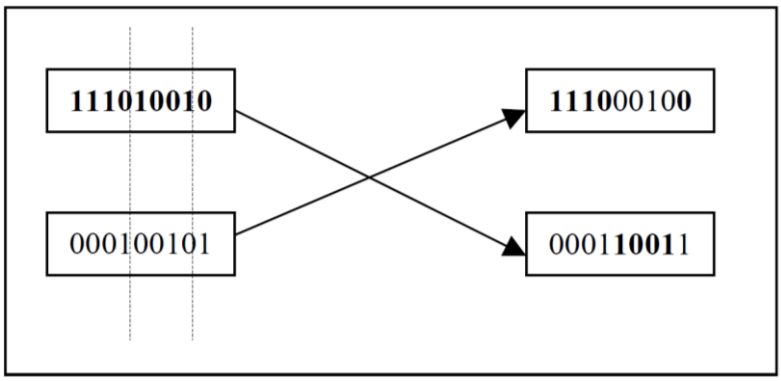
- *turnajová selekce*

  nejprve se náhodně vyberou dva jedinci, s předefinovanou pravděpodobností *p* se pak z těchto dvou vybere jedinec s vyšší hodnotou *fit*, případně s pravděpodobností *1-p* se vybere ten jedinec s nižší hodnotou *fit*.

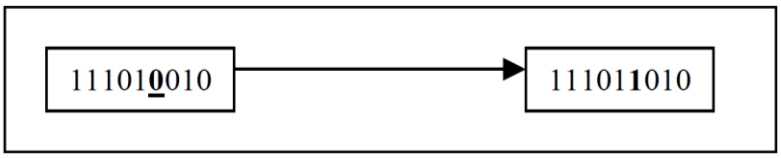
Operátor *křížení* (crossover) vytváří ze dvou rodičů dva potomky. V nejjednodušším případě se provádí tzv. jednobodové křížení (single-point crossover). Při jednobodovém křížení „zdědí“ každý potomek počátek chromozomu (řetězce bitů) od jednoho rodiče a zbytek řetězce od druhého rodiče. Bod křížení je pokaždé volen náhodně (Obrázek 40). Složitější variantou je dvoubodové křížení, kdy se chromozomy rozdělí na tři části, každý potomek pak zdědí „prostředek“ řetězce od jednoho rodiče a „okraje“ od druhého rodiče (Obrázek 41).



Obrázek 40: Jednobodové křížení



Obrázek 41: Dvoubodové křížení



Obrázek 42: Mutace

Při *mutaci* se náhodně zinvertuje jeden bit v řetězci tak, jak je uvedeno na Obrázek 42. Zatímco křížení představuje výrazné změny chromozomů, v případě mutace se jedná jen o drobné modifikace.

Výběr se provádí jednak v případě přímého přenosu jedince z jedné generace do druhé (krok selekce), jednak v případě volby rodičů pro křížení nebo jedinců pro mutaci. Postupuje se přitom tak, aby se počet jedinců v populacích nezvětšoval. V algoritmu z Obrázek 39 se tedy např. přímo přenese $(1 - K) * N$ jedinců (krok 1.1.1) a pro křížení se vybere $K*N/2$ dvojic (krok 1.2.1). Tímto způsobem zůstane zachován počet N jedinců v populaci. Pro mutaci se pak vybere $M * N$ jedinců (K, M a N jsou volené parametry).

Nabízí se otázka, jestli algoritmus pracující výše popsaným způsobem může efektivně řešit optimalizační úlohy nebo úlohy prohledávání. Kladnou odpověď dává tzv. teorém o schématech [Holland 1997][41].

Genetické algoritmy trpí podobnou nectností jako neuronové sítě, tendencí nalézt lokální místo globálního optima. Zde to znamená, že nejlepší jedinec má snahu se reprodukovat nejvíce, což vede k málo diverzifikované populaci. V určitém stádiu výpočtu populace zkonverguje do výsledného stavu, ve kterém už operace křížení a mutace nepřinášejí zlepšování jedinců. Nápravu tohoto jevu může přinést modifikování operace selekce (použít turnajové místo ruletové), další podmínky na podobu rodičů v operaci křížení, nebo snížení hodnoty funkce *fit* v případě výskytu podobných jedinců v populaci.

[41] Hollandův teorém říká že krátká schémata nízkého řádu s nadprůměrnou hodnotou funkce *fit* získávají v následující populaci exponenciálně rostoucí zastoupení. Schématem se přitom myslí jakási šablona, která popisuje množinu vzájemně si podobných čáastí chromozomů. V případě reprezentace chromozomů bitovými řetězci se v zápise schématu objevují "1", "0" a "∗", přičemž znak "∗" reprezentuje libovolnou hodnotu. Tedy např. schema 00∗1 zahrnuje dva řetězce 0001 a 0011. Délkou schématu se myslí vzdálenost mezi první a poslední pevně danou pozicí ve schématu (v našem příkladu je délka 3) a řádem schématu se myslí počet pevně definovaných symbolů (v našem příkladu opět hodnota 3). Schemata reprezentují sekvence bitů, které jsou navzájem provázány v tom smyslu, že příspěvek jednoho bitu k hodnotě funkce *fit* závisí na hodnotách ostatních bitů v sekvenci. Tyto sekvence bývají označovány jako *stavební bloky*. Teorém o schématech tedy říká, že v průběhu evoluce roste zastoupení důležitých stavebních bloků v populaci. Tyto stavební bloky se pak kombinují do výsledného řešení.

5.4.2 POUŽITÍ GENETICKÝCH ALGORITMŮ

Genetické algoritmy nalezly uplatnění v řadě oblastí: numerická optimalizace a rozvrhování, strojové učení, tvorba modelů (ekonomických, populačních, sociálních), apod. Z hlediska dobývání znalostí z databází je zajímavé využití genetických algoritmů přímo pro učení se konceptům[42], nebo použití genetických algoritmů pro optimalizaci neuronových sítí.

Použití genetických algoritmů pro učení se konceptům budeme opět demonstrovat na příkladu, který nás provází celou knihou. Vyjdeme přitom ze systému *GABIL* ([DeJong a kol., 1993]). Nejdůležitější otázkou, kterou je třeba zodpovědět je způsob reprezentování hypotéz. *GABIL* pracuje s hypotézami v podobě pravidel, tedy např.

If konto(nízké) ∧ příjem(nízký) then úvěr(ne)

If konto(vysoké) then úvěr(ano)

pro kódování hypotéz se používá bitový řetězec tak, že pro každý atribut je vyhrazeno tolik bitů, kolik má atribut různých hodnot. 1 na příslušné pozici v řetězci pak znamená, že atribut nabývá tuto hodnotu, 0 znamená, že atribut nenabývá tuto hodnotu. Použití tolika bitů, kolik je hodnot umožňuje kódovat i disjunkce (011 v zápisu atributu vyjadřuje, že atribut nabývá druhé nebo třetí hodnoty) i vyjádřit, že na hodnotě atributu nezáleží (zápis 111). Výše uvedená pravidla lze tedy kódovat jako

100 10 01
001 11 10

kde atribut *konto* nabývá tří hodnot v pořadí *nízké*, *střední*, *vysoké*, atribut *příjem* nabývá dvou hodnot v pořadí *nízký*, *vysoký* a atribut *úvěr* nabývá dvou hodnot v pořadí *ano*, *ne*.

GABIL pracuje podle obecného algoritmu uvedeného na Obrázek 39 s těmito upřesněními:

1. funkce *fit(h)* je druhou mocninou správnosti klasifikace hypotézou h[43]

$$\mathrm{fit(h)} = \left(\frac{a}{a+b}\right)^2$$

2. počet jedinců v populaci je mezi 100 a 1000 v závislosti na konkrétní úloze,
3. parametr K vyjadřující podíl křížení má hodnotu 0.6,
4. parametr M vyjadřující podíl mutací má hodnotu 0.001,

[42] Genetický algoritmus provádí paralelní náhodné prohledávání prostoru hypotéz. Jeho schopnost učit se koncepty je srovnatelná s algoritmy pro tvorbu rozhodovacích stromů i rozhodovacích pravidel.
[43] Hypotéza je tvořena pravidlem. Pro toto pravidlo můžeme vytvořit čtyřpolní tabulku určující, kolik pozitivních příkladů (hodnota a) a kolik všech příkladů (hodnota a + b) toto pravidlo pokrývá.

5. použitý operátor křížení je rozšířením výše uvedeného dvoubodového křížení; provedené rozšíření umožňuje křížit řetězce různých délek,

6. mutace je použita tak, jak je uvedeno výše.

Jiným příkladem použití genetického algoritmu pro učení se konceptům je modifikace systému *CN4* popsaná v [Králík, Brůha, 1998]. Připomeňme, že *CN4* je algoritmus pro tvorbu rozhodovacích pravidel metodou pokrývání množin shora dolů. Jádrem algoritmu je nalezení jednoho pravidla. V původním algoritmu se vhodný předpoklad pravidla vytváří metodou specializace kombinace. Modifikovaná verze, systém *GA-CN4*, používá pro nalezení pravidla genetický algoritmus. Kombinace je kódována podobně jako v systému *GABIL* (lze tedy vytvářet disjunkce hodnot), navíc se ke každému atributu přidá jeden bit umožňující vyjádřit negaci. Tím se značně rozšíří vyjadřovací schopnost nalezených pravidel.

V případě optimalizace neuronových sítí (viz např. [Murray, 1994]) vlastně genetický algoritmus automatizuje proces ručního nastavování parametrů a ladění neuronové sítě. Jedinci v populaci pak odpovídají parametrům jednotlivých konfigurací sítě (počet neuronů, parametry neuronů) a kriteriální funkce *fit* odpovídá chybě sítě.

5.5 Bayesovská klasifikace

Metody bayesovské klasifikace vycházejí z Bayesovy věty o podmíněných pravděpodobnostech. Ačkoliv se tedy jedná o metody pravděpodobnostní, jsou intenzivně studovány v souvislosti se strojovým učením a uplatňují se rovněž v systémech pro dobývání znalostí.

5.5.1 Základní pojmy

K ZAPAMATOVÁNÍ

Bayesův vztah pro výpočet podmíněné pravděpodobnosti že platí hypotéza *H*, pokud pozorujeme evidenci *E* má podobu:

$$P(H|E) = \frac{P(E|H)\ P(H)}{P(E)}$$

Apriorní pravděpodobnost hypotézy *P(H)* odpovídá znalostem o zastoupení jednotlivých hypotéz (tříd) bez ohledu na nějaké další informace. Podmíněná pravděpodobnost *P(H|E)*, též nazývaná *aposteriorní*, vyjadřuje, jak se změní pravděpodobnost hypotézy, pokud víme, že nastalo *E*. *P(E)* vyjadřuje pravděpodobnost evidence (pozorování).

Hypotéz, mezi kterými se rozhodujeme, bývá obvykle více[44]. Nás bude zajímat pro danou evidenci ta nejpravděpodobnější. Pro každou hypotézu H_t, $t=1,...,T$ můžeme spočítat $P(H_t|E)$ a z nich vybrat hypotézu H_{MAP}, která má největší aposteriorní pravděpodobnost (maximum aposteriori probability)

$$H_{MAP} = H_J \text{ právě když } P(H_J|E) = \max_t \frac{P(E|H_t)P(H_t)}{P(E)}$$

Vzhledem k tomu, že nás zajímá pouze to, pro které H_t je hodnota aposteriorní pravděpodobnosti maximální, ale už nás nezajímá konkrétní hodnota, můžeme výpočet poněkud zjednodušit zanedbáním jmenovatele.

$$H_{MAP} = H_J \text{ právě když } P(E|H_J) \cdot P(H_J) = \max_t (P(E|H_t) \cdot P(H_t))$$

V zanedbávání můžeme jít ještě dále tak, že budeme předpokládat, že všechny hypotézy jsou stejně pravděpodobné (a tedy že na $P(H)$ nezáleží). Nalezneme tak hypotézu, která má největší věrohodnost (maximum likelihood):

$$H_{ML} = H_J \text{ právě když } P(E|H_J) = \max_t P(E|H_t)).$$

ŘEŠENÁ ÚLOHA

Pro ilustraci tohoto postupu vezměme opět úlohu poskytování úvěru, tentokrát ale pouze na základě výše příjmu. Přepokládejme, že banka vyhoví u 2/3 žádostí o úvěr; tedy apriorní ravděpodobnosti budou P(půjčit)= 0.667 a P(nepůjčit) = 0.333. Dále předpokládejme, že vysoký příjem mělo 91% klientů, kterým banka půjčila, a nízký příjem mělo 88% klientů, kterým banka nepůjčila. Tedy

$$P(\text{vysoký\_příjem}|\text{půjčit}) = 0.91 \quad P(\text{nízký\_příjem}|\text{půjčit}) = 0.09$$

$$P(\text{vysoký\_příjem}|\text{nepůjčit}) = 0.12 \quad P(\text{nízký\_příjem}|\text{nepůjčit}) = 0.88.$$

Předpokládejme, že posuzujeme klienta s vysokým příjmem. Bude větší pravděpodobnost, že banka půjčí nebo že nepůjčí? Podle Bayesovy věty spočítáme

$$P(\text{vysoký\_příjem}|\text{půjčit}) \times P(\text{půjčit}) = 0.607$$

$$P(\text{vysoký\_příjem}|\text{nepůjčit}) \times P(\text{nepůjčit}) = 0.040$$

[44] V tomto případě se P(E) ve jmenovateli Bayesova vztahu obvykle vyjadřuje jako $\sum_t P(E|H_t)\,P(H_t)$.

Tedy H_{MAP} = půjčit. Víme tedy, která hypotéza je pravděpodobnější, přestože jsme přímo nespočítali aposteriorní pravděpodobnosti obou hypotéz. Tyto pravděpodobnosti získáme, pokud budeme uvedené hodnoty normovat tak, aby jejich součet byl roven 1.

Výhodou bayesovských metod je právě tato schopnost klasifikovat příklady do tříd s určitou pravděpodobností. Tuto pravděpodobnost můžeme interpretovat jako spolehlivost rozhodnutí.

Bayesova věta dává návod jak stanovit vliv jedné evidence na uvažovanou hypotézu. Jak ale postupovat, pokud je evidencí více? Tedy, jak stanovit aposteriorní pravděpodobnost $P(H|E_1, \ldots, E_k)$? Následující podkapitola ukazuje jeden z možných přístupů.

5.5.2 NAIVNÍ BAYESOVSKÝ KLASIFIKÁTOR

DEFINICE

Naivní bayesovský klasifikátor vychází z předpokladu, že jednotlivé evidence $B_1,\ldots,B_K$ jsou podmíněně nezávislé při platnosti hypotézy H [Duda, Hart, 1973].

Tento zjednodušující předpoklad umožňuje spočítat aposteriorní pravděpodobnost hypotézy při platnosti všech evidencí

$$P(H|E_1, \ldots, E_k) = \frac{P(E_1, \ldots, E_k|H) \times P(H)}{P(E_1, \ldots, E_k)}$$

jako

$$P(H|\ E_1,\ldots,E_K) = \frac{P(H)}{P(E_1,\ldots,E_K)} \times \prod_{k=1}^{K} P(E_k|H)\ .$$

V případě klasifikace pomocí naivního bayesovského klasifikátoru tedy budeme hledat hypotézu s největší aposteriorní pravděpodobností H_{MAP}

$$H_{MAP} = H_J \text{ právě když } P(H_J) \times \prod_{k=1}^{K} P(E_k|H_J) = \max_t \left(P(H_t) \times \prod_{k=1}^{K} P(E_k|H_t)\right)$$

Abychom mohli tento způsob klasifikace použít, potřebujeme znát hodnoty $P(H_t)$ a $P(E_k|H_t)$. V kontextu dobývání znalostí z databází můžeme tyto hodnoty získat z trénovacích dat ve

fázi učení[45]. Evidence E_k jsou pak hodnoty jednotlivých vstupních atributů (tedy $E_k=A_j(v_k)$) a hypotézy H_t jsou cílové třídy (tedy $H_t=C(v_t)$).

Na rozdíl od jiných algoritmů učení (rozhodovací stromy, rozhodovací pravidla) se zde neprovádí prohledávání prostoru hypotéz. Stačí jen spočítat příslušné pravděpodobnosti na základě četnosti výskytů hodnot jednotlivých atributů[46]. Tedy pravděpodobnosti $P(H_t)$ a $P(E_k|H_t)$ lze spočítat jako

$$P(H_t) = P(C(v_t)) = \frac{n_t}{n}$$

$$P(E_k \mid H_t) = P(A_j(v_k) \mid C(v_t)) = \frac{n_t(A_j(v_k))}{n_t}$$

Tabulka 11: Trénovací data pro bayesovský klasifikátor

klient	příjem	konto	pohlaví	nezaměstnaný	úvěr
k1	vysoký	vysoké	žena	ne	ano
k2	vysoký	vysoké	muž	ne	ano
k3	nízký	nízké	muž	ne	ne
k4	nízký	vysoké	žena	ano	ano
k5	nízký	vysoké	muž	ano	ano
k6	nízký	nízké	žena	ano	ne
k7	vysoký	nízké	muž	ne	ano
k8	vysoký	nízké	žena	ano	ano
k9	nízký	střední	muž	ano	ne
k10	vysoký	střední	žena	ne	ano
k11	nízký	střední	žena	ano	ne
k12	nízký	střední	muž	ne	ano

Pro náš příklad trénovacích dat z Tabulka 11 budou apriorní pravděpodobnosti různých hodnot cílového atributu *úvěr*:

$$P(\text{úvěr(ano)}) = 8/12 = 0.667$$

$$P(\text{úvěr(ne)}) = 4/12 = 0.333$$

Podobně spočítáme podmíněné pravděpodobnosti $P(A_j(v_k)|C(v_t))$, např.

$$P(\text{konto(střední)}|\text{úvěr(ano)}) = 2/8 = 0.25$$

$$P(\text{konto(střední)}|\text{úvěr(ne)}) = 2/4 = 0.5$$

$$P(\text{nezaměstnaný(ne)}|\text{úvěr(ano)}) = 5/8 = 0.625$$

$$P(\text{nezaměstnaný(ne)}|\text{úvěr(ne)}) = 1/4 = 0.25$$

[45] Jedná se tedy opět o učení s učitelem.
[46] Vzhledem k tomu se bayesovský klasifikátor hodí pro veliké datové soubory.

Pro uchazeče o úvěr, který má střední konto a není nezaměstnaný, spočítáme

$$P(\text{úvěr(ano)})\ P(\text{konto(střední)}|\text{úvěr(ano)})\ P(\text{nezaměstnaný(ne)}\ |\text{úvěr(ano)}) = 0.1042$$

$$P(\text{úvěr(ne)})\ P(\text{konto(střední)}|\text{úvěr(ne)})\ P(\text{nezaměstnaný(ne)}|\text{úvěr(ne)}) = 0.0416$$

Tedy naivní bayesovský klasifikátor zařadí tohoto uchazeče do třídy *úvěr(ano)*. Můžeme si všimnout, že jsme úspěšně klasifikovali neúplně popsaný případ, který by zůstal nezařazen dříve vytvořenými rozhodovacími stromy i pravidly.

5.6 Metody založené na analogii

Metody založené na analogie vychází z následujícího principu: *v neznámé situaci použij to řešení, které se osvědčilo v situaci podobné*. Znalosti jsou tentokrát reprezentovány v podobě databáze již vyřešených problémů. Ve fázi učení se tedy neprovádí generalizace z příkladů. Při usuzování se pak v této databázi hledá nejpodobnější případ, dříve použité řešení někdy musí být adaptováno na novou situaci.

Metody založené na analogii lze použít pro deskriptivní úlohy (segmentaci a shlukování) i pro úlohy klasifikační. Podobně jako na jiných místech tohoto textu, i zde bude větší důraz kladen na klasifikaci.

5.6.1 Podobnost mezi příklady

Klíčovým pojmem je koncept *podobnosti*, resp. *vzdálenosti*. Jak podobnost, tak vzdálenost, se vyjadřuje pomocí metriky.

Df ***DEFINICE***

Metrika je definována jako funkce d: $\mathbf{X} \times \mathbf{X} \rightarrow R$ taková, že

1. $\forall \mathbf{x}_1,\mathbf{x}_2 \in \mathbf{X};\ d(\mathbf{x}_1,\mathbf{x}_2) \geq 0$
2. $d(\mathbf{x}_1,\mathbf{x}_2) = 0 \Leftrightarrow \mathbf{x}_1 = \mathbf{x}_2$
3. $d(\mathbf{x}_1,\mathbf{x}_2) = d(\mathbf{x}_2,\mathbf{x}_1)$
4. $\forall \mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3 \in \mathbf{X};\ d(\mathbf{x}_1,\mathbf{x}_2) + d(\mathbf{x}_2,\mathbf{x}_3) \geq d(\mathbf{x}_1,\mathbf{x}_3)$

Nejpoužívanější metriky již známe z kapitoly o statistických metodách. Jsou to eukleidovská vzdálenost

$$d_E(\mathbf{x_1},\mathbf{x_2}) = \sqrt{\sum_{j=1}^{m} \delta_E(x_{1j}, x_{2j})}, \text{ kde } \delta_E(x_{1j}, x_{2j}) = (x_{1j} - x_{2j})^2$$

a Hammingova vzdálenost (v komunitě strojového učení nazývaná Manhattan, nebo city-block)[1]

$$d_H(\mathbf{x_1},\mathbf{x_2}) = \sum_{j=1}^{m} \delta_H(x_{1j}, x_{2j}), \text{ kde } \delta_H(x_{1j}, x_{2j}) = |x_{1j} - x_{2j}|$$

Podobnost pak můžeme spočítat jako $1/d(\mathbf{x_1},\mathbf{x_2})$ nebo jako $1 - d(\mathbf{x_1},\mathbf{x_2})$. První varianta se používá pro $d(\mathbf{x_1},\mathbf{x_2}) \in R$ (nenormalizovaná vzdálenost), druhá varianta se používá pro $d(\mathbf{x_1},\mathbf{x_2}) \in [0,1]$ (vzdálenost je normalizována např. tak, že je vydělena vzdáleností mezi dvěma krajními hodnotami definičního oboru).

5.6.2 NEJBLIŽŠÍ SOUSED

Metoda klasifikace na základě K-nejbližších sousedů (Knearest neighbour rule) vychází z dříve zmíněných základních principům využití analogie:

- ve fázi učení se neprovádí generalizace
- klasifikace se provádí na základě podobnosti

a přidává ještě třetí princip

- příklady jsou chápány jako body v n-rozměrném prostoru atributů.

Základní podobu algoritmu uvádí Obrázek 43 ([Mitchell, 1997]). Ve fázi učení si systém zapamatuje všechny příklady $[\mathbf{x}_k, y_k]$ z trénovací množiny. Ve fázi klasifikace se pro nový příklad **x** nalezne (za použití zvolené metriky) K nejbližších příkladů, které pak „hlasují“ o zařazení příkladu **x** do třídy.

Algoritmus k-NN

Učení

1. Pro každý příklad $[\mathbf{x}_i, y_i]$
   zařaď $[\mathbf{x}_i, y_i]$ do báze příkladů

Klasifikace

1. Pro nový příklad $\mathbf{x}$
   1.1. Najdi $\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_K$ K nejbližších příkladů z báze příkladů
   1.2. Přiřaď

$$y = y_j \Leftrightarrow \sum_{k=1}^{K} \delta(y_j, y_k) = \max_i \sum_{k=1}^{K} \delta(y_i, y_k) \quad ,$$

kde $\delta(y_i,y_k) = 1$ pro $y_i = y_k$, jinak $\delta(y_i,y_k) = 0$

Obrázek 43: Algoritmus K-nejbližších sousedů

Uvedený algoritmus předpokládá, že cílový atribut je kategoriální, jinými slovy, klasifikujeme příklady do konečného počtu tříd. V případě, že cílový atribut je numerický, počítáme místo nejčastější hodnoty cílového atributu hodnotu průměrnou:

$$\hat{y} = \frac{1}{K}\sum_{k=1}^{K} y_k.$$

V obou případech (diskrétní třídy i třídy spojité) má každý z *K* příkladů „rovný hlas“. To je sice demokratické, ale někdy neefektivní. Proto se používá vážené hlasování (resp. vážený průměr). Jeden příklad vážení příkladů je volit váhu příkladu $\boldsymbol{x}_i$ v bázi jako

$$w_i = \frac{1}{d(x,x_i)^2}.$$

kde $d(x, x_i)^2$ je vzdálenost obou příkladů.

Pokud budeme uvedeným algoritmem na základě trénovacích dat z našeho příkladu o úvěrech klasifikovat nového klienta s charakteristikami

Příjem = 15000
Konto = 32000

pak při použití eukleidovské vzdálenosti a pro parametr *K=1* bude závěr klasifikace *úvěr(ano)*, protože posuzovaný klient má nejblíže ke klientovi *k5* (Tabulka 12).

Tabulka 12:Klasifikace nového příkladu podle jednoho nejbližšího souseda

klient	příjem	konto	úvěr	vzdálenost příkladu
k1	3000	15000	ne	20808.65
k2	10000	15000	ne	17720.05
k3	17000	15000	ano	17117.24
k4	5000	30000	ne	10198.04
k5	15000	30000	ano	2000.00
k6	20000	50000	ano	18681.54
k7	2000	60000	ne	30870.70
k8	5000	90000	ano	58855.76
k9	10000	90000	ano	58215.12
k10	20000	90000	ano	58215.12
k11	10000	100000	ano	68183.58
k12	17000	100000	ano	68029.41

V případě velkého množství příkladů, což je situace typická při dobývání znalostí z databází, ale nelze uvažovat o uložení všech příkladů. Neobejdeme se tedy ve fázi učení bez generalizace. Jednotlivé třídy pak budou reprezentovány centroidy, tak, jak to bylo zmíněno v kapitole o statistických metodách. V nejjednodušším případě budou hodnoty atributů pro centroid reprezentující určitou třídu dán průměrnými hodnotami atributů pro příklady této třídy. Tímto způsobem získáme pro naše data dva centroidy:

C(ano): příjem = 14250, konto = 70625

C(ne): příjem = 5000, konto = 30000

Pro eukleidovskou vzdálenost se při klasifikaci trénovacích dat podle těchto centroidů dopustíme tří chyb (v Tabulka 13 vyznačeno tučně).

To ale neznamená, že neexistují centroidy použitelné pro bezchybnou klasifikaci našich dat. Vzhledem k tomu, že data jsou lineárně separabilní (viz kapitola o neuronových sítích), mohou jako centroidy posloužit každé dva body v prostou atributů, které budou osově souměrné podle rozdělující přímky[47]. Tedy například

C(ano): příjem = 15000, konto = 32000

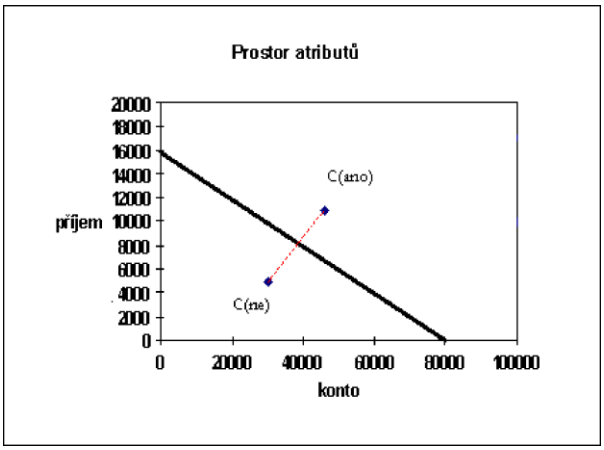
C(ne): příjem = 5000, konto = 30000

[47] Reprezentace dvou lineárně separabilních tříd pomocí jedné lineární diskriminační funkce je tedy ekvivalentní reprezentaci pomocí dvou centroidů symetrických podle této funkce. V případě tříd, které nejsou lineárně separabilní, je třeba jednu třídu reprezentovat více centroidy (viz kapitola o statistice).

Tabulka 13: Klasifikace trénovacích dat podle centroidů – průměrů

klient	příjem	konto	úvěr	vzdálenost od C(ano)	vzdálenost od C(ne)	výsledek klasifikace
k1	3000	15000	ne	56751.24	15132.75	ne
k2	10000	15000	ne	55787.12	15811.39	ne
k3	17000	15000	ano	55692.94	19209.37	**ne**
k4	5000	30000	ne	41664.77	0.00	ne
k5	15000	30000	ano	40631.92	10000.00	**ne**
k6	20000	50000	ano	21411.52	25000.00	ano
k7	2000	60000	ne	16215.83	30149.63	**ano**
k8	5000	90000	ano	21469.82	60000.00	ano
k9	10000	90000	ano	19835.65	60207.97	ano
k10	20000	90000	ano	20210.22	61846.58	ano
k11	10000	100000	ano	29680.85	70178.34	ano
k12	17000	100000	ano	29503.44	71021.12	ano

Polohu těchto centroidů v prostoru atributů ilustruje Obrázek 44. Výsledky klasifikace pomocí těchto centroidů pak ukazuje Tabulka 14.



Obrázek 44: Poloha centroidů

Tabulka 14: Klasifikace trénovacích dat podle centroidů – osově souměrných bodů

klient	příjem	konto	úvěr	vzdálenost od C(ano)	vzdálenost od C(ne)	výsledek klasifikace
k1	3000	15000	ne	20808.65	15132.75	ne
k2	10000	15000	ne	17720.05	15811.39	ne
k3	17000	15000	ano	17117.24	19209.37	ano
k4	5000	30000	ne	10198.04	0.00	ne
k5	15000	30000	ano	2000.00	10000.00	ano
k6	20000	50000	ano	18681.54	25000.00	ano
k7	2000	60000	ne	30870.70	30149.63	ne
k8	5000	90000	ano	58855.76	60000.00	ano
k9	10000	90000	ano	58215.12	60207.97	ano
k10	20000	90000	ano	58215.12	61846.58	ano
k11	10000	100000	ano	68183.58	70178.34	ano
k12	17000	100000	ano	68029.41	71021.12	ano

KONTROLNÍ OTÁZKA

1. Jakým způsobem se dělí trénovací data při použití metody TDIDT pro tvorbu rozhodovacích stromů?
2. Jaká je entropie po zvažovanou část dat, ve které jsou data obě třídy a) zastoupeny stejně a b) je zastoupena pouze jedna ze tříd (předpokládáme 2 třídy v celých trénovacích datech)?
3. V souvislosti s rozhodovacími pravidly, jaký je postup *zdola nahoru*, který odpovídá algoritmům AQ a jak se liší od postupu využívaného např. v algoritmu TDIDT?
4. Co znamená *generalizace* pravidla?
5. Vezmeme-li v potaz adaptivní lineární neuron (Adaline) pro dva vstupní podněty x_1 a x_2, jakého tvaru nabývá hranice, která odděluje vstupní podněty odpovídající výstupu neuronu rovnému 0 a rovnému 1?
6. Chceme-li použít neuronové sítě pro data s kategoriálním atributem Země s možnými hodnotami USA, GB a ČR, jakým způsobem je potřeba data upravit?
7. Jaký je princip genetických algoritmů?
8. Jmenujte tři základní prostředky pro tvorbu nové generace jedinců.
9. Jak zní Bayesovo pravidlo?
10. Z jakého předpokladu vychází Naivní bayesovský klasifikátor?
11. Jak probíhá klasifikace nového příkladu v metodě *K*-nejbližších sousedů?
12. Jak probíhá učení pro metodu *K*-nejbližších sousedů (v základní podobě)?

SHRNUTÍ KAPITOLY

V této kapitole jsme se věnovali vybraným metodám dolování dat, které patří mezi nejčastěji používané metody dolování dat a jejichž implementace lze najít ve většině matematických a statistických softwarových systémech. Seznámili jsme se s metodou pro tvorbu rozhodovacích stromů, rozhodovacích pravidel, metodou založené na umělých neuronových sítích, metodou SVM, dále pak s evolučními algoritmy, metodou klasifikace založenou na Bayesově pravidle a taktéž s metodami založenými na analogii.

ODPOVĚDI

1. Na základě zvoleného kritéria (např. entropie nebo informační zisk) se vybere jeden ze vstupních atributů, podle něj se data rozdělí do tolika tříd, kolik má tento atribut možných hodnot a proces se znovu opakuje, dokud nejsou v dané části dat jen objekty z jedné třídy.

2. a) H = 1 b) H = 0. Všimněte si, že pomocí algoritmu TDIDT se snažíme dosáhnout v co nejmenším počtu kroku entropii rovnou nule (tedy aby v části dat byly jen objekty z jedné třídy).

3. Počáteční hypotéza pokrývající jeden pozitivní příklad (pravidlo je vytvořeno z jednoho řádku trénovacích dat), se postupně zobecňuje tak, aby pokrývala více pozitivních příkladů a žádný negativní příklad.

4. Z předpokladové části pravidla se vyjme část předpokladů, a to tak, aby pravidlo stále odpovídalo trénovacím datům.

5. Přímka.

6. Pomocí *binarizace* vytvoříme tři binární atributy Země_USA, Země_GB, Země_ČR nabývajících pouze hodnot 0 a 1.

7. Uchovat v populaci pouze ty jedince, kteří jsou nejlepší vzhledem ke zvolenému kritériu.

8. Selekce, křížení a mutace.

9. $P(H|E) = \frac{P(E|H)P(H)}{P(E)}$.

10. Naivní bayesovský klasifikátor vychází z předpokladu, že jednotlivé evidence $E_1,\ldots,E_K$ jsou podmíněně nezávislé při platnosti hypotézy H.

11. Nový příklad se zařadí do té třídy, která převažuje v jeho nejbližších K sousedech.

12. Všechna trénovací data se pouze vloží do báze příkladů, tzn. neprobíhá žádná optimalizace např. jako u neuronových sítí. V praxi se nicméně experimentálně určuje nejvhodnější hodnota počtu sousedů K.

6 VYHODNOCENÍ VÝSLEDKŮ

RYCHLÝ NÁHLED KAPITOLY

Vzhledem k množství existujících metod dolování dat je možno pro každá data získat značné množství různých modelů těchto dat. Pro výběr toho nejvhodnějšího modelu, který bychom poté nasadili do praxe, je potřeba tyto modely nějakým způsobem ohodnotit. Tato kapitola se tedy věnuje různými možnostmi hodnocení modelů získaných dolováním dat. Tato kapitola taktéž představuje několik možností, jak získané modely dát vizualizovat a také způsobem, jak je možno různé modely zkombinovat tak, abychom získali model překonávající všechny jednotlivé modely, ze kterých je vytvořen.

CÍLE KAPITOLY

Cílem této kapitoly je seznámit čtenáře s možnostmi hodnocení modelů, které jsme schopni získávat z dat pomocí metod představených v předcházející kapitole. Čtenář se seznámí s několika často používanými kritérii pro hodnocení kvality získaných modelů, jako jsou například celková správnost, správnost pro jednotlivé třídy nebo přesnost a úplnost. Taktéž čtenář získá vzhled do možností vizualizace získaných modelů. Nakonec kapitoly se čtenář dozví o možnostech volby nejvhodnějšího algoritmu pro dolovaní daných dat.

KLÍČOVÁ SLOVA KAPITOLY

správnost, přesnost, úplnost, křivka navýšení, křivka ROC

Vzhledem k množství existujících metod dolování dat je možno pro každá data získat značné množství různých modelů těchto dat. Pro výběr toho nejvhodnějšího modelu, který bychom poté nasadili do praxe, je potřeba tyto modely nějakým způsobem ohodnotit. Následující kapitola se tedy věnuje různými možnostmi hodnocení modelů získaných dolováním dat. Tato kapitola taktéž představuje několik možností, jak získané modely dat vizualizovat.

Důležitým krokem v celém procesu dobývání znalostí je interpretace a evaluace nalezených znalostí. V případě deskriptivních úloh je hlavním kritériem novost, zajímavost, užitečnost a srozumitelnost. Tyto charakteristiky úzce souvisejí s danou aplikační oblastí, s tím, co přinášejí expertům a koncovým uživatelům. Z tohoto pohledu můžeme hovořit o

- zřejmých znalostech, které jsou ve shodě se „zdravým selským rozumem“

příkladem může být pravidlo, že pokud měl pacient problémy v těhotenství, tak se jednalo o ženu; takovéto znalosti mohou v expertovi vzbudit pochybnost o smysluplnosti dobývání znalostí – odborníkovi na KDD ale potvrzují, že použitý algoritmus funguje tak jak má,

- zřejmých znalostech, které jsou ve shodě se znalostmi experta z dané oblasti

příkladem může být pravidlo, že pokud se účet klienta banky pohybuje v záporném zůstatku, má tento klient problémy se splácením úvěru; takovéto znalosti byť nepřinášejí nic nového, ukazují expertovi, že použitá metoda je schopna nalézat v datech znalosti,

- nových, zajímavých znalostech, které přinášejí nový pohled

toto jsou ideální znalosti, které expert hledá,

- znalostech, které musí expert podrobit bližší analýze, neboť není zcela jasné co znamenají i tyto znalosti mohou být pro experta přínosem

- „znalosti“, které jsou v rozporu se znalostmi experta

takováto pravidla, zachycující nejspíše nějaké nahodilé koincidence, expert patrně vyloučí; ovšem pozor, kdoví, jestli se naopak nejedná o zásadní nový pohled na celou oblast.

Hodnocení deskriptivních znalostí nezávisle na aplikační oblasti se opírá především o různé numerické parametry[48]. Zde je ale třeba zdůraznit, že ne vše, co je v datech přesvědčivě prokázáno, má pro experta význam (viz výše uvedené členění).

Pomocí pro hodnocení znalostí ve smyslu *porozumění* znalostem jsou pak různé vizualizační metody letmo zmíněné v příslušné podkapitole. V této části se tedy budeme věnovat především vyhodnocením znalostí pro úlohy typu klasifikace a predikce.

6.1 Testování modelů

Při hledání znalostí pro potřeby klasifikace se obvykle postupuje metodou učení s učitelem. Vychází se tedy z toho, že jsou k dispozici příklady, o kterých víme, do které třídy patří. Metody evaluace jsou pak založeny na testování nalezených znalostí na datech, na možnosti porovnat, jak dobře se nalezené znalosti shodují s informací od učitele. Pro testování se nabízí celá řada variant podle toho, jaká data použijeme pro učení a jaká pro testování:

- testování v celých trénovacích datech

- křížová validace (cross-validation)

[48] V případě asociačních pravidel nás zajímá např. spolehlivost a podpora.

- leave-one-out
- bootstrap
- testování na testovacích datech.

Testování v datech použitých pro učení (celá trénovací data) má nejmenší vypovídací schopnost o tom, jak budou nalezené znalosti použitelné pro klasifikování nových případů. Často totiž může dojít k „přeučení“ (overfitting), kdy nalezené znalosti vystihují spíše náhodné charakteristiky trénovacích dat a neodhalí to podstatné, co lze použít pro generalizaci. Trénovací data tedy nejsou příliš vhodná pro testování nalezených znalostí[49], a proto se obvykle používají data jiná. Otázkou je, jak taková data získat. Jeden problém může být, že dat je k dispozici málo, jiný problém je, že je žádoucí, aby data použitá pro testování se „podobala“ datům trénovacím[50]. Oba problémy umožní řešit různé způsoby výběru trénovacích a testovacích dat.

Při testování metodou *křížová validace* se data dopředu rozdělí např. na 10 částí tak, že vždy jedna desetina se vyjme pro testování a zbylých devět desetin se použije pro učení. Celý tento postup se zopakuje desetkrát a výsledek testování se zprůměruje. Proto se tomuto způsobu testování říká desetinásobná křížová validace (*10 fold cross-validation*).

Variantou tohoto přístupu je metoda *leave-one-out.* Z dat, která jsou k dispozici, se vyjme jeden příklad pro testování a zbylá data se použijí pro učení. Toto se opakuje tolikrát, kolik příkladů máme. Vznikne tedy n souborů znalostí, které se otestují na *n* příkladech[51]. Výsledek testování dává odhad, jak by se znalosti získané ze všech dostupných *n* příkladů chovaly při klasifikování příkladů neznámých.

Při metodě zvané *bootstrap* se příklady vybrané pro učení mohou opakovat. Na rozdíl od křížové validace, kdy jeden příklad se použije buď pro učení, nebo pro testování, zde se tentýž příklad může pro učení vybrat několikrát. Máme-li opět k dispozici *n* příkladů, *n*-krát provedeme výběr s navracením, abychom získali *n* příkladů pro učení. Pravděpodobnost, že příklad bude vybrán je 1/n, pravděpodobnost že vybrán nebude je 1-1/n. Při *n* opakováních je pravděpodobnost, že příklad vybrán nebude

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n} = e^{-1} = 0.368$$

[49] Výjimku z tohoto pravidla představuje metoda pesimistického odhadu na základě trénovacích dat, kterou používá Quinlan ve svém C4.5. Pro každý list stromu počítá počet všech pokrytých příkladů (analogie s hodnotou *a+b* ze čtyřponí tabulky) a počet chybně pokrytých příkladů (analogie s hodnotou *b*). Výskyt chybné klasifikace pak chápe jako náhodný jev (pozorovaný na vzorku příkladů pokrytých listem), který se řídí binomickým rozdělením. Pro zvolenou hladinu významnosti *CF* pak spočítá horní (pesimistický) odhad pravděpodobnosti výskytu chybné klasifikace *UCF(a,b*). Odhad počtu chybných klasifikací v uvažovaném listu je pak UCF(a,b) × (a+b). Součet těchto počtů přes všechny listy je pak odhadem počtu chyb celého stromu.

[50] 3 V tom smyslu, že rozdělení příkladů do tříd v testovacích datech se řídí stejnými zákonitostmi jako rozdělení příkladů do tříd v datech trénovacích, jinými slovy, že je stejný podíl tříd v obou vzorcích.

[51] Jde tedy vlastně o n-fold cross-validation.

Pro rozumně velká data se tedy vybere zhruba 63,2% příkladů pro učení a 36.8% příkladů pro testování. V trénovací množině se budou příklady opakovat, to, co zbude, se použije při testování.

Tomuto poměru zhruba odpovídá *náhodný výběr* 75% příkladů pro učení a 25% příkladů pro testování[52]. Na rozdíl od bootstrapu se opět každý příklad použije jen jednou; pro učení nebo pro testování. Na rozdíl od křížové validace se učení i testování provede jen jednou.

K ZAPAMATOVÁNÍ

Ať tak či onak, cílem testování je určit, v kolika případech se klasifikátor shoduje s učitelem a v kolika případech se dopustil chyb. Tyto údaje bývá zvykem zachycovat v tzv. *matici záměn* (confusion matrix), viz Tabulka 15. V matici jsou ve sloupcích uvedeny informace o tom, jak postupoval při klasifikaci systém využívající nalezené znalosti, v řádcích informace o tom, jak to má (alespoň jak učitel říká) být. Tabulka 15 zachycuje situaci, kdy jde o klasifikaci do dvou tříd, „+“ a „-“[53]. *TP* (správně positivní, true positive) je počet příkladů, které systém správně zařadil do třídy „+“, *FP* (falešně positivní, false positive) je počet příkladů, které systém chybně zařadil do třídy „+“ (patří do třídy „-“) *TN* (správně negativní, true negative) je počet příkladů, které systém správně zařadil do třídy „-“, a *FN* (falešně negativní, false negative) je počet příkladů, které systém nesprávně zařadil do třídy „-“ (patří do třídy „+“).

Tabulka 15: Matice záměn

	Klasifikace systémem	
Správné zařazení	+	-
+	TP	FN
-	FP	TN

Matice záměn sleduje pouze počty správně a nesprávně zařazených příkladů. V řadě případů může být důležitý i typ chyby, kterého se systém dopustil (*FP* vs. *FN*). Jestliže se systém, který hodnotí bonitu klientů banky za účelem rozhodnutí o úvěru dopustí chyby, nastane buď situace, že doporučí půjčku klientovi, který nesplatí, nebo situace, že zamítne půjčku klientovi, který by ji splatil. V prvním případě tak banka prodělá, ve druhém případě banka nevydělá. První chyba je z hlediska banky jistě závažnější[54]. Skutečnost, že různé chyby jsou různě závažné, lze do testování (i učení) zahrnout pomocí tzv. matice cen (cost matrix). Zde se uvede, jaká je cena za různé typy rozhodnutí; čím horší chyba, tím vyšší cena[55]. Při hodnocení znalostí se tedy nemusí

[52] Jedná se o empiricky doporučený poměr trénovacích a testovacích dat.
[53] V případě více tříd je vhodnější zvětšit počet sloupců i řádků tak, aby odpovídal počtu tříd. Často nás totiž nezajímá pouze prosté zjištění, že systém udělal chybu ale i to, kterou chybu udělal.
[54] Ve statistice se v podobných situacích používá pojem chyba prvního druhu a chyba druhého druhu.
[55] Správné rozhodnutí (TP, TN) má obvykle cenu 0.

brát do úvahy prostý počet chybných rozhodnutí, ale i cena těchto chyb (ztráta, způsobená uživateli).

6.1.1 CELKOVÁ SPRÁVNOST

Celková správnost (overall accuracy) – resp. úspěšnost (succesfulness), nebo komplementární celková chyba (overall error) jsou nejjednodušší charakteristiky toho, jak jsou získané znalosti kvalitní. Celková správnost se spočítá jako relativní počet správných rozhodnutí systému

$$\mathrm{Acc} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

celková chyba se spočítá jako relativní počet chybných rozhodnutí systému

$$\mathrm{Err} = \frac{\mathrm{FP} + \mathrm{FN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Celková správnost by měla být v intervalu $[\mathrm{Acc}_{\mathrm{def}}, \mathrm{Acc}_{\mathrm{max}}]$, kde Acc_{def} je správnost systému, který všechny příklady přiřadí k majoritní třídě[56], a Acc_{max} je maximální správnost dosažitelná pro daná data[57].

V případě, že nás při testování zajímá jen počet správných resp. nesprávných rozhodnutí systému, je

$$\mathrm{Err} = 1 - \mathrm{Acc}.$$

Pokud bereme do úvahy i matici cen, výše uvedená rovnost neplatí a je vhodnější použít jako kritérium celkovou chybu spočítanou tak, že se počty chybných rozhodnutí (*FP*, *FN*) vynásobí cenou za příslušný typ chyby:

$$\mathrm{Err} = \mathrm{FP} * c(P,n) + \mathrm{FN} * c(N,p),$$

kde *c(P,n)* je cena zařazení negativního příkladu do třídy „+“ a *c(N,p)* je cena zařazení pozitivního příkladu do třídy „-“.

6.1.2 SPRÁVNOST PRO JEDNOTLIVÉ TŘÍDY

V případě, že třídy jsou v datech rozloženy výrazně nerovnoměrně (např. pouze 5% klientů banky je podezřelých, zbylých 95% je v pořádku), bude celková správnost dávat zkreslený obraz o nalezených znalostech. 95% celková správnost může znamenat, že jsme nerozpoznali žádného podezřelého klienta (95% je implicitní správnost), nebo že jsme rozpoznali všechny podezřelé

[56] Znalosti tohoto systému jsou např. tvořeny pouze implicitním (default) pravidlem.
[57] V případě, že v datech nejsou kontradikce (příklady se stejnými hodnotami vstupních atributů, které se liší v přiřazení ke třídě), je tato správnost 1 (100%). V případě kontradikcí je tato správnost nižší neboť systém zařadí všechny stejně popsané případy to téže třídy.

klienty (a dopustili se chyb při rozpoznávání klientů spolehlivých). V takové situaci je vhodnější sledovat správnost (resp. chybu) pro jednotlivé třídy:

$$\mathrm{Acc}_{+} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$\mathrm{Acc}_{-} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}}$$

6.1.3 Přesnost a úplnost

Přesnost a *úplnost* (precision and recall) jsou pojmy používané v oblasti vyhledávání informací. Hledáme-li např. na webu nějaké dokumenty týkající se určitého tématu (třeba pomocí vyhledávače jako je google), pak :

1. ne všechny nalezené dokumenty se týkají tématu,
2. určitě jsme nenalezli vše co je o tématu k dispozici.

Přesnost nám říká, kolik nalezených dokumentů se skutečně týká daného tématu a úplnost nám říká, kolik dokumentů týkajících se tématu jsme nalezli. Tyto míry shody lze použít i pro hodnocení znalostí:

$$\text{Přesnost} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

$$\text{Úplnost} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

6.1.4 Sensitivita a specificita

Sensitivita a *specificita* (sensitivity a specificity) jsou charakteristiky převzaté z medicíny. V případě nasazení nějakého nového léku nás zajímá, u kolika nemocných pacientů lék zabere (sensitivita), a zda lék zabírá pouze na danou chorobu (specificita). Z matice záměn se tyto hodnoty spočítají jako:

$$\text{Sensitivita} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\text{Specificita} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

A opět můžeme pozorovat shodu používaných kritérií; sensitivita je totéž co úplnost. Sensitivita a specificita se buď uvádějí samostatně, nebo jako vzájemný součin obou čísel.

Doposud jsme hodnotili znalosti na základě jednoho testování; pomocí jednoho čísla[58]. Komplexnější popis chování modelu nám mohou dát následující analýzy. Při hodnocení na základě více testů může jít opět o hodnocení jednoho modelu (křivka učení, lift chart) nebo o hodnocení více modelů[59] (ROC, DEA). Všechny tyto metody používají grafické znázornění výsledků testování.

6.1.5 KŘIVKA UČENÍ

Křivka učení (learning curve) dává do souvislosti počet příkladů v trénovací množině a správnost klasifikace (Obrázek 45). Vychází se z předpokladu, že čím více příkladů je k dispozici ve fázi učení, tím budou nalezené znalosti přesnější. Pro různé počty příkladů tak dostaneme různé hodnoty správnosti při testování (Tabulka 16). Testování se často provádí opakovaně, takže kromě průměrné hodnoty správnosti $\overline{\mathrm{Acc}}$ získáme ještě její chybu (třetí sloupec v Tabulka 16). Tato chyba se obvykle počítá jako

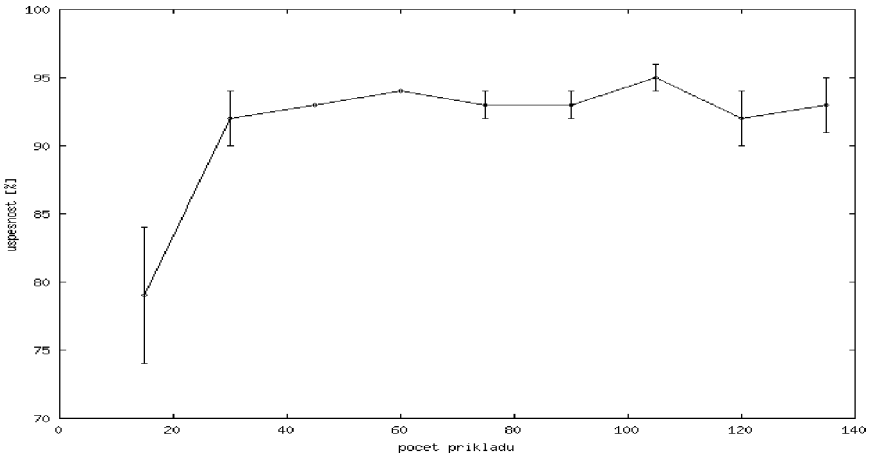
$$\frac{S_{Acc}}{\sqrt{n}}, \quad \text{kde} \quad S_{Acc}^2 = \frac{\sum_i (Acc_i - \overline{Acc})^2}{n-1}.$$

Tabulka 16: Data pro křivku učení

Počet příkladů (n)	Prům. správnost (Acc_i)	chyba
15	79.56%	+- 5.69%
30	92.00%	+- 2.07%
45	93.14%	+- 0.70%
60	94.67%	+- 0.42%
75	93.07%	+- 1.54%
90	93.00%	+- 1.62%
105	95.11%	+- 1.30%
120	92.00%	+- 2.26%
135	93.33%	+- 2.98%

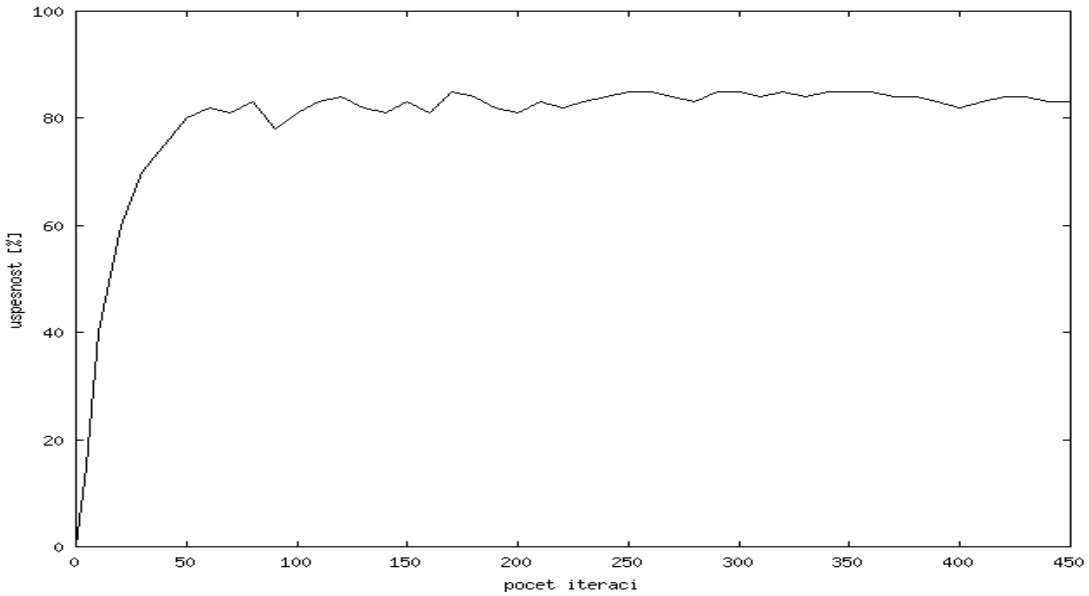
[58] V případě např. desetinásobné křížové validace jednomu číslu odpovídá průměrná hodnota z výsledků na deseti maticích záměn.

[59] Zvolený učící se algoritmus může mít celou řadu parametrů, které ovlivňují nalezené znalosti. Nejmarkantněji je to vidět u neuronových sítí, ale můžeme to pozorovat i u rozhodovacích stromů a pravidel



Obrázek 45: Křivka učení

Jistou analogií tohoto přístupu je křivka učení používaná při učení neuronových sítí jako indikace toho, kdy je možno učení ukončit. Zde se jedná o závislost správnosti na počtu iterací (průchodů týmiž trénovacími daty) – viz Obrázek 46.



Obrázek 46: Závislost správnosti na počtu iterací

6.1.6 KŘIVKA NAVÝŠENÍ

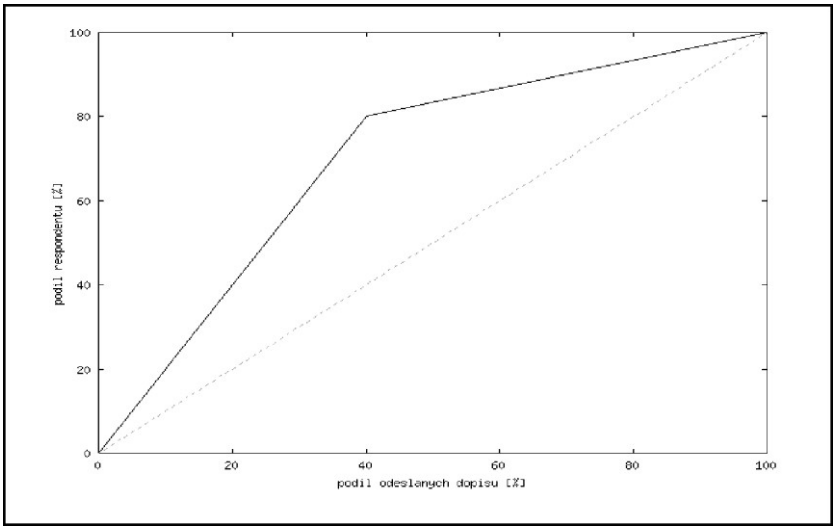
Křivka navýšení (lift curve) se často používá v marketingu. Vezměme si situaci, kdy je třeba poslat klientům nabídku nějakého produktu. Ze zkušenosti víme, že na takovou nabídku odpoví velice málo (řekněme 1%) oslovených zákazníků. To znamená, že většina dopisů s nabídkou je odeslána zbytečně. Při tvorbě modelu bychom tuto skutečnost chtěli vzít v úvahu. To umožní

křivka navýšení, která dává do souvislosti podíl respondentů, kteří odpověděli (*TP*) s podílem odeslaných dopisů (*TP+TN+ FP+FN*). Jak takovou křivku vytvořit? Lze to pouze u modelů, které neprovádějí pouze binární klasifikaci (ano/ne) ale klasifikace je doprovázena numerickou hodnotou, která vyjadřuje, jak moc si klasifikátor věří při svém rozhodnutí pro daný příklad (pravděpodobnost, váha). Lze tedy křivku navýšení vytvořit např. pro neuronové sítě nebo bayesovské klasifikátory.

Nejprve seřadíme příklady použité při testování sestupně podle váhy klasifikace. U každého příkladu máme samozřejmě k dispozici informaci o tom, do které třídy patří (Tabulka 17). Pak pro libovolný úsek setříděných příkladů počínaje příkladem klasifikovaným s nejvyšší vahou vytvoříme dílčí matici záměn s hodnotami *TP'*, *TN'*, *FP'* a *FN'*. Hodnoty *TP'/TP* a *(TP' + TN' + FP' + FN')/(TP +TN + FP + FN)*[60] pak vyneseme do grafu. Křivka navýšení vždy prochází bodem [0,0] (nepošleme-li žádný dopis, nebude žádná odpověď) a [1,1] (pošleme-li dopisy všem, zachytíme všechny respondenty). Model bude tím lepší, čím bude křivka navýšení ležet nad diagonálou reprezentující náhodný výběr. Z křivky navýšení uvedené na Obrázek 47 lze vyčíst, že pošleme-li nabídku 40% nejperspektivnějším klientům, oslovíme 80% všech respondentů. Dosáhli jsme tedy dvojnásobného navýšení odezvy klientů ve srovnání s náhodným výběrem 40% adresátů.

Tabulka 17: Data pro křivku navýšení

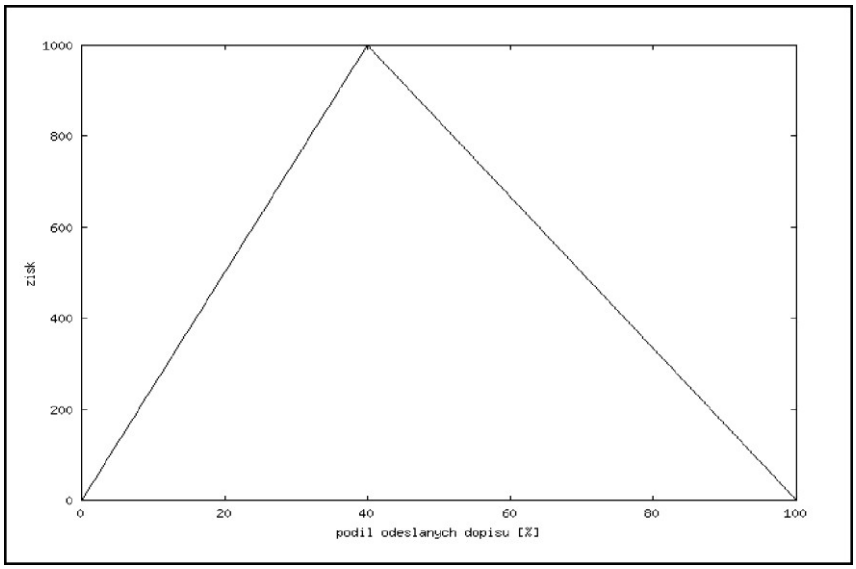
Pořadí	skutečná třída	predikce pro "+"
1	+	0,95
2	+	0,93
3	-	0,93
4	+	0,91
5	+	0,88
...	...	.....



Obrázek 47: Křivka navýšení

[60] Hodnoty TP, TN, FP a FN odpovídají celým testovacím datům.

V případě, že máme k dispozici informace o cenách za odeslání nabídky i o cenách nabízeného produktu, můžeme křivku navýšení převést na křivku návratnosti investic (ROI). Z této křivky je přímo vidět jak se mění náš zisk v závislosti na počtu oslovených klientů (Obrázek 48)[61].



Obrázek 48: Křivka návratnosti investic

6.1.7 KŘIVKA ROC

Křivka ROC (ROC curve) byla převzata pro evaluaci modelů z oblasti radiotechniky (ROC znamená receiver operating characteristic, tedy pracovní charakteristika přijímače) [Provost, Fawcett, 1997]. Tato křivka dává do souvislosti podíl *TP* s podílem *FP*:

$$TP_{\%} = \frac{TP}{TP + FN}$$

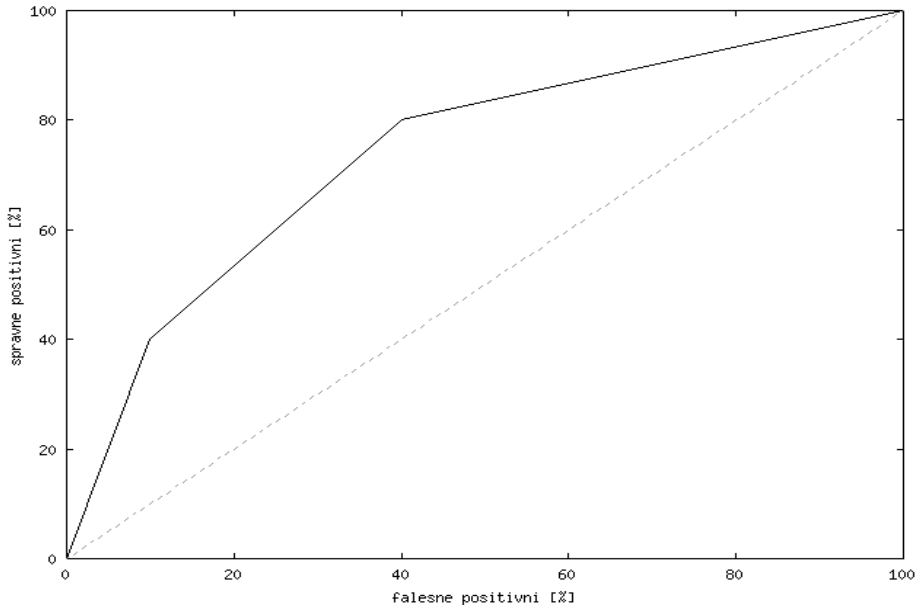
$$FP_{\%} = \frac{FP}{FP + TN}$$

Uvedené charakteristiky již známe; $TP_{\%}$ = Sensitivita, $FP_{\%}$ = 1 – Specificita. Používá se tedy i kritérium $TP_{\%} * (1 - FP_{\%})$.

ROC křivku lze použít u modelů, které klasifikaci doprovázejí váhou resp. pravděpodobností. Křivku vytvoříme tak, že budeme měnit práh, při kterém bude výsledek klasifikace interpretován jako třída '+'[62]. Bodu [0,0] (žádná predikce třídy +) odpovídá práh 1, bodu [1,1] (všechny příklady zařazeny do třídy '+') odpovídá práh 0. Změnou prahu lze simulovat chování modelu v případě změny poměru mezi počty příkladů obou tříd i změny cen za chybnou klasifikaci, ROC křivka tedy dává obraz o chování klasifikátoru bez ohledu na rozdělení tříd a na cenu chyb.

[61] Celý příklad je převzat z demo dat k systému Clementine.
[62] ROC křivku můžeme tedy vytvořit podobně jako křivku navýšení; na základě volby vzorku ze seznamu příkladů uspořádaného podle výsledků klasifikace (predikce '+').

Poznamenejme, že je žádoucí se v grafu pohybovat „vlevo nahoře“, tedy blízko bodu [0,1], který odpovídá bezchybné klasifikaci.



Obrázek 49: ROC křivka

6.1.8 NUMERICKÉ PREDIKCE

Zatím jsme se zabývali situací, kdy cílem modelu je klasifikovat příklady do tříd. V případě numerické predikce19 se jako kritérium správnosti používají např. střední kvadratická chyba (mean-squared error, MSE), odmocnina ze střední kvadratické chyby (root mean-squared error, RMSE), střední absolutní chyba (mean absolute error, MAE), relativní kvadratická chyba (relative squared error, RSE), nebo korelační koeficient.

$$MSE = \frac{(p_1-s_1)^2 + \ldots + (p_n-s_n)^2}{n}$$

$$RMSE = \sqrt{\frac{(p_1-s_1)^2 + \ldots + (p_n-s_n)^2}{n}}$$

$$MAE = \frac{|p_1-s_1| + \ldots + |p_n-s_n|}{n}$$

$$RSE = \frac{(p_1 - s_1)^2 + \ldots + (p_n - s_n)^2}{(s_1 - \bar{s})^2 + \ldots + (s_n - \bar{s})^2}, \qquad \text{kde } \bar{s} = \frac{1}{n}\sum_i s_i$$

$$\rho = \frac{S_{ps}}{\sqrt{S_p^2 S_s^2}}, \qquad \text{kde } S_{ps} = \frac{\sum_i (p_i - \bar{p})(s_i - \bar{s})}{n-1}, \quad S_p^2 = \frac{\sum_i (p_i - \bar{p})^2}{n-1}, \quad S_s^2 = \frac{\sum_i (s_i - \bar{s})^2}{n-1}$$

Ve výše uvedených vzorcích je p_i predikovaná hodnota a s_i skutečná hodnota pro i-tý příklad ze souboru n příkladů tvořících trénovací data.

6.2 Vizualizace

Přestože vizualizace hraje důležitou roli především ve fázi porozumění datům resp. při interpretaci deskriptivních znalostí, můžeme se s ní setkat i při hodnocení modelů určených pro klasifikaci. Jako na jiných místech procesu dobývání znalostí, tak i zde jde o to umožnit expertovi lépe porozumět tomu, co se děje.

6.2.1 VIZUALIZACE MODELŮ

Většina systémů pro dobývání znalostí z databází klade velký důraz na vizualizaci. Ta se projevuje i při vizualizaci modelů. Příkladem může být různý způsob znázornění rozhodovacího stromu od nejjednoduššího textového (systém Weka) až po trojrozměrné "mrakodrapy" okolo kterých je možno kroužit jako v letovém simulátoru (systém MineSet). Podoba rozhodovacích stromů je uvedena v kapitole věnované jednotlivým systémům; zde tuto věc zmiňujeme jen pro úplnost.

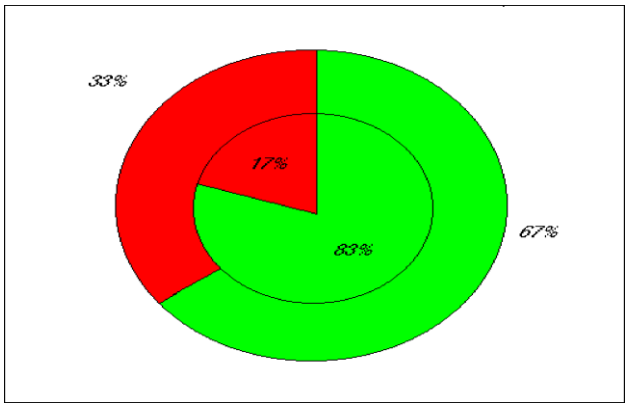
Vizualizovat můžeme např. jednotlivá pravidla. Obrázek 50 ukazuje koláčový graf odpovídající pravidlu

IF nezamestnany(ne) THEN uver(ano)

Hodnoty v grafu jsou převzaty z kontingenční tabulky, která má pro naše demonstrační data tuto podobu:

	uver(ano)	uver(ne)	
nezamestnany(ne)	5	1	6
nezamestnany(ano)	3	3	6
	8	4	12

Uvedený graf názorněji ukazuje, že platnost pravidla 5/6 je větší než relativní četnost třídy *úvěr(ano)* v datech (ta je rovna 8/12) a že tedy toto pravidlo dobře charakterizuje bonitní klienty.



Obrázek 50: Vizualizace jednoho pravidla

6.3 Volba nejvhodnějšího algoritmu

K ZAPAMATOVÁNÍ

Vzhledem k tomu, že neexistuje algoritmus, který by předčil ostatní na libovolných datech[63], dostává se do popředí otázka jak dopředu poznat, který algoritmus zvolit pro danou úlohu. Odpověď můžeme hledat na základě znalosti silných a slabých stránek jednotlivých algoritmů, nebo experimentálně.

Mezi známé charakteristiky algoritmů, které můžeme brát do úvahy, patří např.

- rozdíl mezi způsobem reprezentace příkladů (hodnoty atributů nebo relace),
- rozdíl mezi vyjadřovací silou jednotlivých algoritmů (rozhodovací stromy a pravidla rozdělují prostor atributů rovnoběžně s osami, neuronové sítě nebo diskriminační funkce naleznou i diagonální hranici mezi třídami),
- schopnost práce s numerickými atributy (některé algoritmy vyžadují pouze kategoriální atributy),
- schopnost práce se zašuměnými a chybějícími daty (vyhození vs. extrapolace),
- schopnost práce s maticí cen (ve fázi učení, ve fázi testování, vůbec ne),
- předpoklad nezávislosti mezi atributy (např. naivní Bayesovský klasifikátor),
- ostrá vs. neostrá klasifikace (jen indikátor třídy nebo i pravděpodobnost či váha klasifikace).

KONTROLNÍ OTÁZKA

1. Co je to *křížová validace*?
2. Co jsou to *falešně pozitivní* příklady?
3. Jaký je rozdíl mezi kritérii *Úplnost* a *Sensitivita*?

SHRNUTÍ KAPITOLY

[63] Tato skutečnost je známa pod názvem no free lunch teorem.

V této kapitole jsme věnovali různým možnostem hodnocení modelů získaných dolováním dat a představili jsme si také možnost, jak získané modely dat vizualizovat.

ODPOVĚDI

1. Při křížové validaci se data dopředu rozdělí např. na 10 částí tak, že vždy jedna desetina se vyjme pro testování a zbylých devět desetin se použije pro učení. Celý tento postup se zopakuje desetkrát a výsledek testování se zprůměruje.
2. Jsou to příklady z negativní třídy, které však model nesprávně řadí do třídy pozitivní.
3. Žádný.

LITERATURA

[Aamodt a kol., 1998] Aamodt,A. – Sandtorv,H.A. – Winnem,O.M.: Combining case based reasoning and data mining – a way of revealing and reusing RAMS experience. In: Proc. Safety and Reliability ESREL'98, 1998, 1345-1351.

[Aamodt, Plaza, 1994] Aamodt,A. – Plaza,E.: Case-based reasoning: foundational issues, methodological variations and system approaches. AI Communications, 7(1), 1994, 39-59.

[Aha a kol., 1991] Aha,D.W. - Kibler,D., - Albert, M. K.: Instance-based learning algorithms. Machine Learning, 6, 1991, 37-66.

[Anand a kol, 1996] Anand, S. a kol.: Towards Real-World Data Mining. In: Practical Aspects of Knowledge Management, Schweizer Informatiker Gesellschaft, Basel, 1996.

[Anděl, 1978] Anděl,J.: Matematická statistika, Praha, SNTL/ALFA 1978.

[Bauer, Kohavi, 1999] Bauer,E., and Kohavi,R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. Machine Learning, 36(1/2):105-139, 1999.

[Berka a kol., 1991] Berka,P. – Ivánek,J. - Jirků,P. - Stejskal,B.: Knowledge Acquisition from Data - a Tool for Regional Planning. In: UNIDO/CSFR Workshop of Interactive Decision Support Systems to Industrial and Territorial Planning, Bratislava, 1991

[Berka, 1993] Berka,P.: Discretization of numerical attributes for Knowledge EXplorer. Výzkumná zpráva, Praha, LISP-93-03, 1993, 11s.

[Berka, 1993] Berka,P.: Vybrané znalostní systémy, SAK, SAZE, KEX. Skripta VŠE, Praha 1993, 245s.

[Berka, 1993] Berka,P.: Vybrané znalostní systémy, SAK, SAZE, KEX. Skripta VŠE, Praha 1993.

[Berka, 1993b] Berka, P.: Knowledge EXplorer. A tool for automated knowledge acquisition from data. Tech. Report, Austrian Research Institute for AI, Vienna, TR-93-03, 1993, 23s.

[Berka, 1993c] Berka,P.: Discretization of numerical attributes for Knowledge EXplorer. Výzkumná zpráva, Praha, LISP-93-03, 1993, 11s.

[Berka, 1993d] Berka,P.: A comparison of three different methods for acquiring knowledge about virological hepatitis tests. Tech. Report, Austrian Research Institute for AI, Vienna, TR-93-10, 1993, 29s.

[Berka, 1997] Berka,P.: Recognizing Reliability of Discovered Knowledge. In: (Komorowski, Zytkow eds.) Proc. Principles of Data Mining and Knowledge Discovery PKDD'97, LNAI 1263, Springer, 1997, 307-314

[Berka, 1997] Berka,P.: Towards knowledge integration via knowledge revision: The Knowledge Explorer approach. In: (Nakhaeizadeh, Bruha, Taylor eds.) Proc. ECML'97 Workshop Dynamically Changing Domains: Theory Revision and Context Dependence Issues. Prague, 1997, 1-8.

[Berka, 1999] Berka,P.: Guide to Financial Data Set. In: Workshop notes on Discovery Challenge, PKDD'99, 1999.

[Berka, 2003] Berka, P.: Dobývání znalostí z databází, Academia, 2003.

[Berka, Bruha, 1998] Berka,P. - Bruha,I.: Empirical comparison of various discretization procedures. Int. J. of Pattern Recognition and Artificial Intelligence Vol. 12 No. 7 (1998) 1017-1032.

[Berka, Bruha, 1998a] Berka,P. - Bruha,I.: Empirical comparison of various discretization procedures. Int. J. of Pattern Recognition and Artificial Intelligence Vol. 12 No. 7 (1998) 1017-1032.

[Berka, Bruha, 1998b] Berka,P. - Bruha,I.: Discretization and Grouping: Preprocessing Steps for Data Mining. In: (Zytkow, Quafafou eds.) Proc. Principles of Data Mining and Knowledge Discovery PKDD'98, LNAI 1510, Springer, 1998, 239-245.

[Berka, Iványek, 1994] Berka,P. - Iványek,J.: Automated knowledge acquisition for PROSPECTOR-like expert systems. In. (Bergadano, de Raedt eds.) Proc. ECML'94, Springer 1994, 339-342.

[Berka, Pelikán, 1997] Berka,P. – Pelikán,E.: Data Mining Methods in Prediction. In: PASE'97, Computers in Finance and Economics. 1997, 115-128.

[Berka, Sláma, 1998] Berka,P. - Sláma,M.: Using neural nets to learn weights of rules for compositional expert systems. In: (Mira, Pobil, Ali eds.) Methodology and Tools in Knowledge-Based Systems. Proc. 11th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA-98-AIE, LNAI 1415, Springer, 1998, 511-519.

[Berndt, Clifford, 1996] Berndt,D. – Clifford,J.: Finding Patterns in Time Series: A Dynamic Programming Approach. In: Fayyad et al. Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996, 229-248.

[Biggs a kol., 1991] Biggs,D. – deVille,B. – Suen,E.: A method of choosing multiway partitions for classification and decision trees. Journal oj Applied Statistics, Vol. 18, No. 1, 1991, 49-62,

[Brazdil, Soares, 2000] Brazdil,P. – Soares,C.: A Comparison of Ranking Methods for Classification Algorithm Selection. In: (Mantaras, Plaza eds.) Proc. European Conf. On Machine Learning ECML2000, LNAI 1810, Springer 2000, 63-74.

[Brazdil, Torgo, 1990] Brazdil,P. – Torgo,L.: Knowledge acquisition via knowledge integration. In: Current Tredns in Knowledge Acquisition, IOS Press, 1990.

[Breiman a kol., 1984] Breiman,L. – Friedman,J.H. – Olshen,R.A. – Stone,P.J.: Classification and Regression Trees. Wadsworth, 1984.

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

[Bruha, 1996] Bruha,I.: Rule-induction in machine learning: Some latest enhancements and trends. In: Proc. Artificial Intelligence Techniques AIT'96, Brno, 1996.

[Bruha, Berka, 1997] Bruha,I. – Berka,P.: Knowledge acquisition of rules with continuous classes: Empirical comparison of two methods. In: (Kaylan, Lehman eds.) Proc. European Simulation Multiconference ESM'97, SCS, 1997, Vol.2, 42-46.

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

[Bruha, Kočková, 1994] Bruha,I. – Kočková,S.: A support for decision making: Cost-sensitive learning system. Artificial Intelligence in Medicine, 6 (1994), 67-82.

[Clark, Niblett, 1989] Clark,P. - Niblett,T.: The CN2 induction algorithm. Machine Learning, 3 (1989), 261- 283.

[Cortes, Vapnik, 1995] Cortes,C. – Vapnik,V.: Support vextor networks. Machine Learning, Vol 20, 1995, 273- 297.

[Cost, Salzberg, 1993] Cost,S. - Salzberg,S.: A weighted nearest neighbor algorithm for learning with symbolic features, Machine Learning, 10:1, (1993), 57-78.

[Daelemans a kol., 1996] Daelemans,W. - van den Bosch,A. – Weijters,T.: IGTree: Using trees for compression and classification in lazy learning algorithms. In: D. Aha (ed.) Artificial Intelligence Review, special issue on Lazy Learning, 1996.

[DeJong a kol., 1993] DeJong,K.A. – Spears,W.M. – Gordon,D.F.: Using genetic algorithms for concept learning. Machine Learning, 13, 1993, 161-188.

[Dempster a kol,1997] Dempster,A.P. – Laird,N.M. – Rubin,D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society, Series B, 39(1), 1997, 1-38.

[Dietterich, 1997] Dietterich T.: Machine-learning research: four current directions. AI Magazine, winter 1997, 97-136.

[Diettrich, 2000] Dietterich,T.G.: An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. Machine Learning, 40(2):139-158, 2000.

[Domingos, Pazzani, 1996] Domingos,P. – Pazzani,M.: Beyond independence: conditions for the optimality of the simple bayesian classifier. In: (Saitta ed.) Proc. 13th Int. Conference on Machine Learning ICML'96, 1996, 105-110.

[Duda, Gasching, 1979] Duda,R.O. - Gasching,J.E.: Model design in the Prospector consultant system for mineral exploration. in: Michie,D. (ed.), Expert Systems in the Micro Electronic Age, Edinburgh University Press, UK, 1979.

[Duda, Hart, 1973] Duda,R.O. – Hart,P.E.: Pattern classification and scene analysis. Wiley, 1973.

[Fayyad a kol, 1996] Fayyad,U. – Piatetsky-Shapiro,G. – Smyth,P. – Uthurusamy,R. eds.: Advances in Knowledge Discovery and Data Mining. AAAI Press/MIT Press, 1996, ISBN 0-262-56097-6.

[Fayyad, Irani, 1993] [Fayyad,U. - Irani,K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. 13th Joint Conf. of Artificial Intelligence IJCAI'93, 1993, 1022-1027.

[Fisher, 1987] Fisher,D.H.: Knowledge Acquisition Via Incremental Conceptual Clustering. Machine Learning, 2, 1987, 139-172.

[Friedman a kol., 1977] Friedman,J. – Bentley,J. – Finkel,A.R.: An algorithm for finding best matches in logaritmic expected time. ACM Transactions on Mathematical Software, 3(3), 1977.

[Friedman, 1991] Friedman,J.H.: Multivariate adaptive regression splines. Annals oj Statistics 19(1), 1991, 1- 141.

[Friedman,1998] Friedman,N.: The bayesian structural EM algorithm. In Proc. UAI'98, 1998.

[Gennari a kol., 1989] Gennari, J.H. – Langley,P. – Fisher,D.H.: Models of incremental concept formation. Artificial Intelligence, 40, 1989, 11-61.

[Goldberg, 1989] Goldberg,D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. AddisonWesley, 1989.

[Goodman, Smyth, 1989] Goodman,R.: Smyth,P. The induction of probabilistic rule sets – the ITRule algorithm. In: Proc. Int. Wshop on Machine Learning, Morgan Kaufman, 1989.

[Gunopulos, Das, 2000] Gunopulos,D. - Das,G.: Time Series Similarity Measures. KDD2000 tutorial, 2000.

[Hájek a kol.,1983] Hájek,P. – Havránek,T. – Chytil,M.K.: Metoda GUHA. Automatická tvorba hypotéz. Academia, 1983.

[Hájek, 1985] Hájek,P.: Combining functions for certainty factors in consulting systems. Int.J. Man-Machine Studies 22,1985, 59-76.

[Havránek, 1993] Havránek,T.: Statistika pro biologické a lékařské vědy. Academia, 1993, ISBN 80-200-0080- 1.

[Havránek, 1993] Havránek,T.: Statistika pro biologické a lékařské vědy. Academia, 1993. ISBN 80-200-0080- 1.

[Hebák, Hustopecký, 1987] Hebák,P. – Hustopecký,J.: Vícerozměrné statistické metody s aplikacemi. SNTL, 1987.

[Heckerman, 1995] Heckermann,D.: A tutorial on learning with Bayesian networks. Microsoft Research tech. Report MSR-TR-95-06.

[Hecht-Nielasen, 1991] Hecht-Nielsen,R.: Neurocomputing. Addison Wesley, 1991, ISBN 0-201-09355-3. [Kohonen, 1995] Kohonen,T.: Self-Organizing Maps, Springer, 1995.

[Holland, 1962] Holland,J.H.: Outline for a logical theory of adaptive systems. Journal oj the ACM, 3, 1962, 297-314.

[Holland, 1975] Holland,J.H.: Adaptation in natural and artificial systems. Univ. of Michigan Press, 1975.

[Holte, 1993] Very simple classification rules perform well on most commonly used datasets. Machine Learning, 11, 1993, 63-91.

[Chan, Stolfo, 1993] Chan,P.K. - Stolfo,S.J.: Experiments on Multistrategy Learning by Meta-Learning. In: Proc. 2nd Int. Conf. on Information and Knowledge Management, 1993.

[Chan, Stolfo, 1995] Chan,P.K. – Stolfo,S.J.: Learning arbiter and combiner trees from partitioned data for scaling machine learning. In: Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining. AAAI Press, 1995 39-44.

[Chapman a kol, 2000] Chapman,P. – Clinton,J. – Kerber,R. – Khabaza,T. – Reinartz,T. – Shearer,C. – Wirth,R.: CRISP-DM 1.0 Step-by-step data mining guide. SPSS Inc. 2000.

[Cheeseman, Stultz, 1996] Cheeseman, P. – Stultz,J.: Bayesian classification (AutoClass): Theory and results. In: (Fayyad, Piatetsky-Shapiro, Smyth, Uthurusamy, eds.) Advances in Knowledge Discovery and Data Mining, AAAI/MIT Press, 1996, ISBN 0-262-56097-6.

[Ivánek, Stejskal, 1988] Ivánek,J. - Stejskal,B.: Automatic acquisition of knowledge base from data without expert: ESOD (Expert System from Observational Data). In: Proc. COMPSTAT'88 Copenhagen, Physica-Verlag, 1988, 175-180.

[James a kol., 2013] James, G., Witten, D., Hastie, T., Tibshirani, R.: An introduction to statistical learning. Springer, 2013.

[Jensen, 1996] Jensen, F.: An introduction to bayesian networks. UCL Press, 1996.

[Jiroušek, 1995] Jiroušek,R.: Metody reprezentace a zpracování znalostí v umělé inteligemci. Skripta VŠE, 1995.

[John a kol, 1994] John,G. – Kohavi,R. – Pfleger,K.: Irrelevant features and the subset selection problem. In: Proc. 11th Int. Conf. on Machine Learning, Morgan Kaufmann, 1994, 121-129.

[Karalič, 1992] Karalič,A.: Employing linear regression in regression tree leaves. In: Proc. European Conf. on Artificial Intelligence ECAI'92, Vienna, 1992, 440-441.

[Keogh, Pazzani, 1999] Keogh,E. - Pazzani,M.: Scaling up Dynamic Time Warping to Massive Datasets. In: (Zytkow, Rauch, eds.) Proc. European Konf. On Priniples and Practice of KDD PKDD'99, Springer, 1999, 1- 11.

[Kietz a kol., 2000] Kietz,J.U. - Zücker,R. - Vaduva,A.: Mining Mart: Combining Case-Based-Reasoning and Multi-Strategy Learning into a Framework to reuse KDD-Application. In: Proc. 5th Int. Workshop on Multistrategy Learning MSL2000, 2000.

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

[Kohavi a kol.,1997] Kohavi,R. – Becker,B. – Sommerfeld,D.: Improving simple Bayes. In: (Sommeren, Widmer, eds.) Poster Papers of the 9th European Conf. on Machine Learning ECML'97. 1997.

[Kohavi, 1994] Kohavi,R.: MLC++. A Machine Learning Library in C++, Tech.Rep. CS229B, Stanford Univ. 1994.

[Kolodner, 1993] Kolodner, J.: Case-Based Reasoning. Morgan Kaufman, 1993.

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

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

[Koza, 1992] Koza,J.: Genetic programming: On the programming of computers by means of natural selection. MIT Press, 1992.

[Králík Brůha, 1998] Králík,P. – Brůha,I.: Genetic Learner: Attribute-Based Rule-Inducing Approach. In: Proc. 4th Int. Conf. On Soft Computing Mendel'98. Brno, 1998.

[Kubalík, 2000] Zlepšení funkčních vlastností genetických algoritmů. Disertační práce FEL ČVUT, 2000. [Mařík a kol, 2001] Mařík,V. – Štěpánková,O. - Lažanský,J. a kol.: Umělá inteligence 3. Academia, 2001. [Murray, 1994] Tuning neural networks with genetic algorithms. AI Expert, June 1994, 27-31.

[Lauritzen Spiegelhalter, 1988] Lauritzen,S. – Spiegelhalter,D.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society, Series B, 50, 1988, 157-224.

[Lauritzen, 1996] Lauritzen,S.: Graphical models. Oxford, 1996.

[Lavrac, 2000] Lavrac,N.: Decision Rules. Slajdy, http://www.cs.bris.ac.uk/Teaching/Resources/COMS70300/ slides/RulesHTML/, 2000.

[Lee, Shin, 1994] Lee,C. – Shin,D.: A context-sensitive discretization of numeric attributes for classification learning. In: (Cohn, ed.) Proc: 11th European Conf. on Artificial Intelligence ECAI'94, John Wiley, 1994, 428- 432.

[Liu, 1996] Liu,W.Z.: An integrated approach for different attribute types in nearest neighbour classification. The Knowledge Engineering Review, Vol. 11:3, 1996, 245-252.

[Manilla, Rokainen, 1997] Manilla,H. - Rokainen,P.: Similarity of Event Sequences. In: Proc. 4th Workshop on Temporal Representation and Reasoning TIME'97, 1997.

[Mantaras, 1991] Mantaras,R.L.: A distance-based attribute selection meassure for decision tree induction. Kluwer, 1991.

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

[Michalski, 1969] Michalski,R.S.: On the Quasi-minimal solution of the general covering problem. In: Proc. 5th Int. Symposium on Information Processing FCIP'69, Bled, Yugoslavia, 1969, 125-128.

[Michalski, 1978] Michalski,R.S.: A Planar Geometric Model for Presenting Multidimensional Discrete Spaces and Multiple-valued Logic Functions. Tech. Rep. UIUCDCS-R-78-897, Univ. of Illinois, 1978.

[Michie a kol., 1994] Michie,D. – Spiegelhalter,D.J. – Taylor,C.: Machine learning, neural and statistical classification. Ellis Horwood, 1994, ISBN 0-13-106360-X.

[Michie a kol., 1994] Michie,D. – Spiegelhalter,D.J. – Taylor.: Machine Learning, Neural and Statistical Classification. Ellis Horwood, 1994, ISBN 0-13-106360-X.

[Mitchell, 1996] Mitchell,M.: An Introduction to Genetic Algorithms. MIT Press, 1996.

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

[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

[Murphy, 2000] Murphy,K.: A brief introduction to graphical models and bayesian networks. http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html

[Murphy, Aha, 1994] Murphy,P. - Aha,D.: UCI Repository of Machine Learning Databases. University of California Irvine, Dept. Of Information and Computer Science, 1994.

[Novák a kol.,1992] Novák,M. - Faber,J. - Kufudaki,O.: Neuronové sítě a informační systémy živých organizmů. Grada, Praha, 1992.

[Pecen a kol., 1994] Pecen,L. - Pelikán,E. – Beran,H.: Short-term foreign exchange rate forecasting from high frequency data. In: Proc. International Workshop on Neural Networks in the Capital Markets, Pasadena, CALTECH, USA, 1994.

[Provost, Fawcett, 1997] Provost,F. – Fawcett,T.: Analysis and Visualization of Classifier Performance: Comparison under Imprecise Class and Cost Distributions. In: Proc. KDD-97, AAAI Press, 1997, 43-48.

[Pudil a kol., 1994] Pudil,P. – Novovičová,J. – Kittler,J.: Floating search methods in feature selection. Pattern Recognition Letters, 15, 1994, 1119-1125.

[Pudil, 2001] Pudil,P.: Methodology and Advances in Feature Selection for Statistical Pattern Recognition, Disertace DrSc, UTIA AV ČR, Praha, 2001

[Quinlan, 1979] Quinlan,J.R.: Discovering rules by induction from large collections of examples. In: (Michie, ed.), Expert Systems in the Micro-Electronic Age. Edinburgh University Press, 1979.

[Quinlan, 1986] Quinlan,J.R.: Induction of decision trees. Machine Learning, 1(1), 1986, 81-106.

[Quinlan, 1992] Quinlan,J.R.: Learning with continuous classes. In: Proc. AI'92, Singapore, 1992, 343-348. [Quinlan, 1993] Quinlan,J.R.: C4.5: Programs for machine learning. San Mateo, Morgan Kaufman, 1993.

[Rauch, 1999] Rauch,J.: Získávání znalostí z databází. Podklady pro posluchače semináře IISEM 491 v letním semestru 1998/99, VŠE, 1999.

[Rissanen, 1978] Rissanen,J.: Modeling by shortest data description. Automatica, Vol. 14, 1978, 465-471.

[Russell a kol, 1995] Russell,S. – Binder,J. – Koller,D. – Kanazawa,K.: Local learning in probalistic networks with hidden variables. In: Proc. 14th Int. Joint Conference on Artificial Intelligence IJCAI'95, Morgan Kaufmann, 1995.

[Saitta a kol., 2000] Saitta,L. -Kietz,J.U. - Beccari,G.: Specification of Pre-Processing Operators Requirements. Deliverable, D1.1, IST Project MiningMart, IST-1999-11993, 2000.

[Sakoe, Chiba, 1990] Sakoe,H. - Chiba,S.: Dynamic Programming Algorithm Optimization For Spoken Word Recognition 1990. IEEE Trans. Acoustics, Speech and Signal Proc., Vol. ASSP-26, 1978, 43-49.

[SEMMA,2002] SASS: SEMMA. Internet, http://www.sas.com/products/miner/semma.html, 2002 [5A, 2000] SPSS: Metoda 5A. Internet, http://www.spss.cz/datamin_jak.html, 2000.

[Shafer a kol, 1996] Shafer,J. – Agrawal,R. – Mehta,M.: SPRINT: a scalable parallel classifier for data mining. In: Proc. 22th Very Large Databases Conference VLDB, Morgan Kaufmann, 1996, 544-555.

[Shapire, 1999] Shapire,R.: Theoretical Views of Boosting and Applications. In: Proc.10th Int. Conf. on Algorithmic Learning Theory.

[Shavlik, 1992] Shavlik,J.: A framework for combining symbolic and neural learning. Technical Report no. 1123, Comp. Sci. Dept., Uni. of Wisconsin, 1992.

[Schölkopf, Smola, 2001] Schölkopf,B. - Smola,A.J.: Learning with Kernels. MIT Press, 2001.

[Stanfill, Waltz, 1986] Stanfill,C. – Waltz,D.: Towards memory-based reasoning. Communications of the ACM 29 (12), 1986, 1213-1228.

[Suzuki, 2000] Suzuki,E. (editor): Proc. Int. Wshop of KDD Challenge on Real-world Data. 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining PAKDD-2000, Kyoto, 2000.

[Svátek, 1996] Svátek,V.: Learning with value hierarchies in the ESOD framework. In: Proc. Artificial Intelligence Techniques AIT'96, Brno, 1996.

[Šíma, Neruda, 1996] Šíma,J. – Neruda,R.: Teoretické otázky neuronových sítí. Matfyzpress, 1996.

[Torgo,1995] Torgo,L.: Data fitting with rule-based regresion. In: Proc. Artificial Intelligence Techniques AIT'96, Brno, 1996.

[Towell, Shavlik, 1994] Towell,G. - Shavlik,J.: Knowledge-based artificial neural networks. Artificial Intelligence, 70(1-2), 1994, 119-165.

[Watson, Marir, 1994] Watson,I. – Marir,F.: Case-based reasoning: An review. The Knowledge Engineering Review, Vol. 9:4, 1994, 327-354.

[Wettschereck, Aha, 1995] Wettschereck,D. - Aha,D.: Weighting features. In Proc. 1st Int. Conf. on Case-Based Reasoning, Springer, 1995.

[Widmer, 1994] Widmer,G.: Combining robustness and flexibility in learning drifting concepts. In: (Cohn ed.) Proc: 11th European Conf. on Artificial Intelligence ECAI'94, John Wiley, 1994.

[Widmer, Kubat, 1992] Widmer,G. – Kubat,M.: Learning flexible concepts from streams of examples: FLORA2. In: (Neumann ed.) Proc. 10th European Conf. on Artificial Intelligence ECAI'92, John Wiley, 1992.

[Witten, Frank, 1999] Witten,I.H. – Frank,E.: Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufman, 1999, ISBN 1-55860-552-5.

SHRNUTÍ STUDIJNÍ OPORY

V tomto textu jsme se seznámili s pojmem dolování dat a s ním souvisejícími základními pojmy. Ve druhé kapitole jsme se věnovali fázi přípravy dat, což je nezbytná součást většiny procesů zahrnujících dolování dat, která umožňuje upravit hrubá data získaná přímo z praktických aplikací na data, která jsou již vhodná pro vybranou metodu dolování dat. Po zopakování několika statistických metod a seznámení se s přístupy k automatizaci procesu učení, se text věnuje, ve své hlavní části, základním metodám dolování dat, které často patří mezi nejčastěji používané metody dolování dat, jako jsou např. metody pro tvorbu rozhodovacích stromů, asociačních pravidel, metody založené na umělých neuronových sítích, bayesovské metody nebo metody založených na analogii. Pro výběr nejvhodnějšího praktického modelu získaného z dat jsme se poté v závěrečné kapitole věnovali různým možnostem hodnocení získaných modelů.

Název:	Dolování dat
Autor:	prof. Ing. Petr Berka, CSc. Ing. Jan Górecki, Ph.D.
Vydavatel:	Slezská univerzita v Opavě
	Obchodně podnikatelská fakulta v Karviné
Určeno:	studentům SU OPF Karviná
Počet stran:	121

Tato publikace neprošla jazykovou úpravou.