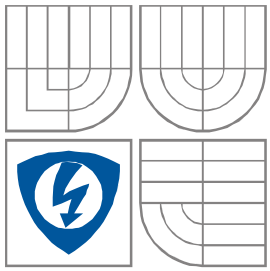


Strojové učení

Garant předmětu:
Ing. Petr Honzík, Ph.D.

Autoři textu:
Ing. Petr Honzík, Ph.D.

Brno **1.10. 2006**

Obsah

Seznam obrázků

Seznam tabulek

Vstupní test

1. Vysvětli pojmy
   a) kvantitativní data (proměnná)
   b) kvalitativní data (proměnná).

2. Jak se dělí kvantitativní data (proměnné)?

3. Jak se dělí kvalitativní data (proměnné)?

4. Co je to nezávislá proměnná?

5. Co je to závislá proměnná?

6. Vysvětli rozdíl mezi pojmy klasifikátor a regresní model.

1 Úvod do strojového učení

Cíle kapitoly:

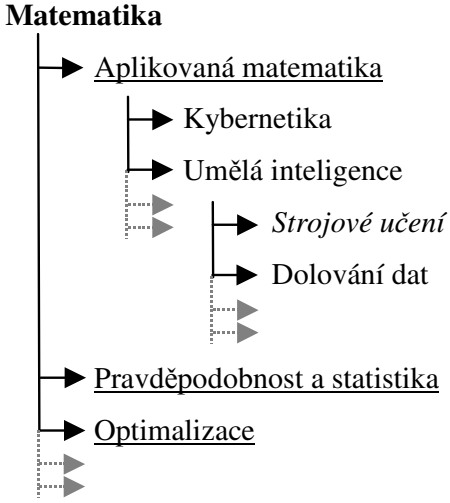
Po nastudování této kapitoly je student znalý zařazení strojového učení mezi ostatní obory a způsobu, jakým ostatní vědní disciplíny strojové učení ovlivňují. Rozumí složení i významu komponent v procesu učení s učitelem, rozlišuje základní typy veličin, modelů, chybových funkcí, testů přesnosti a meta-algoritmů. Je obeznámen se základními teorémy a termíny, jejichž znalost je nezbytná pro další studium oboru strojové učení.

1.1 Zařazení strojového učení

zařazení strojového učení

Strojové učení (SU) je českým ekvivalentem anglického termínu Machine Learning (ML). Jedná se o vědní disciplínu zabývající se algoritmy a programy, které umožňují strojům proces *učení*.

Samotné hierarchické zařazení SU jako kategorie mezi ostatními vědními disciplínami není jednoznačné. Důvodem je jejich vzájemné prolínání se a doplňování. Následující schéma proto není jediným možným způsobem, kam SU zařadit.



Obr. 1.1: Zařazení oboru *strojové učení*

Mezi obory, které významným způsobem zasahují do SU a přitom jsou hierarchicky v odlišné oblasti náleží zejména *statistika* a *optimalizace.*

Statistika nabízí silný matematický aparát využívaný zejména ve fázi předzpracování dat a při vyhodnocování modelů (průběžném i konečném). Termín model lze z pohledu statistiky nahradit termínem *hypotéza* (model je ve své podstatě hypotézou o vztazích a vazbách mezi vstupními a výstupními veličinami v modelovaném procesu či objektu). Optimalizace se zabývá algoritmy, které vedou k nalezení extrému funkce. Tento obor tedy na model pohlíží jako na *funkci*. Cílem optimalizace ve SU je nastavení parametrů modelu

tak, aby při daných vstupních hodnotách byla minimalizována odchylka mezi výstupem požadovaným a získaným z modelu.

umělá inteligence

Jak je ze schématu na obr. 1.1 patrné, umělá inteligence (UI), ang. Artificial Intelligence (AI), je oboru SU nadřazená. Marvin Minský definoval UI jako vědu, která se zabývá tím, jak přimět stroje projevovat se takovým chováním, které by v případě člověka vykazovalo potřebu inteligence. Existuje tzv. Turingů test, který zjednodušeně spočívá v tom, že člověk klade otázky (např. písemnou formou) a dostává odpovědi. Na základě těchto odpovědí se rozhoduje, jestli komunikuje s člověkem nebo strojem. Pokud stroj při tomto testu oklame člověka, projevuje se inteligentně. Je to však opravdu důkaz existence umělé inteligence? Stroj mohl nalézt odpovědi v nějaké databázi, přitom ve skutečnosti vůbec nechápe jejich smysl. Pak hovoříme o tzv. *slabé UI*. *Silná UI* naproti tomu chápe obsah podobně jako člověk. Nedostatečnost Turingova testu na dokázání silné UI byla provedena pomocí tzv. čínského pokoje. Osoba neznalá čínštiny, jen na základě určitých pravidel, korektně odpoví na otázky v tomto exotickém jazyce. Princip tohoto důkazu přirovnal Stanisław Lem ke skládání puzzle obrázku. Podstata pokusu spočívá v tom, že dílky jsou obráceny rubem nahoru, není tedy možné puzzle sestavit na základě obrázku, jehož vzhled je vodítkem při skládání. Přesto je možné v případě, že je každý dílek unikátní (vhodný formální popis jazyka), poskládat puzzle správně tak, aby obrazec (nebo např. text) na lícové straně dával smysl. Tím byla prokázána nedostatečnost Turingova testu na dokázání silné UI. Vývoj UI ovlivňuje nejen akademická obec. Celá oblast literatury označovaná jako science fiction si pohrává s prognostickými myšlenkami o možném vývoji a vlivu UI. Prognózy některých spisovatelů jsou brány velice vážně (A.C.Clark, Stanisław Lem, ...). Smyslem těchto příkladů bylo vysvětlit, že UI se jako obor nezabývá pouze přímou aplikací matematických modelů, ale také filosofickými a etickými důsledky plynoucími z jejich použití.

Skutečná realizace však nakonec končí u algoritmů a konkrétních matematických modelů. Tím se dostáváme do oblasti, která je středem zájmu tohoto předmětu, ke strojovému učení.

strojové učení

Strojové učení je nauka o algoritmech, které umožňují učení umělých objektů. *Učením se* je rozuměno automatické zlepšování se na základě zkušeností. SU se tedy orientuje na matematický aparát, na jehož základě dochází k technické realizaci cílů definovaných v UI. Skutečné meze těchto algoritmů ve vztahu k člověku a jeho chápání včetně ostatních filosofických úvah do SU nespadají.

dolování dat

Dolování dat (ang. Data Mining) je disciplínou definovanou cílem získávat netriviální a potenciálně užitečné informace z dat, případě z databází (Knowledge Discovery in Databases). Jsou definovány určité metodologie (SEMMA, CRISP-DM,...), jak postupovat při zpracování dat. Formálně lze říci, že disciplína dolování dat využívá nástrojů SU.

statistika

Statistika je s algoritmy strojového učení spjata velice úzce. Řada postupů používaných ve SU vychází přímo ze statistických metod. Zcela zásadní rozdíl mezi SU a statistikou spočívá v tom, že podstatou SU je *vytvořit model* (*hypotézu*), který bude predikovat a ve finále skutečně rozhodovat, přičemž proces učení nemusí vést při opakovaném výpočtu ke stejným výsledkům, tedy stejným modelům. Cílem je nastavit model co nejlépe. Statistika je matematický obor, který se zabývá popisem (charakteristiky datových souborů), indukcí

(předpoklady o základním souboru ze vzorků), porovnáváním a testováním (posuzuje hladinu významnosti hypotéz na základě předpokladů). Co do číselných výpočtů je narozdíl od SU jednoznačná. Umožňuje tak absolutní hodnocení, na jehož základě lze provádět různá srovnání. Zjednodušeně řečeno, SU vytváří hypotézy na základě velice pestré škály názorů, nápadů a postupů. K posouzení míry zdařilosti této činnosti (ať už během procesu učení nebo po jeho ukončení) se pak obrací na statistiku jako na metodologii garantující korektnost při absolutním posuzování kvality vytvořených modelů.

optimalizace

Optimalizace je poslední oblastí, která hraje ve SU zásadní roli. SU zahrnuje mnoho různých modelů, ve kterých lze uložit znalost v podobě jejich struktury a parametrů. Nastavení hodnot parametrů závisí na vstupních informacích, tedy trénovacích datech. Postup, jak nastavit model na základě dat, se nazývá optimalizace (spočívá v minimalizaci odchylek rozdílů výstupních hodnot modelu a požadovaných výstupních hodnot).

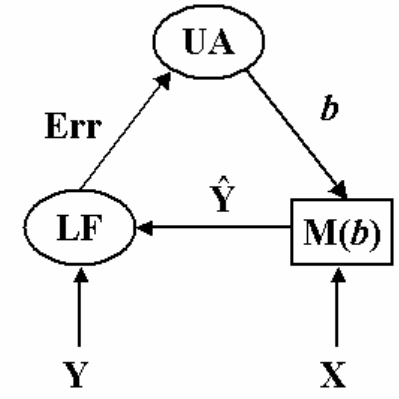
shrnutí

Součinnost uvedených vědních disciplín si lze představit tak, že z databáze SU je vybrán model (matematický objekt, který má schopnost uložit v sobě znalost), optimalizace poskytne postup, kterým se model na základě dostupných dat správně nastaví a statistika pak poslouží jako absolutní měřítko jeho dosažené přesnosti. Celý tento proces spadá do oboru SU.

1.2 Komponenty procesu modelování

základní schéma

Schéma na obrázku 1.2 znázorňuje objekty a funkce, které dohromady tvoří základní komponenty procesu modelování z pohledu strojového učení. Je tvořeno vstupní veličinou $\boldsymbol{X}$, výstupní veličinou Y, predikovanou výstupní veličinou $\hat{Y}$, modelem s parametry $M(\boldsymbol{b})$, chybovou funkcí LF a optimalizačním (učicím) algoritmem UA.



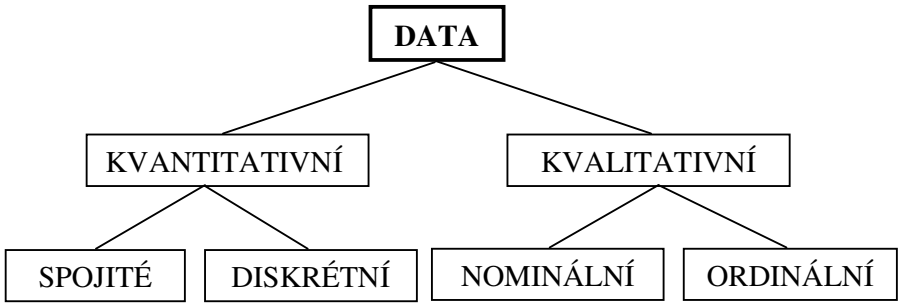
Obr. 1.2: Model učení s učitelem

Jeden cyklus procesu učení s učitelem probíhá následovně: postupně jsou předkládána vstupní data $\boldsymbol{X}$ modelu $M(\boldsymbol{b})$; výstup modelu $\hat{Y}$ je porovnáván s požadovaným výstupem Y; míru vzniklé odlišnosti kvantifikuje chybová funkce LF (ang. loss function) na tzv. chybu Err (ang. error); optimalizační (učicí) algoritmus UA upravuje parametry modelu $\boldsymbol{b}$ na základě informace o chybě Err tak, aby její hodnota byla v příštím cyklu co nejmenší.

veličiny

Vstupní, výstupní i predikované veličiny ($\boldsymbol{X}, Y, \hat{Y}$) mohou být různého typu. Jejich základní dělení znázorňuje obrázek 1.3.

Charakteristické pro kvantitativní veličinu je skutečnost, že její číselná hodnota má nějakou jednotku (metry, procenta, počet lidí v tramvaji,...) a vyjadřuje množství (kvantum) v dané jednotce. Mezi kvantitativními daty pak existuje operace sčítání a odečítání. Kvantitativní veličiny se dále dělí na spojité a diskrétní (obdoba analogové a digitální). Například délka nosníku je veličina spojitá, počet lidí v tramvaji veličina diskrétní.



Obr. 1.3: Typy veličin

Kvalitativní veličiny nenesou číselnou informaci. Dělí se na ordinální a nominální. Prvky v ordinální veličině lze seřadit, ve smyslu orientace je mezi nimi možno definovat relace <, >, =. Často však relace závisejí na kontextu, ve kterém jsou data použita. Například vztaženo ke vzdálenosti od slunce lze množinu planet {*Merkur, Venuše, Země, Mars, Jupiter, Saturn, Uran, Neptun*} považovat za ordinální. Neplatí však, že by vzdálenost mezi Zemí a Marsem byla stejná jako mezi Marsem a Jupiterem. Pokud bychom ze zabývali hmotností planet, pořadí by bylo odlišné. Jestliže kontext neznáme nebo dokonce neexistuje, na pořadí prvků v množině nezáleží. Taková veličina je nominální.

V literatuře se lze setkat s různou symbolikou odlišující výstupní veličiny podle jejich typu. Kvantitativní veličině bývá přiřazován symbol Y ($\hat{Y}$), kvalitativní G ($\hat{G}$).

modely

Podle výše zmíněného dělení veličin na kvantitativní a kvalitativní se dělí i modely podle typu výstupní veličiny na *modely regresní* – výstup je kvantitativní, a *klasifikátory* – výstup je kvalitativní. Běžná je kombinace obou typů modelů dohromady, tzv. regresní klasifikátory. Jejich princip spočívá v tom, že kvalitativní výstupní veličina je nahrazena čísly, která jsou v regresním modelu použita jako kvantitativní veličina. Na spojitém výstupu regresního modelu je následně stanovena tzv. kritická hodnota (práh, mezní hodnota), na jejíž základě (překročení, nedosažení) je klasifikováno. Typická je tato substituce u binárních dat (muž/žena, ano/ne – nahrazeno čísly 0/1 nebo -1/1).

chybové funkce

Vstupy chybové funkce jsou požadovaná výstupní hodnota Y (G) a výstupní hodnota modelu $\hat{Y}$ ($\hat{G}$). Cílem je kvantifikovat míru chyby modelu. Pokud platí rovnost obou výstupních veličin ($Y=\hat{Y}$, $G=\hat{G}$), je chyba nulová. V případě jejich odlišnosti chybu jednoznačně kvantifikuje chybová funkce *LF*. Pokud je výstupní veličina kvalitativní, používají se k určení chyby tzv. *nákladové matice*. Protože

chybová funkce představuje zpětnou vazbu v optimalizačním procesu, je její korektní stanovení důležité. Jejímu určení a výpočtu je ve skriptech věnována celá kapitola.

optimalizační algoritmy

Smyslem optimalizačního algoritmu je hledání extrému funkce. V případě SU je cílem najít takové parametry $\boldsymbol{b}$ modelu $M(\boldsymbol{b})$, při kterých je chyba predikce na trénovacích datech co nejmenší. I toto nastavení má však svoji hranici v podobě tzv. přeučení modelu (viz. další kapitola). Optimalizační algoritmy jsou samostatnou disciplínou v oboru matematiky. V rámci těchto skript bude věnována zvýšená pozornost tzv. genetickým algoritmům, které kopírují proces evoluce podle Darwinovy teorie přírodního výběru. U některých metod SU bude použitý optimalizační algoritmus popsán a vysvětlen, u jiných bude pouze naznačena jeho základní myšlenka.

testy přesnosti modelů

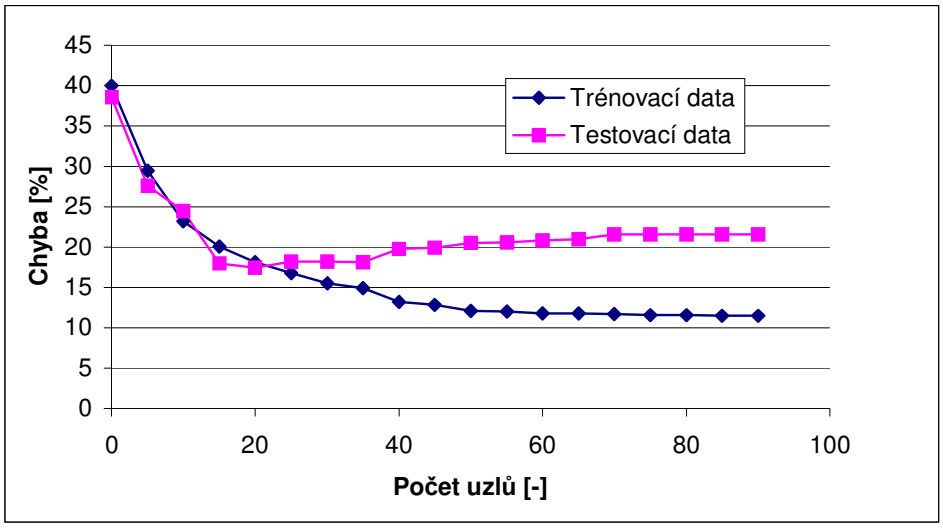
Aby bylo možné ověřit skutečnou přesnost modelu, dělí se dostupná data na data trénovací a testovací. Trénovací data jsou použita pro naučení modelu, testovací na zjištění jeho přesnosti. Problém však spočívá v tom, že model naučený na pouhou část dat (trénovací) nemusí dosahovat přesnosti, které by dosahoval v případě, že by byla použita veškerá dostupná data (trénovací i testovací). Tímto problémem se zabývají testy přesnosti modelů (cross-validatiaon, bootstrap), které umožňují použít na naučení modelu všechna data, přitom však nedochází ke ztrátě informace o skutečné přesnosti modelu.

meta algoritmy

Meta algoritmy se zabývají tím, jak využit k predikci větší počet modelů. Metody se dělí podle toho, jestli pracují se stejnými typy modelů (bagging, boosting) nebo různými typy modelů (bumping, stacking). Jejich použití může ovlivnit i proces samotného učení, ve výsledku zvyšuje přesnost predikce. Největší přínos mají zejména v případech, kdy jednotlivé modely predikují optimálně jen s lokální platností.

1.3 Teorémy a terminologie

přeučení



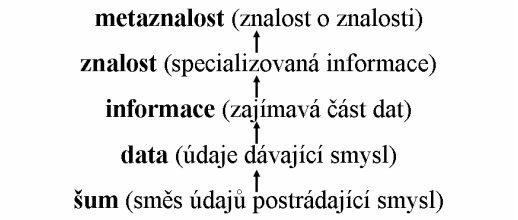
Obr. 1.4: Přeučený model rozhodovacího stromu

Jedním z problémů při učení modelu na dostupných datech je tzv. *přeučení modelu*. Tento stav spočívá v tom, že se model naučí i zdánlivé, ve skutečnosti neexistující souvislosti. Důvodem může být příliš mnoho parametrů modelu (Occamovo ostří) nebo zašuměnost a chyby v trénovacích datech. Vzniklý efekt je patrný z obr. 1.4. V uvedeném příkladě je vytvářen rozhodovací strom. Postupným přidáváním dalších uzlů je možno zvyšovat přesnost klasifikace na trénovacích datech. Z grafu je patrné, že od určitého počtu uzlů se chyba na testovacích datech začala zvětšovat. Příčinou je skutečnost, že byl model na trénovacích datech přeučen, tzn. naučil se závislosti, které ve skutečnosti neexistují a které ve svém důsledku vedou ke zhoršení přesnosti predikce.

ocaamovo ostří

Tento teorém je připisován anglickému filosofu Williamu Ockhamovi (14. století). Z pohledu SU je podstatná následující interpretace: „pokud pro nějaký jev existuje vícero vysvětlení, je lépe upřednostňovat to méně komplikované“. Přeneseněji řečeno, pokud dva různé modely predikují identicky, je správný ten jednodušší (mající méně parametrů).

data



Obr. 1.5: Úrovně záznamů podle typu nesené informace

Na obr. 1.5 je schéma přechodu od šumu k metaznalosti. Nejnižší úrovní z pohledu nesené informace je *šum*. Výběr smysluplných údajů ze šumu tvoří *data*. Data představující konkrétní vlastnost jsou označována termínem *informace*. *Znalost* je specializovaná informace, která má schopnost transformovat informaci v jinou informaci; v případě SU se jedná o model. *Metaznalost* je pak schopnost transformovat znalost v novou znalost nebo informaci (např. rozhodování o tom, který z modelů – znalostí – bude použit).

Dále lze data rozdělit podle toho, jestli mají otevřenou nebo uzavřenou doménu. Pokud má proces nebo objekt konečný počet vlastností, které jsou známy, má *uzavřenou doménu*. V opačném případě, tedy pokud je počet vlastností procesu či objektu neomezený nebo neznámý, hovoříme o tzv. *doméně otevřené*.

indukce, dedukce

Indukce je logický proces, přechod od konkrétního k obecnému (od dat ke znalostem, modelu). Algoritmy SU jsou induktivního charakteru. Jsou platné vždy jen s určitou pravděpodobností. *Dedukce* je přechod od obecného ke konkrétnímu (od znalostí k datům, novým závěrům). Tento proces postupuje od předpokladů k jistým (jednoznačným) závěrům. Pokud jsou závěry v rozporu se skutečností, je vyvrácena platnost předpokladu (např. modelu).

učicí algoritmy

V první řadě rozlišujeme *učení s učitelem* (ang. supervised learning) a *bez učitele* (ang. unsupervised learning). Rozdíl spočívá v tom, jestli je v trénovacích datech obsažena požadovaná výstupní hodnota Y (G) nebo ne. Pokud ano, jedná se o učení s učitelem (učitelem je tedy ona požadovaná výstupní hodnota), v opačném případě se jedná o učení bez učitele. V takovém případě jsou v datech

hledány dosud neznámé souvislosti, jsou hledány tzv. shluky (ang. clusters), tedy oblasti tvořené daty, která jsou si vzájemně podobná. Takto nalezené shluky pak mohou představovat nové třídy nebo hledané objekty.

Další vlastností je schopnost učit se na základě nových dat bez potřeby použít znovu všechna dosud známá trénovací data. Pokud tedy model potřebuje k nové úpravě maximálně *k* předešlých příkladů, aby znalost uloženou ve své struktuře rozšířil o informaci obsaženou v nově příchozí instanci, hovoříme o *inkrementálním učení*. V opačném případě jde o *neinkrementální učení*.

Algoritmy lze také dělit podle toho, zda je možné model upravovat během jeho používání. Pokud ano, hovoříme o *on-line učení*, v opačném případě o *off-line* učení.

generalizace a specializace

Při hledání správné hypotézy na základě trénovacích dat lze postupovat dvěma odlišnými způsoby. Prvním postupem je *specializace*. Nejdříve je vytvořena zcela obecná hypotéza. S přibýváním nových případů je tato hypotéza zpřesňována (specializována) tak, aby eliminovala všechny negativní případy. Opačným směrem se ubírá *generalizace*. Na základě prvního pozitivního trénovacího prvku je vytvořena zcela specifická hypotéza zahrnující pouze tento jediný prvek. S přibýváním nových příkladů je hypotéza rozšiřována tak (generalizace), aby zahrnovala i nové pozitivní příklady. Podrobněji se touto problematikou zabývá induktivní učení.

U generalizace rozlišujeme *správnou generalizaci* (např. „všechna sudá čísla větší než 2 nejsou prvočísla“) *neoprávněnou generalizaci* (např. „přirozená čísla končící na 1 jsou prvočísla“) a *podceněnou generalizaci* (např. „žádné prvočíslo není větší než 100“).

aplikace SU

Mezi typické úlohy využívající SU patří vyhledávání (internetové vyhledávače), lékařské diagnostické systémy, rozlišení nelegálního použití kreditních karet, rozlišení řeči, písma a psaného textu, analýza investičních trhů, expertní systémy, ...

1.4 Souhrn

Strojové učení je jednou z oblastí spadající do umělé inteligence. Zaměřuje se na strukturu, učení a aplikaci matematických modelů schopných automatického učení se. Pro tyto účely uvádí v součinnost především dva další obory, statistiku a optimalizaci.

Hlavní strukturu procesu učení s učitelem tvoří následující komponenty: vstupní data ***X;*** model $M(\boldsymbol{b})$; výstup modelu $\hat{Y}$; požadovaným výstup Y; chybová funkce *LF;* chyba *Err*; optimalizační (učicí) algoritmus *UA;* parametry modelu $\boldsymbol{b}$. Cílem procesu učení s učitelem je takové nastavení modelu, aby co nejlépe odpovídal skutečným vztahům a vazbám v modelovaném procesu nebo objektu.

Specifickými problémy spojenými s učením modelů jsou tzv. přeučení modelu a stanovení skutečné chyby modelu.

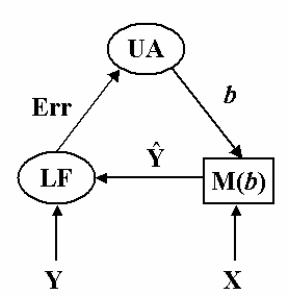
Kontrolní otázky a úlohy

1. Jak se dělí veličiny?
Kvantitativní (spojitá, diskrétní), kvalitativní (nominální, ordinální).

2. Jaké úrovně záznamů podle obsahu rozlišujeme?
Šum, data, informace, znalost, metaznalost.

3. Jaký je rozdíl mezi umělou inteligencí a strojovým učením?
Umělá inteligence se zabývá tím, jak naučit stroje projevovat se podobně jako člověk, jak tento stav dokázat a jaké důsledky z toho společensky vyplývají. Strojové učení je součástí umělé inteligence, které se zabývá matematickým aparátem, tedy matematickými modely a algoritmy umožňující učení se strojů.

4. Nakresli a popiš komponenty procesu učení s učitelem.



*Jeden cyklus procesu učení s učitelem probíhá následovně: postupně jsou předkládána vstupní data **X** modelu M(**b**); výstup modelu Ŷ je porovnáván s požadovaným výstupem Y; míru vzniklé odlišnosti kvantifikuje chybová funkce LF (ang. loss function) na tzv. chybu Err (ang. error); optimalizační (učicí) algoritmus UA upravuje parametry modelu **b** na základě informace o chybě Err tak, aby její hodnota byla v příštím cyklu co nejmenší.*

5. Vysvětli pojmy „occamovo ostří“ a „přeučení modelu“.
Occamovo ostří: pokud dva různé modely predikují identicky, je správný ten jednodušší.
Přeučení modelu: naučení závislostí, které ve skutečnosti neexistují; způsobeno chybami nebo šumem v použitých trénovacích datech.

6. Vysvětli pojmy indukce a dedukce.
Indukce je zobecnění, z konkrétních dat k obecnému pravidlu (model). Dedukce je směr opačný, tedy konkrétní závěr z obecně platného pravidla.

Literatura:

[1] Berry M.J.A., Linoff G.S.: Data Mining Techniques, Wiley Publishing, Inc., 2004. ISBN 0-471-47064-3.

[2] Fielding A.H., Bell J.F.: A review of methods for the assessment of prediction errors in conservation presence/absence models. Foundation for Environmental Conservation. 1997, pp. 38-49.

[3] FRANK, E., HARRELL, J. Regression Modeling Strategies. NY: Springer, 2001. 568 pages. ISBN 0-387-95232-2.

[4] Hastie T., Tibshirani R., Friedman J.: The Elements of Statistical Learning. Springer, 2001. ISBN 0-387-95284-5.
[5] Lem S.: Tajemství čínského pokoje. Mladá fronta, 1999. ISBN 80-204-0826-6.
[6] Machová K.: Strojové učenie, Košice 2002, http://neuronai.tuke.sk/~machova/SU4.pdf
[7] Mitchel T.: Machine Learning. MacGraw Hill Science, 1997. ISBN 0070428077.
[8] Schölkopf B., Smola A.J.: Learning with Kernels. MIT Press, Cambridge, MA, 2002. ISBN 0-262-19475-9.
[9] Theodoridis S.: Pattern Recognition, Elsevier, 2003. ISBN 0-12-685875-6.

2 Chybové funkce

Cíle:

Po nastudování této kapitoly má student jasnou představu o významu chybové funkce a matematických vztazích, kterými ji lze vypočíst. Chápe rozdíly i společné rysy, které jsou mezi metodou nejmenších čtverců a maximální věrohodností, rozumí rozdílnosti v přístupu k chybovým funkcím z pohledu typu vstupních dat (kvantitativní, kvalitativní), umí použít nákladové matice.

chybová funkce

Podstatou chybové funkce je vyjádřit odlišnost mezi modelem a realitou. Stejně, jako se od sebe liší různé typy modelů, liší se i chybové funkce, které jsou pro jejich nastavování používány. Základní dělení chybových funkcí vyplývá z toho, pro jaký typ dat (kvantitativní, kvalitativní) jsou určeny.

definice chybové funkce

Zápisem $(x,y,f(x)) \in X \times Y \times Y$ rozumíme uspořádanou trojici, ve které veličina x představuje vstupní hodnotu, y představuje požadovanou výstupní hodnotu a $f(x)$ představuje predikovanou výstupní veličinu (též $\hat{y}$). Funkce $L: X \times Y \times Y \rightarrow \langle 0; \infty)$, pro kterou platí, že pro $\forall\, x \in X$ a $\forall\, y \in Y$ je $L(x,y,y)=0$, je označována jako chybová funkce.

2.1 Metoda nejmenších čtverců

definice MNČ

Metoda nejmenších čtverců (MNČ) je aproximační metoda, která spočívá v tom, že hledáme takové parametry zvolené funkce, pro které je součet čtverců odchylek vypočtených hodnot od hodnot naměřených minimální.

Výpočet chyby *Err* podle uvedené definice vyjadřuje následující vztah:

$$Err = \sum_{i=1}^{N} \left[y_i - f\left(x_i, \boldsymbol{b}\right) \right]^2 \tag{2.1}$$

kde y je požadovaná výstupní hodnota, x je hodnota vstupní veličiny a $\boldsymbol{b}$ je vektor parametrů modelu $f(x,\boldsymbol{b})$.

lineární MNČ

Lineární MNČ se týká regresních modelů, které jsou lineární ve svých parametrech $\boldsymbol{b}$, jak ukazuje následující rovnice

$$f\left(\boldsymbol{x};\boldsymbol{b}\right) = b_0 + b_1 f_1\left(\boldsymbol{x1}\right) + \ldots + b_N f_N\left(\boldsymbol{xM}\right) \tag{2.2}$$

Hlavní předností modelů lineárních v parametrech je skutečnost, že pro nalezení optimálních hodnot jejich parametrů zpravidla existují známá analytická řešení. Pokud je to možné, bývají z tohoto důvodu linearizovány i modely nelineární. To má většinou za následek, že rozložení chyby měření již není normální, což je implicitní předpoklad pro použití lineární MNČ. Navzdory tomu však bývají výsledky takto upravených závislostí uspokojivě přesné.

Nevýhodou modelů lineárních v parametrech je problematická aproximace složitějších závislostí a citlivost vůči prvkům, které jsou např. vinou chyby měření výrazně vzdáleny od prvků ostatních (ang. outliers). Pokud je použita linearizace složitější závislosti, je třeba brát zřetel na interval, ve kterém toto zjednodušení platí; závislost nelze extrapolovat.

nelineární MNČ

V mnoha případech není linearizovaná aproximace řešení dostatečně přesná nebo vůbec možná a je použit nelineární model. K hledání optimálních hodnot parametrů takových modelů se používá nelineární MNČ.

Hlavní myšlenka výpočtu chyby zůstává stále stejná. Ke stanovení hodnot neznámých parametrů modelu je však třeba použít numerické iterační metody (optimalizační algoritmus), s jejichž použitím souvisí řada komplikací (volba samotné iterační metody, uvíznutí v lokálním extrému, závislost nalezeného řešení na počátečních odhadech parametrů, …).

váhová MNČ

V určitých případech může mít stejná odchylka při odlišné vstupní hodnotě různou váhu. Příkladem je např. studie vztahu mezi předpokládanou a skutečnou cenou stavebního projektu. Stejná odchylka v malém projektu má odlišnou váhu než v projektu rozsáhlém. Typickou ukázkou je také rozdílný rozptyl chyby měření při jednotném rozsahu a různě velkými naměřenými hodnotami. Ukázkou upravené chybové funkce je následující vztah:

$$Err = \sum_{i=1}^{N} \left(y_i - f\left(x_i, \boldsymbol{b}\right)\right)^2 \cdot \frac{1}{x_i^2} \tag{2.3}$$

Váhovou MNČ lze použít v lineárních i nelineárních modelech, jde pouze o modifikaci chybové funkce.

absolutní chyba

Místo kvadrátu odchylky požadované hodnoty lze použít absolutní vzdálenost měřené veličiny od hodnoty funkce modelu. Takovou odchylku však nelze chápat jako spojitou diferencovatelnou veličinu. Výhodou je, že její hodnota roste proporciálně s chybou (na rozdíl od MNČ).

pásmo necitlivosti ε

Další variantou chybové funkce založené na absolutní hodnotě je její rozšíření o pásmo necitlivosti ε. Smyslem je, že chyba v určitém malém okolí ε nebude penalizována. Nechť $\Delta y_i=(y_i-f(x_i,\boldsymbol{b}))$. Vztah pro chybovou funkci pak vypadá následovně:

$$Err\left(x_i, y_i, f\left(x_i, \boldsymbol{b}\right)\right) = \max\left(\left|\Delta y_i\right| - \varepsilon, 0\right) \tag{2.4}$$

polynomické varianty

Dalším krokem mimo rámec MNČ je zvýšení mocniny odchylky z kvadrátu na mocninu vyššího řádu, popřípadě sloučení více různých metod pro výpočet chyby predikce. Jednou variantou je tzv. Huberova robustní chyba (2.5), polynomická chyba (2.6) nebo po částech polynomická chybová funkce (2.7):

$$Err\left(x_i, y_i, f\left(x_i, \boldsymbol{b}\right)\right) = \begin{cases} \frac{1}{2\sigma}\Delta y_i^{\,2} & , pokud \left|\Delta y_i\right| \le \sigma \\ \left|\Delta y_i\right| - \frac{\sigma}{2} & , jinak \end{cases} \tag{2.5}$$

$$Err\left(x_i, y_i, f\left(x_i, \boldsymbol{b}\right)\right) = \frac{1}{d}\left|\Delta y_i\right|^d \tag{2.6}$$

$$Err\left(x_i, y_i, f\left(x_i, \boldsymbol{b}\right)\right) = \begin{cases} \frac{1}{d\sigma^{d-1}}\left|\Delta y_i\right|^d & , pokud \left|\Delta y_i\right| \le \sigma \\ \left|\Delta y_i\right| - \sigma\frac{d-1}{d} & , jinak \end{cases} \tag{2.7}$$

Uvedené vztahy lze dále rozšířit např. o pásmo necitlivosti a získat tak další různé chybové funkce. Jejich uplatnění však souvisí spíše se specifickými

problémy. Z metod popsaných v této podkapitole nachází v praxi největší uplatnění lineární a nelineární MNČ.

2.2 Maximální věrohodnost

pravděpodobnost

Pravděpodobnost je šance, že nastane určitá událost (či množina událostí) vyjádřená lineárně v intervalu od 0 (událost nenastane) do 1 (událost nastane).

Při výpočtu pravděpodobnosti můžeme vyjít ze znalosti základního souboru nebo experimentu. Na jeho základě jsme schopni vyjádřit, s jakou pravděpodobností nastane konkrétní výběrový soubor.

L -věrohodnost

„Věrohodnost je hypotetická pravděpodobnost, že událost, která již nastala, je následkem specifického procesu“.

Podstatou je, že známe výběrový soubor a na jeho základě vyjadřujeme, s jakou pravděpodobností tento výběr nastal za předpokladu zvoleného základního souboru.

věrohodnost vs. pravděpodobnost

Základním rozdílem mezi věrohodností a pravděpodobností spočívá v tom, že pravděpodobnost se vztahuje k budoucím událostem, zatímco věrohodnost k událostem minulým se známými výstupy.

Pravděpodobnost lze vyjádřit jako $P(X|p)$; pravděpodobnost X závisí na parametrech p modelu (základního souboru). Věrohodnost oproti tomu vyjadřuje opačný vztah p a X: $L(p|X)$.

Rozdíl spočívá v tom, že pokud ze známého základního souboru vypočteme pravděpodobnosti všech možných výběrů, jejich součet bude roven 1. Pravděpodobnost nastolení konkrétní události je absolutní. V tomto smyslu však nelze chápat věrohodnost. Její hodnota vyjadřuje pravděpodobnost nastolení konkrétního výběru za předpokladu námi zvoleného základního souboru. Hypotetických základních souborů však lze vymyslet nekonečně mnoho, takže konkrétní hodnotu věrohodnosti je třeba chápat podmíněně, nikoliv absolutně. Vhodná volba základního souboru na základě maximalizace věrohodnosti je podstatou použití věrohodnosti jako chybové funkce.

MLE - odhad maximální věrohodnosti

„Odhad maximální věrohodnosti (MLE – maximum likelihood estimator) je obecná technika sloužící k určení parametrů modelu a vyjádření statistických vztahů v různých, zvláště nestandardních situacích“.

Protože věrohodnost celého výběru se vyjadřuje součinem pravděpodobností nastolení sledované události u dílčích prvků výběru při daném základním souboru, jsou hledanými parametry právě tyto pravděpodobnosti.

věrohodnost v alternativním rozložení

Mějme množinu výběru $y=\{1,0,\ldots\}$, kde 1 znamená nastolení sledované události A, 0 pak její nenastolení. Dále označme N počet prvků výběru a P pravděpodobnost nastolení události A. Cílem je stanovit věrohodnost L při zvoleném P. Její hodnotu vypočteme z následujícího vztahu:

$$L = \prod_{i=1}^{N} P^{y_i} (1-P)^{1-y_i} \tag{2.8}$$

věrohodnost v obecném

V obecném případě je pravděpodobnost P nahrazena pravděpodobnostní funkcí f_p, jejíž hodnota v daném bodě $\boldsymbol{x_i}$ vyjadřuje pravděpodobnost nastolení

případě

události *A*. Typický je výpočet věrohodnosti hypotézy, že veličina ***x*** je daného rozložení (pro konkrétní parametry) nebo že vztah veličin ***x*** a ***y*** popisuje určitá funkce za předpokladu známého rozložení chyby měření těchto veličin. V takových případech odpovídá funkce f_p funkci hustoty použitého rozložení. Funkce f_p může také v binárních klasifikátorech vyjadřovat pravděpodobnost, se kterou veličina ***x*** v konkrétní hodnotě $\boldsymbol{x_0}$ náleží do třídy 0 nebo 1 (logitový model). Vztah pro výpočet věrohodnosti vypadá následovně:

$$L = \prod_{i=1}^{N} f_p(x_i)^{\,y_i}\left[1 - f_p(x_i)\right]^{1-y_i} \tag{2.9}$$

parametrizace funkce f_p

S rostoucím *N* se hodnota *L* zmenšuje. Z důvodu lepší srozumitelnosti se pracuje s logaritmem součinu pravděpodobností. Dále řekněme, že funkce f_p má *p* neznámých parametrů $\boldsymbol{b}=\{b_1,\ldots,b_p\}$. Věrohodnost funkce f_p v konkrétních parametrech ***b*** pak vyjadřuje rovnice:

$$L = \sum_{i=1}^{N} y_i \log\left(f_p(x_i)\right) + \sum_{i=1}^{N} (1 - y_i)\log\left(1 - f_p(x_i)\right) \tag{2.10}$$

maximální věrohodnost a její odhad

ML je taková hodnota $L(\boldsymbol{b})$, která je největší dosažitelná ze všech kombinací parametrů vektoru ***b*** funkce f_p. MLE je proces vedoucí k nalezení ML. Analytický MLE pro běžná rozložení vychází ze znalosti první a druhé derivace *L*. Numerický MLE slouží pro řešení složitějších a vícerozměrných závislostí. Hledání optimálních parametrů ***b*** je v takovém případě proces iterační založený na trial-and-error numerických algoritmech (nejčastěji Newton-Raphsonova metoda).

využití maximální věrohodnosti

Věrohodnost se využívá k ověřování hypotéz, porovnávání různých modelů, kvantifikování míry informace atd. V řadě těchto aplikací bývá využívána poměrná věrohodnost (likelihood ratio). Hlavní ideou je skutečnost, že rozdíl logaritmů věrohodností dvou modelů vynásobený –2 má pro dostatečně velké vzorky dat přibližně χ^2 rozložení. Stupeň volnosti odpovídá počtu určených parametrů za předpokladu, že nulová hypotéza H_0 nemá žádné neznámé parametry. Pro χ^2 pak platí následující vztah:

$$\chi^2 = LR = -2 \cdot \log\left(L\ při\ H_0 / L\ při\ H_1\right) = -2 \cdot \left[\log(L_0) - \log(L_1)\right] \tag{2.11}$$

entropie

Shanonova entropie je používána jako chybová funkce pro kvalitativní výstupní data např. v některých algoritmech sloužících k vytváření rozhodovacích stromů (ID3). Máme-li soubor záznamů o *N* prvcích, každý záznam lze klasifikovat do právě jedné třídy $G_1, \ldots, G_c$ a pravděpodobnost klasifikace do každé ze tříd označíme $p(G_1), \ldots, p(G_c)$, pro výpočet entropie platí:

$$H = -\sum_{i=1}^{c} p(G_i)\log_c p(G_i) \tag{2.12}$$

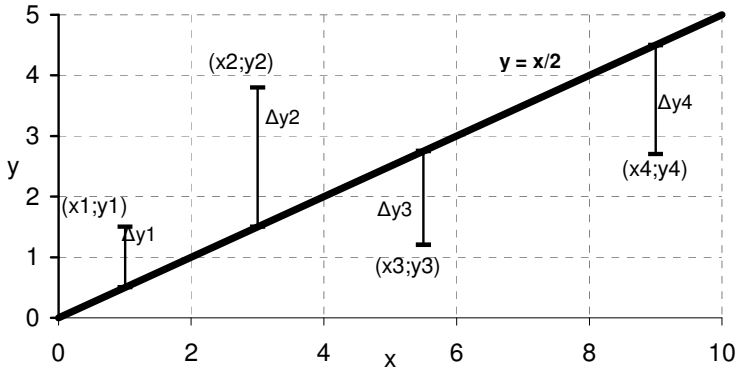
Vzhledem k tomu, že v případě klasifikace je $p(G_i)$ vypočteno jako podíl počtu záznamů N_i náležejících do třídy G_i ku všem záznamům *N*, odpovídá uvedený vztah věrohodnosti $-L$ podělené konstantou, tedy počtem všech záznamů, jak je z následujícího vztahu:

$$H = -\sum_{i=1}^{c} \frac{N_i}{N}\log_c \frac{N_i}{N} = -\frac{1}{N}\sum_{i=1}^{c} N_i \log_c \frac{N_i}{N} = -\frac{1}{N} \cdot L \tag{2.13}$$

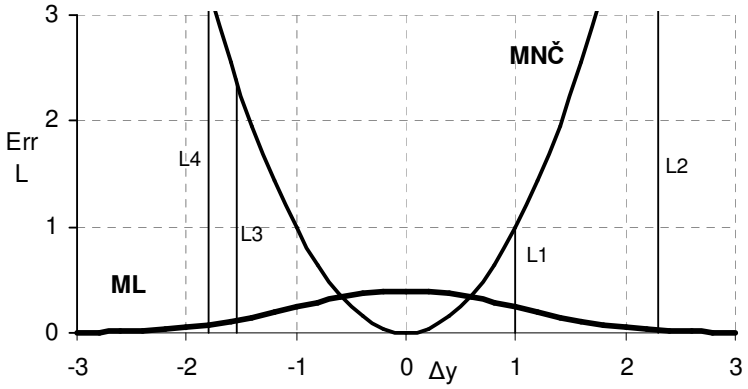
2.3 MNČ vs. MLE

příklad 2.1: lineární model

Na následujících dvou grafech je ideologické schéma výpočtu hodnotící funkce lineárního modelu pomocí MNČ a ML.



Obr. 2.1: Odchylky reálných dat od modelu *Δy*



Obr. 2.2: Transformace *Δy* podle MNČ a ML

Model je vyjádřen funkcí *y=x/2*. Pro 4 body je vyznačena jejich vzdálenost *Δy* od předpokládané hodnoty. Tato hodnota je v dalším grafu transformována do hodnot *L* a *Err*. V případě MNČ se s rostoucím |*Δy*| hodnota *Err* zvětšuje, proto je označována jako chyba; cílem je minimalizovat její hodnotu. Naopak u ML se s rostoucí absolutní hodnotou odchylky *L* zmenšuje. Je označováno jako věrohodnost a cílem je jeho maximalizace. Optimum nalezené na základě ML za předpokladu normálního rozložení chyby je nezávislé na zvoleném σ (pokud má chyba u všech naměřených hodnot stejný rozptyl). Oba postupy (MNČ i ML) vedou v takovém případě ke stejnému řešení.

Pro výpočet hodnotící funkce platí následující vztahy:

MNČ – minimalizace chyby *Err*

$$Err = \sum \Delta y_i^2 \qquad (2.14)$$

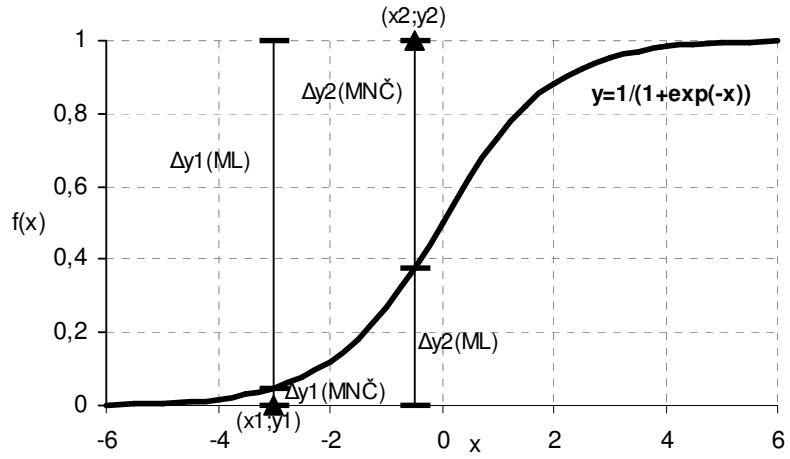
ML – maximalizace věrohodnosti *L*

$$L = \prod h(\Delta y_i) \qquad (2.15)$$

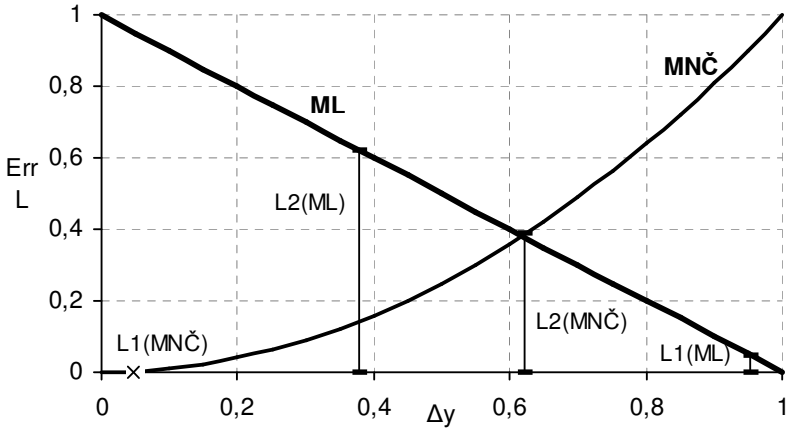
kde $h(\Delta y_i)$ je funkce hustoty normálního rozložení $N(f(x_i),\sigma^2)$.

příklad 2.2: logitový model

Na následujících dvou grafech je ideologické schéma výpočtu hodnotící funkce logitového modelu pomocí MNČ a ML.



Obr. 2.3: Odchylky reálných dat od modelu *Δy*



Obr. 2.4: Transformace *Δy* podle MNČ a ML

Model je vyjádřen logistickou funkcí $y=f(x)=1/[1+exp(-x)]$. V logitovém modelu se však hodnota *Δy* pro jednotlivé hodnotící funkce liší. MNČ se vypočítá stejně jako v předešlém příkladě. Při použití ML je $f(x)$ chápána jako pravděpodobnostní funkce, $f(x_i)=Pr\{y=1|x=x_i\}$. Pokud při daném x_i platí, že $y_i=1$, pak pravděpodobnost této události vyjadřuje skutečná hodnota $f(x_i)$. Pokud se $y_i=0$, pak se pravděpodobnost nastolení této události rovná $1-f(x_i)$. Pro hodnoty odchylky tedy platí, že $\Delta y(ML)=1-\Delta y(MNČ)$. Výpočet hodnotící funkce vyjadřují následující rovnice:

MNČ – minimalizace chyby *Err*

$$Err = \sum \Delta y_i^2 = \\ = \sum \left[f(x_i) - y_i \right]^2 \qquad (2.16)$$

ML – maximalizace věrohodnosti *L*

$$L = \prod \left(1 - |\Delta y_i|\right) = \\ = \prod \left[1 - y_i - f(x_i)(-1)^{y_i}\right] = \\ = \prod \left[f(x_i)^{y_i} (1 - f(x_i))^{1-y_i} \right] \qquad (2.17)$$

společné vlastnosti MNČ a MLE

Základní společnou vlastností obou uvedených metod je skutečnost, že vycházejí z transformace stejné vstupní informace – diference mezi predikovanou a skutečnou hodnotou. Lze tedy vypočíst ohodnocení modelu pro jednotlivý prvek a celková chyba je tvořena souhrnem chyb dílčích. Mějme ohodnocení modelu na základě dvou libovolných prvků tvořených uspořádanou dvojicí $(\boldsymbol{x};\boldsymbol{y})$. Ordinální relace mezi těmito ohodnoceními bude stejná pro obě uvedené metody. V mnoha případech vedou obě metody ke stejnému řešení.

rozdíly mezi MNČ a MLE

Základní rozdíl mezi uvedenými přístupy spočívá v typu transformační funkce, způsobu výpočtu souhrnné chyby modelu a informaci obsažené v celkové chybě modelu.

Transformace vstupní diference Δy je v případě MNČ funkce rostoucí, v případě MLE funkce klesající. Zatímco u MNČ je funkce označována za chybu a je minimalizována, v případě MLE se hovoří o věrohodnosti a její velikost je maximalizována. Souhrnná chyba je u MNČ získána součtem všech dílčích chyb, u MLE je použit součin jednotlivých věrohodností. Z toho také vyplývá, že u MLE má význam absolutní hodnota chyby, která vyjadřuje podmíněnou pravděpodobnost za předpokladu předem zvoleného typu rozložení chyby.

2.4 Další typy a modifikace chybových funkcí

chyba klasifikátoru

Chybu klasifikace lze posoudit jednoduše následující funkcí (obr. 2.5) představující binární chybu klasifikátoru:

$$Err\left(x_i, y_i, f(x_i,\boldsymbol{b})\right) = \begin{cases} 0 & , y_i = f(x_i,\boldsymbol{b}) \\ 1 & , jinak \end{cases} \tag{2.18}$$

Tato funkce však nemá schopnost rozlišovat mezi typem chyby a její závažností (odlišit různé typy chyb). Lze provést následující rozšíření:

$$Err\left(x_i, y_i, f(x_i,\boldsymbol{b})\right) = \begin{cases} 0 & , y_i = f(x_i,\boldsymbol{b}) \\ \tilde{E}rr\left(x_i\right) & , jinak \end{cases} \tag{2.19}$$

Uvedený přístup není stále pro regresní klasifikátory dostačující. Problémem je slučování kvalitativní klasifikace s kvantitativní regresí. Rozdílný přístup mezi těmito dvěma regresními typy modelů je vysvětlen v následujícím odstavci.

regresní klasifikátor, model a binární kódování

U regresního modelu je apriorně předpokládáno, že jeho výstup bude kvantitativní veličina. Proto je taky typickým vstupem jeho chybové funkce rozdíl mezi požadovanou a predikovanou hodnotou, $\Delta y_i = y_i - f(x_i)$.

Smyslem regresního klasifikátoru je klasifikace, často binární. Při použití tradičních parametrických modelů je předem dána kritická hodnota, jejíž překročení výstupem regresního modelu určuje změnu klasifikace. Pokud jsou výstupní třídy nahrazeny prvky z množiny {0;1}, je předpokládána kritická hodnota 0,5 (logitový model). Pokud jsou binární výstupy nahrazeny prvky z množiny {-1;1}, odpovídá kritická hodnota číslu 0 a jako klasifikátor slouží funkce $sgn(y_i)$. V takovém případě je jako vstup chybové funkce používán místo Δy_i součin $y_i.f(x_i)$. Má tu vlastnost, že nabývá kladné hodnoty, pokud jsou

znaménka funkce $sgn(f(x_i))$ a požadovaného binárního výstupu y z množiny {-1;1} stejná. Tak lze odlišit chybnou a správnou klasifikaci regresního klasifikátoru na základě znaménka uvedeného součinu.

lineární a kvadratická chybová funkce

Následující dvě chybové funkce jsou určené pro regresní klasifikaci do tříd {-1;1}. Jedná se o tzv. „soft margin" chybové funkce [43], přičemž první z nich je lineární

$$Err(x_i, y_i, f(x_i, \boldsymbol{b})) = \max(0, 1 - y_i \cdot f(x_i)) \tag{2.20}$$

a druhá kvadratická (obdoba MNČ).

$$Err(x_i, y_i, f(x_i, \boldsymbol{b})) = \max(0, 1 - y_i \cdot f(x_i))^2 \tag{2.21}$$

Jejich průběh v závislosti na $y_i.f(x_i)$ je patrný z grafu na obr. 2.5.

logistická chybová funkce

Další variantou chybové funkce pro regresní klasifikátory je tzv. logistická chyba (obr. 2.5), která vychází ze statisticky odvozené chybové funkce ML. Popisuje ji následující vztah

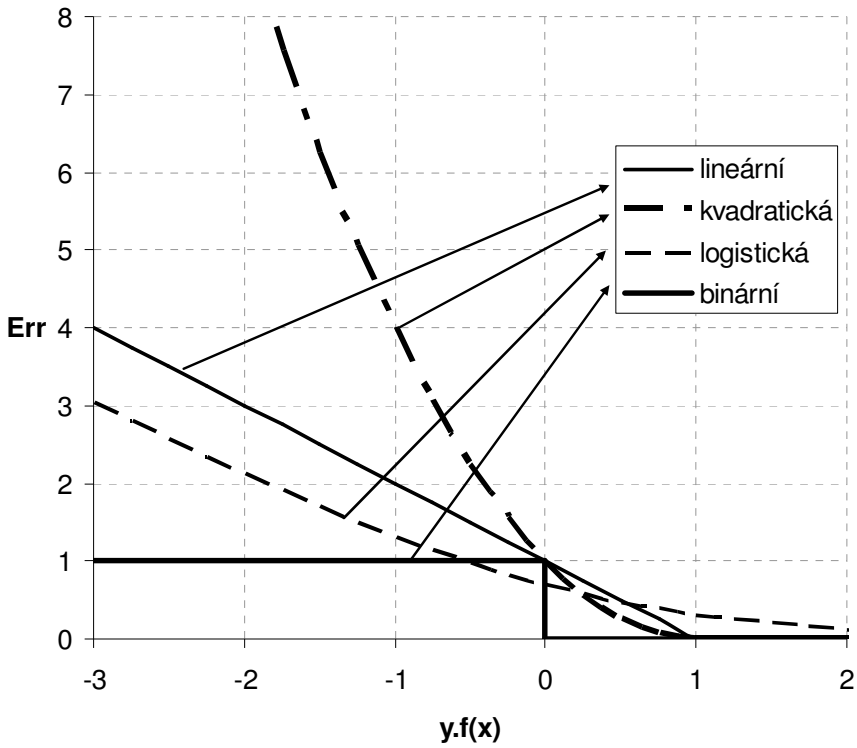
$$Err(x_i, y_i, f(x_i, \boldsymbol{b})) = \ln(1 + \exp(- y_i \cdot f(x_i))) \tag{2.22}$$

Jejím dalším rozšířením je např. tzv. log-log funkce (obr. 2.5):

$$Err(x_i, y_i, f(x_i, \boldsymbol{b})) = 1 - \exp(- \exp(- y_i \cdot f(x_i))) \tag{2.23}$$

průběhy různých chybových funkcí

Průběh výše popsaných chybových funkcí, jejichž vstupní hodnotou je součin $y_i.f(x_i)$, znázorňuje graf na obr. 2.5.



Obr. 2.5: Různé chybové funkce pro regresní klasifikátor

robustní chybové funkce

Robustností je obecně rozuměna necitlivost vůči malé odchylce od idealizovaných předpokladů. Z toho vyplývá, že robustní chybovou funkcí je např. varianta MNČ s pásmem necitlivosti ε (viz. kapitola 2.2.1). Obecně se robustní metody dělí do tří skupin. První vychází z maximální věrohodnosti (M-estimate), druhá používá lineárních kombinací pořadových statistik jako je např. medián (L-estimate) a poslední je založena na použití pořadových testů jako jsou neparametrické korelace a regresní koeficienty (R-estimate).

2.5 Chybová funkce v klasifikátorech

chybná klasifikace

Veličina kvalitativní nemá definovány operace sčítání a odečítání. Není tedy možné na základě hodnot G a $\hat{G}$ vypočítat přímo odchylku ΔG. Jednoduše by bylo možné nahradit chybnou klasifikaci např. hodnotou 1, správnou 0. Tato myšlenka je dobrá, jejím rozvinutím jsou tzv. *nákladové matice*.

nákladová matice

Nákladová matice je výstižně charakterizována již svým názvem. Nese informaci o nákladech spojených s chybnou klasifikací. Odlišením závažnosti chyby lze pak ovlivnit i to, jak bude vytvářený model naučen. Číselné ohodnocení závažnosti chyby však nemusí být triviální problém. Je zpravidla podmíněno znalostí modelované problematiky, k jejímu stanovení může být nutné provést analýzu výstupní veličiny.

binární klasifikace

Častým typem úloh je binární (dichotomní) klasifikace. Jde o rozhodování typu muž/žena, ano/ne atd. Z pohledu statistiky (testování hypotéz) rozlišujeme chybu I. a II. druhu. Chyba I. druhu (též chyba α) spočívá v mylném zamítnutí nulové hypotézy (a přijetí alternativní hypotézy), chyba II. druhu je nezamítnutí chybné nulové hypotézy (tab.2.1). Závažnost těchto dvou chyb se může zásadním způsobem lišit. Např. nulová hypotéza H_0 „pacient nemá akutní zánět slepého střeva“ je v případě chyby 1. druhu (H_0 platí, avšak my tuto hypotézu zamítli) ve svých důsledcích méně závažná než v případě chyby 2. druhu (kdy s chybnou hypotézu nezamítneme – pacient zánět má). Pro představu významu těchto informací si zopakujme, že místo termínu hypotéza, který je používán ve statistice, SU používá termín jiný – *model*.

Tabulka 2.1: Chyba 1. a 2. druhu

	H_0 platí	**H_0 neplatí**
H_0 nezamítnuta	správné rozhodnutí ($p \geq 1-\alpha$)	chyba 2. druhu ($p=\beta$)
H_0 zamítnuta	chyba 1. druhu ($p \leq \alpha$)	správné rozhodnutí ($p=1-\beta$)

Pokud by tedy na základě uvedených znalostí měla být vytvořena nákladová matice, skutečnost větší závažnosti chyby 2. druhu by se v ní měla projevit, jak je patrno z tabulky 2.2.

Tabulka 2.2: Ilustrativní nákladová matice

	Skutečnost: „0“ nemá zánět s.s.	**Skutečnost: „1“ má zánět s.s.**
Predikce: „0“ nemá zánět s.s.	**0**	**15**
Predikce: „1“ má zánět s.s.	**1**	**0**

Máme-li tedy diagnostický model, jehož výstup je roven 0 v případě, že se nejedná o zánět slepého střeva, a v případě opačném je roven 1, bude při učení záměna 0 za 1 15-krát závažnějším pochybením než pochybení obrácené. Stanovení konkrétních hodnot (čísla 1 a 15 jsou pouze pro názornost) může být obtížným úkolem, závisí m.j. na konečných požadavcích na model (poměr senzitivity, specificity a pozitivní prediktivní hodnoty). Podrobnější přehled o této problematice v kontextu ROC analýzy je obsažen v Appendixu 1.

vícerozměrná klasifikace

Nákladová matice pro vícerozměrnou analýzu má rozměry odpovídající počtu tříd, do kterých je klasifikováno. V SU se lze setkat s přístupem, že nulová hypotéza je příslušnost do jedné třídy, alternativní hypotéza je příslušnost do libovolné jiné. Tak je problém klasifikace do k tříd rozdělen na k možných případů. Každý model se vyjadřuje k možnosti příslušnosti prvku do jedné ze tříd, nejvěrohodnější hypotéza je pak vítězná a určuje výstupní hodnotu modelu.

regresní klasifikátory

Regresní klasifikátory jsou pro SU typické (neuronové sítě, podpůrné vektory, ...). Třídy jsou nahrazeny čísly, kvalitativní veličina je akceptována jako kvantitativní. Regresní model je nastaven pomocí MNČ nebo ML. Až na základě vzniklého regresního modelu jsou definovány pomocí kritických hodnot nebo jiného kritéria pravidla, na jejichž základě se výstup regresního modelu převádí zpátky na třídy, které jsou skutečným výstupem modelu. V převodu z kvalitativní veličiny na kvantitativní však může dojít ke změně informace v datech, respektive k jejím neoprávněnému přidání. Z toho důvodu je jako zpětná informace o skutečné přesnosti modelu používána např. ROC analýza.

2.6 Souhrn

Chybová funkce je uspořádaná trojice $(x,y,f(x))$, přičemž její hodnota kvantifikuje míru odlišnosti mezi požadovaným výstupem y a výstupem z modelu $f(x)=\hat{y}$. Chybové funkce se dělí podle typu výstupní veličiny. Pokud je y kvantitativní, používá se metoda nejmenších čtverců nebo maximální věrohodnost včetně celé řady jejich variant. Dalšími typy chybový funkcí jsou např. logistická nebo binární chybová funkce. U kvalitativních výstupních veličin se používají nákladové matice nebo např. entropie.

Kontrolní otázky a úlohy

1. Definuj chybovou funkci.
Funkce $L:X\times Y\times Y \rightarrow \langle 0;\infty)$, pro kterou platí, že pro $\forall\, x\in X$ a $\forall\, y\in Y$ je $L(x,y,y)=0$, je označována jako chybová funkce.

2. Uveď hlavní výhody a nevýhody modelů lineárních v parametrech z pohledu stanovení extrému funkce.
Hlavní předností modelů lineárních v parametrech je skutečnost, že pro nalezení optimálních hodnot jejich parametrů zpravidla existují známá analytická řešení. Nevýhodou modelů lineárních v parametrech je problematická aproximace složitějších závislostí a citlivost vůči prvkům, které jsou např. vinou chyby měření výrazně vzdáleny od prvků ostatních (ang. outliers). Pokud je použita linearizace složitější závislosti, je třeba brát zřetel na interval, ve kterém toto zjednodušení platí; závislost nelze extrapolovat.

3. Jaké znáte chybové funkce?
lineární, nelineární, váhová, robustní (s pásmem necitlivosti), polynomické, logitové, binomické, entropii, věrohodnost

4. Jaký je rozdíl mezi věrohodností a pravděpodobností?
Pravděpodobnost lze vyjádřit jako $P(X|p)$; pravděpodobnost X závisí na parametrech p modelu (základního souboru). Věrohodnost oproti tomu vyjadřuje opačný vztah p a X: $L(p|X)$. Rozdíl tedy spočívá v tom, že pokud ze známého základního souboru vypočteme pravděpodobnosti všech možných výběrů, jejich součet bude roven 1. Pravděpodobnost nastolení konkrétní události je absolutní. Věrohodnost vyjadřuje pravděpodobnost nastolení konkrétního výběru za předpokladu zvoleného základního souboru. Hypotetických základních souborů však lze vymyslet nekonečně mnoho, takže konkrétní hodnotu věrohodnosti je třeba chápat podmíněně, nikoliv absolutně.

5. Jaké jsou společné vlastnosti a rozdíly mezi MNČ a MLE?
Společnou vlastností je fakt, že vycházejí z transformace stejné vstupní informace – diference mezi predikovanou a skutečnou hodnotou. Rozdíly mezi uvedenými přístupy spočívají v typu transformační funkce, způsobu výpočtu souhrnné chyby modelu a informaci obsažené v celkové chybě modelu.

6. Co rozumíme robustností u chybové funkce?
Robustností je obecně rozuměna necitlivost vůči malé odchylce od idealizovaných předpokladů.

7. Co je to nákladová matice?
Nákladová matice nese informaci o nákladech (ztrátách) spojených s chybnou klasifikací.

8. Co je to dichotomní klasifikace?
Binární, tedy dvouhodnotová klasifikace.

Literatura:

[1] Bradley A.P.: The Use of the Area under the ROC Curve in the Evaluation of Machine Learning Algorithms. Pattern Recognition, 30(7): pp. 1145-1159, 1997.

[2] Corets C., Mohri M.: AUC Optimization vs. Error Rate Minimization. Advances in neural information processing systems 16. 2004, Cambridge MA: MIT Press.

[3] FARRAGI, D., REISER, B.: Estimation of the area under the ROC curve. Statistics in Medicine. 2002, 21:3093-3106.

[4] Fielding A.H., Bell J.F.: A review of methods for the assessment of prediction errors in conservation presence/absence models. Foundation for Environmental Conservation. 1997, pp. 38-49.

[5] FRANK, E., HARRELL, J. Regression Modeling Strategies. NY: Springer, 2001. 568 pages. ISBN 0-387-95232-2.

[6] Hastie T., Tibshirani R., Friedman J.: The Elements of Statistical Learning. Springer, 2001. ISBN 0-387-95284-5.

[7] HONZÍK, P. Area under the ROC Curve by Bubble-Sort Approach (BSA) In Automatic Control, Modeling and Simulation (ACMOS'05). 7th WSEAS International Conference on AUTOMATIC CONTROL, MODELING AND SIMULATION (ACMOS '05). Praha: WSEAS, 2005, s. 494 - 499, ISBN 960-8457-12-2

[8] Huber, P.J. 1981, Robust Statistics (New York: Wiley).

[9] Zapletal, J.: Základy počtu pravděpodobnosti a matematické statistiky. VUT Brno, PC-DIR s.r.o., 1995. ISBN 80-214-0711-5.

3 Genetické algoritmy

Cíle kapitoly:

Genetické algoritmy (GA) jsou silným nástrojem pro modelování složitých vícerozměrných nelineárních systémů. Po stručném úvodu o historii GA následuje část věnovaná základní terminologii a principům, na kterých jsou GA založeny. Dále jsou rozebrány jednotlivé operátory a jejich používané varianty. Kapitola je zakončena přehledem základních parametrů GA, doporučením jejich nastavení a uvedením příkladů z praxe.

3.1 Úvod

Genetické algoritmy (GA) jsou součástí evolučního počítání, které spadá do oboru umělé inteligence. Vycházejí z genetiky a jsou inspirovány Darwinovou teorií o evolučním vývoji. Idea evolučního počítání byla představena v roce 1960 I. Rechenbergem.

Zrod GA je datován do roku 1975 a je spojen se jménem John Holland. Tento americký teoretický biolog se zaměřil ve svém výzkumu na nalezení formální teorie adaptace. Vycházel mj. z knihy R.A.Fishera: „The Genetic Theory of Natural Selection“. V ní se mluví o tom, že evoluce, podobně jako učení, je formou adaptace na prostředí. Rozdíl je v tom, že působí v průběhu generací a ne v rámci jednoho života.

John Holland vydává roku 1975 knihu “Adaptation in Natural and Artificial Systems”, ve které shrnuje svůj dlouholetý výzkum a snaží se odpovědět na původně jednoduchou otázku: Proč se navzájem liší jedinci patřící ke stejnému živočišnému druhu? Tato kniha položila základ GA, které byly později rozvíjeny a modifikovány. Významnou roli v tomto výzkumu hráli Hollandovi studenti, které profesor zapojil do zkoumání detailů GA. Experimentovali s jejich parametry a snažili se standardizovat tvar algoritmu. Prvními aplikacemi jsou práce Hollandových studentů (například zjednodušená verze šachů nebo simulace funkcí jednobuněčného organismu).

Jedním z žáků Hollanda byl David Goldberg, který Hollandovu teorii dopodrobna rozpracoval. Roku 1989 vydává další klíčovou knihu “Genetic Algotithms in Search, Optimalization and Machine Learning”. Goldberg se v této knize snaží obhájit GA jako techniku obecně aplikovatelnou na širokou třídu úloh. Publikace věnované GA dodnes vychází z Goldbergových výsledků a opírají se o ně.

Ke konci 70. let došlo k velkému rozvoji klasické umělé inteligence, která svým úspěchem GA zcela zastínila.

V 80.letech opět vrostl zájem o GA především díky pracím Davida Goldberga a dalších. Rozšířila se idea o jejich všemohoucnosti.

V 90. letech došlo k vystřízlivění. Efektivita GA není vždycky tak velká, jak se předpokládalo. Řada vědců se dodnes snaží GA různě modifikovat a vylepšit, aby se problémy vyřešily. Tvůrcem významných modifikací je Zbigniew Michalewicz, autor knihy “Genetic Algorithms + Data Structures = Evolution Programs“ (1996)).

Kromě řady modifikací přinesla 90. léta i nový směr evolučních výpočetních technologií, tzv. genetické programování.

3.2 Základy genetických algoritmů

Kapitola začíná přehledem nejnutnější terminologie. Dále je vysvětlen princip a smysl základních operátorů a operandů přirovnáním k jejich zjednodušeným biologickým předlohám. Nakonec je uveden princip samotného algoritmu.

3.2.1 Přehled základní terminologie

Termíny lze rozdělit do tří skupin na operandy, operátory a ostatní.

Operandy:

Gen – konkrétní část chromozomu, popisuje jednu konkrétní vlastnost.

Generace – populace jedinců v daném časovém okamžiku; nová generace vzniká z rodičů a jejich potomků.

Genom – soubor všech genetických materiálů charakterizujících jednoho jedince (jednoho jedince může charakterizovat i více chromozomů).

Chromozom – obecná genetická informace ve tvaru řetězce; sekvence symbolů.

Jedinec – nositel genetické informace.

Populace – množina jedinců.

Potomek – jedinec vzniklý rekombinací rodičů (křížení a mutace).

Rodič – jedinec vstupující do rekombinace.

Operátory:

Elitismus – konkrétní metoda/technika selekce.

jedinec[] = elitismus(generace);

Fitness (kvalita, účelová funkce, hodnotící funkce, kriteriální funkce, funkce vhodnosti) – síla jedince v generaci; odvozuje se od ní jeho šance na přežití.

číslo = fitness(jedinec);

Kódování – zapsání vlastností pomocí symbolů; základní jsou kódování binární a pomocí reálných čísel.

chromozom = kódování(skutečné vlastnosti – barva, hmotnost, …);

Křížení – konstrukce nových jedinců z vybraných jedinců generace – rodičů.

jedinec = křížení(rodič1,rodič2);

Mutace – změna hodnoty genu v chromozomu.

jedinec = mutace(jedinec);

Selekce (přirozený výběr) – výběr jedinců, kteří přežívají z generace N-1 a stanou se rodiči potomků, s nimiž vytvoří novou generaci N.

rodiče_nové_generace[] = selekce(generace(N-1));

Ukončovací podmínka (zastavovací pravidlo) – definuje konec výpočtu; končí po určitém počtu generací; po *K* krocích (např. *50*) zůstává nejlepší jedinec stále stejný.

boolean = ukončovací podmínka (generace(N));

Ostatní:

Adaptace – přizpůsobení se prostředí

Evoluční počítání – oblast UMI; vychází z Darwinovy teorie; hledání řešení problému na základě mechanizmů přirozeného vývoje;

Genetický algoritmus – postup řešení problému založený na teorii Charlese Darwina; principem je přežití lépe přizpůsobených jedinců.

3.2.2 Princip genetických algoritmů

Hlavním principem GA je napodobovat biologickou evoluci tak, jak je v dnešní době chápána na základě teorie Charlese Darwina. Velice zjednodušeně lze princip evoluce popsat následujícím způsobem.

Každého jedince charakterizují určité atributy (geny), jako jsou např. výška, barva vlasů, počet dioptrií atd. Soubor všech těchto atributů jednoho jedince (genom, u jednoduchých GA a v příkladech těchto skript tvořen jediným chromozomem) určuje jeho celkový charakter. Cenné jsou z hlediska vývoje změny ve vlastnostech, které zvyšují šanci jedince na přežití (adaptace). V GA se míra adaptace na prostředí, ve kterém jedinec žije, vyjadřuje kvantitativně číselnou hodnotou (fitness, hodnotící funkce). Čím je tato hodnota větší, tím je jedinec kvalitnější; je lépe přizpůsoben. Šanci na přežití mají hlavně ti nejlepší, v menší míře ti, kteří měli jednoduše štěstí (selekce). Selekční tlak umožní přežít pouze některým jedincům. Ti se stávají rodiči a mají své vlastní potomky. Tito noví jedinci vzniknou jednak kombinací vlastností svých rodičů (křížení) a také náhodnými chybami, které se během procesu křížení vyskytnou (mutace). Soubor těchto potomků a jejich rodičů (generace) je opět podroben selekčnímu tlaku a ti, kteří přežijí a stanou se opět rodiči, vytvoří novou generaci. Tento cyklus vývoje, který upřednostňuje přežití lépe adaptovaných jedinců, se nazývá evoluce.

Ideové schéma GA lze zapsat následujícím způsobem:

1. **Vytvoření počáteční generace**. Na počátku je třeba první generaci náhodně vygenerovat.
2. **Vyhodnocení počáteční generace**. Zjistíme míru adaptace všech jedinců.
3. **Selekce**. Přežili hlavně nejlépe adaptovaní jedinci a někteří slabší šťastlivci.
4. **Křížení, mutace**. Z přeživších jedinců byli vytvořeny páry rodičů, které mají potomky, s nimiž vytvoří novou generaci.

5. **Vyhodnocení nové generace**. Zjištěna míra adaptace všech jedinců. Kvalita jedinců je často součástí ukončovací podmínky.
6. **Ukončovací podmínka**. Pokud je splněna, konec. Pokud není splněna, pokračuje se od bodu 3.

3.3 Operátory genetických algoritmů

Existuje velké množství různých implementací genetických operací. Cílem této části skript je představit nejběžnější operátory (kódování, křížení, mutaci, selekci, hodnotící funkci) a vysvětlit principy, na jejichž základě fungují.

3.3.1 Kódování

Binární kódování

Kódování problému pomocí binárních čísel patří mezi nejpoužívanější metody. Každý chromozom je tvořen řetězcem konečné délky složeným z nul a jedniček. Existují dvě základní možnosti, jak tento chromozom chápat. Buď každý jednotlivý gen interpretuje binární informaci, zda určitou vlastnost jedinec má či nemá a nebo lze celý chromozom chápat jako binárně zapsané číslo (Obr. 3.1). Pokud známe interval, ve kterém se pohybuje hledané řešení a maximální požadovanou přesnost hledaného parametru, je možné na základě těchto požadavků vytvořit adekvátně dlouhý chromozom.

Chromozom A	101100101100101000000000
Chromozom B	111111100000110000000000

Obr. 3.1: Binárně kódovaný chromozom

Pokud potřebujeme pracovat s vícerozměrnými modely, jejichž parametry tvoří reálná čísla s pohyblivou řádovou čárkou, použití binárního kódování může být příliš složité a nepraktické. V takových případech se nabízí varianta kódování pomocí reálných čísel.

Kódování pomocí reálných čísel

Chromozom se skládá z řetězce reálných čísel, která vyjadřují hodnoty jednotlivých genů (Obr. 3.2). Oproti binárnímu kódování umožňuje tento přístup při stejné numerické přesnosti řešení podstatně kratší a čitelnější zápis. Odpadá také převod mezi binárním a dekadickým zápisem. Nevýhodou je ale větší volnost při hledání vhodných typů ostatních genetických operátorů (zejména křížení a mutace). Jejich správná volba má zásadní vliv na úspěšnost řešení.

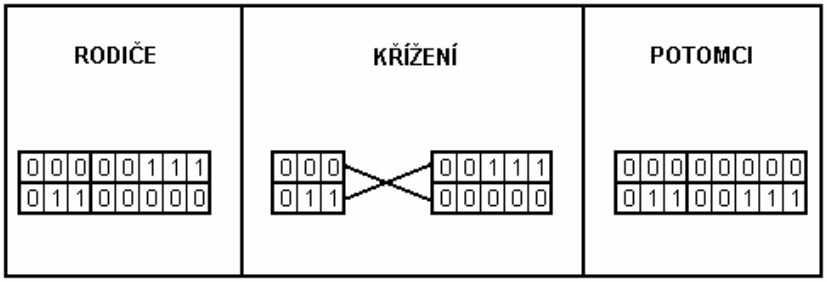
Chromozom A	1.2465	2.6461	4.4992	1.3468	0.2464
Chromozom B	0.6495	4.0216	2.8762	3.4256	3.6289

Obr. 3.2: Reálně kódovaný chromozom

3.3.2 Křížení

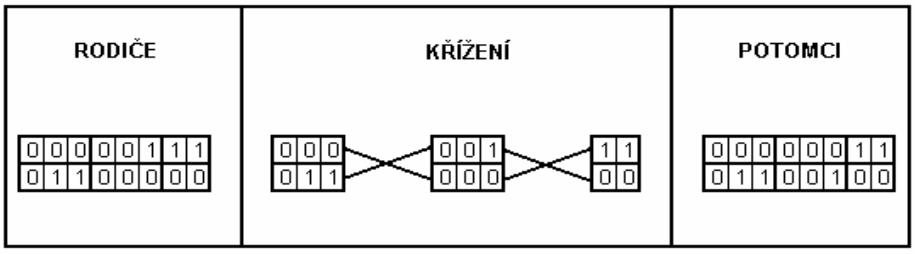
Křížení binárně kódovaných jedinců

Na Obr. 3.3 je ukázán postup mechanismu křížení binárně kódovaných jedinců. Jako rodiče jsou použity chromozomy 00000111 a 01100000. V řetězci délky 8 existuje sedm míst, kde jej lze rozdělit na dvě části. Náhodnou volbou čísla v rozmezí od jedné do sedmi je nejdříve zvoleno místo, kde k rozdělení dojde. Padne-li například číslo 3, znamená to, že první potomek bude mít první tři pozice v řetězci shodné se svým prvním rodičem. Zbývajících 5 pozic bude doplněno informacemi ze shodných pozic druhého rodiče. Přesně naopak je tomu u druhého potomka. Křížením vzniknou dva noví jedinci, kteří nesou nově zkombinované informace svých rodičů.



Obr. 3.3: Křížení jedinců

Další metodou křížení je tzv. dvojité křížení. Princip spočívá v tom, že se rodiče kříží na dvou různých místech. Ta jsou opět určena dvěma náhodnými čísly. Padnou-li například čísla 3 a 6, rodiče se budou křížit nejprve na třetí a poté na šesté pozici (Obr. 3.4).



Obr. 3.4: Křížení jedinců

Chromozomy se mohou štěpit i na více místech. Většinou se však volí pouze jeden nebo dva dělící body. Na základě těchto principů může ze dvou slabých rodičů vzniknout silný potomek.

Křížení jedinců kódovaných pomocí reálných čísel

Při použití kódování pomocí reálných čísel je teoreticky možné použít metody křížení používané u binárního kódování. Každá z reálných hodnot chromozomu však může nabývat hodnot z jiného intervalu a vznikaly by problémy s rozsahem jednotlivých parametrů. Řešení takového problému je možné, leč těžkopádné a v konečné podobě neefektivní. Vhodnější je proto zavést jiné metody křížení. Jejich volba závisí zejména na typu řešeného problému.

Jedna z metod křížení, kterou lze u chromozomu složeného z reálných hodnot použít, je dána následujícím vztahem:

$$\boldsymbol{P} = R \cdot \boldsymbol{A} + (1 - R) \cdot \boldsymbol{B} \qquad (3.1)$$

kde $\boldsymbol{P}$ je potomek vzniklý z křížení rodičů

$\boldsymbol{A}, \boldsymbol{B}$ jsou vybraní jedinci pro křížení (rodiče)

R je náhodně vybrané číslo z intervalu *(0,1)*.

Tímto postupem vznikne první potomek z rodiče $\boldsymbol{A}$ a $\boldsymbol{B}$. Změnou pozice rodičů vznikne druhý jedinec. Metodu lze rozšířit tak, aby byl obor hodnot (možných potomků) větší a tím bylo křížení účinnější. Nový vztah pak může vypadat následovně:

$$\boldsymbol{P} = \left| \frac{|\boldsymbol{A} - \boldsymbol{B}|}{2} + \left| \left(\left(R \cdot K - \frac{K}{2} \right) \cdot |\boldsymbol{A} - \boldsymbol{B}| \right) \right| \right| \qquad (3.2)$$

Přibyl ovšem další parametr K, jehož hodnota by měla být větší než 1, (např. 1,5).

Na základě dále uvedených tří vzorců lze vytvořit ze dvou vybraných rodičů $\boldsymbol{A}$ a $\boldsymbol{B}$ tři nové potomky. Je zjištěna jejich kvalita a dva nejsilnější postoupí do nové generace.

$$\boldsymbol{P}_1 = \frac{\boldsymbol{A} + \boldsymbol{B}}{2} \qquad (3.3)$$

$$\boldsymbol{P}_2 = \frac{3 \cdot \boldsymbol{A} - \boldsymbol{B}}{2} \qquad (3.4)$$

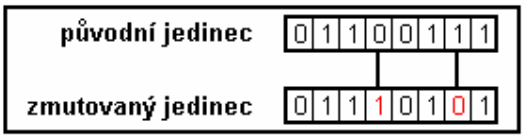
$$\mathrm{P}_3 = \frac{-\boldsymbol{A} + 3 \cdot \boldsymbol{B}}{2} \qquad (3.5)$$

U většiny metod křížení je třeba hlídat meze, aby nedošlo k překročení intervalu definičního oboru parametrů.

3.3.3 Mutace

Mutace binárně kódovaných jedinců

Při použití binárního kódování se s každým genem s předem určenou pravděpodobností provádí operace záměny, což znamená, že pokud je na dané pozici jednička, je mutací změněna na nulu a naopak. Ukázka mechanizmu mutace je uvedena na Obr. 3.5.



Obr. 3.5: Mutace jedinců

Mutací vznikne zcela nový jedinec s novou kvalitou (Tabulka 3.1).

Tabulka 3.1: Kvality jednotlivých jedinců po mutaci

		BIN -> DEC, hodnota x	KVALITA f(x)
Původní jedinec	0 1 1 0 0 1 1 1	103	7,1323
Zmutovaný jedinec	0 1 1 1 0 1 0 1	117	7,3318

Mutace jedinců kódovaných pomocí reálných čísel

Mutace chromozomu skládajícího se z reálných hodnot může být principiálně shodná s mutací binárního chromozomu.

1) číslo na vybrané pozici je zvýšeno nebo sníženo o předem danou hodnotu

2) číslo na vybrané pozici je nahrazeno náhodně vygenerovaným číslem ze zadaného intervalu

Rodič	1.29 5.68 **2.86 4.11** 5.55
Potomek	1.29 5.68 **2.73 4.22** 5.55

Obr. 3.6: Mutace jedinců

Dalším druhem mutace je tzv. dynamická mutace. Z. Michalewicz zavedl nový operátor nazývaný dynamická mutace. Ten mění své vlastnosti v závislosti na čase. Pravděpodobnost mutace je na začátku vyšší a dále s počtem generací klesá podle následujících dvou vzorců:

$$x_i^* = \begin{cases} x_i + \Delta(t, X_{MAX} - x_i) \\ nebo \\ x_i + \Delta(t, x_i - X_{MIN}) \end{cases} \qquad (3.6)$$

$$\Delta(t, y) = y\left(1 - r^{\left(1-\frac{t}{T}\right)^B}\right) \qquad (3.7)$$

kde

x_i je původní hodnota genu před mutací

x_i^* je nová hodnota zmutovaného genu

X_{MAX} je maximální možná hodnota proměnné x_i

X_{MIN} je minimální možná hodnota proměnné x_i

T je maximální počet generací

r je náhodné číslo z intervalu $\langle 0,1 \rangle$

B je parametr nelinearity mutace

3.3.4 Selekce

Vážená ruleta

Jedním ze základních způsobů selekce je výběr pomocí vážené rulety. Váženou ruletu si lze představit jako klasickou ruletu s jedním rozdílem. Členění klasické rulety je rovnoměrné a vymezuje každému možnému losovanému číslu kruhovou výseč o stejném úhlu. U vážené rulety je pravděpodobnost výběru a tedy i velikost kruhové výseče určena kvalitou jedince. Pravděpodobnost, že bude jedinec vybrán se počítá následovně.

$$P_i = \frac{F_i}{\sum_{j=1}^{N} F_j} \qquad (3.8)$$

kde P_i je pravděpodobnost volby výseče na ruletě daného jedince

F_i je kvalita jedince pro kterého výseč počítáme

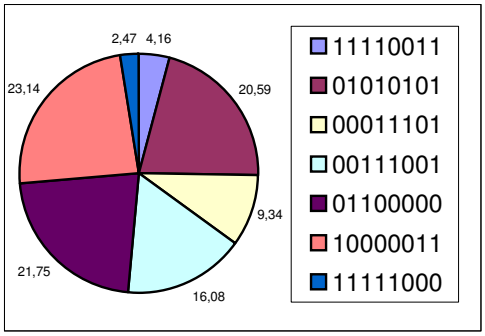
$\sum_{j=1}^{N} F_j$ je suma kvalit všech jedinců v generaci

Rozložení pravděpodobností na ruletě může například vypadat tak, jak je uvedeno v Tabulka 3.2.

Tabulka 3.2: Pravděpodobnosti jednotlivých jedinců

JEDINEC								KVALITA	PRAVDĚPODOBNOST (%)
1	1	1	1	0	0	1	1	1,794	4,16
0	1	0	1	0	1	0	1	8,889	20,59
0	0	0	1	1	1	0	1	4,031	9,34
0	0	1	1	1	0	0	1	6,943	16,08
0	1	1	0	0	0	0	0	9,389	21,75
1	0	0	0	0	0	1	1	9,992	23,14
1	1	1	1	1	0	0	0	1,067	2,47

Rozložení jednotlivých kruhových výsečí podle pravděpodobností vyplývajících z Tabulka 3.2. je na Obr. 3.7.



Obr. 3.7: Procentuální rozložení na ruletě

Představme si, že ruletu sedmkrát roztočíme a získáme tak skupinu nových jedinců. Ta může vypadat jako v Tabulka 3.3:

Tabulka 3.3: Kvality vylosovaných jedinců

JEDINEC								KVALITA
0	0	1	1	1	0	0	1	6,943
0	1	0	1	0	1	0	1	8,889
0	0	0	1	1	1	0	1	4,031
0	1	1	0	0	0	0	0	9,389
1	0	0	0	0	0	1	1	9,992
1	1	1	1	1	0	0	0	1,067
0	1	1	0	0	0	0	0	9,389

Do „vylosované“ skupiny se dostali někteří jedinci vícekrát. Tento fakt odpovídá skutečnosti, že např. pravděpodobnost vylosování jedince *01100000* je přibližně *0,22*. Porovnáním průměrné kvality jedinců původní a vylosované skupiny zjistíme, že došlo k jejímu výraznému nárůstu. Tento výsledek je dán způsobem výběru jedinců pro další generaci. Upřednostněni jsou jedinci s vyšší kvalitou.

Poziční selekce (Rank Selection)

Výše uvedený typ selekce „vážená ruleta“ má jeden závažný nedostatek. Stane-li se, že například nejlepší jedinec zabere 90% z celé rulety, budou mít potom ostatní jedinci velmi malou šanci, že dojde k jejich výběru. Tak může algoritmus velice rychle uvíznout v lokálním extrému. Algoritmy typu poziční selekce existují v řadě variací. Jejich společnou charakteristikou je, že chromozomy v generaci jsou seřazeny podle velikosti hodnotící funkce a dále se již pracuje pouze s pořadím jednotlivých jedinců. Při použití tohoto algoritmu je však třeba v každém cyklu seřadit jedince podle hodnotící funkce.

Nejjednodušší z postupů funguje tak, že jsou na počátku jedinci seřazeni podle hodnotící funkce, jejíž hodnota je pak nahrazena indexem každého z chromozomů. Nejhorší bude mít kvalitu 1, druhý nejhorší bude mít kvalitu 2 a tak pokračujeme dál, až se dostaneme k nejlepšímu, který bude mít kvalitu N (což jest rovno počtu jedinců v generaci). S těmito novými hodnotami pracujeme dále stejně jako v případě vážené rulety. Uvedený typ selekce omezuje zvýhodnění silnějších jedinců, což se může projevit pomalejší konvergencí ke správnému řešení a větším kolísáním průměrné kvality nových generací. Riziko uvíznutí v lokálním extrému je však oproti předešlému typu selekce menší.

Metoda TURNAJ

Jednou z dalších používaných metod je selekce typu TURNAJ. Hlavním důvodem je jednoduchost implementace při zachování kvality selekčního tlaku. Z populace jsou náhodně vybírány dvojice různých jednotlivců, které a se podrobují „souboji o přežití“. Ten spočívá v tom, že jedinec s vyšší kvalitou přežívá a postupuje do další generace. Tím se splňuje základní předpoklad o přežití lepších chromozomů.

Tabulka 3.4: Příklad metody Turnaj

Jedinec	Kvalita
1	4
2	4,2
3	4,05
4	4,17

Souboj	Vítěz
1 z 2	2
3 z 1	3
4 z 2	2
4 z 3	4

Z příkladu (Tabulka 3.4) je patrné, že je-li do souboje vybrán nejsilnější jedinec, tak vždy do nové generace postoupí. Může se však stát, že nebude do žádného souboje vylosován. To lze ošetřit modifikovanou implementací algoritmu nebo použitím tzv. elitismu. Nejslabší jedinec nepostoupí za žádných okolností.

Elitismus

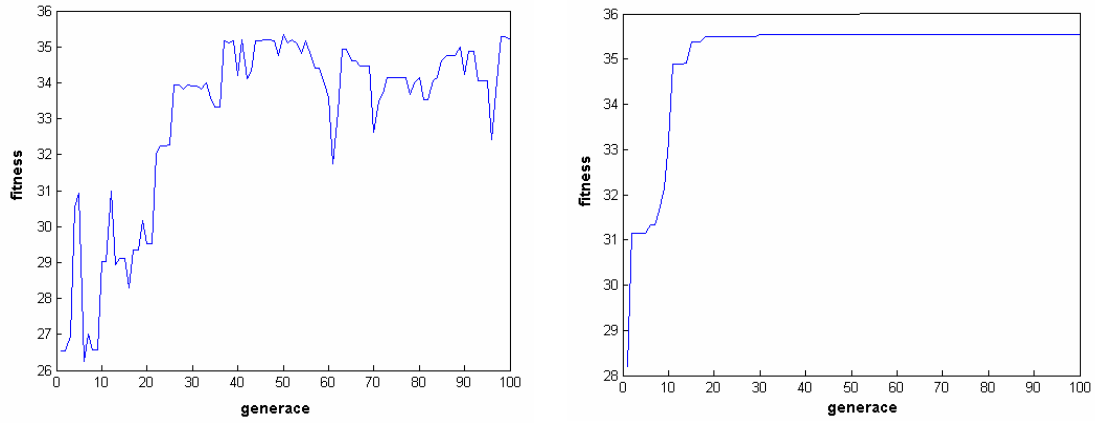
Elitismus je jednoduchá technika, kterou je vybrán určitý počet nejlepších jedinců. Ti se nepodrobují selekčnímu tlaku a postupují do nové generace přímo. Obvykle se volí jeden nebo dva nejlepší jedinci.

Příklad:

Mějme hodnotící funkci (fitness) následujícího tvaru:

$f(x,y) = x + (10 \cdot \sin(5 \cdot x)) + (7 \cdot \cos(4 \cdot y)) + y$, kde $x \in \langle 0,10 \rangle$ a $y \in \langle 0,10 \rangle$.

Hledány jsou takové hodnoty x a y v zadaném intervalu, pro které bude hodnotící funkce nabývat maxima.



Obr. 3.8: a) Bez použití elitismu b) s použitím elitismu

Jak je z Obr. 3.8a patrné, v průběhu výpočtu GA byl nejsilnější jedinec mnohokrát ztracen. Podařilo se vždy najít nového silného jedince, což ovšem není pravidlem. Vzhledem k počtu generací bylo dosaženo i hledaného maxima.

Průběh hledání řešení pomocí elitismu znázorňuje Obr. 3.8b. Elitismus se projeví tak, že před selekčním výběrem, křížením a mutací jedinců je nalezen v populaci nejsilnější jedinec a ten je uložen. Po provedení třech zmíněných operací se v nové populaci nahradí nejslabší jedinec uloženým nejsilnějším jedincem. Takto řešený problém vykazuje lepší výsledky, než když elitismus použit není.

3.3.5 Fitness – hodnotící funkce

Vstupem hodnotící funkce (funkce vhodnosti, kriteriální funkce, fitness funkce) je jedinec (chromozom) a výstupem jeho kvalita. Čím je výstupní hodnota vyšší, tím je jedinec lepší. Volba hodnotící funkce závisí na typu optimalizačního procesu.

Jednoduchým příkladem je hledání hodnot parametrů, při nichž dosahuje zadaná funkce svého maxima. Hodnotící funkcí je pak přímo zadaná funkce.

Příklad:

Mějme chromozom 00110011. Převedeme jej z binárního zápisu na dekadický (jednoduchý celočíselný převod) a získáme hodnotu 51. Dosazením této hodnoty do zkoumané funkce $f(x) = 1 + 0.05 \cdot x + \cos(0.02 \cdot \pi \cdot x)$ obdržíme funkční hodnotu *f(x)* v bodě 51, což je zároveň kvalita uvedeného jedince.

Obvykle je však volba hodnotící funkce složitější. Například k získání kvality jedince v optimalizaci výroby pomocí GA je možno optimalizovat podle doby trvání výrobního procesu, velikosti nákladů, zisku a dalších parametrů. Při návrhu optimální konstrukce mostu lze optimalizovat například podle hmotnosti, nosnosti, nákladů atd. Za těchto okolností je třeba volit hodnotící funkci jako kompromis mezi jednotlivými požadavky. To je obvykle problém, především mají-li požadavky různé priority.

Celkové vyhodnocení generace pak spočívá ve výpočtu kvality všech jedinců v generaci. Navíc se obvykle vypočítávají další důležité statistické parametry (průměrná kvalita jedinců, rozptyl kvalit v generaci, kvalita nejlepšího jedince atd.). Tyto parametry mohou být užitečné při vyhodnocování zastavovacího pravidla a v celkovém hodnocení úspěšnosti algoritmu.

3.4 Praktická aplikace genetických algoritmů

V této kapitole se budeme zabývat nastavením základních parametrů GA. Jak bude ukázáno, jeden problém lze řešit takřka nekonečným množstvím různě nastavených algoritmů. Řada z nastavení přitom povede ke správnému řešení. Přesto i v takových případech musíme brát zřetel na skutečnost, že i se stejným nastavením a stejnou výchozí generací nemusíme získat identické výsledky. Stejně jako v životě, i v genetickém modelování hraje významnou roli náhoda. V poslední části jsou uvedeny některé aplikace z praxe a příklad řešený pomocí GA.

3.4.1 Přehled parametrů GA

Během řešení libovolného problému pomocí GA je nezbytné učinit rozhodnutí o typu operátorů a jejich konkrétním nastavení. Dále je třeba stanovit, s jakou pravděpodobností budou zvolené operátory použity. Parametry GA se většinou rozumí právě zmíněné pravděpodobnosti. Ani ostatním rozhodnutím se však nelze vyhnout a jejich nevhodná volba může výrazně snížit výkonnost celého algoritmu. Před použitím GA na řešení libovolného problému je třeba učinit mj. tato rozhodnutí:

1. Jak budou hledané proměnné **kódovány**?
2. Jak velká bude **počáteční populace**?
3. Jaká bude použita **hodnotící funkce**?

4. Jaký bude použit **typ selekce**?
5. Jaký bude použit **typ křížení**?
6. S jakou **pravděpodobností** bude **křížení** provedeno?
7. Jaký bude použit **typ mutace**?
8. S jakou **pravděpodobností** bude provedena **mutace**?
9. Jaká bude **ukončovací podmínka**?

Jak je zřejmé, stupeň volnosti GA je opravdu veliký. Následující podkapitola nabízí určité vodítko se základními doporučeními. Bohužel neexistuje žádné obecně správné nastavení. Dále je třeba si uvědomit, že významnou roli v GA hraje náhoda (vygenerovaná počáteční populace, pravděpodobnosti použití jednotlivých operátorů atd.). I se stejným počátečním nastavením probíhá pak aproximace řešení různými cestami a konečné nejlepší nastavení hledané proměnné nemusí být při opakovaném pokusu identické nebo bude nalezeno po odlišném počtu generací. Na to je třeba brát zřetel zejména při nastavování ukončovacích podmínek.

3.4.2 Nastavení parametrů GA

1) Jak budou hledané proměnné kódovány?

Najít způsob, jak vhodně kódovat stavy jednotlivých proměnných, může být velice složité. Lze kódovat proměnné kvantitativní (spojité, intervalové) i kvalitativní (ordinální, nominální).

Kvantitativní proměnné (nevíme-li, v jakých mezích se řešení bude pohybovat) se nejlépe kódují pomocí reálných čísel. Binární kódování by bylo při více proměnných nepřehledné a nepraktické (5 proměnných zapsaných po 32 bitech znamená chromozom s 5*32=160 geny).

Pokud je předem znám interval, ve kterém se nachází hodnoty hledaných parametrů a známe požadovanou přesnost řešení, lze použít reálné i binární kódování. V případě binárního kódování by počet kombinací jedné vlastnosti měl být minimálně dvojnásobný oproti počtu kombinací vyplývajících z požadované přesnosti. Např. hledáme-li číslo v intervalu *(–7;12)* s přesností na dvě desetinná místa, existuje *[12-(-7)]/0,01=1900* kombinací. Počet bitů, do kterého takové číslo kódujeme, by měl mít alespoň *2*1900=3600* možných stavů. Nejbližší takový řetězec má 12 bitů, tedy 4096 kombinací. Lze použít i reálné kódování. V jednotlivých operátorech je však třeba ošetřit okrajové podmínky. Křížením i mutací mohou vzniknout proměnné, které se ocitnou mimo rozsah definičního oboru. V případě překročení těchto mezí se proměnná nastavuje na svoji maximální (minimální) hodnotu.

Kvalitativní proměnné lze kódovat například jako intervalové proměnné do reálných čísel. Pokud má proměnná např. 9 možných stavů (9 barev), lze např. zvolit interval <0;9).

Binární proměnné se kódují pomocí binárního kódování.

2) Jak velká bude počáteční populace?

Příliš malá populace způsobí, že proběhne relativně málo křížení, což může mít za následek prohledání jen malé části definičního oboru. Existuje pak vysoké riziko, že algoritmus uvízne v lokálním extrému. Naopak velká populace GA výrazně zpomaluje.

Nejčastěji doporučovaná velikost počáteční populace se pohybuje kolem 20-30 jedinců, lze se však setkat i s doporučeními mezi 50-100 jedinci.

Obecně platí, že má-li chromozom 16 genů, optimální počáteční populace by měla mít méně jedinců než v případě chromozomu tvořeného 32 geny.

3) Jaká bude použita hodnotící funkce?

Pokud je hledán extrém nějaké funkce, jako hodnotící funkce je použita funkce samotná. V případě aproximace bodů funkcí nebo např. hledání nastavení regulátoru se používá běžná chybová funkce (metoda nejmenších čtverců, ITAE kritérium atd.). Často je však volba hodnotící funkce složitější. Například k získání kvality jedince v optimalizaci výroby pomocí GA je možno optimalizovat podle doby trvání výrobního procesu, velikosti nákladů, zisku a dalších parametrů. Při návrhu optimální konstrukce mostu lze optimalizovat například podle hmotnosti, nosnosti, nákladů atd. Za těchto okolností je třeba volit hodnotící funkci jako kompromis mezi jednotlivými požadavky, což může být problém, především mají-li požadavky různé priority.

4) Jaký bude použit typ selekce?

Vážená ruleta je metoda vhodná pro řešení jednodušších problému. U složitých problémů může uvíznout v lokálním extrému. Problém nastává, pokud není známo, v jakém intervalu se pohybuje správné řešení, které je přitom samo o sobě hodnotící funkcí (např. hledáme extrém nějaké složité vícerozměrné funkce). Výpočet podle prvního uvedeného vzorce pak není přímo možný. Lze buď použít nějakou vhodnou substituci (např. jako interval jedné generace použít rozdíl mezi hodnotou kvality nejlepšího a nejhoršího jedince) nebo použít jinou selekční metodu.

Poziční selekce je alternativou vážené rulety. Při selekci používá místo velikosti hodnotící funkce indexy prvků podle ní uspořádané. Předností tohoto přístupu je skutečnost, že algoritmus neuvízne tak snadno v lokálním extrému. Nevýhodou je pomalejší iterace. Pro její urychlení bývá vhodné doplnit tento typ selekce elitismem. Oproti předešlé metodě je vhodnější na řešení složitějších nelineárních funkcí.

Turnaj je poslední z uvedených alternativ. Její upřednostnění před jinou z uvedených metod závisí na konkrétní aplikaci. Nelze ji obecně označit za více či méně vhodnou.

5) Jaký bude použit typ křížení?

V případě binárního kódování se běžně volí křížení v jednom nebo dvou bodech. Vícebodové dělení není běžné.

Pro reálně kódované chromozomy existuje celá řada možných postupů křížení. Ve skriptech byly představeny tři různé metody. U první uvedené rovnice jsou potomci nakombinováni z rodičů tak, že žádný potomek nepřekročí hodnotou svého genu hodnotu, kterou mají oba rodiče (ani směrem nahoru, ani směrem dolů). Pokud by při křížení rodičů měl vzniknout i potomek, hodnota jehož konkrétního genu by byla větší nebo menší než u

obou rodičů, jsou vhodnějšími druhá a třetí metoda křížení (mají-li rodiče hodnoty téhož genu 2 a 3, druhá a třetí metoda umožňuje vznik potomka s hodnotu tohoto genu např. 1,9; 3,7 atd.).

6) S jakou pravděpodobností bude křížení provedeno?

Optimální nastavení pravděpodobnosti křížení je uváděno v rozsahu (0,6;1). Operátor křížení je jediným operátorem, jehož prostřednictvím mohou vzniknou zcela odlišní jedinci. Z tohoto vyplývá, že by pravděpodobnost jeho uplatnění měla být co nejvyšší. Pokud ke křížení nedojde, potomci jsou identičtí se svými rodiči.

7) Jaký bude použit typ mutace?

V případě binárního kódování je mutace otázka nastavení pravděpodobnosti jejího nastolení. Postupuje se od jednoho genu k dalšímu. Pokud k mutaci dojde, je 1 nahrazena 0 nebo obráceně.

Mutace u reálného kódování může být statická (změna o stejnou hodnotu), náhodná (vybírána z předem daného intervalu) nebo dynamická. Dynamická mutace vykazuje mimořádně dobré vlastnosti. Hodnota, o kterou se v případě mutace velikost parametru změní, se s přibývajícím počtem generací zmenšuje. Na začátku způsobuje větší změny a na konci už pouze „dolaďuje“ stávající řešení.

8) S jakou pravděpodobností bude provedena mutace?

Za optimální nastavení pravděpodobnosti mutace jsou udávány hodnoty v intervalu (0,001;0,15).

V případě binárního kódování, kde je číslo reprezentováno 16 bity a pravděpodobnost mutace je 0,05, pravděpodobnost, že nedojde ke změně jednoho parametru je 0,95^16=0,44. Pokud použijeme stejnou pravděpodobnost u reálného kódování, je pravděpodobnost, že ke změně parametru nedojde, rovna 1-0,05= 0,95. Z toho je patrné, že pravděpodobnost mutace volíme v případě reálného kódování v horním intervalu (kdyby měla odpovídat výše uvedeném příkladu s binárním kódováním, volili bychom pravděpodobnost mutace 0,56) a v případě binárního kódování v dolních mezích (0,01). Uvedené hodnoty jsou přirozeně orientační, závisí na typu problému nebo např. na velikosti chromozomu v případě binárního kódování.

9) Jaká bude ukončovací podmínka?

Ukončovací podmínka může být kombinací posouzení kvality nejlepšího jedince (pokud se po určitý počet generací nezměnil) a celkového počtu vytvořených generací. Příkladem může být ukončovací podmínka, která ukončí hledání řešení v případě, že ve 30 po sobě jdoucích generacích nevznikl žádný nový nejlepší jedinec. Vhodné je podmínku doplnit o maximální možný počet generací. Výpočet se tedy ukončí i v případě, že je vytvořena již např. 200 generace.

3.4.3 Příklady aplikací genetických algoritmů

GA nacházejí uplatnění v architektuře, informatice, elektrotechnice, řízení výrobních procesů, managementu atd.

Mezi významné reálné aplikace patří např. návrh motorů letadla Boeing 777. Proudový motor byl nejprve navržen pomocí klasických metod s ohledem na co nejmenší spotřebu paliva. Následně byly použity GA pro detailní doladění některých parametrů. Úspora paliva v důsledku použití GA dosáhla téměř 2,5%. Toto zdánlivě malé zlepšení má ovšem velký ekonomický dopad. Úspora na jedno letadlo za rok činí 2 miliony dolarů.

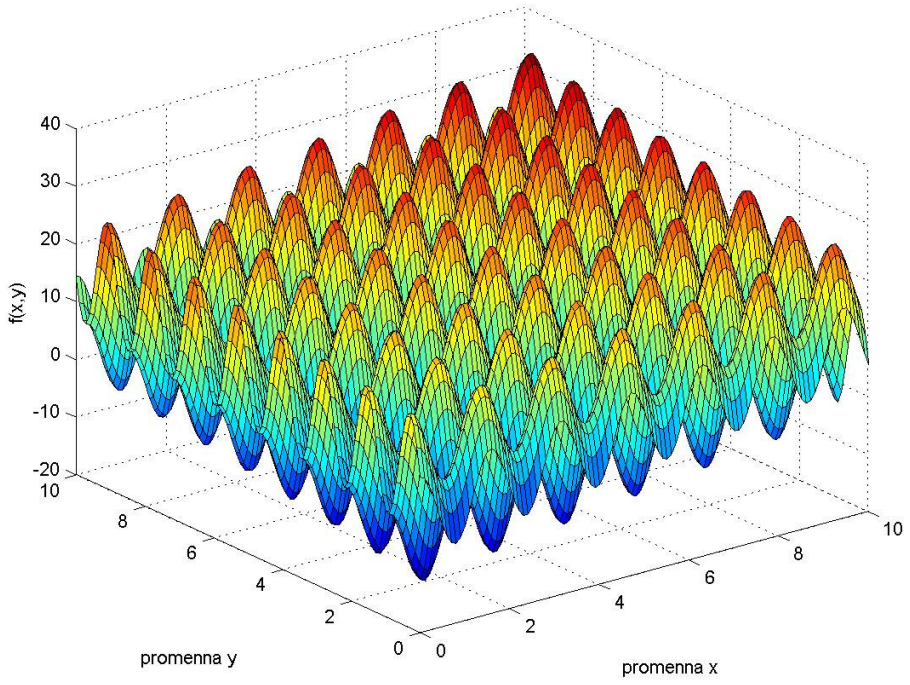
GA jsou používány v USA při rekonstrukci tváří lidí podezřelých ze spáchání zločinu. Do chromozomů jsou zakódovány různé parametry obličeje, jako tvar hlavy, posazení a barva očí, tvar nosu, tváří či úst. Na začátku je vygenerována sada obličejů a svědek zločinu je seřadí podle podobnosti s hledanou osobou. Pomocí selekce rekombinačních operátorů je vygenerována další sada obličejů.

Příklad:

Nalezněte hodnoty parametrů x a y s přesností na 3 desetinná místa, ve kterých nabývá následující funkce svého maxima: $f(x,y) = x + (10 \cdot \sin(5 \cdot x)) + (7 \cdot \cos(4 \cdot y)) + y$, kde $x \in \langle 0,10 \rangle$ *a* $y \in \langle 0,10 \rangle$.

Řešení:

Průběh hledané funkce je na Obr. 3.9 (ve skutečných případech takovou možnost náhledu hledané funkce samozřejmě nemáme).



Obr. 3.9: Průběh funkce f(x,y)

Kódování: proměnné lze kódovat do reálných čísel i binárně. V případě binárního kódování je minimální přípustný počet kombinací hodnot genu větší než *2*10/0,001=20.000*. Tomu odpovídá alespoň 15-ti bitový gen (32.768 kombinací). V příkladu bude použit gen 16-ti bitový.

Hodnotící funkce: jako hodnotící funkce je v takovém případě použita funkce samotná.

Selekce: rozhodování o vhodné selekci je velice obtížné. V tomto příkladě víme, že jsou skládány harmonické průběhy s lineárním nárůstem v obou osách, takže lze očekávat řadu lokálních extrémů (pracujeme s radiány, nikoliv se stupni). Z tohoto důvodu lze upřednostnit při prvních pokusech poziční selekci.

Křížení: v případě binárního kódování jednobodové (je to pouze otázka volby), v případě genu z reálných čísel lze zkusit typ křížení vnitřní soutěž. Pravděpodobnost křížení může být nastavena kolem 0,9. Je to opět otázka experimentu.

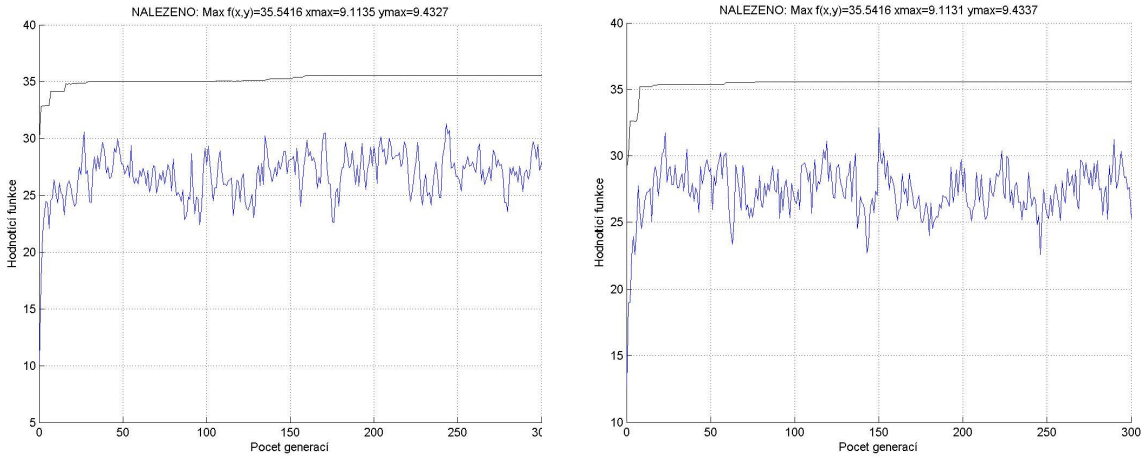
Mutace: v případě binárního kódování byla nastavena pravděpodobnost mutace 0,05. Tomu odpovídá u reálného kódování mutace s pravděpodobností přibližně 0,56; záleží však na zvoleném typu. Nižší pravděpodobnost se použije, pokud je mutace statická a výrazně větší než požadovaná přesnost. V počátku hledání řešení pomáhá rychlejší aproximaci, ale v závěru může znehodnocovat skoro správná řešení. Větší hodnoty lze použít, je-li velikost mutace nastavena na úrovni požadované přesnosti. Projeví se však až na konci aproximace (pokud ovšem dojde k přiblížení se správnému řešení). U mutace dynamické lze použít vyšší pravděpodobnost, protože se svou velikostí přizpůsobuje předpokládané blízkosti správného řešení. To všechno jsou ale podmínky za předpokladu, že bude zvolen vhodný počet generací a ostatních operátorů. A splnění tohoto předpokladu opět nelze garantovat.

Ukončovací podmínka: celkový počet generací roven 300.

Tabulka 3.5: Parametry použité při modelování příkladu:

	Binární kódování	Reálné kódování
Velikost počáteční populace	25	25
Elitismus	Ano	Ano
Typ selekce	Vážená ruleta	Vážená ruleta
Typ křížení	Jednobodové	Vnitřní soutěž
Pravděpodobnost křížení	0,95	0,95
Typ mutace	-	Z intervalu (–0,005;0,005)
Pravděpodobnost mutace	0,05	0,56
Ukončovací podmínka	Počet generací: 300	Počet generací: 300

Průběhy vývoje nejlepšího jedince a průměrné kvality generace jsou pro binární i reálné kódování velice podobné, jak je patrno z Obr. 3.10.



Obr. 3.10: Vývoj nejlepšího jedince a průměrné kvality generace na počtu generací pro a) binární kódování b) reálné kódování.

3.5 Souhrn

GA vycházejí z **evolučního počítání**, které je součástí umělé inteligence. Hlavní myšlenka má kořeny v Darwinově teorii o přirozeném vývoji na základě schopnosti **adaptace**. Podobně jako u biologický předloh jsou vlastnosti jedinců **kódovány** do **genů**, jejichž soubor vytváří **chromozom** jedince. Kódování rozlišujeme **binární** a **kódování do reálných čísel**.

GA funguje tak, že je na počátku je vytvořena nultá **generace**. Pak začíná cyklus, ve kterém je spočtena všem jedincům velikost jejich **hodnotící funkce** (**fitness**); následuje **selekce**, na jejímž základě jsou vybráni **rodiče;** operandy **křížení** a **mutace** pak dávají vzniknout novým **potomkům**; pokud nová generace splňuje **ukončovací podmínky**, je cyklus ukončen, jinak se opět vrací na svůj začátek.

Existuje řada typů genetických operátorů. Jejich správná volba a nastavení zásadně ovlivňují kvalitu konečného výsledku. Velký počet možných nastavení, z nichž jen některá vedou k řešení, představuje hlavní problém při použití GA. Přesto v praxi existuje řada jejich úspěšných aplikací.

Kontrolní otázky a úkoly:

1. Vysvětlete všechny tučně zvýrazněné pojmy obsažené v souhrnu kapitoly.

Řešení viz. Kapitola x.2.1 Přehled základní terminologie.

2. Napiš ideové schéma GA.

Řešení viz. Kapitola x.2.2 Princip genetických algoritmů.

3. V čem je kódování pomocí reálných čísel výhodnější ve srovnání s binárním kódováním?

Jednoduchý a čitelný zápis i složitějších chromozomů, jejichž binární obdoba by ve stejném rozsahu možných řešení byla příliš složitá.

4. Kolik potomků vznikne křížením na třech místech ze dvou binárně kódovaných rodičů?

Dva potomci.

5. Jaká je hlavní nevýhoda typu selekce „vážená ruleta“?

Pokud bude jeden jedinec výrazně lepší než ostatní (zabere např. 90% rulety), aproximace řešení se může výrazně zpomalit, v horším případě uvízne algoritmus v lokálním extrému.

6. Mějme grafický průběh „kvality nejlepšího jedince“ na „řádu generace“. V algoritmu selekce byl použit elitismus. Proč bude průběh této závislosti vždycky funkcí neklesající?

Než dojde k provádění genetických operací s jedinci z generace G(T), je vybrán ten nejsilnější a jeho kopie je uložena. Po ukončení selekce, křížení a mutace je do nově vytvořené generace G(T+1) přidán uložený jedinec. Pokud nevznikl žádný nový silnější chromozom, dodatečně přidaný nejsilnější jedinec z generace G(T) zůstává nejlepším. Úsek v grafu mezi generacemi G(T) a G(T+1) ani neroste, ani neklesá; zůstává konstantní. Pokud vznikl nový silnější jedinec, je jeho kvalita větší a úsek v grafu mezi generacemi G(T) a G(T+1) je rostoucí. Z toho plyne, že průběh závislosti „kvality nejlepšího jedince“ na „řádu generace“ bude při použití elitismu vždy neklesající funkcí.

Literatura

[1] VLACH, Z. Diplomová práce – Analýza a využití genetických algoritmů. VUT FEEC, Brno, 2004.

[2] OBITKO, M. Genetické algoritmy: Matrice života v počítačích [online]. 2001 [cit. 2001-03-14]. Dostupné z: <http://www.scienceworld.cz>.

[3] KAJÁNEK, P., PŘIKRYL, J. Seznámení se s Genetickými algoritmy [online]. Dostupné z: <http://mujweb.atlas.cz/www/prikrylj/Genetickealgoritmy.html>.

[4] NeuroDimension Inc. [online]. 2002.

Dostupné z: < http://www.nd.com/products/genetic/selection.htm >.

[5] OBITKO, M. Genetic Algorithms [online]. 1998.

Dostupné z: < http://cs.felk.cvut.cz/~xobitko/ga/main.html>.

[6] NEKVINDA, M. Genetické algoritmy. Automatizace, 2001, roč. 44, č. 10, s. 614-619.

[7] BOLDIŠ, Petr. : Bibliografické citace dokumentu podle CSN ISO 690 a CSN ISO 690-2 (01 0197) : Část 1 – Citace: metodika a obecná pravidla. Verze 3.2 1999–2002, poslední aktualizace 3.9. 2002.

Dostupné z: < **Chyba! Odkaz není platný.** >.

[8] BOLDIŠ, P. : *Bibliografické citace dokumentu podle CSN ISO 690 a CSN ISO 690-2* (01 0197): Část 2 – Modely a příklady citací u jednotlivých typů dokumentů. Verze 2.5 (2002) 1999–2002, poslední aktualizace 3. 9. 2002. Dostupné z: < http://www.boldis.cz/citace/citace2.pdf >.

[9] DEB, K. Genetic algorithm in search and optimalization : The technique and Application [online]. Dostupné z:

< http://citeseer.nj.nec.com/cache/papers/cs/1833/http:zSzzSzls11-www.informatik.uni-dortmund.dezSz~debzSzpaperszSzisinew.pdf/genetic- algorithm-in-search.pdf >

[10] COLLINS, J., EATON, M. Genocodes for genetic algorithms [online]. Dostupné z: <http://citeseer.nj.nec.com/cache/papers/cs/3818/http:zSzzSzwww.csis.ul.iezSzstaffzSzjcollinszSzmendel97.pdf/genocodes-for-genetic-algorithms.pdf >

[11] REITERMANOVÁ, Z. Genetické algoritmy: úvod do evolučních výpočetních technik [online]. 5.8.2003.

Dostupné z:
<http://ulita.ms.mff.cuni.cz/mff/seznamy/prg022_03_do/Reitermanova_Geneticke_algoritmy/Geneticke%20algoritmy-clanek.doc>

[12] DREIER, F. Genetic Algorithm Tutorial [online]. Červenec 2002.

Dostupné z: < http://www.dreier.cc/extdata/documents/gatutorial.pdf >

[13] NeuroDimension Inc. What is neural network [online]. 2004.

Dostupné z:

< http://www.nd.com/neurosolutions/products/ns/whatisNN.html >.

[14] Tomáš Pevný. Použití genetických algoritmu při učení neuronových sítí a možné aplikace. Technical report, Fakulta jaderná a fyzikálně inženýrská, ČVUT, 2002.

[15] Tomáš Pevný. Neuronové sítě a genetické algoritmy – vybrané plaikace při řešení některých technických problémů. Technical report, Fakulta jaderná a fyzikálně inženýrská, ČVUT, 2003.

[16] SYED, O., SYED, A. The Game of Real Intelligence [online]. Dostupné z:

< http://arimaa.com/arimaa/about/Thesis/Thesis.pdf >.

[17] SINČÁK, P., ANDREJKOVÁ, G. Neuronové siete [online]. Dostupné z:

< http://neuron-ai.tuke.sk/cig/source/publications/books/NS1/html/all.html>.

[18] VLACH, Z. Semestrální práce 2. VUT FEEC, Brno, 2003.

[19] SAHAMI, M. Exhaustive recursion and backtracking [online]. Únor 2004.

Dostupné z:

<http://www.stanford.edu/class/cs106x/handouts/HO25%20Rec%20backtrack%20examples.pdf >

4 Rozhodovací stromy

Cíle kapitoly:

Cílem této kapitoly je uvedení do problematiky rozhodovacích stromů (RS). RS je hierarchický, nelineární systém umožňující uložení znalostí. Tyto znalosti je možné ze struktury RS extrahovat nebo použít k analýze nových dat. V první části kapitoly jsou uvedeny terminologie, základní principy a dělení RS. Následuje popis některých algoritmů pro jejich vytváření a úpravu. Kapitola je zakončena souhrnem a přehledem použité literatury.

4.1 Úvod

Se znalostmi uloženými do stromové struktury se lze setkat v různých vědních disciplínách. Typický je např. botanický klíč, ve kterém lze na základě postupného popisu konkrétních atributů rostliny určit její název. Při vytváření takového klíče je třeba vhodně volit jednotlivé atributy, aby bylo vyhledávání co nejrychlejší a nejefektivnější. Získaná znalost je uložena do struktury rozhodovacího stromu (RS). A právě základům jejich vytváření, interpretaci a použití je věnována tato kapitola.

RS jsou primárně používány pro klasifikaci kvalitativních závislých proměnných na základě vstupních proměnných. Tato metodika je užívána zejména v oblasti „Data Mining“. Principielně je blízká jiným klasickým metodikám (diskriminační analýza, shluková analýza, neparametrická statistika, nelineární odhady). Ve specifických případech jsou uvedené tradiční přístupy vhodnější. Obecně lze ale říci, že RS patří v současné době mezi nejpoužívanější metodiky pro analýzu dat a jejich hierarchickou reprezentaci.

Prvním z významných rysů RS je jejich hierarchická struktura. Jestliže se uzel rozdělí na základě hodnoty atributu na několik dalších uzlů, vytváří se z každého nově vzniklého uzlu nový nezávislý podstrom. Tím se RS liší např. od regresního modelu vytvořeného pomocí diskriminační analýzy.

Dalším významnou vlastností RS je jejich flexibilita. Jako nezávislé proměnné zde mohou vedle sebe vystupovat proměnné kvantitativní i kvalitativní (ordinální i nominální). RS není omezen na posuzování hodnoty jedné proměnné. V uzlu může být rozhodováno na základě vícerozměrné funkce tvořené kombinací vybraných nezávislých proměnných (tato možnost bývá využívána, pokud je klasifikováno do několika málo tříd na základě mnoha nezávislých proměnných). Na druhou stranu lze jednu spojitou proměnnou rozdělit na několik intervalů a vytvořit tak rozvětvenější strom (vhodné, pokud je k dispozici málo nezávislých proměnných a více tříd, do kterých bude klasifikováno).

Posledním významným rysem RS je jejich čitelnost a snadná interpretovatelnost. Pokud je rozhodnutí závažné z hlediska bezpečnosti nebo nákladů, je vždy nezbytné rozumět systému, který dodává podklady k rozhodnutí. To je také zásadní nevýhoda např. neuronových sítí, jejichž složitá struktura znemožňuje dostatečné porozumění jednotlivým parametrům. Z toho důvodu se lze jen těžko spolehnout v závažných rozhodnutích na takový systém, byť se jeví být sebelepší.

Obecně lze tedy říci, že RS je hierarchický systém, do jehož nelineární struktury lze uložit znalosti. Existují algoritmy, které umožňují automatické vytváření RS a extrahují tak

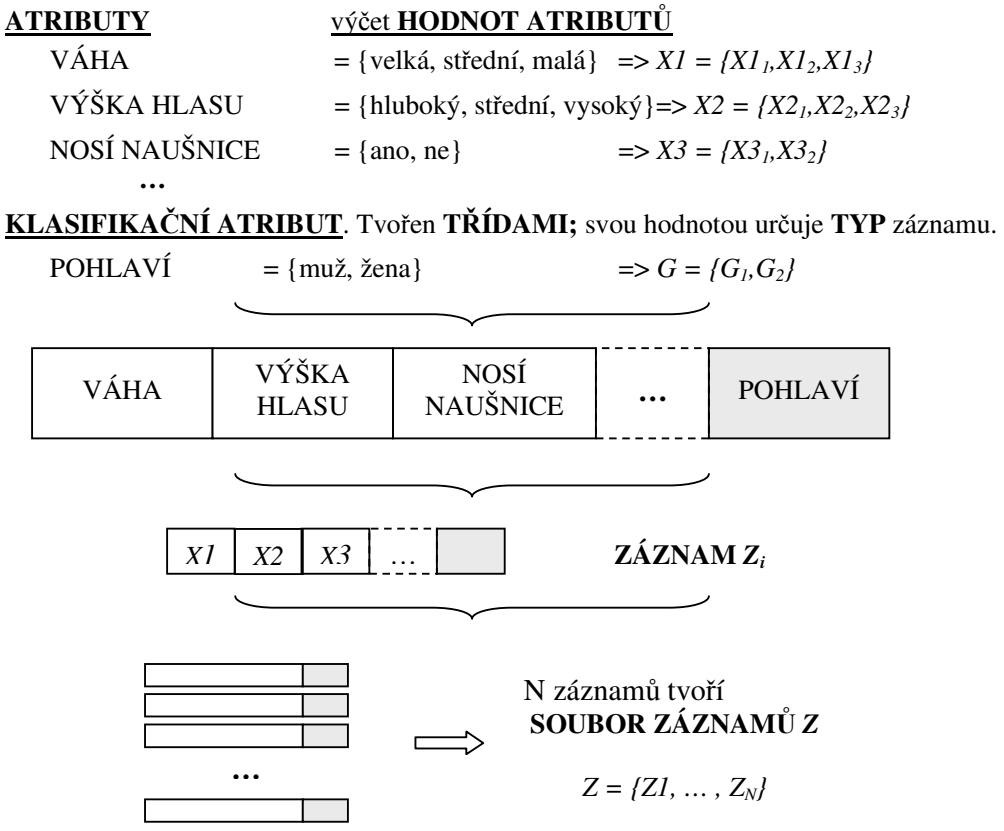
znalosti z dat. Hlavními přednostmi a charakteristickými rysy RS jsou také: hierarchická struktura, nelinearita, srozumitelnost a čitelnost, flexibilita (co do typu analyzovaných dat i co do topologie) a existence algoritmů, které umožňují jejich automatické vytváření.

4.2 Základy rozhodovacích stromů

Kapitola začíná přehledem nejnutnější terminologie. Pokračuje popisem principu fungování RS a je zakončena přehledem základních typů RS a jejich charakteristikami.

4.2.1 Terminologie rozhodovacích stromů

Atribut – vlastnost (Obr. 4.1). Svou hodnotou popisuje nebo charakterizuje prvek nebo sledovaný objekt; hodnotu lze vyjádřit číselně nebo symbolicky - může být kvantitativní nebo kvalitativní.



Obr. 4.1 Analyzovaná data - terminologie

Entropie – míra neuspořádanosti; číselná hodnota v intervalu <0;1>. Cílem je její minimalizace, tedy dosažení úplné uspořádanosti (přesné predikce).

Hloubka stromu – číselná hodnota (Obr. 4.2). Vyjadřuje počet větví, které vedou od kořenového uzlu k nejvzdálenějšímu listu.

Hodnota atributu – číslo nebo symbol (Obr. 4.1). Vyjadřuje konkrétní stav atributu.

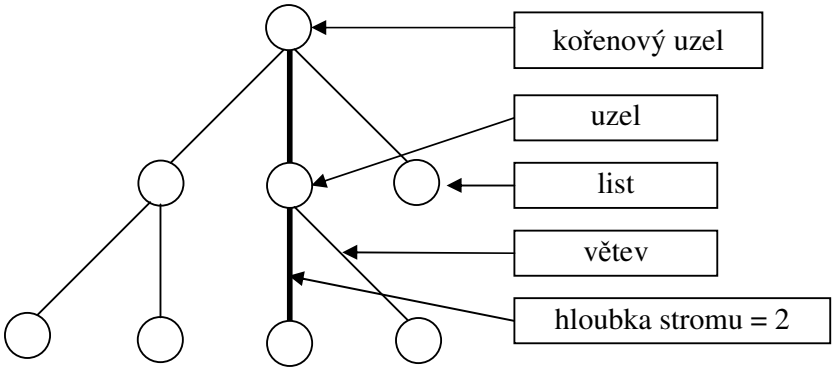
Kořenový uzel – počáteční uzel RS (Obr. 4.2). Jeho hloubka v RS je 0.

List – součást RS (Obr. 4.2); jeho dosažení vede ke klasifikaci nebo predikci hodnoty výstupní veličiny analyzovaného záznamu.

Prořezávání stromu – algoritmus; metoda sloužící k optimalizaci velikosti stromu (zejména přeučeného) jeho zmenšováním na základě např. zvyšování celkové přesnosti, snížení komplexnosti, …

Přeučený strom – RS, který ve své struktuře zohlednil i závislosti, jež jsou ve skutečnosti šumem a nesouvisejí s analyzovanou problematikou.

Rozhodovací strom – matematický model (Obr. 4.2). RS je hierarchický nelineární systém umožňující nalezení a uložení znalostí a jejich využití k analýze nových dat.



Obr. 4.2 Rozhodovací strom

Uzel – součást RS (Obr. 4.2); při jeho průchodu jsou data rozdělena na základě podmínky do větví z něj vycházejících.

Větev – součást RS (Obr. 4.2); je spojnicí mezi dvěma uzly nebo uzlem a listem. Spojuje dva objekty ze sousední hierarchické úrovně.

Záznam – prvek (Obr. 4.1); soubor údajů o jednom objektu složený z vektoru hodnot vstupních atributů a výstupního atributu nebo klasifikátoru.

4.2.2 Princip rozhodovacích stromů

RS je hierarchický systém umožňující uložení znalostí. Významné je, že existují algoritmy, které RS automaticky vytvářejí. Z jejich struktury lze díky srozumitelným

grafickým výstupům pochopit a interpretovat získané znalosti. Je také možno RS převádět do jazykových pravidel.

Než dojde k vytvoření RS, je třeba pochopit a vhodně připravit data, která budou analyzována. Množina těchto dat (soubor záznamů) se skládá z jednotlivých prvků (záznamů). Obecně popsaný záznam je tvořen vstupními atributy (vlastnosti) a jedním výstupním atributem (též klasifikátor). Každý atribut může nabývat určitých hodnot. Konkrétní záznam je pak tvořen hodnotami svých atributů. U kvantitativních vlastností jde o číslo, v případě kvalitativních vlastností o libovolný znak nebo řetězec znaků (Obr. 4.1).

Příklad:

V určitých případech může být obtížné definovat vhodné atributy. Pokud je cílem vytvořit hru kámen-nůžky-papír, která bud hrát proti člověku a použije jako herní systém RS, musíme stanovit, z jakých atributů se bude záznam skládat. Klasifikátor je zřejmý, výstupní hodnoty jsou „kámen“, „nůžky“ a „papír“. Problém však nastane při hledání vstupních atributů. Navrhněte nějaké atributy a definujte jejich definiční obor (hodnoty, které může daný atribut nabývat). Ukázku některých možných řešení naleznete na konci kapitoly.

Při vytváření RS dochází k principielně podobným krokům. Do každého uzlu vstupují určitá trénovací data. Pokud jsou splněny ukončovací podmínky, je na jejich základě rozhodnuto o konečné klasifikaci a z uzlu se stává list. V opačném případě je hledán způsob, jak stanovit co nejvhodnější podmínku, na jejímž základě budou stávající data rozdělena do nových uzlů tak, aby bylo dosaženo co největšího informačního zisku. Podmínka může souviset s binárním nebo vícerozměrným dělením na základě jednoho z atributů. Existují také podmínky vytvořené na základě lineární kombinace více atributů. Topologie RS je jednou z dalších vlastností, na jejichž základě RS rozdělujeme.

Vytvořený RS může být přeučený. To znamená, že ve své struktuře zohlednil i závislosti, které jsou ve skutečnosti šumem a souvisejí nikoliv s analyzovanou problematikou, ale se samotnými trénovacími daty. K ověření, zda RS je či není přeučený, se používají testovací data nebo metoda cross-validation. Přeučené RS se prořezávají. Existuje opět řada metod prořezávání. Některé z nich jsou součástí samotných algoritmů, které RS vytvářejí. Nakonec se RS validují. Validace vyjadřuje přepokládanou chybu modelu.

4.2.3 Rozdělení rozhodovacích stromů

Existuje více druhů RS. Kritériem jejich odlišnosti jsou zejména topologie, typy vstupních a výstupních proměnných. Jeden typ stromu však může být generován různými postupy – algoritmy. Souvislosti mezi typy stromů, konkrétními algoritmy a dalšími vlastnostmi jsou uvedeny v následující tabulce. RS jsou rozděleny do tří skupin – CART, CLS a AID. Vzhledem k dynamickému vývoji jednotlivých algoritmů je třeba upozornit, že se jedná o přehled orientační. Stále dochází k vývoji nových nebo modifikacím stávajících algoritmů.

Tabulka 4.1 Dělení rozhodovacích stromů

Typ RS	CART	CLS	AID
Některé algoritmy	CART, tree (S)	CLS, ID3, C4.5, C5	AID, THAID, CHAID, MAID, XAID, FIRM, TREEDISC
Motivace	statistická predikce	učení se konceptu	hledání komplexních statistických vztahů
Vstupní Proměnná	kvantitativní, kvalitativní	nominální, ordinální	nominální, ordinální, existují verze pro kvantitativní
Výstupní Proměnná	kvantitativní, nominální	nominální	AID, MAID, XAID – kvantitativní THAID, CHAID, TREEDISC – nominální FIRM – kvantitativní i kvalitativní
Topologie	binární	počet větví každého uzlu odpovídá počtu možných hodnot dělícího atributu	všechny typy topologií
Specifika	algoritmus zahrnuje cross-validation a prořezávání	historicky je vývoj v pořadí CLS, ID3, C4.5, C.5	k určení velikosti stromu používány statistické testy, podpora predikce při chybějících hodnotách

4.3 Vybrané algoritmy pro tvorbu rozhodovacích stromů

Následující kapitola je věnována popisu některých algoritmů určených pro vytváření a úpravu RS. V základních rysech jsou popsány algoritmy ID3, CART, prořezávání, používané chybové funkce a typické ukončovací podmínky.

4.3.1 ID3

ID3 je algoritmus pro vytváření RS. Je určen pro klasifikaci na základě nominálních vstupních proměnných. RS se větví podle počtu hodnot konkrétních dělících atributů. Předností ID3 je, že dokáže z velkého množství atributů vybírat ty, které nejvíce zvyšují informační zisk stromu (snižováním celkové entropie). Algoritmus však negarantuje, že nalezený strom je nejlepší možný.

Ideologické schéma algoritmu ID3:

1. Mějme jeden uzel obsahující veškeré záznamy, které budou použity na modelování stromu.
2. Pokud jsou pro tento uzel splněny ukončovací podmínky, stane se uzel listem s odpovídající klasifikací a algoritmus větvení této části stromu je ukončen.

3. V opačném případě je nalezen atribut, jehož informační přínos je největší (rozdělí soubor záznamů nejlépe). Vznikne tak větvením několik nových uzlů, mezi které jsou rozděleny záznamy podle hodnot zvoleného atributu.
4. Na nové uzly je pohlíženo jako na nové počáteční uzly. Na každý uzel se aplikuje tento algoritmus od bodu 1.

Uvedeným postupem je vytvořen rozvětvený strom zakončený listy. Každý list představuje klasifikaci do jedné konkrétní třídy.

Vytváření RS pomocí ID3

Základním pojmem v algoritmu ID3 je tzv. Shanonova entropie. Její hodnota vyjadřuje míru informace v souboru záznamů. Čím je její hodnota vyšší, tím mnohoznačnější je obsah souboru záznamů a tím menší má informační hodnotu. Lze tedy jednoduše říci, že čím je entropie menší, tím jednoznačněji lze předpovědět typ záznamu v souboru. Ideální případ nastane, když je hodnota entropie rovna 0. Soubor se skládá ze záznamů téhož typu (např. všechna auta jsou typu „žlutá“, pokud je klasifikačním atributem „barva auta“).

Mějme soubor záznamů. Každý záznam lze klasifikovat do právě jedné třídy atributu G, který může nabývat hodnot *G1, …, Gc*. Pravděpodobnost klasifikace do každé ze tříd označíme *p(G1), …, p(Gc).* Pak platí:

$$\sum_{i=1}^{C} p(G_i) = 1 \qquad (4.1)$$

Shanonova entropie takového souboru je dána vzorcem:

$$H = -\sum_{i=1}^{C} p(G_i) \log_C p(G_i) \qquad (4.2)$$

V případě klasifikace do dvou tříd (např. ano/ne, …), kde n_1 označíme počet záznamů spadajících do první a n_2 do druhé třídy, lze předešlé vzorce vyjádřit následujícím způsobem:

$$\sum_{i=1}^{2} p_i = p_1 + p_2 = \frac{n_1}{n_1 + n_2} + \frac{n_2}{n_1 + n_2} = 1 \qquad (4.3)$$

$$H(S) = -\sum_{i=1}^{2} \frac{n_i}{n_1 + n_2} \log_2 \frac{n_i}{n_1 + n_2} = -\frac{n_1}{n_1 + n_2} \log_2 \frac{n_1}{n_1 + n_2} - \frac{n_2}{n_1 + n_2} \log_2 \frac{n_2}{n_1 + n_2} \qquad (4.4)$$

Při vytváření RS je důležité vybrat atribut, který bude entropii co nejvíce minimalizovat. Proto jsou postupně vybírány jednotlivé atributy a uzel *S* je na jejich základě rozdělen. Vybrán je atribut, který rozvětvil strom z uzlu *S* tak, že celková entropie nově vzniklých uzlů je co nejmenší. Při vyjádření celkové entropie více uzlů je třeba zohlednit i počty trénovacích záznamů připadajících do nových uzlů. Máme-li tedy uzel *S* a atribut *A = {a1,…,aM}*, podle kterého uzel rozvětvíme, přičemž v každé z *M* větví je *N1,..,NM* prvků, celková entropie nově vzniklých uzlů se vyjádří následujícím vztahem:

$$H(S, A) = \sum_{i=1}^{M} \left(\frac{N_i}{\sum_{j=1}^{M} N_j} \cdot H(S_i) \right) \qquad (4.5)$$

Informační zisk *I(A)* tohoto dělení vyjadřuje vzorec:

$$I(A) = H(S) - H(S,A) \qquad (4.6)$$

Postupným prozkoušením dělení podle všech možných atributů je vybrán ten, jehož informační zisk je největší. S nově vzniklými uzly se pracuje dále stejným způsobem. Tak je vytvořen RS algoritmem ID3.

Důležité je dodat, že pro účely vypočitatelnosti entropie ve všech možných stavech je dodefinován logaritmus v hodnotě 0 a je v ní roven 0, tedy log(0)=0 (funkce logaritmus v této hodnotě není definována, limituje k nekonečnu).

Příklad

V následující tabulce jsou zaznamenány výsledky pozorování, zda se dotyčná osoba spálila při pobytu na sluníčku. Úkolem je najít první dělící atribut pomocí algoritmu ID3.

Osoba č.	Barva vlasů	Výška	Váha	Opalovací krém	Spálila se
1	blond	průměrná	malá	ne	ano
2	blond	velká	průměrná	ano	ne
3	hnědé	malá	průměrná	ano	ne
4	blond	malá	průměrná	ne	ano
5	zrzavé	průměrná	velká	ne	ano
6	hnědé	velká	velká	ne	ne
7	hnědé	průměrná	velká	ne	ne
8	blond	malá	malá	ano	ne

Vypočítáme celkovou entropii pro jednotlivé atributy a jako nejlepší označíme ten parametr, jehož *H(S,A)* bude nejmenší.

Rozdělení podle atributu „Barva vlasů“ rozdělí osoby do 3 skupin. V první skupině (hodnota atributu = blond) jsou 4 osoby, ve druhé skupině (hnědé) jsou 3 osoby a v poslední skupině (zrzavé) je jedna osoba. Osob je celkem 8. Entropie *H(S,Barva vlasů)* se vypočítá následujícím způsobem:

$$H(S, Barva\ vlasů) = \frac{4}{8}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right) + \frac{3}{8}\cdot 0 + \frac{1}{8}\cdot 0 = 0{,}5$$

Podobným způsobem se spočítají i zbývající 3 entropie:
H(S,Výška) = 0,69
H(S,Váha) = 0,94
H(S,Opalovací krém) = 0,61

Z výsledků je patrné, že nejvyšší entropie (a tedy i nejnižší informační zisk) přineslo dělení podle váhy (což lze ostatně i očekávat). Nejnižší entropii vykazuje dělení podle barvy vlasů. Rozhodovací strom by tedy začal dělením podle atributu „Barva vlasů“. Z počátečního uzlu povedou 3 větve.

4.3.2 CART

CART (Classification and Regression Trees) algoritmus byl uveden roku 1984 v práci L. Briemana. Stromy vytvořené touto metodou mají binární topologii. Znamená to, že z každého uzlu vycházejí právě dvě větve. Algoritmus umí pracovat s libovolnými vstupními proměnnými (kvalitativní, kvantitativní). Jak vyplývá z názvu algoritmu „CART“, stromy slouží ke klasifikaci i regresi. Výstupní proměnné mohou tedy býti kvalitativní i kvantitativní.

Použití algoritmu lze doporučit, je-li třeba vytvořit strom s binárním větvením. Další předností je možnost váhování jednotlivých klasifikačních tříd a vytvoření matice nákladů (cost matrice) pro penalizační funkci. V matici nákladů jsou charakterizovány výdaje spojené s chybnou klasifikací do konkrétní třídy. Jinými slovy, u klasifikačních RS lze rozlišit závažnost chyby podle typu třídy, do níž byl prvek chybně zařazen. Skutečnost, že prvek třídy A byl klasifikován jako typ B, může být penalizována odlišně, než chybné zařazení toho samého prvku do třídy C. Dalšími vlastnostmi jsou např. možnost prořezávání stromu na základě komplexnosti, přímá aplikace zastavovacích pravidel atd.

Základní verze CART algoritmu funguje na principu nejvhodnějšího rozdělení záznamů podle jednoho z atributů. Mohou však nastat situace, kdy je tento přístup nevhodný. Ukázkou je případ, kde je čtverec o rozměru 10x10 rozdělen úhlopříčkou na dvě poloviny. Každý prvek je charakterizován dvojicí souřadnic (x,y) a je zařazen do kategorie „+“ nebo „o“ podle toho, v které půlce čtverce rozděleného úhlopříčkou se nachází; stačí jediná podmínka daná rovnicí tvořenou lineární kombinací obou parametrů a prvky jsou klasifikovány bezchybně. Vzhledem k tomu, že CART pracuje pouze s jednotlivými proměnnými, jím vytvořený RS nikdy nenalezne zcela jednoduché správné řešení a bude jej pouze velice krkolomně aproximovat.

Ideologické schéma algoritmu

1. Získej základní nastavení a informace o datech (priorní pravděpodobnosti, matici nákladů, váhy, …).

2. Vypočti údaje o uzlu.
   2.1 Klasifikační RS.
      2.1.1 Pravděpodobnost, že prvek dosáhne daného uzlu v rámci celého RS.
      2.1.2 Pravděpodobnosti jednotlivých klasifikací v daném uzlu.
      2.1.3 Hodnotu penalizační funkce pro jednotlivé klasifikační třídy.
   2.2 Regresní RS.
      2.2.1 Chybu v uzlu pomocí MNČ.

3. Zvaž jestli rozdělit uzel (nesplnění podmínek = ukončení algoritmu).
   3.1 Klasifikační RS – podmínky, jejichž splnění vede k dělení.
      3.1.1 Záznamů v uzlu je více než minimální požadovaný počet.
      3.1.2 Záznamy v uzlu nejsou jedné třídy.
   3.2 Regresní RS – podmínky, jejichž splnění vede k dělení.
      3.2.1 Záznamů je více než minimální požadovaný počet.
      3.2.2 Celková chyba uzlu (určená MNČ) je větší než minimální chyba.

4. Urči pro větvení nejlepší proměnnou.
   4.1 Najdi u každé proměnné nejlepší kritickou hodnotu (minimální chyba).

4.2 Vyber proměnnou maximalizující informační zisk (bývá používán GINI index).

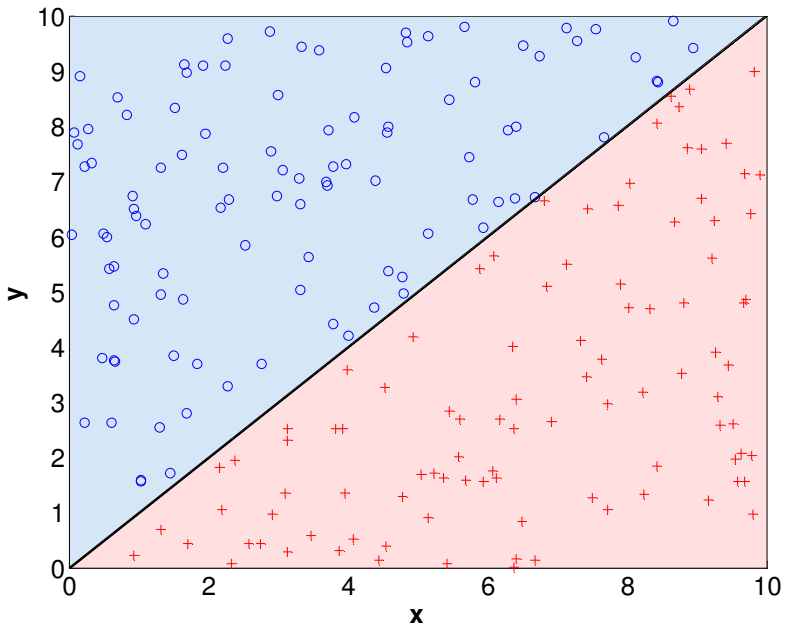
5. Proveď dělení a s oběma novými uzly a pokračuj od bodu 2.

6. Proveď prořezání stromu. Projdi od spodu všechny uzly. Pokud není chyba uzlu větší než suma chyb jeho potomků (uzlů v jeho větvích), větve odstraň a z uzlu udělej list.

Příklad:

Mějme čtverec o rozměrech 10x10, který je rozdělen úhlopříčkou na horní a dolní polovinu. Prvky ležící v horní polovině klasifikujeme jako „o“, prvky v dolní polovině jako „+“. Náhodně ve čtverci zvolíme body a přiřadíme jim příslušnou třídu. Zpětně se pak pomocí algoritmu CART pokusíme vytvořit strom, který vygeneruje pravidla pro klasifikaci těchto prvků. Cílem úlohy je na názorném příkladu ukázat jednoduchý problém, se kterými se metody používající jednu nejlepší proměnnou (univariate split) na klasifikaci nedokáží rozumně vypořádat.

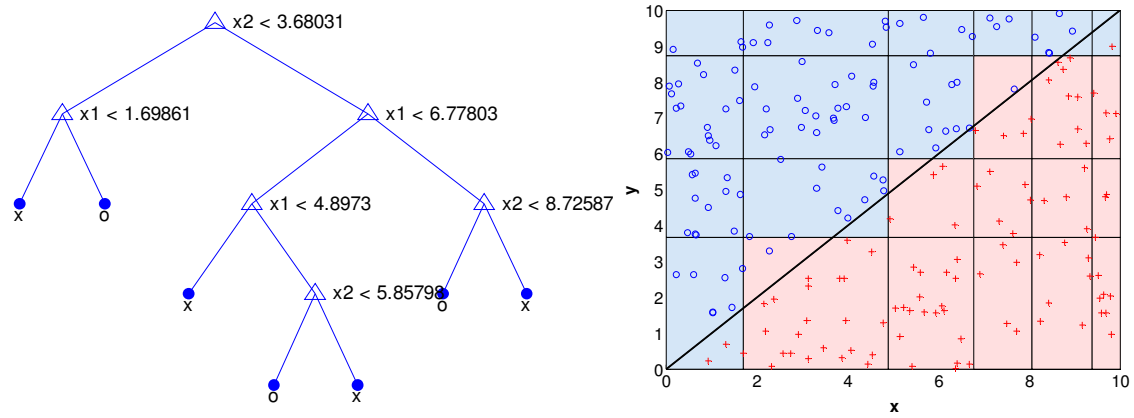
Na Obr. 4.3 je graf, ve kterém jsou vyneseny náhodně vygenerované prvky. Úhlopříčka vedená středem je funkcí, na jejímž základě lze prvky přesně klasifikovat. Úhlopříčka je popsána rovnicí $y=x$. Jako kriteriální funkce pro klasifikaci pak slouží funkce $f(x,y) = x - y$. Klasifikace se mění, je-li hodnota $f(x,y)$ větší (menší) než nula.



Obr. 4.3 Vygenerované body určené ke klasifikaci

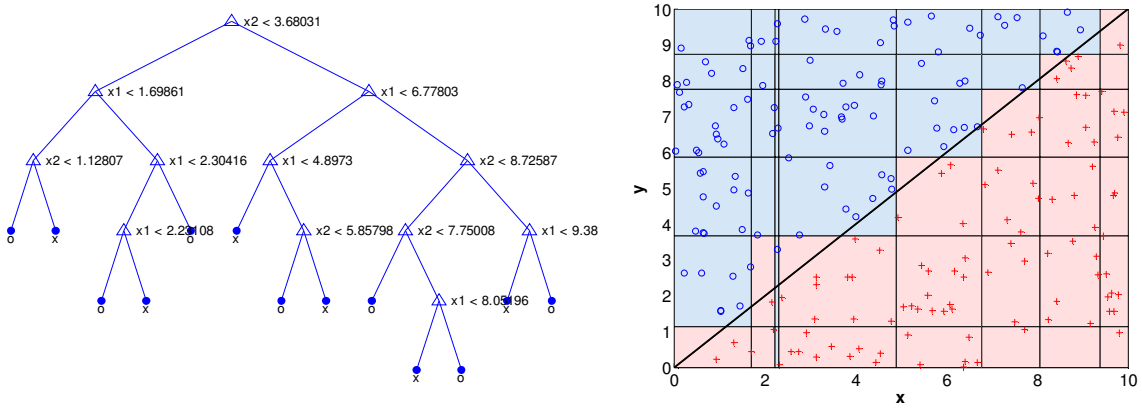
Nastavovány jsou parametry N (počet záznamů), splitmin (při kolika uzlech je ještě povoleno dělení uzlu) a sumDat (kolik procent předkládaných vzorů klasifikuje chybně – zašuměnost dat). Následující grafy znázorňují vytvořený RS a rozdělení plochy čtverce ze stromu vyplývající (světle modrá plocha znamená, že body v této oblasti jsou RS klasifikovány jako třída „o“, světle červené plochy jsou RS označeny jako třída „+“).

První dvojice grafů na následujícím obrázku byla algoritmem CART vytvořena při nastavení N=200, splitmin=10 a sumDat=0.



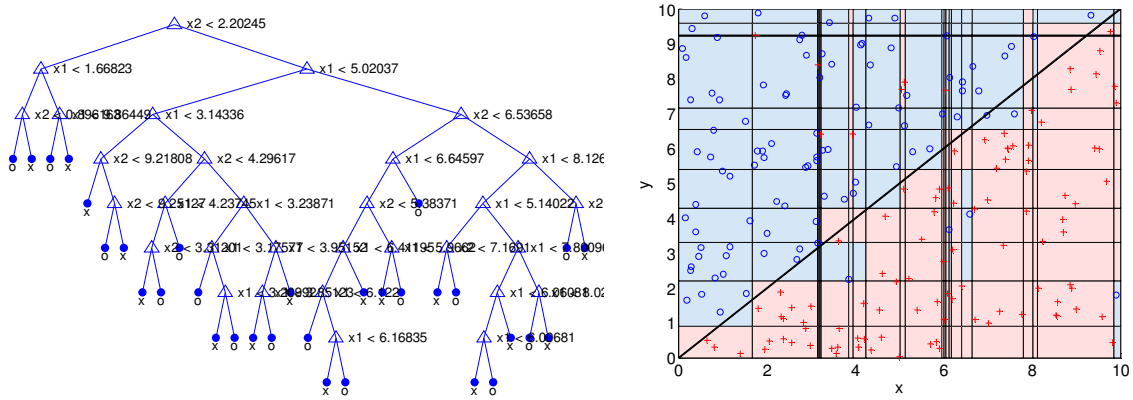
Obr. 4.4 a) Vygenerovaný rozhodovací strom. b) Predikce (barevné plochy) na základě dat.

Následující dvojice grafů znázorňuje situaci, kdy je nevhodně nastaven parametr splitmin=2. Strom je přeučený. Vzhledem k tomu, že data nejsou zašuměna, vede tato skutečnost ke zlepšení klasifikace.



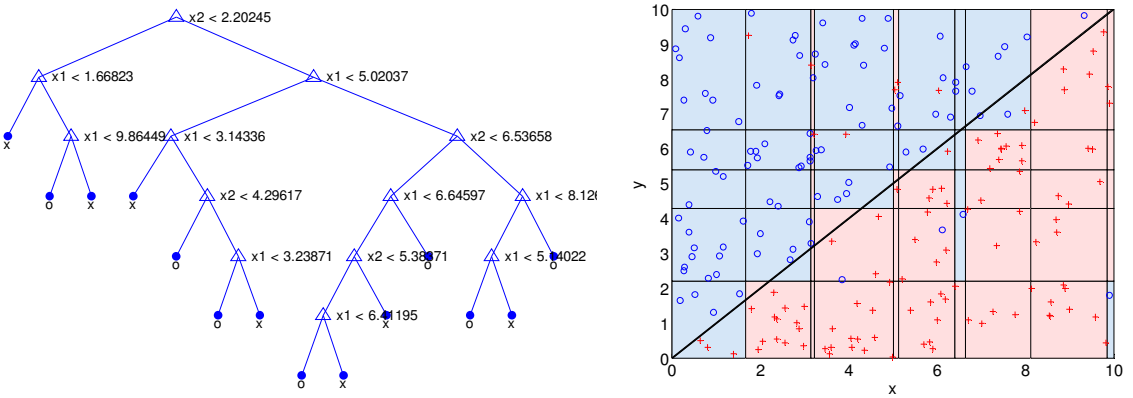
Obr. 4.5 a) Vygenerovaný rozhodovací strom. b) Predikce (barevné plochy) na základě dat.

Předposlední dvojice grafů znázorňuje situaci, kdy jsou data zašuměna v 5%, sumDat=5. Při nastavení splitmin=2 je z grafu zřejmé, že RS je přeučený a klasifikuje i podle zašuměných dat.



Obr. 4.6 a) Vygenerovaný rozhodovací strom. b) Predikce (barevné plochy) na základě dat.

Poslední dvojice grafů znázorňuje situaci při sumDat=5 a splitmin=10. Jak je patrno, RS se při přísnější ukončovací podmínce se zašuměnými daty vypořádal mnohem lépe.



Obr.

Obr. 4.7 a) Vygenerovaný rozhodovací strom. b) Predikce (barevné plochy) na základě dat.

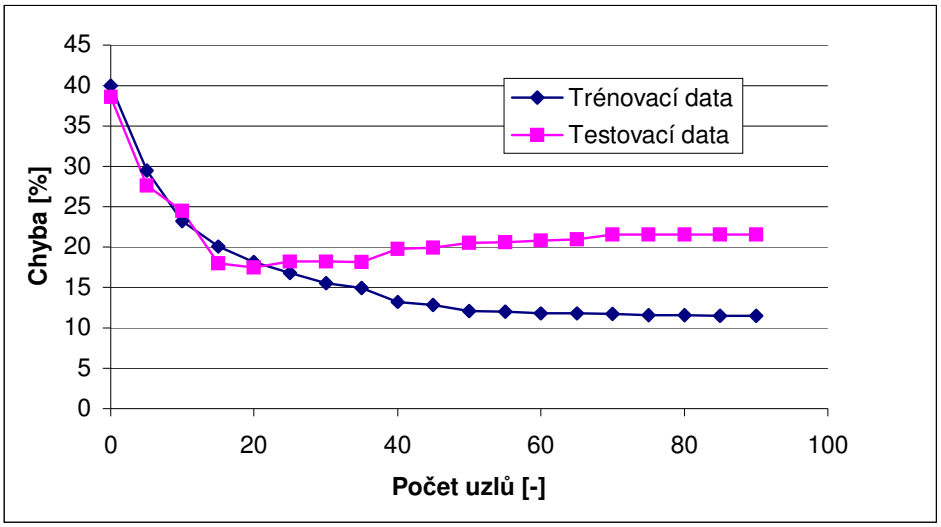
Obecně se žádný z vygenerovaných RS nepřiblížil (ani nemohl) skutečnému, velice jednoduchému řešení tvořenému lineární kombinací obou proměnných *x* a *y*. V praxi jsou data vždycky zašuměna a hlavně neznáme skutečné závislosti, které mezi daty existují. Smyslem tohoto příkladu je poukázat na to, že i když jsou RS velice složité a tedy nevyhovující, neznamená to, že mezi daty neexistuje velice jednoduchý vztah řešící hledanou závislost.

4.3.3 Další algoritmy pro generování RS

Prořezávání stromů

Strom vygenerovaný s důrazem na přesnost klasifikace může být **přeučený** (angl. overfitting). Prakticky to znamená, že ve stromu existují listy, do kterých spadá např. pouze

jeden z trénovacích záznamů. Takový strom sice klasifikuje trénovací data velice přesně, ale při klasifikaci nových záznamů dochází k častým chybám (Obr. 4.8). Příčinou je skutečnost, že strom z trénovacích dat extrahoval i souvislosti, které jsou náhodné nebo souvisejí se šumem a nepřesnostmi v datech. Jednou z možností, jak tyto chyby ošetřit, je provedení tzv. **prořezávání stromu**.



Obr. 4.8 Přeučený strom

Používány jsou metody, které buď zjednodušují rozhodovací strom během jeho generování nebo už vygenerovaný RS postupně zjednodušují. Techniky prořezávání se dále dělí podle toho, zda pracují pouze s trénovací množinou nebo i s testovacími daty. O volbě metody pak rozhoduje, zda je k dispozici dostatečné množství dat.

V případě, že je k dispozici dostatečné množství nezašuměných dat na vytvoření testovací množiny, lze použít tzv. **redukci chyby**. Redukce chyby je metoda využívající testovacích dat k prořezávání. Funguje tak, že nejdříve rozdistribuuje testovací data do jednotlivých listů. Pro každý nelistový uzel *S* stromu *T* je zjištěno, jak by se změnila kvalita klasifikace, kdyby byl *S* nahrazen svým nejlepším listovým uzlem. Jestliže nový strom s tímto upraveným uzlem klasifikuje stejně nebo lépe než strom původní a *S* neobsahuje žádný podstrom s touto vlastností, je *S* nahrazen daným listem. Tak se pokračuje, dokud nelze provést žádnou další úpravu, která by vedla ke zlepšení rozhodovacího stromu (menší velikost, lepší klasifikace). Podmínkou dobré funkčnosti tohoto modelu je kvalitní testovací množina dat (dostatečné zastoupení všech běžných typů dat). Jinak může dojít k odstranění části stromu, která pouze nebyla dostatečně zastoupena v testovacích datech.

V případě, že není dostatečné množství dat na vytvoření testovací množiny, používají se postupy založené na metodě cross-validation. Z trénovacích dat je vytvořeno *V* disjunktních podmnožin záznamů. Strom je generován *V*-krát, vždy z dat bez jedné podskupiny. Ta je použita jako testovací vzorek. Všech *V* stromů se analyzuje podle ceny, komplexnosti nebo na základě maximální věrohodnosti. Podle těchto údajů se strom prořezává a také se určuje jeho věrohodnost.

Chybové funkce

Jako hlavní chybové funkce jsou používány Entropie a Gini index. Při prořezávání bývá používána tzv. Micclassification rate.

Entropii vyjadřuje následující vztah:

$$Entropie = -\sum_{i=1}^{c} p(G_i)\log_2 p(G_i) \qquad (4.7)$$

$p(G_i)$ pravděpodobnost klasifikace do konkrétní třídy
c počet tříd, do kterých je klasifikováno

Gini index se počítá podle vzorce:

$$Gini = 1 - \sum_{i=1}^{c} p(G_i)^2 \qquad (4.8)$$

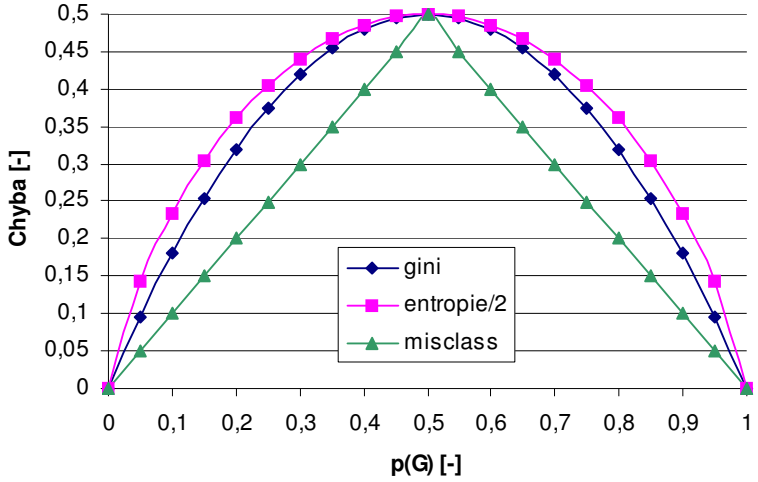
$p(G_i)$ pravděpodobnost klasifikace do konkrétní třídy
c počet tříd, do kterých je klasifikováno

Misclassification rate vyjadřuje vztah:

$$Misclass = 1 - p(G_J) \qquad (4.9)$$

$p(G_J)$ pravděpodobnost klasifikace do G_J – nejpočetněji zastoupené třídy

Srovnání průběhů těchto chybových funkcí znázorňuje Obr. 4.9. Hodnota entropie je podělena dvěma, aby byla lépe porovnatelná s Gini indexem.



Obr. 4.9 Srovnání průběhů chybových funkcí při binární klasifikaci

Ukončovací podmínky

Třídit do naprosté přesnosti je většinou nesmyslné, zejména u zašuměných dat. Takový postup vede k přeučení RS. Jednou z ukončovacích podmínek je nastavení minimálního počtu záznamů, při kterém ještě může dojít k větvení (např. 10). Pokud je počet prvků v uzlu menší, je uzel přeměněn na list s příslušnou klasifikací. Další možností je ukončení větvení, pokud je jedna třída v uzlu zastoupena ve větším poměru než stanovené minimum (např. 90%). Další omezení mohou souviset s maximální povolenou hloubkou stromu (např. max. 5). Podmínky vždy velice úzce souvisí s typem algoritmu a dat, se kterými je pracováno.

4.4 Souhrn

Rozhodovací strom je hierarchický nelineární systém umožňující nalezení a uložení znalostí a jejich využití k analýze nových dat. Rozhodovací stromy jsou primárně používány pro klasifikaci kvalitativních závislých proměnných na základě vstupních atributů.

Hlavními přednostmi a charakteristickými rysy rozhodovacích stromů jsou: **hierarchická struktura**, **nelinearita**, **srozumitelnost a čitelnost**, **flexibilita** (co do typu analyzovaných dat i co do topologie) a **existence algoritmů**, které umožňují jejich automatické vytváření. Příkladem algoritmů jsou např. ID3, CART nebo CHAID.

Rozhodovací strom se skládá z **kořenového uzlu** a dalších **uzlů** a **listů**. **Větve** spojují dva objekty (uzel-uzel nebo uzel-list) ze sousední hierarchické úrovně. Při průchodu uzlem jsou data rozdělena na základě podmínky do větví z uzlu vycházejících. Podmínka se může týkat hodnoty jednoho nebo kombinace více atributů. Dosažení listu při průchodu rozhodovacím stromem vede ke klasifikaci nebo predikci hodnoty výstupní veličiny analyzovaného záznamu.

Dalšími důležitými pojmy jsou **přeučený strom** (extrahoval chybné znalosti ze šumu či chyb v trénovacích datech), **prořezávání** (metody ke zmenšování přeučených nebo příliš komplexních stromů) a **ukončovací podmínka** (určuje, kdy algoritmus přestává dále vytvářet rozhodovací strom; špatné nastavení vede k přeučení).

Kontrolní otázky a úlohy

1. Cílem je vytvořit hru kámen-nůžky-papír, která bude hrát proti člověku a použije jako herní systém RS. Výstupní hodnoty jsou „kámen“, „nůžky“ a „papír“ (předpověď, jak bude hrát protihráč). Navrhněte nějaké vstupní atributy a definujte jejich definiční obor (hodnoty, které může daný atribut nabývat).

Atribut: počet výher v posledních třech kolech.
Výčet hodnot: 0,1,2,3.
Poznámka: může se ukázat, že pokud hráč vícekrát prohraje, probouzí se v něm agresivita a intuitivně více volí kámen nebo nůžky a minimálně papír.

Atribut: předcházející kolo [hrac,pocitac].
Výčet hodnot: 11,12,13,21,22,23,31,32,33 (1-kámen,2-nůžky,3-papír).
Poznámka: může se ukázat, že hráč má tendenci volit v dalším kole symbol, který v předešlém kole nebyl, nebo v případě výhry zkouší opakovat stejnou variantu.

Atribut: který symbol padl v posledních 10 tazích nejčastěji.
Výčet hodnot: 1,2,3,12,13,23 (10 není dělitelné 3, tedy nemůže nastat situace 123).
Poznámka: hráč má svůj oblíbený symbol, který používá nejčastěji.

2. Vysvětli pojmy „přeučený strom“, „prořezávání“ a „ukončovací pravidlo“.
Viz. Přehled základní terminologie.

3. Vyjmenuj 3 základní kategorie RS a urči hlavní atribut, který tyto stromy odlišuje.
CART, CLS, AID. Rozdíl spočívá zejména v topologii RS, dále v typu vstupních a výstupních dat.

4. Napiš vzorec vyjadřující Shanonovu entropii?
Viz. kapitola ID3.

5. V jakých případech selhává algoritmus CART?
Algoritmus CART selhává, pokud je vhodná dělící podmínka tvořena kombinací více atributů – např. při popisu plochy nebo prostoru (viz. kapitola CART – příklad).

Literatura

[20] Mařík, V., Štěpánková, O., Lažanský, J.: Umělá inteligence (1), Academia, Praha 1993

[21] Lewis R. J.: An Introduction to Classification and Regression Tree (CART) Analysis, http://www.saem.org/download/lewis1.pdf

[22] Machová K.: Strojové učenie, Košice 2002, http://neuron-ai.tuke.sk/~machova/SU4.pdf

[23] Wilkinson L.: Tree Structured Data Analysis: AID, CHAID and CART, http://www.spss.com/research/wilkinson/Publications/c&rtrees.pdf

[24] CHAID analysis, http://www.statsoft.com/textbook/stathome.html

[25] CART trees, http://www.statsoft.com/textbook/stathome.html

[26] Classification trees, http://www.statsoft.com/textbook/stathome.html

5 Učení založené na instancích

Cíle kapitoly:

Učení založené na instancích (IBL) patří mezi jednoduché a přitom velice účinné nástroje strojového učení. Po nastudování této kapitol je čtenář znalý základních principů IBL, což jsou jejich přednosti, nedostatky a omezení. Rozumí pojmu metrika a ovládá různé způsoby jejího vyjádření. Je znalý základních pravidel souvisejících s výběrem, ukládáním a vyhledáváním záznamů a základních algoritmů založených na principech IBL.

5.1 Úvod

princip IBL

Metody typu IBL (Instance Based Learning) fungují na základě podobnosti nového případu s případy již známými. Přestože jsou tyto metody principielně jednoduché, umožňují aproximovat i značně složité funkce s kvalitativní i kvantitativní výstupní hodnotou.

Z pohledu algoritmů IBL je proces predikce výstupní hodnoty nového prvku opravdu jednoduchý. Mějme nový prvek Z, který je popsaný vstupními veličinami $\boldsymbol{x}_Z$. Chceme určit jeho výstupní hodnotu y_Z. Máme uložena trénovací data, která jsou taktéž popsána stejnými atributy $\boldsymbol{x}$, známe u nich však i správné výstupní hodnoty y. Definujeme operaci *vzdálenost*, jejímž vstupem jsou dva prvky popsané vstupními atributy $\boldsymbol{x}$ a výstupem je kvantitativní vyjádření jejich vzájemné nepodobnosti, tedy vzdálenosti. Na základě této operace je v databázi uložených trénovacích záznamů nalezen předem daný počet prvků, které se svým popisem $\boldsymbol{x}$ nejvíce podobají prvku Z, který je určen vektorem $\boldsymbol{x}_Z$. Z této nově vytvořené množiny nejpodobnějších případů se bude následně určovat výstupní hodnota y_Z. Algoritmus tohoto postupu může principielně probíhat dvěma odlišnými způsoby:

- na základě **výstupních hodnot** y **nalezených sousedů** (popřípadě i jejich vzdáleností od prvku Z) je určena výstupní hodnota, tedy y_Z;
- na základě **atributů** $\boldsymbol{x}$ **a** y **popisujících nalezené sousedy** je vytvořen lokálně platný model (např. lineární, rozhodovací strom, neuronová síť,...), který generalizuje znalost v okolí neznámého prvku a bude sloužit k určení výstupní hodnoty y_Z; při nastavování parametrů může být součástí chybové funkce též vzdálenost sousedů od klasifikovaného prvku.

Společným rysem obou přístupů je *nalezení určitého počtu z uložených prvků, které jsou nové instanci nejpodobnější.* Metody zahrnující tento princip spadají do kategorie algoritmů pracujících na základě instancí, tedy IBL nebo též MBR (memory based reasoning).

charakteristiky IBL

Metody IBL se řadí mezi tzv. *líné* (ang. lazy) metody. Pro líné metody je specifické, že nevytvářejí žádný model. Výpočet sloužící k predikci výstupu provádějí až ve chvíli, kdy je předložen nový prvek k analyzování. Proto jsou metody IBL schopny dobře aproximovat závislosti uvnitř prostoru vymezeném trénovacími daty. Na druhou stranu nejsou schopny dobré extrapolace, tedy predikce mimo oblast pokrytou uloženými záznamy. Opakem k metodám líným

jsou metody *dychtivé* (ang. eager).

Algoritmy IBL představují určitý přechod mezi učením s učitelem a bez učitele. Protože je známa požadovaná výstupní hodnota u každého trénovacího prvku, je IBL jednoznačně algoritmus *s učitelem*. Pokud se však podíváme na základní algoritmus k-NN (k nearest neighbor), k žádnému „učení“ nedochází. Trénovací data jsou jednoduše uložena a predikce nového prvku je spočtena „průměrem“ nejbližších sousedů.

Metody IBL lze použít pro libovolné typy vstupních a výstupních dat (kvantitativní i kvalitativní), včetně jejich vzájemných kombinací.

V případě klasifikace mohou být algoritmy IBL citlivé na počet prvků spadajících do jednotlivých tříd. Při použití většího počtu sousedů ke klasifikaci může početněji zastoupená třída přehlasovat třídu početně méně zastoupenou.

Algoritmy IBL odvozují predikci neznámého prvku na základě k nejbližších sousedů. Pokud k odpovídá počtu všech trénovacích prvků, jedná se o *globální* metodu (výpočetně náročné). Pokud je (typicky) k menší než počet všech trénovacích prvků, jedná se o *lokální* metodu.

výhody IBL

Hlavní výhody IBL jsou:

- Možnost *lokálně ovlivňovat přesnost* metody (např. uložení více záznamů nebo jejich znásobení).
- *Proces učení* je omezen pouze na ukládání záznamů – je tedy *velice rychlý*. Nejsou požadovány žádné významné apriorní znalosti o vztazích mezi proměnnými.
- *Inkrementální charakter* – protože nedochází ke generalizaci, je možné při získání nových dat tato jednoduše přidat. Není přitom ani problém, pokud by nové prvky umožňovaly např. klasifikovat i do dosud neznámé třídy.
- Předností IBL je také skutečnost, že místo odhadu *jedné funkce* popisující celý prostor řešení je vytvářen *nový jednoduchý model* pro každou nově klasifikovanou instanci. To umožňuje řešit i velice složité a komplexní problémy pomocí jednoduchých aproximací s lokální platností.

nevýhody IBL

Z principu IBL však vyplývají i určitá omezení:

- Předpověď výstupní hodnoty nového prvku může být *časově náročná*. Čím více trénovacích prvků je uloženo, tím náročnější je nalezení *k* nejbližších sousedů. Vzniká tak i otázka, jak data co nejlépe uložit, aby bylo nalezení sousedů co nejrychlejší.
- Metody IBL jsou velice *citlivé na irelevantní atributy*. Pokud budou mezi 10 atributy pouze 3 relevantní a metodě IBL bude předloženo všech 10 atributů, metoda bude predikovat velice nepřesně.
- V případě použití metody *k* nejbližších sousedů (*kNN*) nedochází ke generalizaci (k té dochází až během klasifikace nové instance). Nevzniká obecný model, *znalost* uložená v datech *není pro člověka srozumitelná* a jednoduše interpretovatelná, i když metoda predikuje úspěšně.
- Zcela zásadní význam má *správná definice vzdálenosti* mezi dvěma prvky, což může být v případě kombinace více atributů různého typu obtížné.

5.2 Vzdálenost dvou prvků

proč potřebujeme metriku

Mějme nový prvek *q*, který je popsán svými *V* vlastnostmi $a_1(q),...,a_V(q)$. Hledáme jeho výstupní hodnotu *y*. Podstatou metod založených na instancích je nalezení určitého počtu prvků z trénovací databáze, které jsou co nejpodobnější prvku *q*. Na základě výstupních hodnot nalezených „nejbližších sousedů" následně usuzujeme, jaký bude asi nejpravděpodobnější výstup prvku nového. Jak ale určit, kteří sousedé jsou ti nejbližší? Aby mohlo k jejich výběru dojít, je třeba definovat vhodnou *metriku*, která bude vyjadřovat vzájemnou vzdálenost dvou libovolných prvků.

definice metriky *d*

Metrikou bude rozuměna funkce d: $\mathbf{X} \times \mathbf{X} \rightarrow \mathrm{R}$ taková, že

1. $\forall x_1, x_2 \in \boldsymbol{X} : d(x_1, x_2) \geq 0$
2. $d(x_1, x_1) = 0$
3. $d(x_1, x_2) = d(x_2, x_1)$
4. $\forall x_1, x_2, x_3 \in \boldsymbol{X} : d(x_1, x_2) + d(x_2, x_3) \geq d(x_1, x_3)$

normalizace kvantitativního atributu

Pokud používáme více vstupních atributů, které popisují různé veličiny v různých měřítcích, nastává typicky problém, kdy jedna veličina je např. řádově v tisících, druhá v setinách. Aby bylo možné vzdálenosti v rámci takových veličin srovnat, je nutné jednotlivé atributy normalizovat tak, aby byly hodnoty jejich metrik vzájemně srovnatelné. U kvantitativních veličin jsou nejčastěji používány následující dva typy normalizace:

Normalizace rozsahem (rozsah veličiny v intervalu $\langle 0;1 \rangle$). Pokud *max* označíme největší hodnotu, kterou může veličina mít (nebo která je mezi trénovacími daty) a *min* je minimální možná hodnota, je výpočet *i*-tého prvku upraven následujícím vztahem:

$$a_{norm}(x_i) = \frac{a(x_i) - max}{max - min} \tag{5.1}$$

Normalizace rozptylem (99% prvků v intervalu $\langle -3;3 \rangle$). Pokud σ označíme rozptyl veličiny a δ její střední hodnotu, normalizace i-tého prvku je provedena podle následujícím vztahem:

$$a_{norm}(x_i) = \frac{a(x_i) - \delta}{\sigma} \tag{5.2}$$

normalizace a metrika kvalitativního atributu

To také souvisí s kombinací atributů kvantitativních a kvalitativních, kde kvalitativním musíme přiřadit konkrétní číselné hodnoty vyjadřující míru podobnosti. Jejich volba souvisí s typem použité normalizace (respektive s rozsahem normalizovaných veličin). Jednoduše lze definovat vzdálenost tak, že jsou-li dva prvky stejné třídy, je vzdálenost vždy 0, jinak může být např. 1. Vzdálenosti mezi jednotlivými třídami (zejména při vícerozměrné klasifikaci) lze však vyjádřit i složitější tabulkou vzdáleností. Za příklad nám může sloužit množina jazyků – čeština, ruština, maďarština, finština a čínština. Tabulka 1 vyjadřuje podobnost jazyků.

Tabulka 5.1: Metrika pro podobnost jazyků

	CZ	**RU**	**HU**	**FI**	**CN**
CZ	0	0,2	0,8	0,8	1
RU	0,2	0	0,8	0,8	1
HU	0,8	0,8	0	0,3	1
FI	0,8	0,8	0,3	0	1
CN	1	1	1	1	0

Jed o ilustrativní příklad, zasvěcený lingvista by s uvedenými hodnotami souhlasit nemusel. Podstatné ale je, že musí stále platit 4. definice metriky.

metriky

Metrik, které splňují uvedené podmínky, je celá řada. Snad nejčastěji používaná je euklidovská vzdálenost, nicméně nelze tvrdit, že její použití je za všech okolností tím nejlepším řešením. Existuje celá řada dalších možností. V následujících vztazích je V počet atributů (tedy vstupních veličin), přičemž zápisem $a_i(x_j)$ rozumíme i-tý atribut prvku x_j.

Euklidovská vzdálenost

$$d(x,z)=\sqrt{\sum_{j=1}^{V}\left(a_j(x)-a_j(z)\right)^2} \tag{5.3}$$

Manhattanská (Hammingova) vzdálenost

$$d(x,z)=\sum_{j=1}^{V}\left|a_j(x)-a_j(z)\right| \tag{5.4}$$

Překrytí (overlap)

$$d(x,z)=\sum_{j=1}^{V}\partial\left(a_j(x),a_j(z)\right) \tag{5.5}$$

$$\textbf{\textit{kde}}\ \partial\left(a_j(x),a_j(z)\right)\begin{cases}=0, \textbf{\textit{pokud}}\ a_j(x)=a_j(z)\\ =1, \textbf{\textit{pokud}}\ a_j(x)\neq a_j(z)\end{cases}$$

Chebychevova vzdálenost

$$d(x,z)=\max_{\forall j\in V}\left|a_j(x)-a_j(z)\right| \tag{5.6}$$

tečná vzdálenost

Tečná vzdálenost (ang. tangential distance) je specifickou metrikou používanou zejména při rozlišování psaného textu. Umožňuje eliminovat různé typy transformací obrázků (roztažení, zkosení, rotaci, ...).

5.3 Výběr a uložení prvků, vyhledání nejbližších sousedů

jak urychlit nalezení nejbližších sousedů

Na výběru prvků a způsobu jejich uložení závisí rychlost klasifikace a také míra dosažené generalizace. Výběr z trénovacích instancí se provádí tak, že jsou vymazány ty, které predikují nesprávně statisticky významně často. K ukládání vhodných záznamů se pak používá hierarchická struktura v podobě stromové struktury. Postupnou dekompozicí jsou na základě jednotlivých atributů vytvářeny podprostory. Hledání nejbližších sousedů se pak omezuje na prohledávání pouze určitých podprostorů, ve kterých je výskyt nejbližších sousedů nejpravděpodobnější.

V následujícím textu jsou popsány metody IB1, IB2, IB3 a IB4. Dále je uveden základní algoritmus pro strukturované ukládání a vyhledávání instancí označované jako k-d trees (ang. *k*-dimensional trees).

výběr prvků IB1

Nejjednodušší z algoritmů. Všechny trénovací instance jsou uloženy. Neztrácí se tedy žádná informace.

Nedostatkem je vysoká náročnost na paměť. Zároveň se s rostoucím počtem uložených instancí zvyšuje výpočetní náročnost při predikci výstupní hodnoty nového prvku.

výběr prvků IB2

Modifikovaná verze IB1. Uloženy jsou pouze ty trénovací instance, které jsou již uloženými chybně klasifikovány. Pokud je nově předložený příklad klasifikován správně, je ihned zapomenut (ukládají se tedy pouze chybně zařazené prvky).

Výhodou je skutečnost, že je tak výrazně snížena náročnost na paměť při zachování dobré přesnosti. To ale jenom za podmínek, že data nejsou zašuměná a neobsahují chybné prvky. Ty jsou totiž prakticky vždy uloženy jako chybně klasifikované.

výběr prvků IB3

Princip IB3 rozvíjí myšlenku IB2 tím, že dříve, než dojde k vyřazení konkrétní instance se archivuje informace o tom, kolikrát predikovala dobře a kolikrát špatně. Tyto údaje se vyhodnocují statistickým významovým testem. Používají se dva prahy důvěry. Práh pro akceptování a práh pro odmítnutí. Podle umístění vůči těmto dvěma prahům se uložená data dělí do tří skupin (akceptované, pozorované a zamítnuté záznamy). Principielně je sledováno, zda je úspěšnost klasifikace jednotlivé instance statisticky významně větší než frekvence její třídy v trénovacích datech.

Metoda IB3 je robustní vůči šumu. Na druhou stranu každá výjimka výpadá jako instance obsahující šum a nedostane se mezi akceptované záznamy.

IB3 - příklad

Mějme instanci *Z*. Třída, kterou reprezentuje, je v trénovacích datech zastoupena z 30%, tedy při náhodném výběru jednoho prvku z trénovacích dat je pravděpodobnost volby této třídy $p=0,3$. Instance *Z* byla pro klasifikaci prvků použita 50-krát. Princip IB3 spočívá v tom, že hranici pro akceptaci splňuje *Z* tehdy, pokud bylo podle něj klasifikováno dobře významně vícekrát, než kdyby šlo o pouze náhodnou volbu (tedy $0,3.50=15$). Při hladině významnosti $\alpha=5\%$ je pro akceptaci nutné, aby prvek *Z* klasifikoval správně alespoň 23-krát. Pro zamítnutí je doporučována hladina významnosti $\alpha=12,5\%$. Z toho lze vypočíst, že pokud naopak *Z* klasifikoval správně maximálně 10-krát, je jeho úspěšnost

natolik špatná, že je jako potenciální vzor zamítnut. Pokud se počet úspěšných klasifikacích Z pohybuje v těchto mezích, je prvek dále pozorován a jeho přesnost je stále vyhodnocována.

výběr prvků IB4

IB4 rozšiřuje IB3 v tom smyslu, že nepředpokládá stejnou závažnost všech atributů. Váha atributu je označována jako *relevance*.

Tento algoritmus je úspěšný v případech, kdy je záznam popsán větším počtem atributů.

význam ukládání

Pokud jsou již vybrány instance, které budou použity ke klasifikaci, je výhodné uložit je do stromové struktury, která umožní rychlé nalezení nejbližších sousedů. Je třeba si uvědomit, že při klasifikaci nového prvku je vždy znovu prováděna generalizace pro nový bod na základě jeho k nejbližších sousedů. Jejich nalezení přitom může představovat i 99% výpočetního času potřebného ke konečné predikci. Důležitost efektivního hledání je tedy zřejmá.

k-d stromy

Základní algoritmus pro ukládání instancí se jmenuje k-d trees. Instance jsou ukládány do stromové struktury, takže algoritmus nalezení souseda má složitost log_2N, kde N je počet uložených instancí. Každý uzel obsahuje informace o dělící veličině, prahu (tedy dělícím kritériu) a neutrální zóně prahu, která neobsahuje žádné prvky. Každý test uložený v uzlu dělí množinu trénovacích dat na dvě části. Pokud je strom striktně binární a má hloubku d, pak bude mít 2^d listů. Má-li být ve stromu uloženo N prvků, musí být splněno, že $2^d \geq N$.

k-d stromy – vytváření

Vytváření k-d stromu je prováděno v následujících krocích:

- je-li jeden případ, konec
- zvol v případě prvního dělení libovolnou veličinu, v případě následujících dělení libovolnou veličinu odlišnou od veličiny předcházející
- uspořádej trénovací prvky podle vybrané veličiny; vyber dvě prostřední instance a jejich průměrnou hodnotu ulož jako *práh*; ulož také vzdálenost mezi prahem a oběma body – *hranice* nebo též *neutrální zóna* (je stejná na obě strany)
- rozděl všechny instance podle prahu na dvě podmnožiny
- pokračuj v tomto dělení, dokud není každá koncová množina jednoprvková

Existují i odlišné varianty pro tvorbu k-d stromů. Např. práh nemusí být průměr dvou středních hodnot, ale může procházet přímo prostředním prvkem (mediánem). Takový strom je pak ve výsledku menší (z pohledu hloubky). Dále existují postupy pro dynamickou úpravu stromu – vymazání a přidání nového prvku. V obou případech však již není garantována balancovaná struktura stromu, což může mít za následek delší dobu potřebnou k nalezení nejbližšího souseda. Pokud je přidáno větší množství prvků, může být vhodné strom vytvořit znovu.

k-d stromy – vyhledávání

Hledání nejbližšího souseda probíhá principielně následujícím způsobem:

- porovnej nový prvek q s prahem současného uzlu; výsledek určuje množinu podobných prvků
- pokračuj předešlým postupem v procházení stromu, až je určen konečný prvek; tento prvek je potenciální nejbližší soused (teoreticky zde může vyhledávání skončit, nalezený prvek však nemusí být nejbližší soused)

- urči vzdálenost *d* mezi prvkem *q* a nalezeným potenciálním nejbližším sousedem
- pokračuj nyní směrem nahoru tak, že postoupíš k uzlu rodičovskému (o řád vyššímu)
- porovnej vzdálenost *d* se vzdáleností prvku *q* od *prahu + hranice*; pokud je *d* menší, druhou větev stromu lze vyloučit (prvek jí určený nebude bližší než dosud nalezený nejbližší soused); pokud však bude vzdálenost menší, je třeba zjistit vzdálenost mezi prvkem *q* a novou instancí z této větve stromu; je-li nová vzdálenost menší než *d*, je určen nový potenciální nejbližší soused a je stanoveno nové *d*
- takto se dále pokračuje směrem nahoru; pokud druhá větev rodičovského uzlu (strom je striktně binární) než ze které se přichází může obsahovat bližší prvek, je třeba nový podstrom předešlým postupem prohledat (porovnání prahů, zamítnutí podstromu nebo procházení do hloubky, nalezení možných sousedů)
- nakonec se tímto postupem dojde do nejvyššího uzlu, u kterého se rozhoduje stejným způsobem; tento postup vede k nalezení nejbližšího souseda

Jak je z algoritmu patrno, nalezení *potenciálního* nejbližšího souseda je opravdu jednoduché a rychlé. Nalezení skutečného nejbližšího souseda se však může mnohonásobně prodloužit. Přesto je tento postup nesrovnatelně rychlejší než obyčejné porovnávání.

Nakonec je třeba zmínit, že rychlost prohledávání ovlivňuje např. vhodná volba dělícího atributu. Také velice záleží na tom, jestli analyzovaný prvek leží v blízkosti více prahů (prohledávání se prodlouží) nebo je ve zcela specifické oblasti, což povede k rychlému zamítnutí všech alternativních řešení.

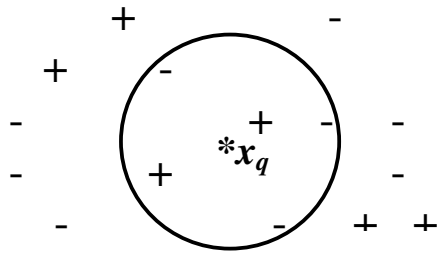
5.4 Algoritmy IBL

rozhodnutí předcházející volbou algoritmu

Mezi základní problémy patří již zmíněná *vhodná volba atributů* (irelevantní atributy výrazně ovlivňují klasifikaci), stanovení *správné metriky* (důležité při kombinaci různých typů veličin, zejména kvalitativních a kvantitativních). Pokud strukturovaně uložíme nezbytně nutná data, jsme připraveni predikovat hodnotu nových prvků.

stanovení vhodného počtu sousedů

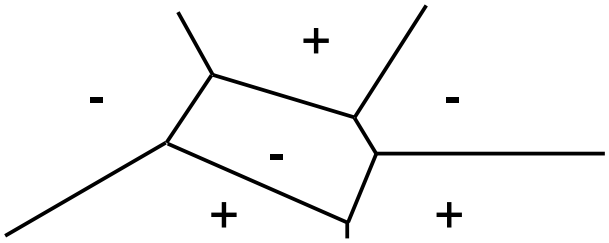
Po zvolení konkrétního typu algoritmu je třeba rozhodnout o tom, na základě kolika *k* sousedů bude predikce prováděna. Jedním z přístupů je použití metody cross-validation. Data jsou rozdělena např. na dvě poloviny, přičemž jedna půlka dat je uložena a druhá je použita ke klasifikaci. Postupným vyzkoušením různých hodnot *k* je stanoven optimální počet sousedů. Tento postup může být výpočetně a tedy i časově náročný. Na obrázku () je znázorněno, jak velikost k ovlivňuje výslednou klasifikaci neznámého prvku $\boldsymbol{x_q}$. Pokud *k=1*, je výsledná klasifikace „+“. Pokud je však *k=5*, je výstupní třída „-“.



Obr. 5.1: k-NN při *k=5*

Voroného graf

Předešlý obrázek představuje klasifikaci jedno prvku. Podobně lze určit třídu pro každý bod v grafu. Pokud *k=1*, pak určením třídy všech prvků dojde k rozdělení plochy na polygony, které vymezují oblasti určující prvky jednotlivých tříd na základě hodnoty nejbližšího souseda. Tomuto grafu se říká Voroného graf (ang. Voronoi diagram), viz. obrázek .



Obr. 5.2: Voroného graf

5.4.1 K-Nearest Neighbour

základní informace

Jedná se o základní algoritmus založený na podobnosti s nejbližšími sousedy. Pokud je rozhodováno na základě jediného nejbližšího souseda (*k=1*), používá se označení nearest neighbour.

trénování

K definici vzdálenosti mezi dvěma instancemi je používána euklidovská vzdálenost. Ve své základní podobě představuje učení pouhé uložení všech trénovacích dat. Vstupní vektor hodnot bude označován $\boldsymbol{x}$, výstupní hodnota $f(\boldsymbol{x})$.

algoritmus k-NN kvalitativní

Základní algoritmus rozhoduje o výsledné klasifikaci tak, že zjistí třídy všech *k* nejbližších sousedů. Do třídy, která je v tomto výběru nejpočetněji zastoupena je zařazen prvek klasifikovaný.

Mějme nový neznámý prvek $\boldsymbol{x}_q$, jehož výstupní třídu chceme určit, přičemž existuje *V* možných tříd. Pak výstupní klasifikaci metodou k-NN vypočteme podle následujícího vztahu:

$$\hat{f}(\boldsymbol{x}_q) = \arg\max_{v \in V} \sum_{i=1}^{k} \partial(v, f(\boldsymbol{x}_i)) \tag{5.7}$$

kde $\delta(a,b)=1$, pokud $a=b$, jinak $\delta(a,b)=0$.

V případě, že je početně stejně zastoupeno více tříd, rozhoduje o klasifikaci nejbližší soused, jehož výstupní hodnota patří do jedné z tříd představující možné řešení.

algoritmus k-NN kvantitativní

V případě kvantitativní výstupní veličiny je výstupní hodnota vypočtena jako průměr k sousedů, tedy:

$$\hat{f}(\boldsymbol{x}_q) = \frac{\sum_{i=1}^{k} f(\boldsymbol{x}_i)}{k} \tag{5.8}$$

váhová metoda k-NN

Váhová metoda vychází z předpokladu, že čím je soused vzdálenější od nové instance, tím menší má vliv na její konečnou výstupní hodnotu. Typická váha w je převrácená hodnota čtverce vzdálenosti, tedy:

$$w_i = \frac{1}{d(x_q, x_i)^2} \tag{5.9}$$

Pro klasifikační algoritmy pak platí úprava vztahu () následujícím způsobem:

$$\hat{f}(\boldsymbol{x}_q) = \arg\max_{v \in V} \sum_{i=1}^{k} w_i \cdot \partial(v, f(\boldsymbol{x}_i)) \tag{5.10}$$

V případě, že mezi nejbližšími soused jsou prvky přímo identické, došlo by ve výpočtu w k dělení nulou. V takové situaci je výstupní hodnota určena přímo hodnotou prvku identického. Pokud existuje takových prvků víc, rozhoduje opět nejpočetněji zastoupené řešení nebo další nejbližší soused. Náhoda jako rozhodovací kritérium nebývá používána, metoda by přestala být deterministická (v identickém příkladě by při opakování výpočtu mohlo dojít k rozdílné predikci).

Vztah pro výpočet výstupní veličiny při kvantitativní predikci vypadá následovně:

$$\hat{f}(\boldsymbol{x}_q) = \frac{\sum_{i=1}^{k} w_i \cdot f(\boldsymbol{x}_i)}{\sum_{i=1}^{k} w_i} \tag{5.11}$$

poznámky

Při použití váhové metod lze použít globální variantu (tedy výpočet přes všechny trénovací záznamy), což u metody bez váhy možné není.

Váhová metoda k-NN je vysoce efektivní induktivní inferenční metoda používaná pro mnoho praktických problémů. Je odolná k výskytu šumu v trénovacích datech a velmi efektivní, pokud je k dispozici dostatek trénovacích dat.

Vzdálenost mezi instancemi se počítá pomocí všech atributů (všech os euklidovského prostoru). Tím se liší od metod založených na pravidlech a rozhodovacích stromech, kde se pro vytváření hypotézy používá jen podmnožina

atributů. Z toho ale také vyplývá vysoká citlivost na irelevantní atributy (ang. curse of dimensionality). K řešení tohoto problému se používá váhování samotných atributů. Tím je měněn jejich rozsah a tedy i jejich možný vliv na konečnou vzdálenost od ostatních prvků. V extrémních případech může být vhodné některé atributy zcela vyloučit.

5.4.2 Lokálně vážená regresní metoda

princip

Lokálně vážená regresní metoda využívá nejbližších sousedů k nastavení modelu s lokální platností, respektive s platností omezenou na novou instanci. Běžně bývá používána lineární či kvadratická funkce, může však být použita např. neuronová síť nebo jiný libovolný model. Vážená je metoda tehdy, když je chyba každého souseda váhována na základě jeho vzdálenosti od nové instance. Díky tomuto přístupu lze i jednoduchými funkcemi aproximovat i velice složité závislosti.

Hlavní rozdíl mezi tímto přístupem a k-NN spočívá v tom, že k-NN vezme sousedy, zjistí jejich výstupní hodnoty, vzdálenosti od analyzovaného bodu a na základě těchto údajů určí výstupní hodnotu.

$$\hat{f}(\boldsymbol{x}_q) = g\left(f(\boldsymbol{x}_1), d(\boldsymbol{x}_q,\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_k), d(\boldsymbol{x}_q,\boldsymbol{x}_k)\right) \tag{5.12}$$

Oproti tomu lokálně vážená regresní metoda se pokouší na základě nejbližších sousedů provést lokální generalizaci závislosti v podobě modelu či funkce g, která předpovídá výstupní hodnotu nového bodu na základě hodnot jeho jednotlivých atributů $a_1, \ldots, a_V$.

$$\hat{f}(\boldsymbol{x}) = g\left(a_1(\boldsymbol{x}), \ldots, a_V(\boldsymbol{x})\right) \tag{5.13}$$

upravené chybové funkce

Označme D množinu všech trénovacích záznamů. Běžně používaná chybová funkce založená na nejmenších čtvercích vyjadřuje celkovou chybu následujícím způsobem:

$$E = \frac{1}{2} \cdot \sum_{x \in D} \left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) \right)^2 \tag{5.14}$$

Aby byla chyba platná jen v omezení na nejbližších k sousedů, je její hodnota určena následujícím vztahem:

$$E(\boldsymbol{x}_q) = \frac{1}{2} \cdot \sum_{i=1}^{k} \left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) \right)^2 \tag{5.15}$$

Zde se však neprojevuje vliv vzdálenosti jednotlivých prvků oproti analyzovanému bodu. Funkci $K(d(x_q,x_i))$ označujeme jako penalizační funkci. Její hodnota se zmenšuje s rostoucím d (tedy chyba prvku blízko analyzovanému bodu ovlivňuje parametry modelu víc než bod vzdálenější). V ideálním případě je taková funkce použita následujícím způsobem:

$$E(\boldsymbol{x}_q) = \frac{1}{2} \cdot \sum_{x \in D} \left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) \right)^2 \cdot K\left(d(\boldsymbol{x}_q,\boldsymbol{x}_k)\right) \tag{5.16}$$

Složitost výpočtu v takovém případě ovlivňuje lineárně počet prvků v množině D. Z toho důvodu se používá aproximaci přes nejbližší sousedy:

$$E(\boldsymbol{x}_q) = \frac{1}{2} \cdot \sum_{i=1}^{k} \left(f(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}) \right)^2 \cdot K\left(d(\boldsymbol{x}_q, \boldsymbol{x}_k)\right) \tag{5.17}$$

poznámky

Hlavní nevýhodou tohoto přístupu je skutečnost, že každý nalezený model je platný pouze pro jeden jediný bod oboru hodnot. Použití složitějších modelů pro lokální aproximaci může být časově velice náročné (např. učení neuronové sítě). Nicméně prakticky jsou lineární či kvadratické aproximace v případě dostatečného počtu trénovacích prvků uspokojivě přesné.

5.4.3 Obecně regresní modely

kernelové metody

Mezi zástupce patří např. radiální bázové funkce (RBF, ang. radial basis functions). Vzhledem ke struktuře modelu bývá tato metoda řazena mezi neuronové sítě, svými vstupy však spadá i do IBL. Principielně jde o to, že definiční obor je popsán pomocí tzv. kernelových funkcí K_i. Lineární kombinace jejich funkčních na základě vah w_i pak určuje konečný výstup. Při tvorbě RBF modelu je třeba vhodně zvolit jádra (respektive body c_i, ve kterých bude jádrová funkce definována). Vstupem jádrové funkce je pak vzdálenost neznámého prvku x_q a jádra kernelové funkce d. Výstup se zpravidla nepočítá ze všech funkcí, ale z funkcí v předem definovaném okolí analyzovaného prvku (pro příliš vzdálené body je funkční hodnota kernelové funkce prakticky nulová).

$$\hat{f}(\boldsymbol{x}) = w_0 + \sum_{u=1}^{k} w_0 K_u \left(d\left(c_u, x_q\right)\right) \tag{5.18}$$

Protože metoda používá vzdálenost bodu od jádra d a pracuje s určitým počtem okolních jádrových funkcí, bývá RBF zařazováno mezi metody založené na instancích.

ostatní modely

Další přístupy se opírají o tradiční modely (kvadratické, exponenciální a jiné regresní funkce, rozhodovací stromy, neuronové sítě, SVM, ...). Pro každý nový prvek jsou však tyto modely nastaveny znovu na základě trénovacích dat – nejbližších sousedů analyzovaného prvku. Vznikají tak lokální aproximace využívající generalizační přednosti vybraných modelů. Podstatnou nevýhodou tohoto přístupu je časová náročnost spojená s predikcí každého nového prvku – musí být nalezen příslušný počet nejbližších sousedů, načež je vytvářen model. Tento proces např. u neuronových sítí může být časově příliš náročný. To však záleží na typu problému. Predikce volebních preferencí a řízení polohového servopohonu vykazuje odlišné časové nároky.

5.5 Souhrn

typické vlastnosti

IBL se liší od ostatních přístupů k modelování zejména tím, že není vytvářena obecná hypotéza (model) nad celým prostorem trénovacích dat. Model je nahrazen prostým ukládáním známých případů. Ve fázi modelování je pak na základě určitého počtu nejbližších sousedů nového neznámého prvku vytvářen jedinečný, lokálně platný model, zpravidla velice jednoduchý (lineární, kvadratický). Přesto je touto cestou dosahováno vysoké přesnosti i při řešení velice složitých a komplexních systémů.

Největší úskalí při použití IBL představuje skutečnost, že veškeré podstatné výpočty jsou prováděny až při analýze neznámého prvku. Jde zejména o vyhledání nejbližších sousedů (v případě velké databáze může být časově velice náročné) a nastavení modelu (v případě, že je použita např. neuronová síť). Další nebezpečí spočívá v přítomnosti irelevantních atributů, které přesnost predikce výrazně zhoršují; metoda sama si s nimi však poradit nedovede.

příprava dat

Prvním z kroků je eliminace irelevantních atributů, které účinnost algoritmů založených na instancích významně znehodnocují. Dále je třeba zvolit vhodnou metriku, která umožní vyjádření vzájemné vzdálenosti mezi prvky. Na jejím základě budou hledáni nejbližší sousedé nového neznámého prvku. Přepokládáme pak, že neznámý prvek bude mít podobnou výstupní hodnotu (případě bude patřit do stejné třídy) jako jemu nejpodobnější známé prvky.

trénování modelu

Trénování IBL spočívá v uložení trénovacích záznamů. Ukládat všechny známé prvky by mohlo být časově velice náročné, proto existují jednak algoritmy (IB1, IB2, ...), které slouží k výběru pouze doopravdy potřebných trénovacích záznamů, dále pak algoritmy (k-d trees), které umožňují ukládat vybrané prvky do hierarchické stromové struktury tak, aby bylo nalezení nejbližších sousedů výpočetně co nejméně náročné.

predikce

Konečná predikce pak probíhá tak, že je nalezen určitý počet trénovacích prvků, které jsou nové instanci nejpodobnější. Předpokládaný výstup nového prvku je pak odvozen z výstupních hodnot sousedů – buď jako průměr, nejčetnější zastoupená třída, nebo je vytvořen ze sousedů lokálně platný model (neuronová síť, rozhodovací strom, ...), který je po vlastním naučení použit pro predikci výstupní hodnoty neznámé instance. Tento postup však může být časově značně náročný. Na druhou stranu umožňuje dosažení vysoké přesnosti i v případě modelování velice složitých nelineárních závislostí.

Kontrolní otázky a úlohy

1. V čem (v jakých parametrech) je uložena znalost algoritmů IBL?
Přímo v samotných uložených trénovacích datech. Během procesu učení je jediným ovlivňovaným parametrem výběr těchto instancí (v nejjednodušším případě můžou být uložena všechna data).

2. Jaké jsou hlavní přednosti a omezení metod IBL.
Přednosti: schopnost modelovat i velice složité a nelineární systémy, inkrementální, možnost lokálního zpřesnění, jednoduchý proces učení.

Omezení: citlivost vůči irelevantním atributům, výpočetně náročný proces vybavování, člověkem těžko interpretovatelný model, problém jak vhodně definovat metriky atributů.

3. Co je to metrika?

Metrika je funkce d: $\mathbf{X} \times \mathbf{X} \rightarrow \mathrm{R}$. *Jejím výstupem je reálné číslo vyjadřující vzdálenost mezi dvěma prvky. Vzhledem k tomu, že prvek může být popsán více různými atributy (kvantitativními, kvalitativními), je nutné nejdříve stanovit metriku v rámci jednotlivých veličin, normalizovat ji a pak stanovit metriku pro celé instance (např. jaká je vzdálenost mezi instancemi pudl {váha:4kg, počet nohou:4, barva: bílá } a kosem {váha:0,2kg, počet nohou:2, barva: černá}). Nevhodně zvolená metrika zapříčiní, že vzdálenost bude např. s velkou převahou určovat pouze jeden atribut (přeceněný), nebo se prakticky vůbec neprojeví (podceněný).*

4. Co je to normalizace?

Normalizace je úprava rozsahu veličiny. Garantuje velikost určitých parametrů (např. rozsah možné hodnoty atributu, průměrnou hodnotu a rozptyl, ...).

5. K čemu slouží metody IB1, ..., IB4?

Jedná se o algoritmy, které provádějí v trénovacích datech výběr instancí, které má smysl uložit pro účely budoucí predikce. Jedná se prakticky o učicí algoritmy. Na rozdíl od metody IB1 – kdy jsou uloženy všechny instance, metody IB2, ... poněkud sofistikovanějšími postupy eliminují instance, jejichž uložení je buď zbytečné, nebo dokonce celkovou kvalitu predikce zhoršují.

6. K čemu slouží algoritmus k-d tree?

Algoritmus k-d tree slouží k uložení vybraných trénovacích dat do hierarchické stromové struktury za účelem snadnější hledání nejbližších sousedů nové instance

7. Lze metodu k-NN použít i k predikci kvantitativní veličiny?

Ano

8. V čem spočívá princip lokálních regresních modelů na bázi IBL?

Termín lokální naznačuje, že model je aproximací mající platnost jen v určitém omezeném prostoru. Regresní model (může být i klasifikátor) je pak libovolný model schopný predikce. Princip IBL se uplatňuje tak, že v okolí nové instance je nalezen určitý počet sousedů. Na rozdíl od metody k-NN se však tato data použijí na naučení specifického modelu, který pak slouží ke klasifikaci nové instance. Komplikací s tím související je skutečnost, že pro další nový prvek budou nalezeny jiné nejbližší prvky, bude tedy nutno vytvořit opět nový, lokálně platný model. Tento přístup může být (podle typu použitého modelu) výpočetně značně náročný.

5.6 Literatura

[1] Berry M.J.A., Linoff G.S.: Data Mining Techniques, Wiley Publishing, Inc., 2004. ISBN 0-471-47064-3.

[2] FRANK, E., HARRELL, J. Regression Modeling Strategies. NY: Springer, 2001. 568 pages. ISBN 0-387-95232-2.

[3] Hastie T., Tibshirani R., Friedman J.: The Elements of Statistical Learning. Springer, 2001. ISBN 0-387-95284-5.

[4] Mitchel T.: Machine Learning. MacGraw Hill Science, 1997. ISBN 0070428077.

[5] Schölkopf B., Smola A.J.: Learning with Kernels. MIT Press, Cambridge, MA, 2002. ISBN 0-262-19475-9.

[6] Theodoridis S.: Pattern Recognition, Elsevier, 2003. ISBN 0-12-685875-6.

APPENDIX A

A.1 Klasifikátory – ROC analýza

historie ROC

ROC analýza má své počátky ve druhé polovině 20. století. Původně byla určena pro detekci signálů. Jednoduchost a srozumitelnost vedly k jejímu rozšíření mezi diagnostické systémy zejména v oblasti medicíny. Následovaly aplikace ve strojovém učení a data miningu.

V posledních letech došlo k nárůstu publikací zabývajících se ROC analýzou. Středem zájmu je plocha pod ROC, tzv. „area under the ROC“ (AUC). Jde o skalární charakteristiku vyjadřující predikční kvalitu klasifikátoru. Vzhledem ke své jednoduchosti nemůže AUC vyjádřit komplexnější požadavky. V důsledku toho může být při porovnání klasifikátorů její hodnota zavádějící, což však bezprostředně souvisí s její chybnou interpretací, respektive přeceněním její informační hodnoty. Navzdory tomuto nedostatku se však díky své jednoduchosti a nezávislosti na typu rozložení prosazuje ve stále větším počtu aplikací. Pro úplnost je třeba dodat, že AUC není ničím zcela novým. Její obdobu lze nalézt v některých neparametrických statistikách (Wilcoxonův test pořadí, Gini index, Mann-Whitneyův test, Somer's D_{xy}).

AUC byla postupně využívána k vyhodnocení kvality predikce klasifikátorů a porovnávání různých modelů. To vedlo k pokusům o její využití jako chybové funkce a dále rozšíření na vícerozměrné klasifikace (ROC je ve své původní podobě definováno pro binární modely). Zejména poslední dvě oblasti, AUC jako chybová funkce a vícerozměrné AUC, jsou tématem publikací v posledních letech.

členění kapitoly

Následující podkapitoly popisují základy ROC analýzy. Zabývají se postupně kontingenčními tabulkami, grafem ROC a plochou pod ROC pro binární a vícerozměrné modely. Na tyto základy navazují kapitoly o aproximačních metodách a uplatnění ROC analýzy.

A.2 Čtyřpolní tabulka

kontingenční tabulky

Ke zjištění a posouzení vztahu mezi dvěma veličinami se používají tzv. kontingenční tabulky. Kategorie jedné veličiny určují řádky a kategorie druhé veličiny sloupce. Do jednotlivých buněk tabulky jsou zaneseny četnosti výskytů prvků spadajících do příslušných kategorií obou veličin. Pokud má jedna veličina r a druhá s kategorií, je vytvořena tabulka o rozměru $r \times s$. Tabulka o rozměru 2×2 je označována jako čtyřpolní tabulka.

typy výsledků binární klasifikace

Binární klasifikace je typickou úlohou, jejíž výsledky lze zapsat do čtyřpolní tabulky. Jednou veličinou jsou skutečné hodnoty a druhou veličinou predikované hodnoty. Existují celkem 4 typy možných výsledků. Pokud je klasifikováno do tříd formálně označených 0 a 1, jsou to typy výsledků uspořádané dvojice (1,1), (1,0), (0,1) a (0,0), kde první číslo znamená klasifikaci modelu a druhé číslo

skutečnou hodnotu. Z kombinací četností výskytů jednotlivých typů výsledků lze vyjádřit základní parametry predikčních vlastností modelu.

čtyřpolní tabulka

Uvedené četnosti výskytů jednotlivých typů výsledků zapsané do čtyřpolní tabulky jsou uvedeny v tabulce 2.2.

Tabulka A.1: Rozdělení výsledků binární klasifikace

		Skutečnost	
		positivní	negativní
Predikce	positivní	**A (TP)**	**B (FP)**
	negativní	**C (FN)**	**D (TN)**

Hodnota A bývá označována také jako TP (true positives) a její hodnota udává četnost výskytu daného typu výstupu. Podobně je tomu i u ostatních hodnot. Cílem modelu je maximalizovat velikost hodnot A a D a minimalizovat četnost chybných klasifikací B a C.

základní parametry

Z uvedené tabulky se určují parametry, z nichž některé dále uvádím:

Senzitivita (TPR – true positives rate): počet správně pozitivně klasifikovaných / počet všech, u kterých k události došlo.

$$senzitivita = \frac{A}{A+C} \tag{A.1}$$

Specificita: počet správně negativně klasifikovaných / počet všech, u kterých k události nedošlo.

$$specificita = \frac{D}{B+D} \tag{A.2}$$

Pozitivní prediktivní hodnota: počet správně pozitivně klasifikovaných / počet všech pozitivně klasifikovaných.

$$PPV = \frac{A}{A+B} \tag{A.3}$$

Negativní prediktivní hodnota: počet správně negativně klasifikovaných / počet všech negativně klasifikovaných.

$$NPV = \frac{D}{C+D} \tag{A.4}$$

Přesnost (accuracy): počet všech správně klasifikovaných / počet měření.

$$presnost = \frac{A + D}{A + B + C + D} \tag{A.5}$$

senzitivita, specificita, PPV

Senzitivita je číslo udávané v intervalu ⟨0;1⟩ nebo v procentech ⟨0;100⟩, které udává poměr úspěšně zachycených pozitivních událostí vůči všem skutečným pozitivním událostem. Pokud by tedy byly všechny testované prvky označeny za pozitivní, bylo by dosaženo maximální senzitivity, tedy 1. To by se však projevilo např. na specificitě, která určuje počet úspěšně zachycených negativních událostí vůči všem skutečně negativním událostem – její hodnota by byla 0.

Dalším významným ukazatelem je pozitivní prediktivní hodnota (PPV). Její hodnota určuje, jaké procento z pozitivně označených prvků bylo skutečně pozitivní. Pokud je výrazný nepoměr mezi počtem pozitivních a negativních událostí, může se stát, že i při relativně vysoké senzitivitě a specificitě (90%) bude PPV velice nízká, což může být závažný nedostatek. Např. je-li zachyceno 9 z 10 pozitivních a 450 z 500 negativních případů, bude TP=9, FP=50, FN=1 a TN=450. Pak je senzitivita i specificita rovna 90%, ale PPV je pouhých 15%.

A.3 Graf ROC

binární klasifikátor

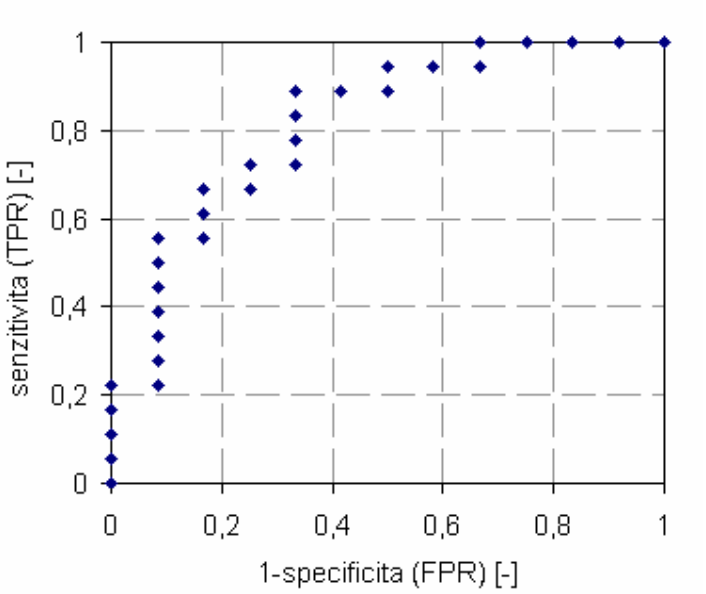
Kvalitu dichotomního klasifikátoru lze vyjádřit pomocí čtyřpolní tabulky a parametrů uvedených v předešlé kapitole. Používaným grafickým znázorněním je zanesení těchto informací do grafu ROC. Na ose *x* je FPR (False Posivite Rate), tedy *1-specificita* a na ose *y* TPR (True Positive Rate), tedy *senzitivita*. Jeden klasifikátor odpovídá v grafu ROC jednomu bodu. Zjednodušeně pak v grafu ROC platí, že čím je bod blíže hornímu levému rohu, tím je klasifikátor lepší.

regresní klasifikátor

Problém regresních klasifikátorů spočívá v tom, že výstupem regresního modelu je spojitá veličina a až rozdělení dat podle stanovené kritické hodnoty rozhodne o konečné klasifikaci. Spojitý výstup modelu je často v intervalu ⟨0;1⟩. Ne vždy však výstupní hodnota představuje absolutní pravděpodobnost dané klasifikace. A pokud ano, pak zpravidla za předpokladu určitého rozložení dat, jehož volba je dalším stupněm volnosti v procesu modelování.

graf ROC pro regresní klasifikátor

Mějme dvojice hodnot $(f(\boldsymbol{x}_i), g_i)$, přičemž hodnota $f(\boldsymbol{x}_i)$ je výstupem regresního modelu a je to spojitá veličina, zatímco g_i je očekávaný binární výstup. Ve výsledném klasifikátoru bude určena kritická hodnota, jejíž srovnání s $f(\boldsymbol{x}_i)$ rozhodne o binární predikci klasifikátoru. Při hledání nejvhodnější kritické hodnoty je možné postupovat tak, že její hodnotu stanovíme menší než nejmenší prvek $f(\boldsymbol{x}_i)$. Pak ji zvětšíme tak, že bude mezi nejmenším a předposledním nejmenším prvkem atd. Nakonec bude její hodnota větší než největší hodnota $f(\boldsymbol{x}_i)$. Pokud byl soubor dat tvořen *N* různými záznamy, bude tímto postupem vytvořeno *N+1* kritických hodnot. Pokud je pro každou takto stanovenou kritickou hodnotu vypočtena příslušná senzitivita a specificita, lze tyto údaje zanést do grafu ROC. ROC slouží k přehlednému zobrazení predikčního potenciálu modelu a může vypadat následovně:



Obr. A.1: Graf ROC

interpretace grafu ROC

Následující odstavce popisují některé důležité vlastnosti ROC:

- Graf není spojitý, ale diskrétní. Jednotlivé body odpovídají konkrétním klasifikátorům, tedy modelům. Ty vznikají u regresních klasifikátorů tak, že výstupní kvantitativní veličině $y=f(\boldsymbol{x})$ je určena kritická hodnota K_r, v porovnání se kterou ($y_i<K_r$, $y_i \geq K_r$) je rozhodnuto o binární klasifikaci jednotlivých prvků. Výsledná klasifikace tedy závisí na velikosti K_r. Je-li v trénovací množině N prvků, existuje celkem $N+1$ možných způsobů klasifikace v závislosti na K_r. Podstatné je, že jsou-li výstupní prvky y_i uspořádány podle velikosti, stejný klasifikátor vznikne volbou libovolného K_r z intervalu $(y_i,y_{i+1}\rangle$. Graf ROC je tedy nezávislý na typu rozložení veličiny y, je neparametrický.
- Zjednodušeně je obecným cílem při klasifikaci dosažení co největší senzitivity a zároveň specificity. Optimálním bodem je tedy levý horní roh (0;1). V tomto místě klasifikátor predikuje se stoprocentní přesností. V bodě (0;0) je specificita maximální, ale senzitivita nulová – klasifikátor označil všechny testované prvky za negativní. V bodě (1;1) je nulová specificita a senzitivita rovna jedné – klasifikátor označil všechny testované prvky za pozitivní.
- Všechny klasifikátory, jejichž hodnota se nachází v ROC na úhlopříčce $y=x$ klasifikují náhodně a nemají žádný informační přínos. Pokud klasifikátor reprezentuje bod pod touto úhlopříčkou, model klasifikuje hůř než náhodně. Jednoduše lze říci, že klasifikuje „obráceně“. Negace výstupu takového modelu mění umístění jeho bodu v grafu ROC na místo symetrické podle diagonály $y=x$.
- Z grafu lze vyčíst více informací. V úloze z oblasti medicíny může být cílem např. dosažení co největší senzitivity (95%) i za cenu nízké specificity (50%) a pozitivní prediktivní hodnoty. Takovou situací může být použití levné a neškodné léčby, která je aplikována mezi rizikovými pacienty takřka plošně. Naopak může nastat situace, kdy je třeba dosáhnout vysoké pozitivní

prediktivní hodnoty (90%), což se zpravidla projeví sníženou senzitivitou (30%) a v důsledku toho vyšší specificitou. Příkladem může být implantace kardiostimulátoru. Podmínkou není umožnit velice drahou léčbu všem pacientům, u kterých by k infarktu během určitého období skutečně došlo. Podmínkou je garance, že pokud již někomu kardiostimulátor bude implantován, jedná se o skutečně ohroženého pacienta. Podobné příklady lze nalézt i v mnoha jiných oborech.

nezávislost na pravděpodobnostní funkci

Použití ROC analýzy vychází z vyjádření míry vzájemné uspořádanosti pozitivních a negativních případů v závislosti na vysvětlující veličině. Jediným předpokladem je, že s rostoucí hodnotou vysvětlující veličiny roste pravděpodobnost pozitivní (negativní) klasifikace. Výstupy v intervalu $\langle 0;1 \rangle$ mohou být tedy transformovány libovolnou ryze monotónní funkcí bez vlivu na výslednou kvalitu ROC analýzy. Z uvedené vlastnosti vyplývá nezávislost této metodiky na tvaru pravděpodobnostní funkce za uvedené podmínky (ryzí monotónnost).

nezávislost na typu rozložení

Další vlastností ROC je nezávislost na četnosti zastoupení jednotlivých tříd, což vede opět k nezávislosti na typu rozložení. Mějme model, který pozná pozitivní případy v 80% a negativní v 90%. Pokud by byly klasifikovány pouze pozitivní případy, je celková přesnost 80%. Pokud by v testované množině byly pouze negativní případy, je celková přesnost 90%. Se změnou poměrného zastoupení pozitivních a negativních případů se bude měnit celková přesnost klasifikace (accuracy), přestože model zůstává stále stejný a stejně účinný. Změna v četnosti zastoupení však na senzitivitu ani na specificitu vliv nemají, poroto na ní ROC nezávisí.

parametrická ROC křivka

Kromě neparametrického grafu ROC, který je tvořen jednotlivými body, existuje tzv. parametrický graf ROC. Jedná se o křivku proloženou jednotlivými body neparametrického ROC grafu. Její parametry vycházejí z předpokládaného rozložení dat, což je její nevýhoda v porovnání s neparametrickou ROC křivkou. Např. data s gaussovským (normálním) rozložením jsou v grafu ROC reprezentována tzv. binormální křivkou, která má dva parametry (rozdíl mezi průměrem prvků s klasifikací 0/1 a poměr rozptylů mezi prvky s klasifikací 0/1).

vícerozměrný graf ROC

S vícerozměrnou klasifikací roste komplexnost řešení. Je-li na základě dané veličiny x klasifikováno do n tříd, existuje při klasifikaci n^2 možných typů predikce, přičemž pouze n je správných a n^2-n chybných. Grafické znázornění ROC ve dvourozměrném grafu již není možné.

Jedním z možných řešení je vytvoření n různých grafů ROC pro každou třídu zvlášť. Postupně je každá ze tříd c_i použita jako referenční – pozitivní a všechny ostatní třídy jako negativní. Tak je z n-rozměrné kontingenční tabulky vytvořeno n dvojrozměrných tabulek a pro každou z nich lze v závislosti na kritické hodnotě vytvořit ROC graf. Nedostatkem tohoto přístupu je citlivost na rozložení analyzované veličiny.

Další možnou variantou je vytvoření všech možných neuspořádaných dvojic ze tříd, do kterých se klasifikuje a pro každou dvojici vykreslení grafu ROC. Výhodou je nezávislost na typu rozložení analyzované veličiny, nevýhodou narůstající počet grafů, které je třeba vytvořit. Ten je roven $n(n-1)/2$ oproti pouhým n grafům z předešlého přístupu, tedy kvadratická komplexnost oproti lineární. To se promítá i do výpočetní náročnosti algoritmů využívajících tyto grafy.

Souhrnně lze říci, že analýza vícerozměrného regresního klasifikátoru na základě odečítání z grafu ROC je nepraktická pro svou komplikovanost a nebývá používána. Názornost a jednoduchost platící pro binární klasifikaci již neplatí. Graf sám o sobě je však pouze jedním výstupem ROC analýzy vhodným zejména pro vizuální prezentaci. Významná číselná charakteristika grafu ROC je velikost plochy pod jeho křivkou, tzv. AUC (Area Under ROC) a ta je skalární nezávisle na počtu tříd, do kterých se klasifikuje. Jejím výpočtem a využitím se zabývají následující podkapitoly.

A.4 Plocha pod grafem ROC – AUC

význam a interpretace AUC

Pro ROC obecně platí, že čím blíže se body charakteristiky nacházejí u levého horního rohu, tím je model přesnější. Jako charakteristika „predikčního potenciálu“ modelu se proto používá plocha pod ROC (AUC). Její hodnota se pohybuje v intervalu $\langle 0;1 \rangle$. Hodnota 0,5 odpovídá náhodné klasifikaci. AUC vyjadřuje kromě plochy pod křivkou také míru uspořádanosti prvků. „AUC je ekvivalentní pravděpodobnosti, že náhodně vybraný pozitivní prvek bude zařazen výše než náhodně vybraný negativní prvek“. Z toho také vyplývá podobnost s některými neparametrickými statistickými testy.

přednosti AUC

AUC je skalární charakteristika z intervalu $\langle 0;1 \rangle$ vyjadřující jedním číslem celkový predikční potenciál regresního klasifikátoru. Nepotřebuje pro své vyčíslení určení konkrétní kritické hodnoty, tedy ani konečné klasifikace. Narozdíl od MNČ nebo ML je však nezávislá na typu rozložení dat. Volba nevhodné distribuční funkce může výslednou velikost chyby určené na základě těchto funkcí významným způsobem ovlivnit.

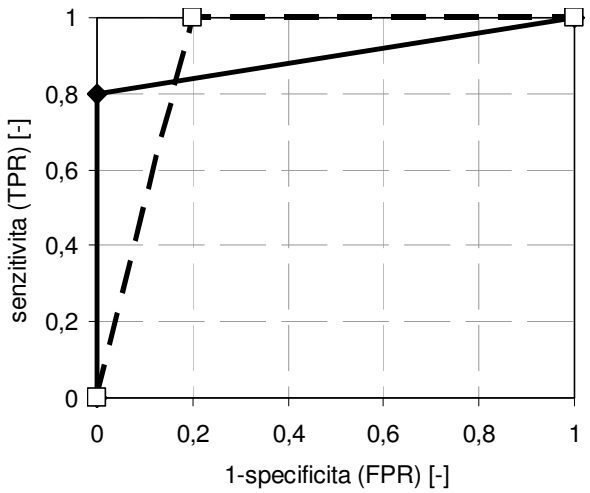
Vstupní veličina sloužící pro vytvoření ROC křivky a výpočet AUC musí být typu ordinálního. Z toho vyplývá další výhoda. Jako vstup lze použít veličinu typu „míra rizika“ (nulová, malá, střední, velká, úplná).

AUC je počítána ze senzitivity a specificity. Protože změna četnosti jednotlivých tříd v testovacích datech nemá vliv na velikost těchto charakteristik (oproti např. přesnosti – accuracy), je na četnosti nezávislá i AUC. Pokud tedy model predikuje pozitivní událost s odlišnou úspěšností než událost negativní, neprojeví se tato skutečnost na velikosti AUC, i když se v různých studiích budou počty prvků jednotlivých tříd měnit.

omezení AUC

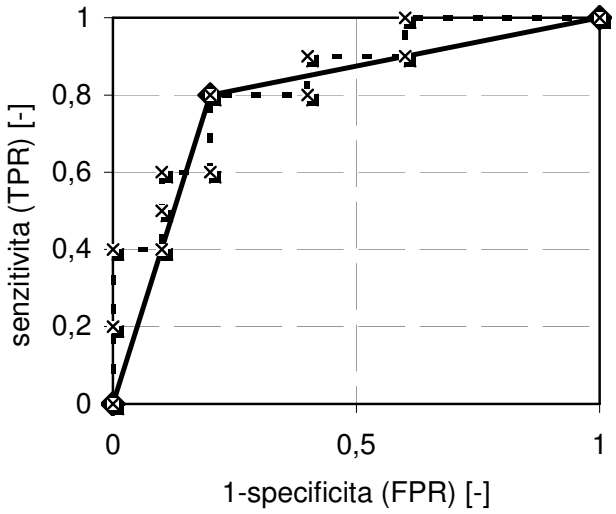
V případě, že je znám typ rozložení dat, má AUC menší výpovědní hodnotu než test parametrický. To už je patrné ze skutečnosti, že vstupní veličina, i když je kvantitativní, je při výpočtu použita pouze jako ordinální, čímž dochází ke ztrátě informace.

Dále nelze obecně říci, že čím je hodnota AUC větší, tím je model lepší nebo přesnější. V případě kvality modelu je rozhodující, jaká informace je upřednostňována, případně jaká je váha chybné klasifikace. Obě charakteristiky v na grafu v obr. 2.7 mají stejnou hodnotu AUC. Vhodnost modelu závisí na závažnosti (váze) chyby I. a II. druhu. Rozdíl mezi modely však není z velikosti AUC patrný.



Obr. A.2: Dvě různé ROC křivky se stejnou AUC

Velikost AUC vyjadřuje celkově kvalitu vztaženou ke všem možným klasifikátorům, které lze vytvořit. Z následujícího grafu je zřejmé, že plocha pod přerušovanou křivkou bude větší než pod křivkou vyznačenou plnou čarou. Podle obecných pravidel je však nejlepší možný klasifikátor v obou případech stejný (bod nejbližší levému hornímu rohu je v obou průbězích identický). Graf s větší plochou má tu přednost, že predikuje lépe v případech, kdy bude odlišena závažnost chyby I. a II. druhu.



Obr. A.3: Dvě ROC křivky s různým AUC a stejným optimálním bodem

parametrická ROC

Body tvořící ROC křivku bývají aproximovány parametrickou funkcí. Popis ROC se tak zjednoduší na typ funkce a její parametry. Pokud je např. předpokládáno, že analyzovaná data mají normální rozložení, ROC křivka bude mít binormální rozložení, které je určeno dvěma parametry. Jedná se však o aproximaci ROC podmíněnou předpokladem o rozložení dat.

AUC pod parametrickou ROC

Obecně platí, že parametrická AUC (vypočtená z parametrické ROC křivky) se liší od neparametrické AUC. Důvodem je skutečnost, že parametrická AUC je buď konvexní (v případě, že plocha je větší než 0,5) a pak je její plocha větší, nebo konkávní (AUC je menší než 0,5) a pak je její velikost menší než plocha neparametrické AUC. AUC vypočtená z parametrické ROC křivky je vždy aproximací podléhající přepokládanému typu rozložení dat.

výpočet binární AUC

Nejjednodušší algoritmus s výpočetní komplexností $O(n^2)$ vychází ze vzájemného porovnání všech prvků s rozdílnou klasifikací, jak uvádí následující vztah:

$$AUC = \frac{1}{n^+ n^-} \sum_{j=1}^{n^+} \sum_{k=1}^{n^-} g\left(x_j^+ - x_k^-\right) \tag{A.6}$$

V uvedeném vztahu odpovídají symboly n^+/n^- počtu prvků klasifikovaných jako pozitivní/negativní, x_j^+/x_k^- určuje velikost *j*-tého/*k*-tého prvku výstupní veličiny regresního modelu a $g(x)$ je heavisidova funkce.

Algoritmus s komplexitou $O(n.log_2 n)$ v první fázi uspořádá prvky podle veličiny x (tím je dána celková linearitmická komplexita) a až následně pokračuje podle následujícího vzorce:

$$AUC = \frac{1}{n^+ n^-} \sum_{j=1}^{N} \left(n_{=j}^- n_{>j}^+ + \frac{n_{=j}^- n_{=j}^+}{2} \right) \tag{A.7}$$

kde N je celkový počet prvků, $n^-_{=j}/n^+_{=j}$ je počet negativních / pozitivních prvků se stejnou velikostí x a $n^+_{>j}$ je počet pozitivních prvků s indexem větším než j (prvky jsou seřazeny podle velikosti x).

SE pro AUC

Rozptyl chyby (SE) hodnoty AUC závisí na typu rozložení. Uvedená aproximace v případě jiného než normálního rozložení uvádí rozptyl větší:

$$SE = \sqrt{\frac{A(1-A) + (n_p - 1)(Q1 - A^2) + (n_n - 1)(Q2 - A^2)}{n_p n_n}} \tag{A.8}$$

kde pro *Q1* a *Q2* platí, že

$$Q1 = \frac{A}{2-A} \qquad Q2 = \frac{2A^2}{1+A} \tag{A.9}$$

A.5 Ekvivalenty a aproximace AUC

ekvivalenty AUC

AUC je interpretována jako plocha pod ROC křivkou. Existuje však hned několik jiných charakteristik a testů, které mají obdobné vlastnosti a jsou vzájemně ekvivalentní. Patří sem Gini index, Somersovo D_{xy}, Mann-Whitneyův test a Wilcoxonův test.

gini index

Gini index je používán zejména v oblasti ekonomiky k vyjádření nerovnosti v příjmech obyvatel různých států. Slovy ROC analýzy je Gini index dvojnásobkem plochy mezi ROC křivkou a hlavní diagonálou. Z grafické interpretace AUC vyplývá následující vztah:

$$AUC = \frac{Gini\ coefficient + 1}{2} \tag{A.10}$$

somersovo D_{xy}

Somersovo D_{xy} určuje míru asociace, tedy do jaké míry asociuje *x* proměnnou *y*.

Nejdříve je vytvořena tabulka rozměru 3×3. Je definováno $\Delta x = Xi - Xj$ a $\Delta y = Yi - Yj$. Z *i* a *j* jsou vytvořeny všechny páry, pro které platí, že *i>j*, a do následující tabulky jsou zapsány počty vzniklých kombinací (tab. 2.3)

Tabulka A.2: Počty prvků v souboru se stejnou a rozdílnou diferencí

	Δx>0	**Δx=0**	**Δx<0**
Δy>0	**T1**	**T2**	**T3**
Δy=0	**T4**	**T5**	**T6**
Δy<0	**T7**	**T8**	**T9**

Definovány jsou následující hodnoty ***C***=*T1+T9*, ***D***=*T3+T7*, ***E***=*T4+T5+T6* a ***F***=*T2+T8*, ze kterých lze vypočítat následující neparametrické koeficienty míry asociace:

$$\begin{aligned} &Kruskal\text{-}Wallis\ \gamma = && (C-D)/(C+D) \\ &Somer's\ D_{xy} = && (C-D)/(C+D+F) \\ &Kendall's\ \tau = && (C-D)/(C+D+E+F) \end{aligned} \tag{A.11}$$

Následující odstavce interpretují význam jednotlivých koeficientů:

Kruskal-Wallis γ: zachycuje pouze situace, kdy se Δx i Δy liší od nuly. Pokud ale pro jedno Δx existuje více různých hodnot y, koeficient tuto informaci nezachytí. Koeficient určuje korelaci, ne regresi.

Somersovo D_{xy}: podstatná je složka F, která rozšiřuje korelační vztah o jednostrannou závislost – regresi. Absolutní hodnota SD se snižuje, pokud je Δx rovno nule, ale Δy není. To znamená, že nezávislá veličina je stejná a závislá se mění. Takový případ lze interpretovat jako např. nedostatečnou citlivost.

Kendallovo τ: kromě složky F je zde i složka E, která je nenulová, pokud pro Δy, které bylo nulové, máme různé hodnoty Δx. Toto číslo však již vyjadřuje, do jaké míry lze asociovat i zpětně *x* z *y*, což není smyslem regrese.

Mezi neparametrickým testem míry asociace „Somersovo D_{xy}“ a plochou pod ROC křivkou platí následující vztah:

$$AUC = \frac{D_{xy} + 1}{2} \tag{A.12}$$

Mann-Whitneyův a Wilcoxonův test

V literatuře zabývající se ROC analýzou je popsána podoba mezi AUC a dvěma neparametrickými testy: Wilcoxonův test pořadí a Mann-Whitneyův test. Podoba vyplývá z interpretace AUC, které je „mírou pravděpodobnosti, že v náhodného páru složeného z pozitivního a negativního případu budou tyto oba korektně rozlišeny“. Z Mann-Whitneovy statistiky U lze vypočíst AUC podle následujícího vztahu:

$$AUC = \frac{n^+ n^- - U}{n^+ n^-} \tag{A.13}$$

Řešení vstupního testu

1. a) **Kvantitativní data** (proměnné) **vyjadřují skutečnou hodnotu měřitelné vlastnosti** (kvantum). Zpravidla mají nějaký rozměr (fyzikální, %, ...).
   Např. hmotnost (kg), úrok (%), atd.

   b) **Kvalitativní data** (proměnné) **vyjadřují příslušnost do třídy nebo pořadí prvků.** Nemají zpravidla žádný rozměr. Pokud je lze uspořádat, skutečné rozdíly mezi různými sousedními hodnotami nemusí být stejné. Jsou to tedy i všechny proměnné vyjadřující pořadí (jsou uspořádané, ale neznáme skutečné rozdíly mezi sousedními hodnotami).
   Např. proměnná „barva“ (nelze vyjádřit matematickou operaci rozdíl: „žlutá“ - „modrá“ = „?“) nebo proměnná „pořadí zvířat podle hmotnosti“ s výčtem hodnot {morče, pes, nosorožec} (lze sice určit pořadí, nicméně skutečný rozdíl hmotností mezi morčetem a psem není totožný s rozdílem mezi nosorožcem a psem, přestože podle pořadí je rozdíl vždy roven jedné).

2. Kvantitativní data (proměnné) se dělí na **diskrétní** a **spojité.**

3. Kvalitativní data (proměnné) se dělí na **ordinální** (existuje relace větší, menší – tedy např. všechna pořadí) a **nominální** (neexistuje relace větší, menší, ... – např. proměnná „barva“).

4. Mějme matematický model se vstupy a výstupy. **Nezávislá proměnná = vstupní proměnná**.

5. Mějme matematický model se vstupy a výstupy. **Závislá proměnná = výstupní proměnná**.

6. **Klasifikátor** „klasifikuje“ do tříd. Jeho výstupní proměnná je tedy kvalitativní. **Regresní model** má kvantitativní výstupní proměnnou (v praxi je bohužel možné setkat se i s výkladem obecnějším; regresním modelem bývá označován libovolný model mající vstupy i výstupy).