ROBUST'2000, 267 – 283

OPTIMÁLNÍ KLASIFIKAČNÍ STROMY

PETR SAVICKÝ, JAN KLASCHKA A JAROMÍR ANTOCH

Abstrakt. Classification and regression trees have been traditionally grown by recursive partitioning, i.e. by a top-down search for "locally optimal" splits. The "local", or "one-step", optimization of splits can to some extent, using the present power of computer hardware, be substituted by the full optimization of whole trees. In this paper, two bottom-up tree optimization algorithms are outlined. Since our procedures guarantee the minimum classification error on the training data only, some results of exeriments aimed at the investigation of the generalization properties of the optimal trees are presented.

Резюме: Методы анализа данных при помощи моделей в форме дерева развиваются с 60-ых лет нашего века. Классификационные и регрессионные дерева конструируемые в прошлых десятилетиях не обладали глобальной оптимальностью, поскольку вычислительная техника не позволяла больше чем оптимизацию индивидуальных ветвлений. Благодаря неудержимому прогрессу технологии, оптимизация деревьев все-таки стала в самом конце тысячелетия реализируемой задачей. В статье изучаются два алгорифма построения классификационных деревьев обладающих минимальной ошибкой классификации используемых в процессе конструкции дерева данных. Точность построенных нашими алгорифмами деревьев при классификации других независимых данных обсуждается на основе эксперимента.

1. Úvod: Stromy včera a dnes

Na obr. 1 je naznačena úloha analýzy dat: Dva typy objektů, kolečka a křížky, mají být od sebe odděleny na základě hodnot vysvětlujících proměnných X_1 a X_2. Na obr. 2 je klasifikační strom, který problém dokonale řeší. Na první pohled je tedy vidět, že úloha je pro klasifikační stromy jako stvořená. Na druhý pohled už věc není tak jednoduchá. Klasifikační strom, který je perfektním řešením, existuje, ale není jisté, jestli by se jej obvyklými metodami konstrukce klasifikačních stromů podařilo najít. Větvení podle X_1 v kořeni totiž nepřináší žádný bezprostřední efekt – pro $X_1 = 0$ i $X_1 = 1$ jsou relativní četnosti koleček a křížků stejné. Teprve následné rozvětvení podle X_2 vede k oddělení obou druhů objektů. Záleží tedy na tom, jakým způsobem se řešení v množině všech možných klasifikačních stromů hledá. Pokud metoda sleduje jen bezprostřední efekt větvení "jeden krok vpřed", může "minout" velmi dobrý klasifikátor.

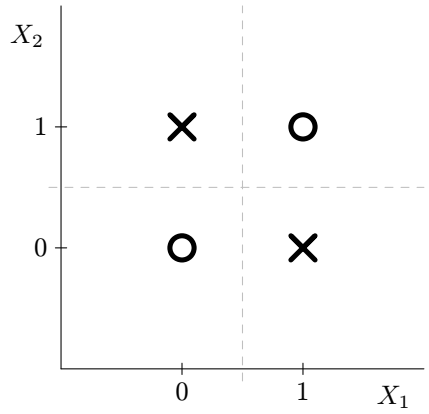
Prakticky všechny metody analýzy dat založené na stromech, které se objevily od 60. let do současnosti, mají jedno společné: *Vytvářejí stromy "shora dolů"*. Nejdříve vyberou vhodné větvení kořenového uzlu a určí jeho "potomky" (dva nebo více, podle toho, o jakou metodu se jedná), potom hledají co nejlepší větvení pro tyto

2000 *Mathematics Subject Classification.* Primary 62H30; Secondary 62-07.

Klíčová slova. Klasifikační stromy, kombinovaná ztráta, optimální stromy, algoritmy, procedury zdola-vzhůru.

Práce prvních dvou autorů byla podporována grantem GA ČR 201/00/1482, práce třetího autora grantem GA ČR 201/00/0769 a MSM 113200008.

"potomky", atd. Přitom je dobře známo, že stromy, které tímto způsobem vznikají, nejsou obecně "globálně" optimální. Jinými slovy: Pokud existuje vynikající klasifikační strom, k jehož konstrukci jsou třeba, jako v situaci na obr. 1, kroky, jejichž efekt se neprojeví okamžitě, může se snadno stát, že tento strom nebude nalezen, a přednost dostane méně kvalitní klasifikátor.

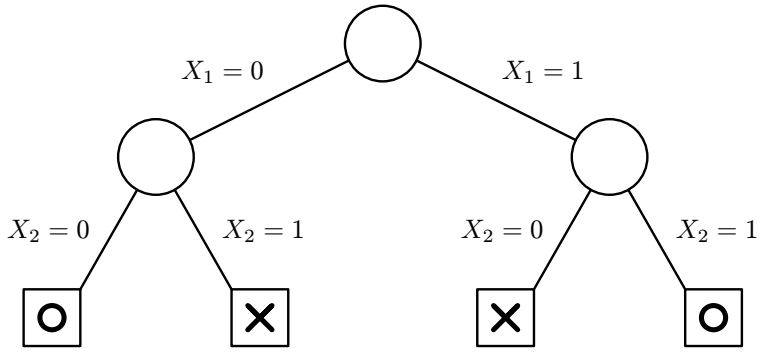


Obr. 1. Jednoduchá modelová situace: Kolečka a křížky se mají oddělit na základě hodnot proměnných X_1 a X_2.

Breiman *et al.* (1984) obhajují tento přístup ve své klasické monografii o klasifikačních a regresních stromech (viz paragraf 2.5.8) slovy: *"At this stage of computer technology, an overall optimal tree growing procedure does not appear feasible for any reasonably sized data set."* Také Gelfand *et al.* (1991) říkají o optimalizaci stromů skepticky: *"... it is clear that computing cost has some relevance in classification tree design: nobody seriously suggests designing truly optimal trees (optimizing over all possible sequences of splits, etc.)."*

Výkonnost počítačů od poloviny 80. let (i od začátku 90. let) řádově vzrostla, takže dnes si můžeme *do určité míry* dovolit "pěstovat" stromy optimálně. V tomto článku budou popsány dva nové algoritmy umožňující při nepříliš velkém počtu vysvětlujících proměnných konstruovat klasifikační stromy, které *mají mezi všemi možnými binárními klasifikačními stromy téže velikosti nejmenší chybu klasifikace* na trénovacích datech. Naše dosavadní experimentální zkušenosti ukazují, že stromy, které jsou v uvedeném smyslu optimální, mají často také dobré generalizační vlastnosti (tj. dobře se chovají na datech, která nebyla použita ke konstrukci stromu). V několika příkladech klasifikačních úloh, kde vztah mezi prediktory a závisle proměnnou je složitý, byly generalizační vlastnosti optimálních stromů prokazatelně lepší než u stromů vytvořených "klasicky". Není úplně jasné, jestli by Breiman *et al.* uznali, že v úlohách, které jsme dnes schopni pomocí optimálních klasifikačních stromů řešit, se vyskytují "any reasonably sized data sets", popř. zda by Gelfand *et al.* shledali, že optimální stromy navrhujeme "seriously".

Kromě klasické (zejména Breiman *et al.*, 1984) a novější (např. Siciliano, 1998) "stromové" literatury inspirovaly náš výzkum také práce z poslední doby (např. Pijls & Bioch, 1999), které reprezentují nový trend: *Některé problémy analýzy dat jsou sice velmi časově a prostorově složité (např. jsou NP-těžké, popř. mají exponenciální složitost), ale "rozumně velké" úlohy se dají v únosném čase řešit, když se veškerá data uloží do operační paměti.*



Obr. 2. Klasifikační strom, který řeší úlohu z obr. 1.

2. Základní pojmy a značení: Krychle, stromy a chyby

Budeme se zabývat klasifikačními modely s $P \geq 1$ prediktory (nezávisle proměnnými) $X_1, \ldots, X_P$ nabývajícími hodnot z $\{0, 1\}$ a $K \geq 2$ třídami $C_1, \ldots, C_K$. (Zobecnění pro vícehodnotové prediktory je možné např. s použitím pomocných binárních proměnných.)

Data jednotlivého případu sestávají z *vektoru prediktorů* $\mathbf{x} = (x_1, \ldots, x_P) \in \{0, 1\}^P$ a *kódu třídy* $C \in \{C_1, \ldots, C_K\}$. Množina $\{0, 1\}^P$ je *prostor prediktorů*.

Velmi frekventovaným pojmem je v této práci *krychle*. *Krychlemi* zde nazýváme takové podmnožiny $B = A_1 \times \cdots \times A_P$ prostoru prediktorů, že pro každý index i je buďto $A_i = \{0\}$, nebo $A_i = \{1\}$, nebo $A_i = \{0, 1\}$.

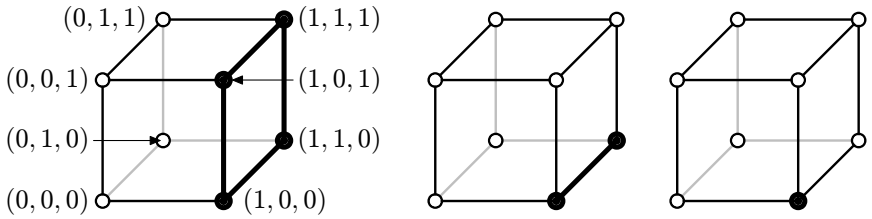
Protože zápis krychlí ve formě kartézského součinu je těžkopádný, používáme spíše reprezentaci krychlí čísly (kódy) v trojkové soustavě. Je-li $B = A_1 \times \cdots \times A_P$ krychle, odpovídá jí číslo s ternárním rozvojem $a_1 \ldots a_P$, kde pro $i = 1, \ldots, P$ je $a_i = 0$, když $A_i = \{0\}$, $a_i = 1$, když $A_i = \{1\}$ a $a_i = 2$, když $A_i = \{0, 1\}$. (Toto kódování krychlí využívá také program, o kterém budeme referovat v paragrafu 3, při výpočtu adres oblastí v paměti počítače přidělených krychlím.)

Několik ilustrativních příkladů krychlí (pro $P = 3$) je na obr. 3.

Symbol $\mathcal{B}_P$ bude značit množinu všech 3^P krychlí $B \subseteq \{0, 1\}^P$. Krychle B je *podkrychlí* krychle B', když $B \subseteq B'$. V termínech ternárních reprezentací $a_1 \ldots a_P$ a $a'_1 \ldots a'_P$ krychlí B a B' to znamená, že pro $i = 1, \ldots, P$ je $a_i = a'_i$, kdykoli $a'_i \neq 2$.

Důvod, proč se krychlemi zabýváme, spočívá v tom, že kterýkoli uzel libovolného klasifikačního stromu můžeme ztotožnit s krychlí, a naopak kterákoli krychle připadá alespoň teoreticky v úvahu jako možný uzel klasifikačního stromu. Pokud cesta od kořene k nějakému uzlu v diagramu popisujícím klasifikační strom vede přes nějaká větvení, mají některé proměnné v uzlu pevně dané hodnoty 0 nebo 1 (podle toho, kde si příslušná cesta "na rozcestích vybere" nuly nebo jedničky), ostatní proměnné jsou pak v uzlu "volné" a mohou zde nabývat (ačkoli v datech ne vždy nabývají) obou hodnot z $\{0, 1\}$. Krychle jsou jakési "stavební kameny", které jsou k dispozici pro konstrukci stromu jako potenciální uzly.

Cílem našich metod samozřejmě je sestrojit klasifikátor stromového typu, který přiřazuje kód třídy každé kombinaci hodnot prediktorů, která se může vyskytnout. Jako pomocný technický prostředek konstrukce však potřebujeme "parciální klasifikátory", které kód třídy přiřazují jen těm hodnotám vektoru prediktorů, které patří do určité krychle. O těchto "parciálních klasifikátorech" budeme mluvit jako o *stromech pro danou krychli*.



Obr. 3. Příklady krychlí (při $P = 3$). Zleva doprava jsou zvýrazněny krychle s ternárními kódy 122, 120 a 100.

Na binární klasifikační strom pro krychli B se můžeme dívat jako na popis jistého rozkladu B na podkrychle spolu s funkcí definovanou na B, která je na každé z těchto podkrychlí konstantní a nabývá hodnoty z $\{C_1, \ldots, C_K\}$. Je-li T strom, značíme příslušnou funkci f_T a její definiční obor D_T. Formálněji a přesněji řečeno, binární klasifikační strom T je:

(i) buďto *triviální strom*, když D_T je krychle a f_T je konstantní funkce s hodnotou z $\{C_1, \ldots, C_K\}$ definovaná na D_T,

(ii) nebo *složený strom*, který lze definovat jako dvojici stromů $T = (T_L, T_R)$, kde D_{T_L} a D_{T_R} jsou disjunktní a $D_{T_L} \cup D_{T_R}$ je krychle. Potom platí $D_T = D_{T_L} \cup D_{T_R}$ a funkce f_T se rovná f_{T_L} na D_{T_L} a f_{T_R} na D_{T_R}.

Stromový diagram odpovídající triviálnímu stromu sestává z jediného uzlu, který je současně kořenem i listem (koncovým uzlem). Diagram složeného stromu $T = (T_L, T_R)$ obsahuje krychli D_T jako kořenový uzel, a kromě kořene další uzly, totiž uzly stromů T_L a T_R.

Ilustrativní příklad stromu pro krychli, která není totožná s prostorem prediktorů, je uveden na obr. 4.

Asi každý tradiční výklad o klasifikačních stromech uvádí, že strom je v uzlu, který není uzlem koncovým, rozvětven podle hodnot některého prediktoru. To samozřejmě platí i v případě stromů pro krychle, ale dosud zde nebylo explicitně řečeno, jak je prediktor, podle nějž je strom v uzlu rozvětven, dán. Formálně lze tento prediktor určit následujícím způsobem: Je-li krychle B uzlem stromu T_0, a není přitom uzlem koncovým, je B kořenovým uzlem složeného stromu $T = (T_L, T_R)$, který dostaneme tak, že funkci f_{T_0} zúžíme na $D_T = B$. Podmínka, že D_{T_L}, D_{T_R} i $D_T = D_{T_L} \cup D_{T_R}$ jsou krychle, přičemž D_{T_L} a D_{T_R} jsou disjunktní, může být splněna jedině tak, že ternární kódy $a_1^L \ldots a_P^L$, resp. $a_1^R \ldots a_P^R$, resp. $a_1 \ldots a_P$ krychlí D_{T_L}, D_{T_R} i D_T jsou shodné až na jednu – řekněme i-tou – pozici, kde $a_i = 2$ a $\{a_1^L, a_1^P\} = \{0, 1\}$. Strom T_0 je pak v B rozvětven podle prediktoru X_i.

Říkáme, že strom T zařazuje (klasifikuje) vektor prediktorů $\mathbf{x} = (x_1, \ldots, x_P) \in B$ do třídy C, když $f_T(\mathbf{x}) = C$. Obdobně říkáme také o jednotlivém případu, že jej strom T zařazuje (klasifikuje) do třídy C, když T klasifikuje do C vektor prediktorů $\mathbf{x} = (x_1, \ldots, x_P) \in B$ tohoto případu.

Velikost triviálního stromu T je $|T| = 1$. Pro složený strom $T = (T_L, T_R)$ definujeme velikost $|T|$ rekurzivně vztahem $|T| = |T_L| + |T_R|$. (Všimněme si, že $|T|$ se rovná počtu listů stromového diagramu odpovídajícího stromu T.)

Množinu všech stromů T pro krychli B, tj. stromů, pro něž platí $D_T = B$, budeme značit $\mathcal{T}(B)$. Místo složitějšího $\mathcal{T}(\{0,1\}^P)$ budeme jako rovnocenný symbol používat $\mathcal{T}$. Symbolem $\mathcal{T}_m(B)$ budeme označovat množinu všech stromů pro krychli B, jejichž velikost je rovna m (argument B budeme vynechávat, pokud B je celý prostor prediktorů).

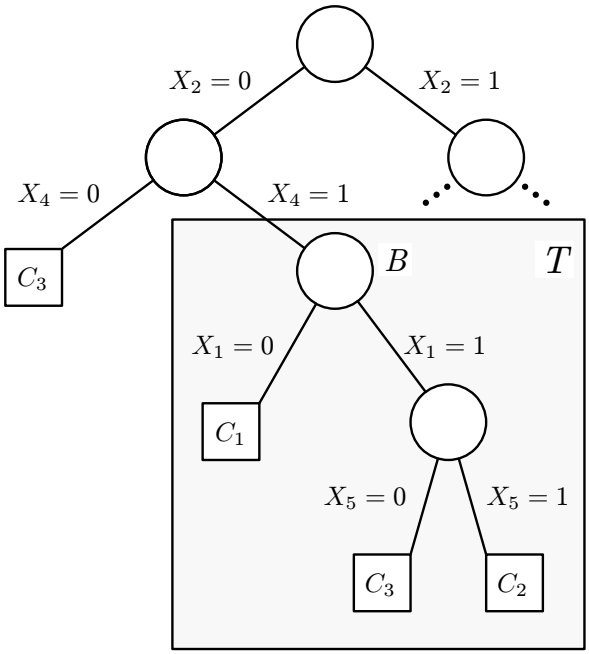
Je-li $\mathcal{L}$ datový soubor (datová množina) a B krychle, bude $\mathcal{L}_B$ značit množinu těch případů z $\mathcal{L}$, jejichž vektor prediktorů patří do B.

Nechť je dán datový soubor $\mathcal{L}$ velikosti $N > 0$, krychle $B \in \mathcal{B}_P$ a klasifikační strom $T \in \mathcal{T}(B)$. Nechť $N_i > 0$ je pro $i = 1, \ldots, K$ počet těch případů z $\mathcal{L}$, které patří do třídy C_i. Nechť $N_{ij}(B)$ pro $i, j = 1, \ldots, K$ je počet těch případů z $\mathcal{L}_B$, které patří do třídy C_i a současně jsou stromem T klasifikovány do C_j. Pro $i, j = 1, \ldots, K$ značíme symbolem Z_{ij} ztrátu při klasifikaci (chybné, když $i \neq j$) jednoho případu ze třídy C_i do C_j. Nechť $\pi_1, \ldots, \pi_K$ jsou apriorní pravděpodobnosti tříd[1].

Chyba (nepřesnost) $R(T|\mathcal{L}_B)$ klasifikátoru T na $\mathcal{L}_B$ je definována jako

$$R(T|\mathcal{L}_B) = \sum_{i=1}^{K} \frac{\pi_i}{N_i} \sum_{j=1}^{K} Z_{ij} N_{ij}(B). \tag{1}$$

Všimněme si, že pro $T = (T_L, T_R) \in \mathcal{T}(B)$, $T_L \in \mathcal{T}(B_L)$ a $T_R \in \mathcal{T}(B_R)$ platí $R(T|\mathcal{L}_B) = R(T_L|\mathcal{L}_{B_L}) + R(T_R|\mathcal{L}_{B_R})$.



$$\begin{aligned} B &= \Big\{ \big(X_1, \ldots, X_5\big) \in \{0,1\}^5; X_2 = 0, X_4 = 1 \Big\} \\ &= \quad \{0,1\} \times \{0\} \times \{0,1\} \times \{1\} \times \{0,1\} \equiv 20212_3 \end{aligned}$$

Obr. 4. Naznačený diagram klasifikačního stromu, $P = 5$, $K = 3$. V rámečku je znázorněn strom T pro krychli $B = D_T$ s ternárním kódem 20212. Tento kód říká, že cesta od celého prostoru prediktorů, kódovaného 22222, ke krychli B ve stromovém diagramu vede přes volby $X_2 = 0$ a $X_4 = 1$. (Kód však neinformuje o tom, v jakém pořadí tyto volby nastávají.) Fukce f_T nabývá na krychlích s kódy 00212, 10210 a 10211, které jsou koncovými uzly stromu T a tvoří rozklad krychle B, hodnoty C_1, C_3 a C_2. Strom T je složený a lze jej zapsat jako $T = (T_L, T_R)$. Strom T_L je triviální a jeho kořenem a jediným listem je krychle s kódem 00212. Strom T_R je složený, jeho kořenem je krychle s kódem 10212 a má dva listy s kódy 10210 a 10211.

[1]Viz např. Breiman *et al.*, 1984, kde jsou uvedeny různé způsoby zadání a použití apriorních pravděpodobností implementované v programu *CART*.

Často (a jmenovitě v příkladech, které budou uvedeny v paragrafu 6) pokládáme $\pi_i = N_i/N$ pro $i = 1, \ldots, K$ (tj. apriorní pravděpodobnosti ztotožňujeme s relativními frekvencemi tříd v datech), $Z_{ii} = 0$ pro $i = 1, \ldots, K$ a $Z_{ij} = 1$ pro $i \neq j$, kde $i, j = 1, \ldots, K$. Chyba (1) se pak redukuje na

$$R(T|\mathcal{L}_B) = \frac{1}{N} \sum_{i=1}^{K} \sum_{j=1}^{K} N_{ij}(B) - \frac{1}{N} \sum_{i=1}^{K} N_{ii}(B).$$

Pro $B = \{0,1\}^P$ se tedy jedná o "obyčejnou" relativní četnost chybně klasifikovaných pozorování v souboru[2].

3. Optimalizační algoritmy

Při tradičním "pěstování" stromů tzv. rekurzivním dělením (recursive partitioning) se nejdříve hledá nejlepší větvení pro kořenový uzel (tj. rozklad prostoru prediktorů na dvě podkrychle), potom se hledají nejlepší větvení (tedy rozklady) pro podkrychle získané v prvním kroku, atd. Ve chvíli, kdy se vybírá nejlepší rozklad nějaké krychle, nejsou nejlepší rozklady jejích podkrychlí zpravidla ještě známy. (Vzácný příklad pokusu o "pohled o více kroků dopředu" se najde např. v práci Friedman, 1979.)

Pro optimalizační přístup, navržený poprvé v článku Savický *et al.* (2000), je charakteristické, že se hledá (v jistém smyslu) nejlepší strom mezi *všemi možnými stromy* dané velikosti. Naše algoritmy konstruují stromy (na rozdíl od klasických metod) "zdola nahoru". V průběhu konstrukce optimálního stromu (pro celý prostor prediktorů) se postupně přiřazují optimální stromy všem krychlím z $\mathcal{B}_P$. Pořadí, v jakém se krychle probírají, je takové, že ve chvíli, kdy se vybírá nejlepší strom (a tím také větvení) pro nějakou krychli, jsou už známy optimální stromy přiřazené všem podkrychlím dané krychle. Výběr větvení pro kořenový uzel (tj. pro krychli $B = \{0,1\}^P$) není při konstrukci stromu úvodním, ale závěrečným krokem.

Za to, že naše algoritmy skutečně vedou k cíli, vděčíme tvrzením, která nám dovolují při hledání optimálního složeného stromu $T = (T_L, T_R)$ v $\mathcal{T}(B)$ nebo $\mathcal{T}_m(B)$ optimalizovat levou a pravou část stromu T_L a T_R odděleně. (Kdybychom se o žádnou takovou větu nemohli opřít, museli bychom množinu $\mathcal{T}(B)$ nebo $\mathcal{T}_m(B)$ prohledávat exhaustivně, což by z hlediska výpočetní složitosti bylo, jak je dnes v módě říkat, "o něčem jiném".) Tato tvrzení budou později uvedena jako věty 2 a 3.

3.1. Minimalizace kombinované ztráty (Algoritmus I).

Navrhli a implementovali jsme algoritmus, který pro datový soubor $\mathcal{L}$ a pro danou hodnotu *parametru složitosti* $\alpha \geq 0$ sestrojí klasifikační strom minimalizující *kombinovanou ztrátu* (viz Breiman *et al.*, 1984, kde se používá termín *cost-complexity measure*) definovanou jako

$$R_\alpha(T|\mathcal{L}) = R(T|\mathcal{L}) + \alpha|T|.$$

Připomeňme, že při prořezávání založeném na kombinované ztrátě (minimal cost-complexity pruning) v metodě *CART* (viz Breiman *et al.*, 1984) se minimalizuje tentýž výraz. Je zde ovšem podstatný rozdíl: Hledání nejlepšího stromu se v metodě *CART* omezuje na *množinu stromů získaných prořezáváním jednoho konkrétního stromu*. Náš algoritmus naproti tomu minimalizuje kombinovanou ztrátu *v množině všech možných stromů*.

[2] V tomto případě mluvíme právem o chybě, zatímco obecně by bylo vlastně správnější používat spíše termín *ztráta*.

Předpokládejme, že platí $R_\alpha(T^*|\mathcal{L}) = \min_{T\in\mathcal{T}} R_\alpha(T|\mathcal{L})$, kde $\alpha > 0$ a $|T^*| = m$. Potom současně platí také $R(T^*|\mathcal{L}) = \min_{T\in\mathcal{T}_m} R(T|\mathcal{L})$. Můžeme proto "zkoušením" různých hodnot parametru α počítat $\min_{T\in\mathcal{T}_m} R(T|\mathcal{L})$ pro některé hodnoty m a současně konstruovat stromy, které tato minima realizují.

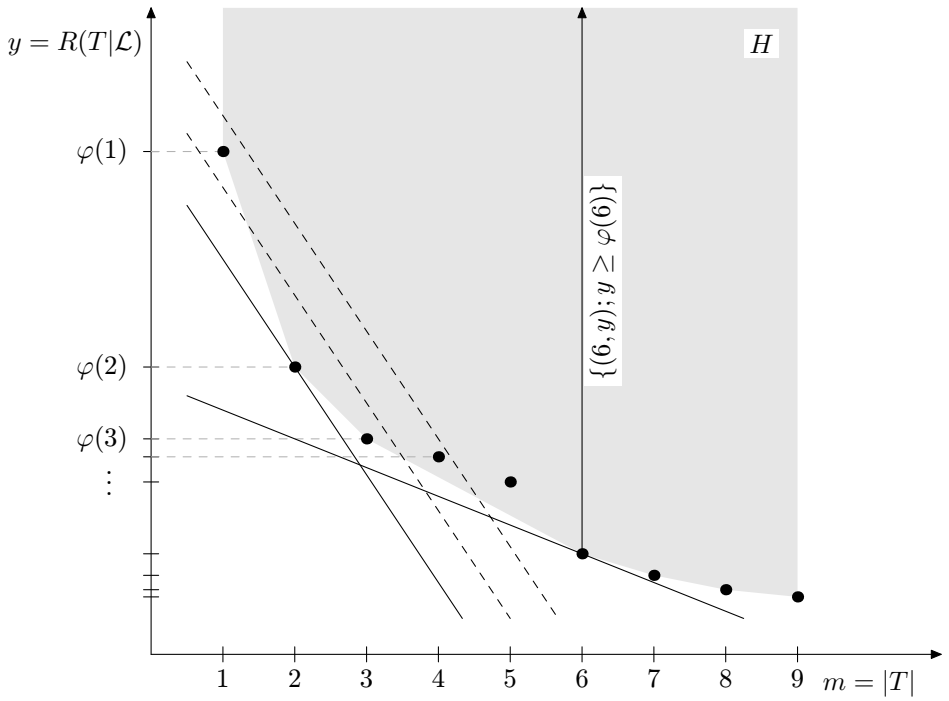
Stromy, které minimalizují $R_\alpha(T|\mathcal{L})$ pro nějaké α, resp. velikosti takových stromů, musejí splňovat následující geometrickou podmínku.

Věta 1. Nechť pro všechna m, pro která je $\mathcal{T}_m$ neprázdná množina, je symbol $\varphi(m)$ definován jako

$$\varphi(m) = \min_{T\in\mathcal{T}_m} R(T|\mathcal{L}).$$

Nechť H je konvexní obal množiny $\{(m, y); \varphi(m) \text{ je definováno}, y \geq \varphi(m)\}$. Nechť T^* minimalizuje $R(T|\mathcal{L})$ v množině $\mathcal{T}_m$. Potom T^* minimalizuje $R_\alpha(T|\mathcal{L})$ v $\mathcal{T}$ pro nějaké $\alpha \geq 0$ právě tehdy, když bod $\big(m, \varphi(m)\big)$ leží na hranici množiny H.

Důkaz věty naznačíme obrázkem 5. Do bodů $\big(m, \varphi(m)\big)$ v grafu se zobrazují stromy s minimální hodnotou $R(T|\mathcal{L})$ při dané velikosti. Všem stromům pevné velikosti $m > 0$ (pokud existuje alespoň jeden) pak odpovídají vesměs body na polopřímce $\{(m, y); y \geq \varphi(m)\}$. Dané $\alpha \geq 0$ určuje směrnici přímek tvořících rovnoběžný svazek, přičemž $R_\alpha(T|\mathcal{L})$ nabývá téže hodnoty pro všechny stromy T, které se zobrazují do bodů téže přímky svazku. Minimalizace $R_\alpha(T|\mathcal{L})$ při daném $\alpha \geq 0$ je vlastně hledáním přímky odpovídajícího svazku, na kterou se zobrazuje nějaký strom a která leží co nejblíže počátku souřadnic. Řešením je proto bod na hranici množiny H.



Obr. 5. Ilustrace věty 1. Minimalizací $R_\alpha(T|\mathcal{L})$ pro dané $\alpha \geq 0$ na datovém souboru $\mathcal{L}$ dostaneme strom, který se zobrazuje do bodu na hranici konvexní množiny H. Vzhledem k tomu, že v situaci na obrázku body $\big(m, \varphi(m)\big)$ pro $m = 4$ a $m = 5$ leží ve vnitřku množiny H, nemůže $R_\alpha(T|\mathcal{L})$ pro žádné $\alpha \geq 0$ minimalizovat strom velikosti 4 nebo 5.

Algoritmus minimalizující kombinovanou ztrátu $R_\alpha(T|\mathcal{L})$ můžeme popsat následujícím způsobem.

Algoritmus I

Vstup: Specifikace proměnných, data $\mathcal{L}$, α.
Výstup: Strom T^* minimalizující $R_\alpha(T|\mathcal{L})$.
Popis:

- Pro každou krychli B z $\mathcal{B}_P$ obsahující alespoň jedno pozorování se provedou následující kroky, přičemž krychle se probírají v takovém pořadí, že libovolná krychle přijde na řadu až po všech svých podkrychlích:
  (1) Vybere se třída $C \in \{C_1, \ldots, C_K\}$ tak, aby veličina $R_\alpha(T|\mathcal{L}_B)$ pro triviální strom T (s $D_T = B$ a $f_T = C$) byla minimální.
  (2) Pro každou takovou dvojici B_L,B_R disjunktních podkrychlí krychle B, že jak B_L, tak B_R obsahuje alespoň jedno pozorování z $\mathcal{L}$ a $B = B_L \cup B_R$ (pokud taková dvojice existuje), se posoudí strom $T = (T_L, T_R)$, kde T_L a T_R jsou stromy přiřazené B_L a B_R.
  (3) Ze stromů posuzovaných ve dvou předcházejících krocích se vybere strom T, který minimalizuje $R_\alpha(T|\mathcal{L}_B)$. (Pokud $R_\alpha(T|\mathcal{L}_B)$ minimalizuje více stromů, vybere se kterýkoli z nich.) Vybraný strom minimalizuje R_α v množině $\mathcal{T}(B)$. Informace, který strom byl vybrán, se uloží.
- Po dokončení cyklu je zkonstruován strom T^* přiřazený $B = \{0,1\}^P$. Ten je výstupem procedury.

Algoritmus I je založen na oddělené optimalizaci levé a pravé části složeného stromu, a tedy na následující větě.

Věta 2. Nechť B je krychle a $\mathcal{L}$ je datový soubor. Nechť strom $T^* = (T_L, T_R)$ minimalizuje pro dané $\alpha \geq 0$ kombinovanou ztrátu $R_\alpha(T|\mathcal{L}_B)$ v $\mathcal{T}(B)$. Nechť B_L a B_R jsou kořenové uzly stromů T_L a T_R (tj. $B_L = D_{T_L}$ a $B_R = D_{T_R}$). Potom T_L a T_R minimalizují $R_\alpha(T|\mathcal{L}_{B_L})$ v $\mathcal{T}(B_L)$ a $R_\alpha(T|\mathcal{L}_{B_R})$ v $\mathcal{T}(B_R)$.

Důkaz. Tvrzení je jednoduchým důsledkem vztahu $R_\alpha(T^*|\mathcal{L}_B) = R_\alpha(T_L|\mathcal{L}_{B_L}) + R_\alpha(T_R|\mathcal{L}_{B_R})$. Kdyby např. strom T_L neminimalizoval $R_\alpha(T|\mathcal{L}_{B_L})$ v $\mathcal{T}(B_L)$, musel by existovat takový strom $T'_L \in \mathcal{T}(B_L)$, že $R_\alpha(T'_L|\mathcal{L}_{B_L}) < R_\alpha(T_L|\mathcal{L}_{B_L})$. Potom by ale pro strom $T' = (T'_L, T_R)$ platilo $R_\alpha(T'|\mathcal{L}_B) < R_\alpha(T^*|\mathcal{L}_B)$, což odporuje předpokladu, že T^* minimalizuje $R_\alpha(T|\mathcal{L}_B)$ v $\mathcal{T}(B)$.

Poznámka: Povšimněme si, že dosadíme-li za α nulu, sestrojí algoritmus I strom s nejnižší hodnotou R v $\mathcal{T}$. Pokud vezmeme velmi malé kladné α, dostaneme *nejmenší* strom mezi těmi, které v $\mathcal{T}$ minimalizují R. Speciálně, pokud existuje alespoň jeden "přesný" strom, tj. strom klasifikující správně všechna pozorování z daného datového souboru, lze uvedenou metodou nalézt *nejmenší* takový strom.

3.2. **Minimalizace chyby při dané velikosti stromu (Algoritmus II).** Druhý z našich nových algoritmů konstruuje pro $M \geq 1$ posloupnost stromů minimalizujících ztrátu $R(T|\mathcal{L})$ na souboru $\mathcal{L}$ v množinách $\mathcal{T}_m$ pro $m = 1, 2, \ldots, M$. Tento algoritmus vytváří na rozdíl od předcházejícího algoritmu stromy *všech* velikostí od 1 do M, tedy i takových velikostí, pro které není splněna geometrická podmínka z věty 2. Daní za tuto výhodu oproti algoritmu I je vyšší prostorová a časová složitost.

Popis algoritmu II uvedeme ve zhuštěné podobě, protože algoritmy I a II jsou velmi podobné.

Algoritmus II

Vstup: Specifikace proměnných, data $\mathcal{L}$, maximální velikost stromu M.
Výstup: Stromy $T_1^*, \ldots, T_M^*$ minimalizující $R(T|\mathcal{L})$ v $\mathcal{T}_1, \ldots, \mathcal{T}_M$.
Popis:

- Krychle se probírají ve stejném pořadí jako v případě algoritmu I. Popis algoritmu se liší jen v bodech 2 a 3, kde jsou tyto změny:
  - $2'$. Pro každou takovou dvojici B_L,B_R disjunktních podkrychlí krychle B, že jak B_L, tak B_R obsahuje alespoň jedno pozorování z $\mathcal{L}$ a $B = B_L \cup B_R$ (pokud taková dvojice existuje), a pro každou dvojici kladných celých čísel m_L, m_R, pro něž $m_L + m_R \leq M$, se posuzují stromy $T = (T_L, T_R)$, kde T_L a T_R jsou stromy velikostí m_L a m_R přiřazené krychlím B_L a B_R. Výsledný strom je kandidátem při výběru stromu velikosti $m_L + m_R$ pro danou krychli.
  - $3'$. Strom z kroku 1 se použije pro velikost 1. Pro každé $m = 2, 3, \ldots, M$ se mezi kandidáty velikosti m z kroku $2'$ vybere strom minimalizující $R(T|\mathcal{L}_B)$ (přičemž případné shody se řeší libovolně). Informace o vybraných stromech všech uvažovaných velikostí se uloží.
- Posloupnost M stromů přiřazených krychli $\{0,1\}^P$ je výstupem algoritmu.

Algoritmus II je podobně jako algoritmus I založen na oddělené optimalizaci levé a pravé části složeného stromu. Věta, která takový postup umožňuje, má tentokrát následující podobu.

Věta 3. Nechť B je krychle, $\mathcal{L}$ je datový soubor a m je celé kladné číslo. Nechť strom $T^* = (T_L, T_R)$ minimalizuje chybu klasifikace $R(T|\mathcal{L}_B)$ v $\mathcal{T}_m(B)$, kde $|T_L| = m_L$ a $|T_R| = m_R$. Nechť $B_L = D_{T_L}$ a $B_R = D_{T_R}$. Potom T_L a T_R minimalizují $R(T|\mathcal{L}_{B_L})$ v $\mathcal{T}_{m_L}(B_L)$ a $R(T|\mathcal{L}_{B_R})$ v $\mathcal{T}_{m_R}(B_R)$.

Důkaz je založen na vztahu $R(T^*|\mathcal{L}_B) = R(T_L|\mathcal{L}_{B_L}) + R(T_R|\mathcal{L}_{B_R})$, jinak je analogický důkazu věty 2.

3.3. **Software a algoritmická složitost.** Algoritmy I and II byly implementovány (v témže programu) v jazyce C++. Program ve zdrojovém tvaru se nachází na URL `http://www.cs.cas.cz/~savicky/trees`. (Tamtéž jsou umístěny také programy generující data pro experimenty.) Současná verze programu je určena pouze pro výzkumné účely – není nijak "přátelská k uživateli", nemá žádné grafické rozhraní a ovládá se parametry z příkazové řádky a řídicích textových souborů.

Velikost paměti, kterou potřebuje algoritmus I, je přibližně $c\, 3^P K$, přičemž konstanta c závisí na implementaci a není příliš velká. (Připomeňme, že P je počet prediktorů a K je počet tříd.) Algoritmus II spotřebuje v porovnání s algoritmem I přibližně M-krát více paměti. Doba výpočtu je u obou algoritmů zhruba přímo úměrná potřebné velikosti paměti (takže když se úloha vejde do operační paměti, lze se také dočkat výsledku).

Jako konkrétní příklad můžeme uvést úlohu rozpoznání parity (viz paragraf 6), kde $P = 10$ a $K = 2$. Na Pentiu III s operačním systémem Linux spotřeboval algoritmus I pro 50 různých hodnot parametru α při 1000 trénovacích a 10000 testovacích případech 9 sekund a 1.6MB RAM. Výpočet algoritmem II na stejných datech pro $M = 50$ velikostí stromů trval 12 sekund a vyžadoval 58.2MB RAM.

4. K čemu je optimalita dobrá?

Otázka v nadpisu nevypadá na první pohled logicky, tedy alespoň pokud se zabýváme takovými problémy analýzy dat, kde se aplikace klasifikačních stromů jeví jako adekvátní a naše algoritmy nenarážejí na hardwarové meze použitelnosti. Na druhý pohled však věci tak jasné nejsou. Naše algoritmy garantují, že vytvořené stromy budou mít nejmenší možnou chybu při dané hodnotě parametru (parametru složitosti α nebo velikosti stromu) na daném trénovacím datovém souboru. To, co nás však zpravidla nejvíce zajímá, je chování stromu na datech, která nebyla při vytváření klasifikačního modelu použita, tedy (jazykem převzatým z oblasti strojového učení, kterému zde místy dáváme přednost před "dialektem" statistickým) generalizace. Co když optimální stromy dosahují nízké chyby proto, že do modelu jsou zahrnuty i nahodilé vlastnosti dat, které se v nových datech s velkou pravděpodobností neobjeví? Pak by se optimalita (v tom smyslu, v jakém jí dosahujeme) mohla stát nevýhodou, a z hlediska generalizace by mohly být optimální stromy horší než stromy s vyšší chybou na trénovacích datech (sestrojené podle jiného principu, např. některou z metod jako *CART* nebo *C4.5*).

Poznamenejme, že k tomu, abychom mohli přesněji říci, co je myšleno "chybou na jiných datech", musíme vyslovit určité pravděpodobnostní předpoklady – např. že data, jak trénovací, tak "jiná", jsou náhodným výběrem z nějakého sdruženého rozdělení vektoru prediktorů a kódu třídy. Pak můžeme definovat chybu klasifikátoru T nejen relativně, tj. *na daném datovém souboru*, ale "absolutně", jako tzv. *skutečnou* chybu

$$R(T) = \sum_{i=1}^{K} \pi_i \sum_{j=1}^{K} Z_{ij} p_{ij},$$

kde π_i pro $i = 1, \ldots, K$ a Z_{ij} pro $i, j = 1, \ldots, K$ mají týž význam jako v (1) a p_{ij} je pro $i, j = 1, \ldots, K$ (podmíněná) pravděpodobnost, že strom T klasifikuje pozorování ze třídy C_i do C_j. Pokud $Z_{ii} = 0$ pro $i = 1, \ldots, K$ a $Z_{ij} = 1$ pro $i \neq j$, kde $i, j = 1, \ldots, K$, platí

$$R(T) = 1 - \sum_{i=1}^{K} \pi_i p_{ii},$$

což není nic jiného, než pravděpodobnost chybné klasifikace náhodně vybraného pozorování.

O vztahu mezi chybou klasifikátoru na trénovacích datech a skutečnou chybou (tedy generalizací) lze filosoficky spekulovat, lze jej vyšetřovat matematicky, a můžeme jej také studovat experimentálně.

Souvislost s filosofií zde skutečně existuje: Jednou z pohnutek, proč jsme se pustili do vytváření optimálních stromů, je alespoň určitý stupeň apriorní důvěry (ne stejný u všech autorů) v tzv. *princip Occamovy břitvy*. Tím rozumíme heuristický předpoklad, že ze dvou teorií, které vysvětlují nějaký jev, je třeba dát přednost té jednodušší. V interpretaci běžné ve strojovém učení to pak znamená, že mezi modely, které se stejnou chybou popisují trénovací data, bude ten nejjednodušší obvykle mít nejlepší generalizační vlastnosti. Konkrétně v případě klasifikačních stromů lze citovat Quinlana (1986): *"Given a choice between two decision trees, each of which is correct over the training set, it seems sensible to prefer the simpler one on the grounds that it is more likely to capture structure inherent in the problem. The simpler tree would therefore be expected to classify correctly more objects outside the training set."*

Úlohu optimalizace stromů sice neformulujeme přímo tak, že bychom minimalizovali velikost stromu při dané chybě, ale de facto řešíme podobnou úlohu: Některé ze stromů, které mají při daném počtu listů minimální chybu, jsou současně také stromy s minimální velikostí při dané chybě. (Pro některé optimální stromy, které jsou výstupem našich algoritmů, mohou menší stromy s totožnou chybou existovat.) Podle principu Occamovy břitvy tedy lze očekávat, že optimalizace chování na trénovacích datech povede ke stromům s dobrými generalizačními vlastnostmi.

Vyšetřovat generalizační vlastnosti optimálních stromů matematicky zatím příliš neumíme. Určitou možnost nabízejí odhady skutečné chyby klasifikace v modelu PAC-learning (probably almost correct learning), viz např. Blumer *et al.* (1987).

Experimentálnímu vyšetřování generalizačních vlastností optimálních stromů na několika příkladech klasifikačních úloh je věnován paragraf 6.

5. Správně velké stromy

Každý z algoritmů popsaných v paragrafu 3 dává jako výsledek různě velké stromy v závislosti na parametru – v případě algoritmu II je parametrem samotná velikost, u algoritmu I parametr složitosti α určuje velikost stromu nepřímo. Cílem konstrukce stromu v praktických situacích zpravidla bude sestrojit jeden klasifikační model, nikoli celou sérii. Obvykle však chybí nějaké jasné vodítko, podle kterého bychom předem dovedli určit "správnou" hodnotu parametru, tj. "správnou" velikost stromu. Rozhodnout by zřejmě měly generalizační vlastnosti. (Příliš malé stromy nepostihnou všechny zákonitosti "za daty", příliš velké stromy budou vedle těchto zákonitostí odrážet i nahodilé vlastnosti dat.)

To samozřejmě není náš objev – s problémem stanovení vhodné velikosti stromu se nějak vyrovnávají prakticky všechny dosavadní metody založené na stromech. Některé přidávají ke stromu nová větvení, dokud to přináší (v nějak definovaném smyslu) dostatečný pokles chyby, potom je konstrukce ukončena. Jiné metody postupují odlišně: V první fázi se také postupně přidávají nová větvení, ale kritérium, podle kterého se tento proces zastaví, je velmi mírné, takže se vytvoří velmi velký strom. Ve druhé fázi se tento strom prořezává – studují se stromy, které vzniknou odstraněním většího či menšího počtu větvení z původního velkého stromu (prořezané podstromy). U každého z těchto prořezaných podstromů se odhaduje kvalita generalizace, a na základě těchto odhadů se pak vybere nejvhodnější velikost stromu.

Všimněme si, jakým způsobem se generalizační vlastnosti prořezaných podstromů odhadují v rámci metody *CART*. Jeden způsob je založen na rozdělení datového souboru na dvě části, z nichž jedna slouží k vytvoření velkého stromu a jeho prořezaných podstromů, zatímco na druhou část jsou klasifikační modely reprezentované prořezanými podstromy aplikovány a chyba na těchto datech slouží jako odhad skutečné chyby[3]. Druhý způsob, tzv. křížová validace (cross-validation), je o něco složitější. K vytvoření velkého stromu T^* a jeho prořezaných podstromů jsou použita všechna data. Pro účely odhadu skutečné chyby prořezaných podstromů stromu T^* je datový soubor rozdělen na $V \geq 2$ přibližně stejně velkých disjunktních částí $\mathcal{L}_1, \ldots, \mathcal{L}_V$ (běžně se pracuje s $V = 10$). Postupně pro $v = 1, \ldots, V$ se ze souboru vyjme $\mathcal{L}_v$, na základě ostatních dat je sestrojen pomocný velký strom T_v^* a posloupnost jeho

[3]Máme určitou terminologickou výhradu k tomu, že první části dat se v literatuře o metodě *CART* říká "learning sample" a druhé "test sample": Druhá část dat je použita, byť jiným způsobem než prvá, k výběru modelu, který bude výstupem metody, a neslouží tedy výhradně k otestování vlastností modelu vytvořeného bez přihlédnutí k těmto datům.

prořezaných podstromů, a $\mathcal{L}_v$ se použije k odhadu skutečné chyby prořezaných podstromů stromu T_v^*. Z těchto odhadů vypočtených pro $v = 1, \ldots, V$ se posléze způsobem, který zde nebudeme blíže popisovat (viz Breiman *et al.*, 1984), zkonstruuje odhad chyby prořezaných podstromů stromu T^*.

Ať už se odhady chyb prořezaných podstromů sestrojí tím, či oním způsobem, zbývá ještě vybrat na jejich základě jeden "nejnadějnější" prořezaný podstrom. Nejjednodušší je uznat za nejlepší ten strom $T_\bullet$, který dává nejnižší odhad $\widehat{R}(\cdot)$ skutečné chyby. Protože však zpravidla existuje více podstromů se "skoro stejnými" hodnotami odhadu skutečné chyby, považuje se za lepší vybrat spíše nejmenší z těch podstromů T, pro něž se $\widehat{R}(T)$ liší od $\widehat{R}(T_\bullet)$ nejvýše o nějaký zvolený s-násobek směrodatné odchylky odhadu $\widehat{R}(T_\bullet)$. (Výpočet této směrodatné odchylky viz Breiman *et al.*, 1984.) Nejběžnější je volba $s = 0$ (v tom případě je výstupem metody strom $T_\bullet$) – pak se mluví o prořezávání podle pravidla 0 SE, nebo $s = 1$, potom jde o pravidlo 1 SE.

Tento exkurs týkající se metody *CART* jsme uvedli proto, že v otázce výběru "správně velkého" stromu hodláme do značné míry postupovat analogicky. Současná verze programu, v němž jsou implementovány algoritmy I a II, však zatím poskytuje jedinou možnost: Stromy různé velikosti se konstruují pomocí dat, která někdy označujeme jako *g-data* ("g" jako "growth"), a konečným výsledkem je strom s minimálním chybou na jiných datech (validačních datech, *v-datech*). Máme v plánu rozšíření možností programu o další varianty, přednostně pak o křížovou validaci.

6. Numerické experimenty

Víme už, že u stromů vytvořených našimi algoritmy nelze a priori garantovat, že nebudou mít horší generalizační vlastnosti než "neoptimální" stromy produkované klasickými metodami. V několika numerických experimentech jsme proto zkoušeli, zda a v jakých úlohách získáme naší optimalizací výhodu oproti metodě *CART*.

6.1. Typy experimentálních dat.

(1) **Dekadické číslice (DČ).** Jde o známou úlohu, kterou poprvé studovali v souvislosti s klasifikačními stromy Breiman *et al.* (1984), a která se vyskytuje v různých sadách problémů, na nichž se běžně testují klasifikační metody. Každá z číslic $C \in \{0, 1, \ldots, 9\}$ je kódována pevným vektorem $(\xi_1^C, \ldots, \xi_7^C) \in \{0, 1\}^7$, který popisuje kombinaci rozsvícených a zhasnutých segmentů na jednoduchém displeji (viz obr. 6). Pro jednotlivý případ v datech se generuje kód třídy (neboli číslice) C jako relizace náhodné veličiny s rovnoměrným rozdělením na $\{0, 1, \ldots, 9\}$. V datech není dostupný samotný vektor $(\xi_1^C, \ldots, \xi_7^C)$, ale jen jeho modifikace $(x_1, \ldots, x_7) \in \{0, 1\}^7$, která se z $(\xi_1^C, \ldots, \xi_7^C)$ získá tak, že pro každé $i = 0, \ldots, 9$ se nezávisle náhodně vybere mezi možnostmi $x_i = \xi_i^C$ a $x_i = 1 - \xi_i^C$, jež mají pravděpodobnost 0.9, resp. 0.1. V klasifikační úloze se tedy určuje číslice z "poškozených" údajů o rozsvícených segmentech displeje – každá jednotlivá informace o segmentu je s pravděpodobností 10% chybná.

(2) **Vysoké a nízké číslice (VNČ).** Jedná se o variaci úlohy DČ. Data jsou generována stejně jako v případě DČ, ale číslice $0, \ldots, 9$ jsou transformovány na kód třídy 0 u "nízkých" číslic $0, \ldots, 4$ a na 1 u "vysokých" číslic $5, \ldots, 9$.

(3) **Parita a zavádějící proměnné (PZP).** Tato úloha, stejně jako úloha následující, nesouvisí s problémem rozpoznávání dekadických číslic. Pro jednotlivý případ v datovém souboru se nejdříve generují hodnoty $\xi_1, \ldots, \xi_5$

jako realizace vzájemně nezávislých náhodných veličin s rovnoměrným rozdělením na $\{0,1\}$. Kód třídy $C \in \{0,1\}$ je definován jako parita součtu těchto pěti hodnot, tj. $C = 1$, když $\sum_{i=1}^{5} \xi_i$ je liché číslo, jinak $C = 0$. V datech nejsou dostupné hodnoty $\xi_1, \ldots, \xi_5$, ale jejich modifikace $x_1, \ldots, x_5$ získané z $\xi_1, \ldots, \xi_5$ obdobným způsobem, jako se v úloze DČ upravují čísla $\xi_1, \ldots, \xi_7$ na $x_1, \ldots, x_5$ – s tím rozdílem, že alternativy $x_i = \xi_i^C$ a $x_i = 1 - \xi_i^C$ mají pravděpodobnost 0.978, resp. 0.022. (Kód třídy C se tudíž rovná paritě součtu $\sum_{i=1}^{5} x_i$ s pravděpodobností cca 0.9.) V datech jsou dále hodnoty $x_6, \ldots, x_{10}$ pěti tzv. "zavádějících" prediktorů $X_6, \ldots, X_{10}$. Každá z hodnot x_i, $i = 6, \ldots, 10$ je náhodně a nezávisle na ostatních (ale v závislosti na známém kódu třídy C) generována tak, že s pravděpodobností 0.6 je $x_i = C$ a s pravděpodobností 0.4 platí $x_i = 1 - C$. Každý z pěti "zavádějících" prediktorů $X_6, \ldots, X_{10}$ sám o sobě umožňuje správně predikovat kód třídy C v 60% případů a jejich kombinováním lze přesnost zvýšit na cca 68%. Prediktory $X_1, \ldots, X_5$ jsou *jednotlivě* pro stanovení kódu třídy stejně užitečné jako házení mincí, protože při dané hodnotě kterékoli z těchto veličin jsou jevy $C = 0$ a $C = 1$ stejně pravděpodobné. Kombinací prediktorů $X_1, \ldots, X_5$ však lze získat predikci správnou v přibližně 90% případů.

(4) **Interval a šumové proměnné (IŠP).** Kód třídy C je podobně jako u dat PZP funkcí součtu "skrytých" proměnných $\xi_1, \ldots, \xi_7$, které jsou generovány jako realizace vzájemně nezávislých náhodných veličin s rovnoměrným rozdělením na $\{0,1\}$. Jedná se však tentokrát o jinou funkci součtu, než je parita, konkrétně o charakteristickou funkci intervalu: Pokládáme $C = 1$, když $3 \leq \sum_{i=1}^{7} \xi_i \leq 4$, jinak $C = 0$. V datech nejsou hodnoty $\xi_1, \ldots, \xi_7$, ale jejich modifikace $x_1, \ldots, x_7$, které jsou z $\xi_1, \ldots, \xi_7$ odvozeny stejně jako v úloze PZP, včetně pravděpodobností 0.978, resp. 0.022 jevů $x_i = \xi_i^C$, resp. $x_i = 1 - \xi_i^C$ pro $i = 1, \ldots, 7$. Data dále obsahují hodnoty tří "čistě šumových" proměnných X_8, X_9 a X_{10}, které mají vesměs rovnoměrné rozdělení na $\{0,1\}$ a jsou nezávislé na kódu třídy, na ostatních prediktorech i navzájem.

6.2. **Struktura experimentů.** V každém z experimentů jsme použili uměle generovaná data představující náhodný výběr ze sdruženého rozdělení odpovídajícího jednomu z výše uvedených typů dat.

Pomocí datového souboru $\mathcal{L}_1$ velikosti 500 jsme vždy sestrojili

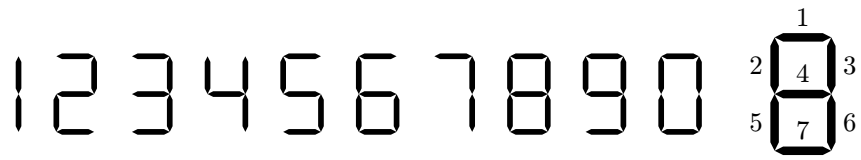
(i) algoritmem I stromy $T_1, \ldots, T_J$ pro J hodnot α tvořících geometrickou posloupnost ($J = 500$ v experimentech DČ, VNČ a IŠP, $J = 50$ u PZP),

(ii) algoritmem II stromy $T'_1, \ldots, T'_{J'}$ velikosti 1 až J' (v experimentech DČ, VNČ a PZP bylo $J' = 50$, u IŠP pak $J' = 200$).

V dalším kroku jsme ze stromů $T_1, \ldots, T_J$, resp. $T'_1, \ldots, T'_{J'}$ vybrali stromy T^*, resp. $T^{*\prime}$ minimalizující chybu na validačním souboru $\mathcal{L}_2$ velikosti 500. Skutečnou chybu stromů T^* a $T^{*\prime}$ jsme pak odhadli na testovacím souboru velikosti 10 000.

Tento postup jsme u dat typu DČ, VNČ a IŠP opakovali (pokaždé s jinými daty) 100krát. Výsledky experimentu PZP byly natolik jasné, že stačilo 10 opakování.

Pro porovnání jsme tatáž data analyzovali metodou *CART*, přičemž soubor $\mathcal{L}_1$ byl použit pro konstrukci (pěstování) stromu, soubor $\mathcal{L}_2$ pro prořezávání a testovací soubor velikosti 10 000 pro odhad chyby klasifikace výsledných (prořezaných) stromů. Jako kritérium pro volbu nejlepšího větvení jsme v experimentu DČ, kde bylo 10 tříd, použili tzv. twoing, v ostatních případech (kde byly třídy jen 2) pak Giniho míru. Při prořezávání jsme vyzkoušeli pravidla 0 SE a 1 SE.



Obr. 6. Dekadické číslice na displeji se sedmi segmenty (ilustrace k experimentům DČ a VNČ). Vpravo je znázorněno číslování segmentů – segmentu č. 1 odpovídá prediktor X_1, atd.

Poznamenejme, že při aplikaci optimalizačních algoritmů i metody *CART* jsme za odhady apriorních pravděpodobností tříd vesměs brali relativní frekvence tříd v datech a každý chybně klasifikovaný případ jsme penalizovali stejně – veličina R je proto v našich experimentech totožná s podílem chybně klasifikovaných případů.

6.3. **Výsledky experimentů.** V Tabulce 1 jsou pro každý typ dat a každou ze čtyř metod konstrukce stromů uvedeny vybrané statistické charakteristiky empirického rozdělení velikostí stromů a chyby stromů na testovacím souboru velikosti 10 000 při 100, popř. 10 opakováních.

Ve všech čtyřech experimentech dosahoval algoritmus II lepších výsledků než algoritmus I. Stejně tak v metodě *CART* bylo vesměs úspěšnější prořezávání podle pravidla 0 SE než podle pravidla 1 SE.

V experimentu DČ naše algoritmy nepřinesly žádné přesvědčivé zlepšení oproti metodě *CART* (twoing, pravidlo 0 SE). Algoritmus II dosáhl na testovacím souboru v 53 případech ze 100 nižší a ve 47 případech vyšší chyby než uvedená nejúspěšnější varianta metody *CART*. Vyjádřeno způsobem obdobným referátu o výsledku zápasu v košíkové, algoritmus II zvítězil nad metodou *CART* 53:47. Algoritmus I s metodou *CART* dokonce prohrál 47:52 při jedné shodě. (V obou případech je rozdíl hodnocený znaménkovým testem statisticky nevýznamný na hladině 5%.) Problém DČ je zřejmě pro *CART*, popř. jiné klasické metody konstrukce klasifikačnách stromů snadný, a nebude patřit k těm úlohám analýzy dat, kvůli kterým stojí za to algoritmy konstruující optimální stromy vyvíjet.

V experimentu VNČ byla výhoda, kterou jsme oproti metodě *CART* získali, prokazatelná – algoritmus II porazil *CART* (Giniho míra, pravidlo 0 SE) v poměru 62:37 při jedné shodě, algoritmus I pak 62:38 (v obou případech se jedná při hodnocení znaménkovým testem o statisticky významný výsledek na hladině 5%). Zároveň je však třeba konstatovat, že rozdíl mezi chybami dosahovanými pomocí optimálních a klasických stromů není nikterak velkolepý. Přičítáme to tomu, že pro *CART* je problém VNČ sice o něco obtížnější než DČ, ale také není příliš těžký.

V experimentu PZP je naše převaha naprosto jasná: Naše algoritmy byly schopny najít "dobře skrytou" závislost ve tvaru funkce parity. *CART* naproti tomu preferoval v kořeni a úrovních blízkých kořeni větvení podle "zavádějících" proměnných, rozdrobil tak soubor na malé podmnožiny, a v těch nebyl s to zákonitost založenou na paritě odhalit. Optimální stromy tak měly na testovacím souboru vesměs chybu mezi 10 a 13%, zatímco chyba stromů vytvořených metodou *CART* (Giniho míra, pravidlo 0 SE) nikdy neklesla pod 30%.

V úloze IŠP, která je přirozenější než PZP, byly naše algoritmy také výrazně úspěšnější než metoda *CART*. Algoritmus II porazil *CART* v poměru 84:16, algoritmus I pak zvítězil o něco skromněji 70:30 (v obou případech jde při hodnocení znaménkovým testem o statisticky významné výsledky). Metoda *CART* je sice schopna na některých datových souborech dosáhnout přesnosti srovnatelné s nejúspěšnějšími

optimálními stromy, ale chyby mají velkou variabilitu. U optimálních stromů je variabilita chyb menší, takže např. třetí kvartil rozdělení chyb u algoritmu II je nižší než první kvartil u metody *CART*.

Data		Optimalizace				CART			
		Alg. I		Alg. II		0 SE		1 SE	
		R %	listů	R %	listů	R %	listů	R %	listů
Dekad.	min	25.31	10	25.30	14	24.85	10	25.03	10
číslice	Q1	26.34	22	26.29	22	26.49	22.25	27.17	12.25
	med	26.93	28	26.78	27	26.88	30	27.98	17
(100	Q3	27.64	33.75	27.38	32	27.48	36	29.00	20.75
souborů)	max	29.90	39	28.82	49	29.88	50	30.18	42
Vysoké/	min	13.22	5	13.22	5	13.50	7	13.73	6
nízké	Q1	14.16	9.25	14.01	10	14.42	13	14.95	8
číslice	med	14.76	13	14.66	13	14.90	18	15.64	10
(100	Q3	15.24	18	15.17	18	15.39	27	16.80	15
souborů)	max	17.16	27	17.16	48	17.73	40	18.33	32
Parita a	min	10.06	30	10.06	32	32.67	6	33.76	4
zavádějící	Q1	10.06	32	10.06	32	33.72	8.50	34.47	6
proměnné	med	10.60	32	10.56	32	34.89	41.50	35.00	7.50
(10	Q3	12.05	34.50	11.63	34.75	35.44	180.75	36.08	17.25
souborů)	max	13.89	36	12.17	38	36.19	204	37.69	90
Interval	min	14.93	40	11.90	50	12.05	59	12.27	36
a šumové	Q1	17.78	80	16.06	96.25	18.68	119	19.43	77
proměnné	med	19.20	95.50	17.47	167	20.75	135	22.24	108
(100	Q3	20.38	100.75	18.52	186	23.62	157	24.86	134.25
souborů)	max	24.56	115	21.37	200	31.41	197	33.72	185

Tab. 1. Výsledky experimentů. Minimum, maximum, medián (med) a 1. a 3. kvartil (Q1, Q3) empirického rozdělení chyby na testovacím souboru (R %) a velikosti stromů ("listů").

7. Optimální stromy zítra

Způsob hledání "správně velkého" stromu použitý v dosavadních experimentech rozhodně není optimální. V nejbližší době hodláme implementovat křížovou validaci, která nám umožní hospodárněji využít data a usnadní nám mj. práci s reálnými daty, kde nemůžeme (na rozdíl od dat umělých) dle vlastního uvážení zvětšovat počet pozorování.

V popisu algoritmů I a II se skrývá neřešený problém. Chybu při dané velikosti nebo kombinovanou ztrátu často minimalizuje ne jeden strom, ale velký počet stromů. O tom, který z nich se stane výstupem algoritmu, rozhodují jednak pravidla typu "při shodě má přednost první (poslední) kandidát", jednak mikroskopické numerické chyby (související s konečnou přesností reálných čísel v počítačích). Kdybychom dovedli podle nějakých charakteristik (jako např. hloubka, rovnoměrnost počtů pozorování v listech, atp.) odhadnout, které ze stromů se stejnou či skoro stejnou chybou na trénovacích datech budou mít lepší generalizační vlastnosti, mohli bychom takovou znalost zabudovat do našich procedur. Vytipování takových charakteristik je jedním z námětů pro budoucí výzkum.

Velikost operační paměti, kterou potřebuje současná verze programu, roste jako 3^P, kde P je počet binárních prediktorů. Faktor 3^P souvisí s tím, že se rezervuje paměť pro každou ze 3^P podkrychlí prostoru prediktorů $\{0,1\}^P$. Při sofistikovanější

implementaci, konkrétně kdybychom přidělovali paměť jen takovým krychlím, do kterých patří nějaká trénovací data, by asymptotika spotřeby paměti byla o něco příznivější – faktor 3^P by byl nahrazen faktorem $N2^P$, kde N je počet různých hodnot vektoru prediktorů v datech (tj. číslo ne větší než velikost trénovacího datového souboru).

Podívejme se na důsledky. Dnes jsme schopni řešit na dostupných počítačích úlohy s maximálně cca 15 prediktory. Při velikosti trénovacího souboru kolem 500 platí zhruba stejný limit i pro implementaci s faktorem $N2^P$ místo 3^P. Extrapolujeme-li do budoucnosti pravidlo, že výkonnost počítačů včetně velikosti RAM se zdvojnásobuje přibližně každých 18 měsíců, potřebujeme nyní k rozšíření našich možností o jeden prediktor cca 2.4 roku, lepší implementací můžeme tuto dobu zkrátit na 1.5 roku. Úlohy s 20 prediktory pak budou jedním způsobem řešitelné asi za 12 let, druhým už za 7 a půl roku, se 30 prediktory buďto za 36 let, nebo jen za 22 a půl roku. Poslední cifra dává na rozdíl od předposlední určitou šanci, že se někteří autoři dočkají řešitelnosti úloh se 30 binárními prediktory během svého aktivního života.

V praktických úlohách se ovšem zpravidla vyskytují kategoriální prediktory s větším počtem hodnot než 2 a prediktory s uspořádaným oborem hodnot. Problém obsahující takové proměnné lze formálně převést na úlohu s pomocnými binárními prediktory, počet těchto pomocných proměnných však rychle přesáhne naše současné, popř. i budoucí možnosti. Abychom z optimalizace stromů učinili široce použitelný praktický nástroj, potřebovali bychom maximální počet binárních prediktorů zvýšit nejméně na 100 (čtenář si může spočítat, jak dlouho by to trvalo)[4]. Chceme-li v rozumném čase takových parametrů dosáhnout, nezbývá nám nic jiného než od plné optimalizace upustit. Hodláme se tudíž v budoucnosti věnovat studiu heuristických postupů, které budou na jedné straně výpočetně méně náročné než optimalizace, ale na druhé straně co možná zachovají převahu nad klasickými metodami tam, kde optimalizace poskytuje podstatnou výhodu. Na plnou optimalizaci pak můžeme pohlížet jako na prostředek, který nám umožní na úlohách, kde plná optimalizace je realizovatelná, zkoumat, nakolik se výsledky heuristických postupů výsledkům optimalizace skutečně blíží.

O praktické užitečnosti optimalizace stromů a příbuzných metod (založených na heuristické aproximaci optimalizace) se však bude rozhodovat ještě jinde: Experimenty uvedené v paragrafu 6 ukazují, že v některých úlohách, kde vztah mezi prediktory a závisle proměnnou je dosti komplikovaný, dává optimalizace výrazně lepší výsledky než klasické metody, ale jinde pro změnu získáme optimalizací velmi málo nebo nic. Úlohy, kde optimalizace přináší podstatnou výhodu, dnes umíme uměle konstruovat. Teprve budoucnost ale ukáže, zda se vyskytují v nezanedbatelné míře také mezi problémy reálnými.

Literatura

Blumer A., Ehrenfeucht A., Haussler D. & Warmuth M. (1987). Occam's Razor. *Information Processing Letters* **24**, 377–380.

Breiman L., Friedman J.H., Olshen R.A. & Stone C.J. (1984). *Classification and Regression Trees*. Belmont CA: Wadsworth.

[4]Nejstarší z autorů, J. A., s poněkud již dětinskou umíněností trvá na poznámce, že pro něj je $2^{100} = \infty$, ať si třeba celý svět věří, že $2^{100} = 1267650600228229401496703205376$.

Friedman J.H. (1979). A tree-structured approach to nonparametric multiple regression. In: *Smoothing Techniques for Curve Estimation. Lecture Notes in Math.* **757**, (eds. T. Gasser & M. Rosenblatt), 5-22. Berlin: Springer-Verlag.

Gelfand S.B., Ravishankar C.S. & Delp E.J. (1991). An iterative growing and pruning algorithm for classification tree design. *IEEE Trans. on Pattern Analysis and Machine Intelligence* **13**, *2*, 163-174.

Pijls W. & Bioch J.C. (1999). Mining frequent itemsets in memory-resident databases. Manuscript, `http://www.few.eur.nl/few/people/pijls/`

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

Savický P., Klaschka J. & Antoch J. (2000). Optimal classification trees. In: *COMPSTAT 2000, Proceedings in Computational Statistics* (eds. J.G. Bethelem & P.G.M. van der Heiden), 427-432. Heidelberg: Physica-Verlag.

Siciliano R. (1998). Exploratory versus decision trees. In: *COMPSTAT 98, Proceedings in Computational Statistics* (eds. R. Payne & P. Green), 113-124. Heidelberg: Physica-Verlag.

ÚI, AV ČR, Pod Vodárenskou Věží 2, 182 07 Praha
E-mail: `savicky@cs.cas.cz, klaschka@cs.cas.cz`

UK MFF, KPMS, Sokolovská 83, 186 75 Praha
E-mail: `jaromir.antoch@karlin.mff.cuni.cz`