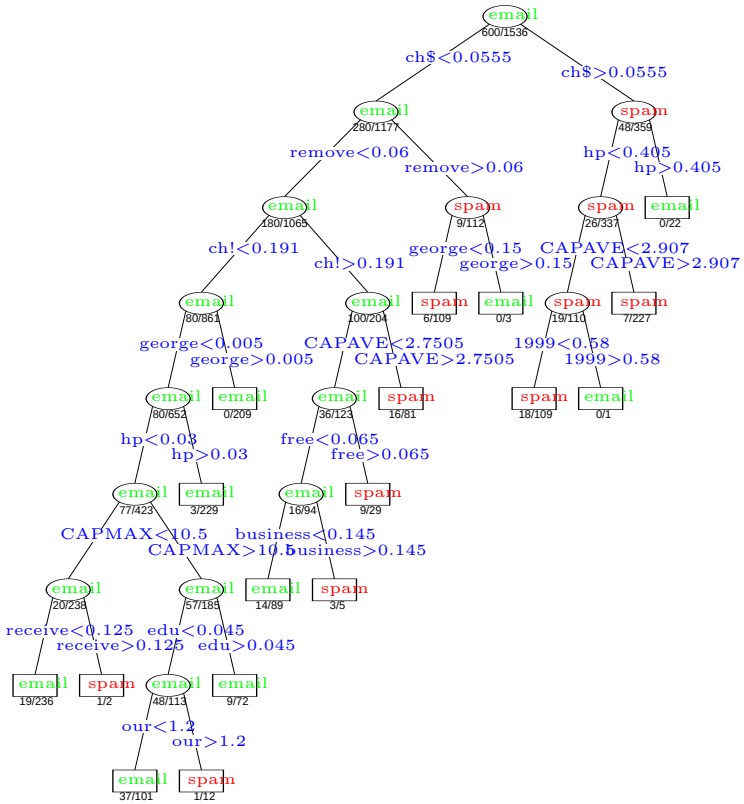
Rozhodovací stromy

Rozhodovací strom pro daný cílový atribut G je kořenový strom tvořený z

- kořene a vnitřních uzlů označených atributem; ze kterého vede jedna hrana pro každou možnou hodnotu tohoto atributu;
- listy jsou označeny předpokládanou hodnotou cílového atributu G za předpokladu, že ostatní atributy nabývají hodnot na cestě od kořene do listu. Pokud se některé atributy na cestě nevyskytují, na jejich hodnotě nezáleží.



Idea konstrukce

Základ algoritmu tvorby rozhodovacího stromu z dat je následující:

- vyber atribut; vytvoř z něj uzel a rozděl data podle hodnoty tohoto atributu
- pro každou hodnotu atributu vytvoř podstrom z dat s odpovídající hodnotou
- pokud data obsahují jen jednu hodnotu cílové třídy či pokud došly atributy k dělení, vytvoř list s hodnotou nejčetnější cílové třídy.

Otázkou je, jak vybírat atribut k dělení.

Entropie

Po míře entropie rozdělení hodnot daného atributu A (tj. míře nejistoty, negativní míře informace) chceme, aby:

- byla nula, pokud jsou všechny hodnoty cílové třídy stejné
- byla největší, pokud je stejně hodnot všech cílových tříd (tj. nevíme nic)
- aby rozhodnutí ve dvou krocích vedlo ke stejnému výsledku jako rozhodnutí naráz, tj.

$$E([2,3,4]) = E([2,7]) + \frac{7}{9} \cdot E([3,4])$$

Toto splňuje pouze **entropie** $E([p_1, \ldots, p_n]) = -\sum_{i=1}^{n} p_i \log p_i$, logaritmus se bere většinou dvojkový.

Pozn. nemusíme normalizovat, pak dostaneme entropii násobenou součetem všech p_i.

Pokud chceme uvést, přes který atribut entropii počítáme, používáme dolní index, např. E_A, resp. E_G pro cílový atribut.

Entropie pro dvouhodnotový atribut

vodorovná osa: p_i, svislá: entropie.
$Gini = 1 - \sum_i (p_i)^2$

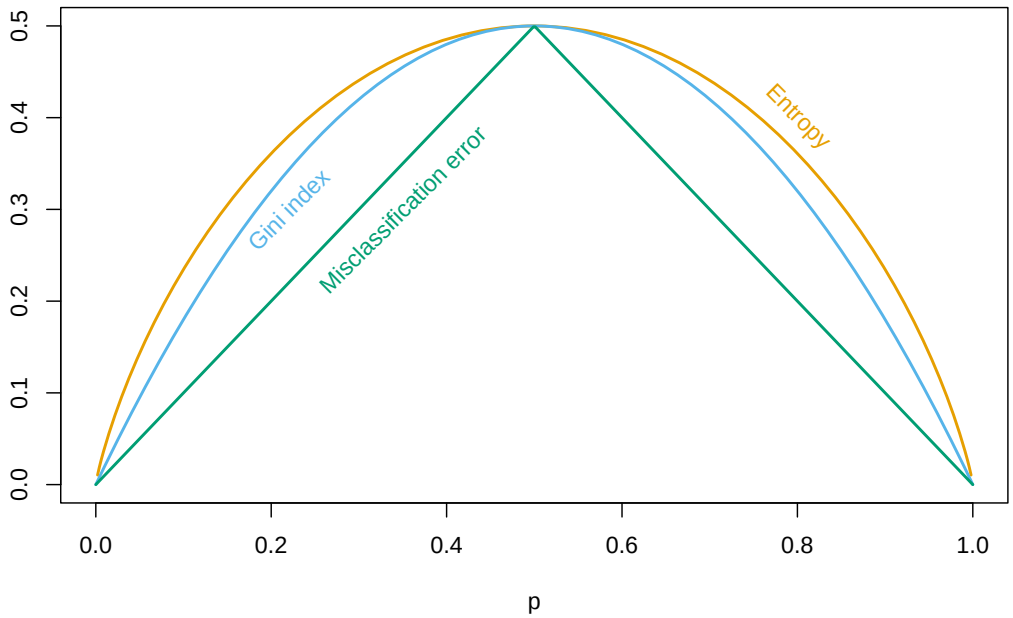


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* $(0.5, 0.5)$.

ID3 algorithmus

Uzel, který dáme do kořene (pod)stromu, vybíráme podle maximálního **informačního zisku** (information gain), definovaného pro množinu dat *data* a atribut X_j jako:

$$Gain(data, X_j) = E_G(data) - \sum_{x_j \in X_j} \frac{|data_{X_j=x_j}|}{|data|} E_G(data_{X_j=x_j})$$

kde $data_{X_j=x_j}$ je podmnožina *data*, kde atribut X_j má hodnotu x_j, entropie je definovaná

$$E_G(data) = \sum_{g \in G} -\frac{|data_{G=g}|}{|data|} \cdot log_2 \frac{|data_{G=g}|}{|data|} = \sum_{i=1}^{|G|} -p_i \cdot log_2 p_i$$

kde p_i je počet dat v *data* patřící do třídy g_i dělený celkovým počtem dat v *data*.

ID3 algorithm(*data*, *G* cíl, *Attributes* vstup. atributy)

Vytvoř kořen *root*
Pokud mají všechna data stejné g, označ kořen g a konec,
Pokud došly *Attributes*, označ *root*
nejčastější hodnotou g v *data* a konec
jinak
X_j= atribut z *Attributes* s maximálním $Gain(data, X_j)$
označ *root* atributem X_j
pro každou hodnotu x_j atributu X_j,
přidej větev pod *root*, odpovídající testu $X_j = x_j$
$data_{X_j=x_j}$= podmnožina *data*, kde $X_j = x_j$
Je–li $data_{X_j=x_j}$ prázdné, přidej list označený
nejčastější hodnotou g v *data* a konec
jinak přidej podstrom $ID3(data_{X_j=x_j}, G, Attributes \setminus \{X_j\})$
vrať *root*

Penalizace mnohahodnotových atributů

Identifikační kód záznamu má nulovou entropii – každé ID má jen jedna instance a ta má jednoznačnou hodnotu. Přímá volba atributu podle maximálního informačního zisku by tedy volila ID atribut.
To ale není dobré pro predikci, protože to nezobecňuje na nové příklady.
Vícehodnotové atributy proto penalizujeme započítáním informace obsažené v samotném dělení podle atributu, bez ohledu na cílovou třídu.
Např. ID kód pro N příkladů bude mít informaci obsaženou v atributu:
$E([1, 1, \ldots, 1]) = -(\frac{1}{N} \cdot \log \frac{1}{N}) \cdot N = \log N$.
Pro výběr atributu se pak používá gain ratio – podíl informačního zisku a entropie daného atributu.
V praxi se navíc přidávají ad–hoc testy na vyhození ID, i když vyjde nejlepší, a na omezení přílišné závislosti na entropii atributu tím, že vyberu maxmální gain ratio jen tehdy, pokud je i information gain aspoň tak dobrý jako průměrný information gain přes všechny testované atributy.

Prořezávání

Snažíme se zabránit přeučení, odstraňujeme přebytečné uzly Occamova břitva

- preferujeme jednodušší model (jsou–li srovnatelně dobré)
- penalizujeme složité modely
- ...

- **postprunning** – nejdřív vytvoříme strom, pak ho prožezáváme;
- **preprunning** – ukončit tvoření stromu dříve, než dojdeme na konec.

I když pre–prunning vypadá lákavěji (než zbytečně tvořit dál a prořezávat), častěji se používá postprunning, protože nezamítne kombinaci atributů, z nichž žádný jednotlivec není užitečný, ale jako celek užiteční jsou. Není jasné, jestli lze nalézt preprunning strategii, která by byla stejně dobrá, jako postprunning.

Prořezávání

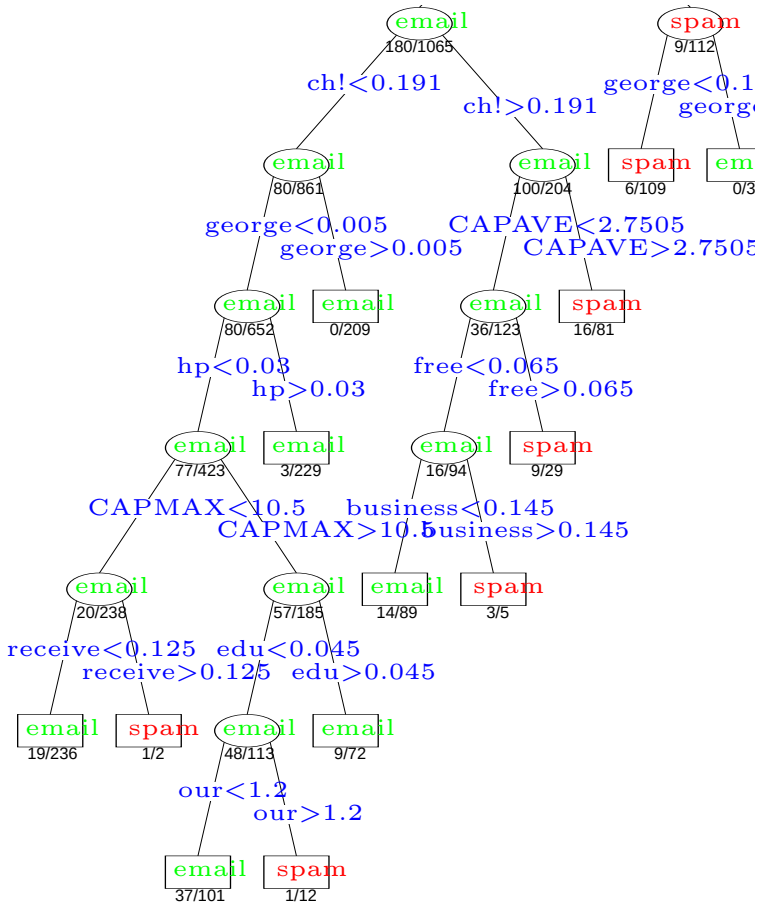
Postprunning uvažuje dvě operace,

- **subtree replacement** – vyberu podstrom a nahradím ho listem; to určitě sníží přesnost na trénovacích datech, ale může to zlepšit predikci na nezávislých testovacích datech. Postupujeme od listů ke kořeni, v každém uzlu buď podstrom nahradíme, nebo necháme.
- **subtree raising** – komplexnější a je otázkou, je–li třeba. Ale je použité v C4.5. Jde o vynechání jednoho uzlu, nahrazení ho jeho podstromem a překlasifikováním příkladů, které patřily do zbylých podstromů. V praxi se to zkouší jen pro nejnavštěvovanější větev.

Odhad chyby

- **reduced error prunning** – nechám trochu dat na prořezávání, ale bohužel učím strom na méně datech.
- odhadovat chybu na základě trénovacích dat – použito v C4.5, motivováno statistikou, v praxi funguje.

C4.5 navíc nevybere atribut, který má skoro jen jednu hodnotu – vyžaduje aspoň dvě hodnoty s aspoň dvěma příklady s tím, že ta druhá dvojka se dá nastavit a měla by se zvětšit, máme–li hodně dat s velkým šumem.



Odhad chyby z trénovacích dat

Vezměme data prošlá do daného bodu, nejčastější třída reprezentuje predikci. Tím máme určitý počet chyb *Err* z počtu instancí *N* v daném uzlu. Předpokládejme, že skutečná (true) pravděpodobnost je *q* a těch *N* instancí bylo generováno Bernoulliho procesem s parametrem *q*, ze kterých je *Err* chybných.
Vzhledem k tomu, že odhad je založen na trénovacích datech, děláme pesimistický odhad a místo výpočtu konfidenčního intervalu bereme jako odhad horní hranici tohoto intervalu.

V C4.5 je default hladina významnosti (confidence level) $\alpha = 0.25$ hledáme z tak, aby:

$$P\left[\frac{q-f}{\sqrt{\frac{q(1-q)}{N}}} > z\right] = \alpha$$

kde $f = \frac{Err}{N}$ je pozorovaná frekvence chyby. Pro $\alpha = 0.25$ je $z = 0.69$, nižší α vede k drastičtějšímu prořezání. z použijeme pro výpočet pesimistického odhadu frekvence chyby *err*

$$err = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

Nebo přibližně:

$$err \approx f + z\sigma \approx f + z \cdot \sqrt{\frac{f(1-f)}{N}}$$

Příklad.

Numerické atributy

64	65	68	69	70	71	72	75	80	81	83	85
yes	no	yes	yes	yes	no	no,yes	yes,yes	no	yes	yes	no

U tohoto atributu připadá v úvahu 11 dělicích bodů, ale pokud neuvažujeme dělení mezi dvěma stejnými hodnotami cílového atributu, tak jen 9. Pro každý řez zvláš» můžeme zjistit informační zisk, např. pro řez v 71.5 dostaneme

$$E([9,5]) - E([4,2],[5,3]) = E([9,5]) - (\frac{6}{14} \cdot E([4,2]) + \frac{8}{14} \cdot E([5,3]))$$

$= 0.940 - 0.939$ bitů.

Testů podle numerického atributu může být na cestě z kořene do listu víc – narozdíl od diskrétního atributu, kde rovnou dělíme podle všech možných hodnot.

Chybějící hodnoty

Ignorovat celou instanci často nemůžeme, protože by nám nezbyla žádná data.Navíc, pokud chybějící atribut nenese informaci pro rozhodování, tak je tahle instance stejně dobrá jako všechny ostatní.
Pokud **fakt, že údaj chybí, není náhodný,** pak je vhodné chybění hodnoty považovat za další možnou hodnotu atributu. např. plat neuvedou ti, kdo ho mají hodně malý či hodně velký.
Pokud ale samotný **fakt chybění nenese informaci** (např. neměřil porouchaný teploměr), pak je lépe nakládat jinak.
Možné řešení je **rozdělení instance na kousky** (numericky vážit podle počtu trénovacích instancí jdoucích jedotlivými větvemi), dojít k listům, a váženě zkombinovat predikce na listech.
Podobně používáme **váhy pro výpočet informačního zisku** a informač. poměru a pro dělení instancí podle chybějícího atributu.

Složitost algoritmu

CART

- Mějme N instancí o p atributech. Předpokládejme hloubku stromu $O(logN)$, tj. že zůstává rozumně hustý. Vytvoření stromu potřebuje $O(p \cdot N \cdot logN)$ času. (Na každé hloubce se každá instance vyskytne právě jednou, je $logN$ hloubek, v každém uzlu zkoušíme p atributů, nejhorší případ stromu (nevyvážený) potřebuje $O(p \cdot N \cdot N)$ času.
- Složitost subtree replacement je $O(N)$, složitost subtree raising je $O(N(logN)^2)$.
- Celková složitost tvorby stromů je $O(p \cdot N \cdot logN) + O(N(logN)^2)$.

MARS

- $N \cdot m^2 + p \cdot m \cdot N$ přidání jedné funkce k m existujícím s p prediktory.
- Celkově $O()NM^3 + pM^2N$.

Stromy pro numerickou predikci

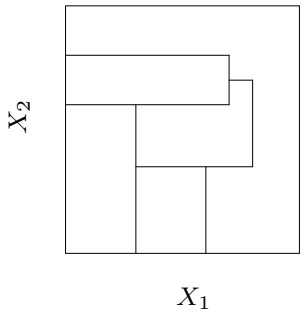
listy stromů obsahují numerické hodnoty, rovné **průměru trénovacích příkladů v listu**.

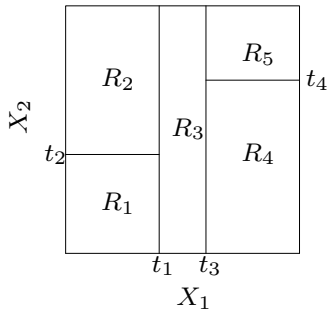
pro výběr atributu k dělení zkoušíme všechny řezy, vybíráme **maximální zlepšení střední kvadratické chyby**.

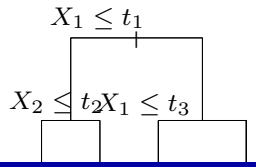
Tyto stromy se nazývají také regresní stromy, protože statistici nazývají regrese proces pro predikci numerických hodnot.

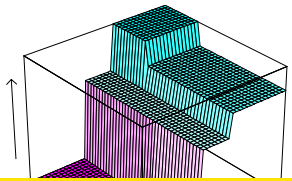
Můžeme i kombinovat regresní přímku a rozhodovací strom tím, že v každém uzlu bude regresní přímka – takový strom se nazývá model tree.

CART CART dovoluje multivariate decision trees, kdy se rozhodujeme na základě kombinace více parametrů (tj. řez nemusí být rovnoběžný s osou). Často tvoří přesnější a menší stromy, ale pracuje daleko déle a stromy se hůře interpretují.









Stromy pro numerickou predikci

- Identifikátory regionů R_m, uvnitř modelujeme konstantou c_m, průměrem.

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$
$$\hat{c}_m = ave(y_i | \ x_i \in R_m).$$

- Myopicky hledáme nejlepší řez přes všechny proměnné j a body dat s :

$$\min_{j,s}[min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2]$$

  - Vnitřní minimalizace řeší průměr $\hat{c}_1 = ave(y_i | \ x_i \in R_1)$.

Reduced Cost Prunning

$$
\begin{aligned}
\hat{c}_m &= \frac{1}{N_m} \sum_{x_i \in R_m} Y_i \\
Q_m(T) &= \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2, \\
C_\alpha(T) &= \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|
\end{aligned}
$$

Pravidla ze stromů

- Ze stromu můžeme vytvořit pravidla napsáním pravidla pro každý list.
- Ta pravidla ale můžeme ještě zlepšit tím, že uvažujeme každý atribut v každém pravidle a zjistíme, jestli jeho vynecháním pravidlo nezlepšíme (tj. nezlepšíme chybu na validačních datech resp. pesimistický odhad chyby na trénovacích datech).
- To funguje celkem dobře, ale dlouho, protože pro každý atribut musíme projít všechny trénovací příklady. Algoritmy tvořící pravidla přímo bývají rychlejší.
- Výsledná pravidla setřídíme podle klesající úspěšnosti.

Ohodnocení úspěšnosti klasifikace

Základní statistiky uváděné při ohodnocování modelů pro klasifikaci (např. ve Weka) vycházejí z tzv. matice záměn (confusion matrix):

správná třída \ klasifikace	+	-
+	TP – true positive	FN – false negative
-	FP – false positive	TN true negative

Česky se říká správně/falešně positivní/negativní.

Základní míry ohodnocení modelu

celková správnost	accurancy	$Acc = \frac{TP+TN}{TP+TN+FP+FN}$
chyba	error	$Err = \frac{FP+FN}{TP+TN+FP+FN}$
přesnost	precision	$Prec = \frac{TP}{TP+FP}$
úplnost, sensitivita	recall, sensitivity	$Sensit = Rec = \frac{TP}{TP+FN}$
specificita	specificity	$Specificity = \frac{TN}{TN+FP}$

Různá cena chyby

Raději predikuji infarkt i tomu, kdo ho nedostane, než nevarovat toho, kdo ho dostane.

$L_{kk'}$ matice ceny za chybnou klasifikaci k jakožto k'

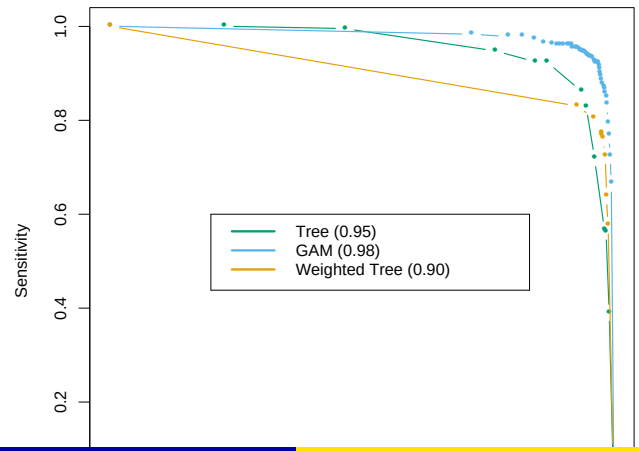
- pro vícekategoriální klasifikaci modifikujeme $Gini = \sum_{k \neq k'} L_{kk'} \hat{p}_{mk} \hat{p}_{mk'}$
- pro dvouhodnotovou vážíme prvky třídy k $L_{kk'}$ krát
- v listu pak klasifikujeme $k(m) = argmin_k \sum_l L_{lk} \hat{p}_{ml}$

Slabší stránky CART

- nestabilita stromů: pro trochu jiná data mohu dostat naprosto jiný strom; lze zmírnit průměrováním přes více stromů (bagging)
- řezy pouze kolmo na osy; dalo by se rozšířit, ale pak se hůře optimalizuje
- výsledek není hladký, ale schodovitý. Pokud předpokládám hladkou cílovou funkci, je to divné. MARS bude hladký (Multivariate Adaptive Regression Splines)
- špatně podchytí aditivní strukturu

$$Y = c_1 I(X_1 < t_1) + c_2 I(X_2 < t_2) + \ldots + c_k I(X_k < t_k) + \epsilon$$

po prvním dělení bude v různých větvích dělit různě, globální vzorec z toho nikdo nevykouká a nejspíš ani strom neodhalí (pro nedostatek dat u listů). MARS toto zvládne.



Lineární regrese

- Cíl: aproximovat funkci $f(x)$, kde x je n–rozměrný vektor, pomocí lineární funkce

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{n} x_j \hat{\beta}_j$$

- Není–li $X^T X$ singulární, dostaneme jednoznačné řešení

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

- a odhad $\hat{y}$ pro dané x_i je $\hat{y}(x_i) = x_i^T \hat{\beta}$

MARS Multivariate Adaptive Regression Splines

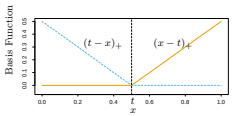


FIGURE 9.9. *The basis functions $(x-t)_+$ (solid orange) and $(t-x)_+$ (broken blue) used by MARS.*

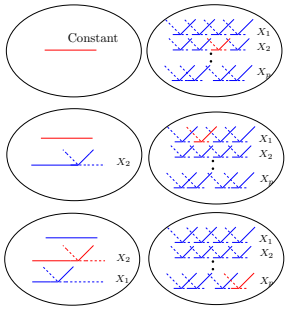


FIGURE 9.10. *Schematic of the MARS forward model-building procedure. On the left are the basis functions currently in the model: initially, this is the constant function $h(X) = 1$. On the right are all candidate basis functions to be considered in building the model. These are pairs of piecewise linear basis functions as in Figure 9.9, with knots t at all unique observed values*

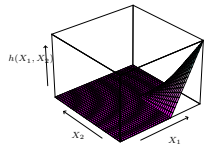


FIGURE 9.11. *The function $h(X_1, X_2) = (X_1 - x_{51})_+ \cdot (x_{72} - X_2)_+$, resulting from multiplication of two piecewise linear MARS basis functions.*

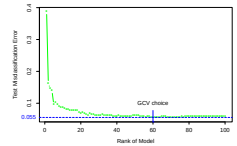


FIGURE 9.12. *Spam data: test error misclassification rate for the MARS procedure, as a function of the rank (number of independent basis functions) in the model.*

- zobecnění postupné lineární regrese i zobecnění CART
- pro každou vstupní proměnnou a každý datový bod vytvoříme dvojici funkcí báze
- $(x-t)_+$ a $(t-x)_+$, kde to $+$ značí nezápornou část, zápornou ořízneme nulou. Tuto dvojici nazýváme zrcadlový pár.
- máme tedy množinu funkcí
- tedy $2Np$ funkcí, jsou–li všechny vstupní hodnoty různé

MARS – pokračování

- model je tvaru

$$f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m h_m(X)$$

  kde $h_m(X)$ je funkce z $\mathcal{C}$ nebo součin libovolného počtu funkcí z $\mathcal{C}$
- pro pevně zvolená h_m spočteme koeficienty β_m standardní lineární regresí (minimalizujeme součet kvadrátů reziduí)
- funkce h_m volíme postupně

MARS – volba báze

- Začneme s konstantní $h_0 = 1$, funkci přidáme do modelu $\mathcal{M} = \{h_0\}$.
- Uvažujeme součin každé funkce z modelu $\mathcal{M}$ s každou dvojicí v $\mathcal{C}$

$$\hat{\beta}_{M+1} h_\ell(X) \cdot (X_j - t)_+ + \hat{\beta}_{M+2} h_\ell(X) \cdot (t - X_j)_+, h_\ell \in \mathcal{M}$$

  vybereme ten, co nejvíce sníží trénovací chybu (vždy dopočteme regresní koeficienty $\hat{\beta}$).
- Opakujeme, dokud $\mathcal{M}$ nemá předem daný počet členů
- protože je model často přeučený, zas ubíráme, vždy tu, co nejméně zvýší trénovací chybu. Máme tak posloupnost modelů $\hat{f}_\lambda$ pro různé počty parametrů λ.
- vybereme tu, co minimalizuje zobecněnou krosvalidaci (abychom se nemuseli namáhat krosvalidaci počítat)

$$GCV(\lambda) = \frac{\sum_{i=1}^{N} (y_i - \hat{f}_\lambda(x_i))^2}{(1 - M(\lambda)/N)^2}.$$

- $M(\lambda)$ je počet efektivních parametrů v modelu, tj. počet funkcí h_m (označím r) plus počet uzlů K, zkušenost radí násobit počet uzlů třikrát, tj. $M(\lambda) = r + 3K$.

Z MARSu CART

- I když se nezdá, souvisí.
- místo po částech lineární zvolíme po částech konstantní funkce $I(x - t > 0)$ a $I(x - t \leq 0)$
- Pokud funkci modelu h_m použijeme pro násobení, tak jí z modelu vymažeme, aby nešla znovu použít. Tím zajistíme, že se použije maximálně pro jedno násobení, tj. jen pro jedno dělení – a máme binární strukturu stromu.
- a máme CART.

"Hon na maxima" PRIM = Bump Hunting

Patient Rule Induction Method

- iterativně hledáme oblasti, kde je Y velké; pro každou oblast vytvoříme pravidlo
- CART po cca. $\log_2(N) - 1$ řezech přijde o data, PRIM si může dovolit cca. $-\frac{\log(N)}{\log(1-\alpha)}$.
  Pro $N = 128$ a $\alpha = 0.1$ to je 6 a 46 resp. 29, protože počty pozorování musí být celé.

PRIM Pravidla pro regresi

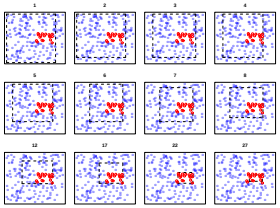


FIGURE 9.7. *Illustration of PRIM algorithm. There are two classes, indicated by the blue (class 0) and red (class 1) points. The procedure starts with a rectangle (broken black lines) surrounding all of the data, and then peels away points along one edge by a prespecified amount in order to maximize the mean of the points remaining in the box. Starting at the top left panel, the sequence of peelings is shown, until a pure red region is isolated in the bottom right panel. The iteration number is indicated at the top of each panel.*

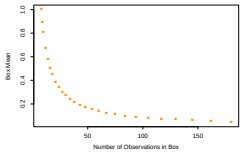


FIGURE 9.8. *Box mean as a function of number of observations in the box.*

- Vezmi všechna data a prostor obsahující všechna data, $\alpha = 0.05$ nebo 0.10
- Najdi X_j a jeho horní či dolní okraj, jehož oříznutím o $\alpha \cdot 100$ procent pozorování vede k největší střední hodnotě zbytku.
- Opakuj 2. dokud zbývá aspoň 10 pozorování.
- Rozšiř oblast v libovolném směru, pokud to zvýší střední hodnotu.
- Vyber z oblastí generovaných 1 až 4 tu (ten počet pozorování), který je nejlepší při krosvalidaci. Nazvěme odpovídající oblast B_1.
- Odstraníme data v B_1 z databáze a opakujeme 2 až 5, vytvoříme B_2, atd., dokud je libo.