Лекция 10. Избягване на преспецификация при регресионни дървета и локални регресионни дървета

10.1. Увод

В предишната лекция бяха разгледани методи за създаване на регресионни дървета, базирани на алгоритъма за рекурсивно разделяне на обучаващото множество от примери. Като следствие от това, при нарастване на дървото изборът на най-доброто разбиване се основава на все по-малки и по-малки подмножества от примери. По тази причина изборите на разделяния в низките нива на дървото стават често статистически ненадеждни, макар че изчисляваната оценка за извадковата грешка на дървото продължава да намалява. Като краен резултат често се получава едно много голямо дърво, което няма добро обобщаване върху нови, тестови примери. Този ефект се нарича преспецификацията на дървото към обучаващите данни, тъй като построеното дърво отразява някои зависимости, които присъстват в обучаващата извадка, а не в самата проблемна област, от която тези данни са били взети. Този факт служи като мотивация да бъдат разработвани модели за подрязване на регресионни дървета, които, в най-общия случай, се свеждат до два метода – *предварителното подрязване,* спиращо нарастване на дървото, когато по-нататъшното разделяне се разглежда като ненадеждно, и *завършващото подрязване*, когато премахването на ненадеждни части на дървото става след завършване на процеса на построяване на голямо и вероятно преспецифицирано дърво.

Предварителното подрязване има очевидни предимства от изчислителната гледна точка в сравнение със завършващото. Обаче, то може да доведе до прекалено рано спиране на ръста на дървото, които води до получаване на суб-оптимални дървета. По тази причина най-често използвания метод за избягване на преспецификацията е завършващото подрязване на регресионни дървета. Някои от тези методи са предмет на настоящата лекция.

Процесът на получаване на "най-добро по размер" регресионно дърво може да се разглежда като процес на търсене в пространство на всички възможни поддървета, които могат да бъдат построени чрез изрязване от началното, прекалено голямо дърво. От тази гледна точка завършващото подрязване на едно регресионно дърво се свежда до решаване на два основни въпроса: 1) какъв метод трябва да се ползва за организиране на претърсването на голямото пространство от всички възможни подрязани дървета и 2) какъв метод да бъде използван за оценяване на възможни алтернативи, възникващи по време на търсене.

Съществуващите подходи към завършващото подрязване на регресионни дървета могат да бъдат разбити на два големи класа – едно-стъпкови и дву-стъпкови. При едно-стъпковите методи (които се използват по-често) алгоритмът обхожда последователно всички възли в дървото (от корена към

листата или обратно), решавайки във всеки възел дали да го изреже или не в зависимост от прилагания критерий за оценяване. При дву-стъпковите методи от начало се генерира определена последователност от подрязани дървета, от които на втората стъпка се избира "най-доброто" като крайния модел. Макар че едно-стъпковият подход изглежда по-ефективен от изчислителната гледна точка, дву-стъпковият дава по-голяма гъвкавост, тъй като генерираните на първата стъпка дървета представляват алтернативни модели с различен баланс между точността на модела и неговата сложност. Освен това изборът на "най-доброто" дърво става на по-глобално ниво (между алтернативните модели), докато при едно-стъпковият метод той е локален, тъй като решението за изрязването се взима във всеки възел от началното дърво.

10.2. Едно-стъпкови методи за завършващото подрязване на регресионни дървета

10.2.1. Подрязване, базирано на m-приближения

Системата RETIS (Karalic 1992) използва един метод за подрязване, базиран на алгоритъма N&B (Niblett and Bratko 1986), който обхожда всички възли на началното дърво T_{max}, от листата – до корена, като във всеки нетерминален възел $t \in T_{\max}$ сравнява грешката на t и претеглената грешка на под-дърво с корен в t (T_t). Теглата се определят чрез пропорцията на примери, минаващи по всеки клон на t. Ако грешката на t е по-малка от грешката на T_t, то дървото T_{max} се изрязва във възела t.

Критичната част на този алгоритъм е получаване на надеждната оценка за грешки. За тази цел авторите използват една Бейсова техника, известна като m-приближение или m-оценител (*m-estimator*), която оценява някой неизвестен параметър θ на общото разпределение чрез следващата комбинация от априорни и апостериорни знания:

$$mEst(\theta) = \frac{n}{n+m} P(\theta \mid D_n) + \frac{m}{n+m} P(\theta) \qquad (10.1)$$

където:

$P(\theta)$ е наша априорна оценка на параметъра θ.

$P(\theta|D_n)$ – апостериорна оценка на този параметър (базирана на извадката данни D_n с размер n, а

m е параметър на този метод за оценяване.

В контекста на LS-регресионни дървета, получаване на грешката на едно дърво изисква изчисляване на средна квадратична грешка на всяко от неговите листа. Извадковите оценки за средна и средна квадратична стойности на грешките се получават на база на примерите, покрити от листото l и се изчисляват по формулите:

$$\bar{y}(D_l) = \frac{1}{n_l} \sum_{i=1}^{n_l} y_i, \quad u \; MSE_{\bar{y}}(D_l) = \frac{1}{n_l} \sum_{i-1}^{n_l} (y_i - \bar{y}(D_l))^2 \quad (10.2)$$

където:

$$D_l = \{< \mathbf{x}_i, y_i > \in l\},$$
$$n_l = \# D_l$$

Обикновено е доста сложно да бъдат получени данни за априорната вероятност на оценявания параметър. Стандартната процедура за избягване на тази трудност се състои в използването за тази цел на цялата обучаваща извадка. В нашия случай това означава, че априорните вероятности на средната стойност на Y и на MSE се получават чрез техните оценки върху цялото обучаващо множество. Апостериорните оценки се получават на базата на съответната извадка от данни в листото.

От тук получаваме m-приближение на средната стойност на Y в листото l:

$$mEst(\bar{y}) = \frac{n_l}{n_l + m}\frac{1}{n_l}\sum_{D_l} y_i + \frac{m}{n_l + m}\frac{1}{n}\sum_{D} y_i = \frac{1}{n_l + m}\sum_{i=1}^{n_l} y_i + \frac{m}{n(n_l + m)}\sum_{i=1}^{n} y_i \qquad (10.3)$$

Съответно, m-приближение на средна квадратична грешка в листото l се задава от:

$$mEst(MSE_{\bar{y}}) = \frac{n_l}{n_l + m}\frac{1}{n_l}\sum_{D_l}(y_i - mEst(\bar{y}))^2 + \frac{m}{n_l + m}\frac{1}{n}\sum_{D}(y_i - mEst(\bar{y}))^2 =$$
$$\frac{1}{n_l + m}\sum_{i=1}^{n_l} y_i^2 + \frac{m}{n(n_l + m)}\sum_{i=1}^{n} y_i^2 - (mEst(\bar{y}))^2 \qquad (10.4)$$

Основният проблем при използването на m-приближения е избор на подходящата стойност за параметъра m, която зависи от конкретната проблемна област. За тази цел могат да бъдат използвани методи на крос-валидация, при която се оценява определено множество от стойности на този параметър и се избира тази, която води до най-добра предсказващата точност на дървото. Обаче подобен подход е доста скъп от изчислителна гледна точка.

10.2.2. Подрязване, базирано на принципа MDL

Системата CORE (Robnik-Sikonja and Kononenko 1998) също използва N&B алгоритъм за подрязване на дървета, описан в предишния раздел. Обаче, вместо сравняване на оценките на грешки във всеки възел t и в неговото под-дърво T_t, CORE използва като ръководство за приемане на решение дали да изрязва някой възел в дървото *принципа за минимална дължина на описание (MDL – Minimum Description Length)*, вече ни познат от лекциите по МС.

Основната идея на MDL-принципа е, че най-простото обяснение на наблюдаваният феномен е най-вероятно да бъде правилно. MDL-принципът може да се опише като предпочитане на такава теория, която минимизира сумата от дължината на описание на теорията плюс дължината на описание на данни чрез тази теория:

$$h = \arg\max_{h \in H} (L(h) + L(D \mid h)) \qquad (10.5)$$

където:

h е теория (или хипотеза), принадлежаща към пространството от възможни теории *H*;

D е множеството от данни

L(.) представя дължина на двоичното представяне (кодиране)

Съгласно този принцип, може да бъде предпочетена една по-къса теория, която допуска няколко грешки върху обучаващите данни, пред една по-голяма теория, която перфектно покрива тези данни.

Авторите предлагат следната схема на двоичното кодиране на регресионни дървета. Двоичният код на едно регресионно дърво се състои от кода на модела и на неговите грешки. Кодирането на дървото се представя като последователността от битове, кодиращи всеки от неговите възли. За всеки възел се кодира неговият тип (дали това е листо или разделящ възел) и неговото съдържание (регресионният модел на листото, например средната *Y* стойност или разделящия тест). Двоичният код на листото се състои от кодирането на използвания модел и кодирането на грешка на този модел. Например, ако в някое листо на LS-дърво имаме набор от примери със следните стойности на Y: {25, 30, 60, 70} със съответната средната стойност 46.25, то броят на битове, необходим за кодирането на това листо е еквивалентно на дължината на кодирането на следните реални числа:

CodeLen(46.26) + CodeLen(46.25-25) + CodeLen(46.25-30) + ...

Кодирането на реалните числа се прави по метода на Rissanen (1982), при който реалното число се дели със зададената точност ε, а полученото цяло число се кодира като двоичен низ. Дължината на кода на низа, съответстващ на някое цяло число, се определя по следната формула:

$$CodeLen(0) = 1$$
$$CodeLen(n) = 1 + \log_2(n) + \log_2(\log_2(n)) + ... + \log_2(2.865064)$$

Като сумирането включва само положителни членове.

Кодирането на разделящите възли зависи от типа на разделяне. Първата част на кода прави това разделение, докато последващите битове съответстват на кода на разделянето. Например, за кодиране на разделянето на някой номинален атрибут във вида $X_i \in S$, където *S* е множеството от стойности от областта $Dom(X_i)$, е необходимо да кодираме информацията за това, кой атрибут се тества и какво е множеството от стойности, участващо в разделянето. Това става по следната схема:

$$CodeLen(X_i \in S) = \log_2(\#A) + \#Dom(X_i) \quad (10.6)$$

където *A* е множество от атрибути.

Кодирането на разделянето на непрекъснат атрибут е подобно на описаното по-горе, само вместо множеството от стойности е необходимо да бъде кодирана

разделящата граница (праг) между възможния диапазон на стойностите на тествания атрибут:

$$CodeLen(X_i \le V) = \log_2(\#A) + \log_2\left(\frac{range(X_i)}{\varepsilon}\right) \quad (10.7)$$

където:

$range(X_i)$ е диапазонът на стойностите на X_i , а

ε е исканата точност на разделящата граница

Предложената схема на кодиране има два параметъра – точността, използвана за кодиране на грешките в листата, и точността за кодиране разделяния по непрекъснати атрибути. Техните стойности могат да бъдат определени чрез процедурата на крос-валидацията.

И така, алгоритъм за подрязване, използван в CORE, обхожда дървото с помощта на алгоритъма N&B и във всеки възел t сравнява дължината на неговият двоичен код с дължината на кода на съответното под-дърво T_t. Ако последната е по-голяма, то голямото дърво T_{max} се подрязва във възела t.

10.2.3. Подрязване в M5

Системата M5 (Quinlan 1993) използва метод за обхождане на дървото, подобен на N&B алгоритъм. В листата на дървото M5 използва линейни регресионни модели, по тази причина решението за подрязване на дървото се управлява от критерий, който е различен от тези, използвани в RETIS или CORE. За всеки възел t M5 построява линеен модел, използвайки примерите от този възел и включвайки в модела само тези атрибути, които се проверяват в под-дървото T_t . След това получената стойност на грешка на построения модел се умножава по един евристично избран наказателен коефициент $(n_t + v)/(n_t - v)$, където n_t е броят на примери в t, а v е броят на атрибути, включени в линейния модел. Получената оценка за грешката след това се сравнява със същата оценка за под-дървото T_t и ако последната е по-голяма, под-дървото се изрязва.

10.3. Дву-стъпкови методи за завършващото подрязване на регресионни дървета

Както следва от самото наименование, тези методи се състоят от две основни стъпки – построяване на поредица от алтернативни подрязани дървета и избор от тях на "оптималното" дърво. Резултатите, получени на всяка от тези стъпки, зависят от използваният критерий за избор.

10.3.1. Построяване на поредица от алтернативни дървета

За пръв път дву-стъпковият подход към завършващото подрязване на регресивни дървета бил използван в системата CART (Brieman et al. 1984). От първоначалното дърво T_{max} CART генерира една последователност от вложени

подрязани дървета, използвайки алгоритъм наречен *грешка-сложност (Error-Complexity)*. Това е един итеративен алгоритъм, който започва с T_{max} като първия елемент от последователността (T_0) и генерира всяко следващо подрязано дърво в последователността чрез намирането на възел $t \in T$, който минимизира критерия:

$$g(t,T) = \frac{Err(t) - Err(T_t)}{\#\widetilde{T}_t - 1} \quad (10.8)$$

където:

T е текущото дърво в последователността,

$T_t \in T$ е под-дърво с корен във възел t, а

$\#\widetilde{T}$ е броят на листа в това под-дърво

На практика, описаният алгоритъм минимизира мярката грешка-сложност:

$$EC_\alpha(T) = Err(T) + \alpha \times \#\widetilde{T} \quad (10.9)$$

където α се нарича параметър на сложността и определя цената на всяко листо.

Като резултат, алгоритъмът генерира една параметрична последователност от вложени подрязани дървета $T(\alpha) = <T_0, T_1, ..., T_n>$, като последното дърво е самият корневият възел. Всяко дърво в тази последователност се характеризира чрез различната стойност α_i, които не е нищото друго, а последователните стойности на функцията *g(t, T)*. Brieman е доказал, че всяко дърво T_i в тази последователност е оптимално по отношение на EC_α критерия в интервала $[\alpha_i, \alpha_{i+1})$.

Един по-прост вариант на този критерий е бил предложен от Bohanec и Bratko (1994). Техният метод се състои в избирането на възела *t*, който води до най-малкото увеличение на извадковата грешка:

$$\min_t (Err(t) - Err(T_t)) \quad (10.10)$$

И в двата случая, алгоритъмът за генериране на последователност от вложени подрязани дървета трябва да прегледа всички възли на текущото дърво, т.е. е силно зависим от размера на дървото. Обаче, в CART е предложен по-ефективен алгоритъм, който има средна изчислителна сложност $O(\#\widetilde{T} \log \#\widetilde{T})$.

Много лесен и ефективен метод за генериране на алтернативни подрязани дървета е предложил Луис Торго в своята система RT (1999). Напомням, че основният мотив за подрязване на регресионните дървета е ненадеждността на оценки за истинската грешка на дървото, получавани на базата на малки по размер извадки от обучаващи данни. Следователно, колко по-малко примери покрива един възел, толкова потенциално ненадеждна става оценката за неговата грешка. Това разсъждение води до алгоритъма, в който всяко следващо вложено подрязано дърво се получава от предишното чрез изрязване на най-ненадеждния възел, т.е. възел *t*, който минимизира:

$$\min_t (n_t) \quad (10.11)$$

Освен своята простота, предложеният от Торго подход, наречен от него *алгоритъм за най-малката статистическа поддръжка (LSS – Lowest Statistical Support)*, има големи изчислителни предимства, тъй като последователността, в която ще бъдат изрязвани възли, може да бъде получена само чрез едно минаване по първоначалното дърво (а на практика – дори в процеса на неговото построяване!). Изрязването на някой възел не променя това подреждане, тъй като броят на примери, покрити от един възел, остава непроменен. Това означава, че, за да генерира последователността от алтернативни подрязани дървета с LSS-алгоритъм, трябва само да получим един списък от възлите, подредени в нарастващ ред по броя на покрити от тях примери, и след това да премахваме последователно всеки възел от този списък, получавайки следващо подрязано дърво.

Интензивните експерименти с различни бази данни, проведени от Л. Торго, показаха, че LSS метод генерира по-точни дървета от други описани методи.

10.3.2. Сравняване на алтернативни подрязани дървета

И така, след като беше построена редица от алтернативни подрязани дървета, е необходимо да дефинираме метод за избирането на "най-доброто" от тях в качество на финалното решение. Очевидно е, че критерият за "най-доброто" трябва да се базира на някаква надеждна оценка за *истинската* (т.е. върху неизвестни на системата примери) грешка на всяко алтернативно дърво. Тъй като всички дървета са LS-регресионни дървета, то на нас ни трябва метод за оценка на истинската средно-квадратична грешка на всяко дърво.

За решаването на тази задачи можем с успех да приложим вече описаните по-горе методи, оценяващи дървета чрез m-приближения на грешките или чрез прилагането на MDL принципа. И в двата случая за получаването на добри резултати трябва да използваме процедурата за крос-валидацията, за да намерим най-добрите параметри на алгоритъма (константата m – в първия случай, или константите ε – във втория).

Процедурата за k-крос-валидация се използва и в системата CART за получаването на оценките за истинските грешки на поредицата от алтернативни дървета. Основната идея на k-крос-валидация (k-CV) се състои в разбиването на зададеното обучаващо множество D на k непресичащи се и приблизително равни по размер подмножества $D_1, \ldots, D_k$. За всяко подмножество D_i се построява избраният регресионен модел $r_i(\boldsymbol{\beta}, \mathbf{x})$ (в нашия случай LS-дърво), използвайки $D \backslash D_i$ като обучаващи данни. Моделът се тества върху множеството от примери D_i. Същият процес се повтаря за всяко подмножество. Накрая, за оценка на истинската грешка се приема грешката на крос-валидацията, изчислена като средната грешка на тези k модела.

Напомням, че в CART алгоритъмът "грешка-сложност" създава една параметрична последователност от вложени дървета $T(\alpha) = <T(\alpha_0), \ldots, T(\alpha_n)>$, където чрез $T(\alpha_i)$ означаваме дървото, асоциирано със съответната α стойност. Чрез процедурата за k-крос-валидация CART, освен тази основна последователност, генерира по същия алгоритъм допълнителни k параметрични

последователности $T^1(\alpha)$, ..., $T^k(\alpha)$. За всяко едно дърво от всяка от тези последователности имаме изчислена надеждна оценка за неговата истинска грешка, изчислена върху съответното тестово множество. Възниква въпрос: как тези оценки могат да бъдат използвани за намирането на надеждното приближение за истинските грешки на дървета от основната последователност?

Отговорът, който предлага CART, е една евристична процедура, която за всяко дърво $T(\alpha_i)$ от основната последователност намира k "най-сходни" дървета (по едно от всяка от допълнителните последователности $T^1(\alpha)$, ..., $T^k(\alpha)$) и определя оценката за истинската грешка на $T(\alpha_i)$ като средно на оценки за истинските грешки на тези k дървета. По-точно, това става по следния начин: за всяко дърво $T(\alpha_i)$ се изчислява стойността $\alpha_i' = \sqrt{\alpha_i \alpha_{i+1}}$, след което от допълнителните последователности $T^i(\alpha)$ се намират дървета с α стойности, най-близки до α_i'. За избор на финалното дърво в CART са предложени два варианта. При първият, като крайното решение се избира дървото $T(\alpha_i)$ с най-малката стойност на изчислена по описаният по-горе начин оценка за истинската грешка. При вторият, се избира *най-малкото дърво* със стойността на оценката за истинската грешка в интервала $[\hat{Err}_l, \hat{Err}_l + SE(\hat{Err}_l)]$, където $\hat{Err}_l$ - е най-малката стойност на тази оценка, а $SE(\hat{Err}_l)$ е стандартното отклонение на тази оценка. Този метод е известен под името *правило 1 – SE* и толерира по-прости дървета, макар, по-някога, за сметка на по-лоша класификационна точност.

Както се вижда от описаното, предложената процедура няма теоретична обосновка и доста скъпа от изчислителната гледна точка. Един интересен начин да избегнем скъпата процедура за крос-валидация е предложен от Л. Торго. Той се базира на известна от математическа статистика (и курса по МС) *интервалната оценка* на оценяваният параметър. Тя предоставя определен *доверителен интервал*, в който със зададена степен на сигурност x% попада истинската стойност на наблюдавания параметър. Изчисляването на доверителния интервал може да се направи на базата на наблюдаваната стойност на този параметър (т.е. само на базата на обучаващата извадка). В нашият случай това означава, че ние можем да получим доверителният интервал за истинската грешка на всяко листо в дървото на база на неговата извадкова грешка.

Тъй като при построяване на дървото се използва критерий за минимизиране на средна квадратична грешка, то грешката, асоциирана със всяко листо, може да се разглежда като едно приближение на дисперсията на Y стойностите на примери, покрити от това листо. Статистическата теория утвърждава, че извадковото разпределение на дисперсията може да се разглежда като една случайна величина, подчиняваща се на χ^2 (чи-квадрат) разпределение, ако оригиналната величина следва нормалното разпределение. Съгласно свойствата на χ^2 разпределение 100x(1-α)% доверителен интервал за истинската дисперсия може да бъде изчислен чрез извадковата дисперсия и размер на извадката по следната формула:

$$\left(\frac{(n-1)s_Y^2}{\chi^2_{\alpha/2,n-1}}, \frac{(n-1)s_Y^2}{\chi^2_{(1-\alpha/2),n-1}} \right) \quad (10.12)$$

където:

s_Y^2 е извадковата дисперсия (получена в конкретното листо)

$\chi^2_{\alpha,n}$ е табличната стойност на χ^2 разпределение за зададеното ниво на сигурност α и n степени на свобода (броя на примери във листото)

Тази формулировка се базира върху предположение за нормалното разпределение на целевия атрибут *Y*, което не може да бъде гарантирано за повечето от реални проблемни области. Обаче, в нашият случай, е важна не точността на оценката, а гаранция, че тези оценки коректно подреждат кандидати – подрязани алтернативни дървета.

χ^2 разпределение е несиметрично, което означава, че средната точка на интервала, определен по ф-ла (10.12), не съответства на истинската средната стойност на извадковата дисперсия. Разликата между тези две стойности намалява с нарастване на степени на свобода, тъй като χ^2 разпределение апроксимира нормалното, когато броят на степени на свободата е достатъчно голям. Това означава, че с нарастване на размера на извадката средната точка на интервала от ф-ла (10.10) ще се приближава към точка, означаваща средната оценка на истинската дисперсия, а това е точно този вид предпочитания, върху които се базират повечето от алгоритми за подрязване на дървета. Те "наказват" оценките, получените в листата на големи дървета (покриващи само няколко примера) при сравнение с оценки за възли, разположени по-високо в дървото.

Тези съображения накараха Л. Торго да предложи използването на средната точка на доверителния интервал от ф-ла (10.12) да бъде използвана като една по-надеждна оценка за дисперсията във всеки възел на дървото. Това води до следният оценител на истинската стойност на средна квадратична грешка (MSE) във възела *t*:

$$ChiEst(MSE(t)) = MSE(t) \times \frac{n_t - 1}{2} \times \left(\frac{1}{\chi^2_{\alpha/2,n-1}} + \frac{1}{\chi^2_{(1-\alpha/2),n-1}} \right) \quad (10.13)$$

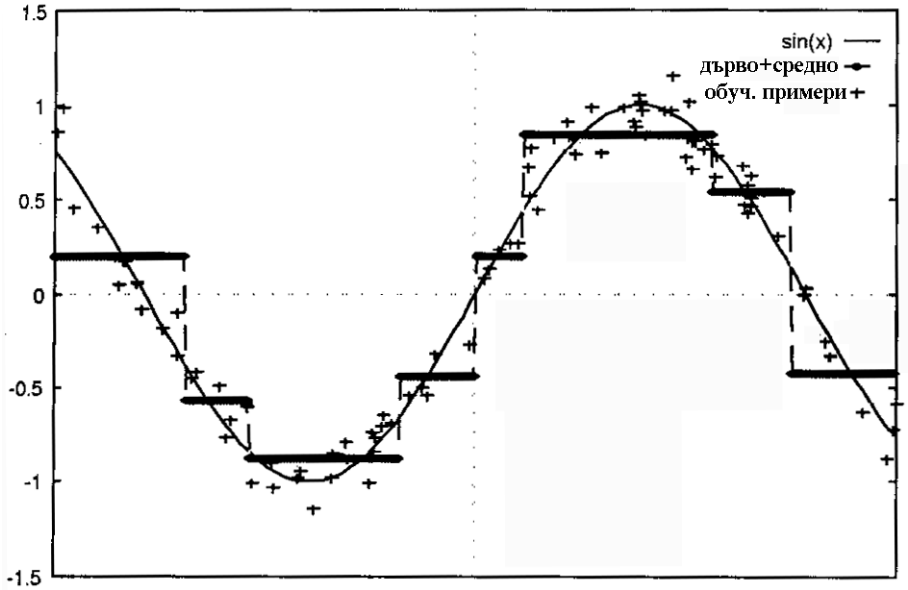
$\frac{n_t - 1}{2} \times \left(\frac{1}{\chi^2_{\alpha/2,n-1}} + \frac{1}{\chi^2_{(1-\alpha/2),n-1}} \right)$ може да се разглежда като един коригиращ фактор за MSE във възела *t*.

Подобната стратегия е използвана в C4.5 (Quinlan) за подрязване на класификационни дървета, където към оценка за извадковата класификационна точност на възела се прилага определен "коригиращ" фактор, базиращ на биномното разпределение на грешката.

Интензивни експерименти, проведени от Л. Торго, показаха че предложеният от него метод за избор на финалното дърво чрез критерия (10.13) при нивото на достоверност 95% постига много добри резултата в сравнение със всички други описани дву- и едно-стъпкови методи за подрязване на LS-регресионни дървета и е много ефективен от изчислителна гледна точка.

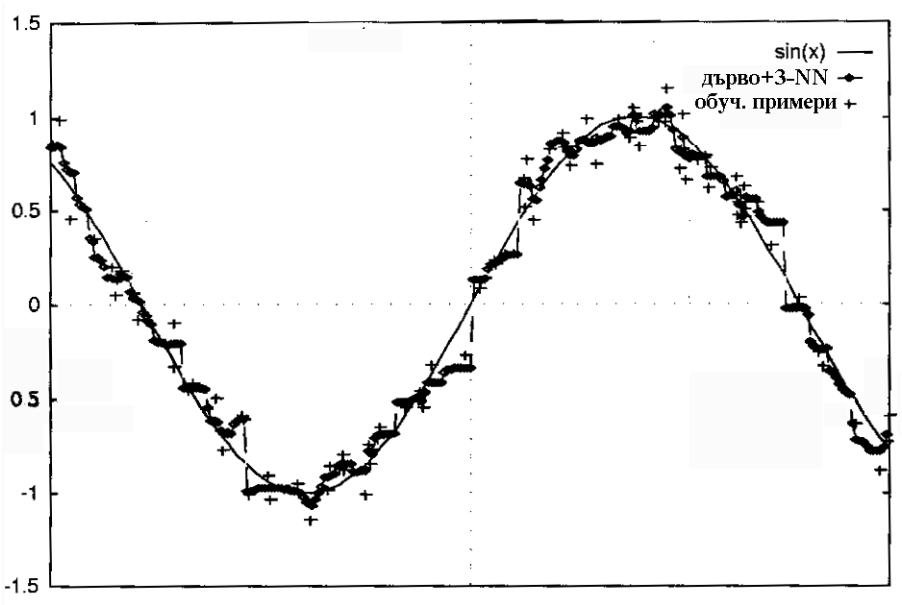
10.4. Локални регресионни дървета

В предишната лекция ние сме разгледали методи за построяване на регресионни дървета, които предполагат един постоянен регресионен модел в техните листа. За LS-дървета, минимизиращи критерия за средна квадратична грешка, тази константа е средната стойност на целевата променлива *Y*, изчислена върху обучаващите примери, покрити от съответното листо. Апроксимацията, която се прави от подобни типове регресионни дървета, представлява една доста негладка функция. Да разгледаме, като пример, каква апроксимация дава едно регресионно дърво за целевата функция *Y = sin(X)*. Да предположим, че обучаващото множество се състои от 100 примера, случайно генерирани с използване на функцията *Y = sin(X) + N(1,0)*, където *N(1,0)* е шум, представен като функцията (гаусиана) на нормалното разпределение с средната стойност 0 и дисперсията 1. Апроксимацията, получена от едно LS-дърво е показана на Фиг. 1. Тя е една стъпаловидна функция, която в някои място има доста голями точки на разрива на непрекъснността. Тези свойства обикновено се разглеждат като основни недостатъци на регресионните дървета.



Фиг. 10.1. Апроксимацията на функция y = sin(x) от едно LS-дърво

Един от възможни начини да минимизира този ефект е използването в листата на регресионни дървета на по-гладки модели. Например, на Фиг. 2 е показана апроксимация, получавана от едно регресионно дърво, научено от същите обучаващи данни, в чиито листа място средните *Y* стойности са използвани модели на ядра (виж предишната лекция).



Фиг. 10.2. Апроксимацията на функция y = sin(x) от едно LS-дърво с модели на ядра (3 най-близки съседи) в листата

Самите локални регресионни модели създават доста гладки апроксимиращи функции. Обаче, те имат няколко сериозни недостатъка, измежду които слаба разбираемостта от страна на потребителя, необходимостта от запазване на всички обучаващи данни и голяма изчислителна сложност при голям обем на обучаващи данни (което е нормалната практика при ИЗД). Освен това, те срещат големи затруднения в проблемните области, където съществуват големи разриви на непрекъснатата, тъй като предполагат, че близко лежащи точки имат сходни стойности на целевия атрибут.

Потенциалните предимства на интегриране на регресионниге дървета с локалните модели са очевидни. От една страна, използването на локалните модели в листата на регресионните дървета ще доведе до по-гладка (и по-точна!) апроксимиращата функция. От друга страна, използването на локалните модели *само* в листата на регресионното дърво ще позволи значително по-ефективно изчисляване на предсказвания, правени от локалните модели, тъй като ограничи търсенето на най-близките съседи само до примери, покрити от конкретното листо. Освен това се повиши разбираемостта на локалните модели и тяхната способност за работа с проблемните области с големи разриви на непрекъснатата.

10.4.1. Възможни методи за интеграция

Възможните подходи за интеграция на регресионните дървета с локалните регресионни модели са три:

1. Използване на локалните модели в листата на дървета още в процеса на научаване на регресионни дървета, т.е. при тяхното създаване и подрязване.
2. Създаване на “стандартни” регресионни дървета и използване на локални модели в техните листа само на фазата на подрязването на дървета.

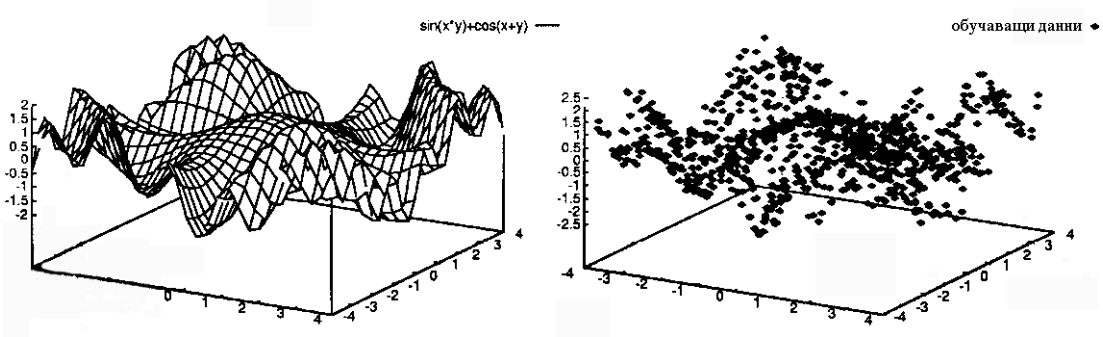
3. Научаване (създаване и подрязване) на стандартни регресионни дървета и използване на локалните модели само за предсказване (определяне на стойността на целевия атрибут на неизвестен пример).

Първата алтернатива изглежда най-последователна от теоретичната гледна точка, тъй като избор на най-доброто разделяне зависи от избрания модел в листата на дървото. Този подход е използван системата RETIS (Karalic 1992), която интегрира глобалният линеен регресионен модел в листата на LS-дървото. Обаче, създаване на такъв интегриран модел е задача с голяма изчислителна сложност, особено за проблемните области с голям брой непрекъснати атрибути и голямо количество на обучаващи примери.

Втората алтернатива е значително по-ефективна от първата, тъй като разглежда по-сложни модели, съдържащи се в листата на породеното дърво, само при неговото подрязване. Този подход е използван в системата M5 (Quinlan 1992), къде в листата се използват глобални линейни регресионни модели.

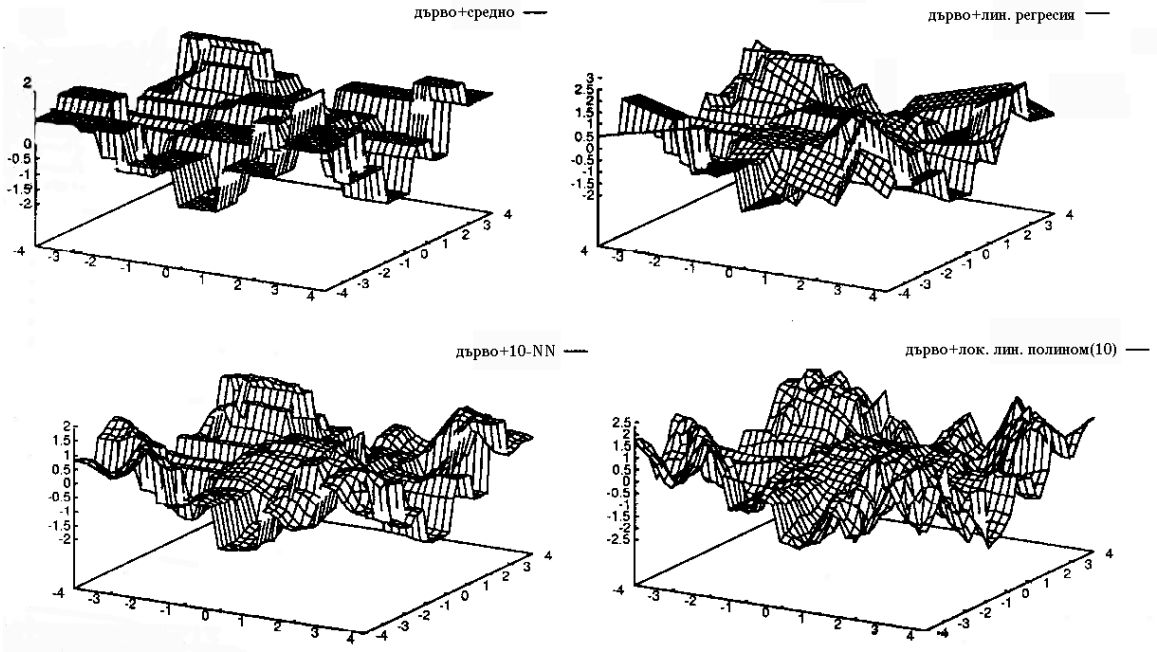
Третата алтернатива е приложени в системата RT (Torgo 1999). Основните предимства на този подход е гъвкавостта (във всяко едно листо на финалното дърво могат да бъдат приложени различни модели) и ефективността. При този подход наученото дърво се разглежда като едно начално приближение към апроксимираната регресионна повърхност (добре разбираемо за потребителя), което след това се доуточнява чрез прилагане на по-сложни модели в листата на дървото. В своята дисертация Торго изследвал използването в листата на дървото на следните видове регресионни модели (всички те са разгледани в предишната лекция):

- Моделите на ядрата (регресионен аналог на методи за k-най-близкия съседи)
- Локални линейни регресионни модели и
- Полу-параметрични (частично линейни) модели.



Фиг.10.3. Целевата функция и обучаващите данни

Всички тези модели се използват само на фазата на предсказване. Тъй като моделите са локални, те включват намирането на най-близките обучаващи примери до тестовия пример. Тази задача се прави само за обучаващите примери, асоциирани (покрити) с конкретното листо. При използването на частично линейни модели изчисляването на параметрите **β** на линейната компонента се прави само един път за всяко от листата (не е за всеки тестовия пример). Получените регресионни дървета той нарича *локални регресионни дървета*.



Фиг. 10.4. Апроксимации, получени чрез интегриране с различни модели

Илюстрацията на ефекта от интегрирането на регресионни дървета с локални регресионни модели е показано на Фиг. 10.3 и 10.4. Фиг. 10.3 показва функцията $f(X,Y) = sin(X \times Y) + cos(X+Y)$ и множеството от 1000 случайно генерирани обучаващи примери, създадени чрез използване на тази функция плюс гаусовия шум.

Обучаващите данни са използвани за научаване на дървото, след което са построени различни апроксимации, което то прави за целевата функция при използване на различни регресионни модели в листата – средните Y стойности, глобални линейни полиноми (горната част на фигура), модели на ядра и локални линейни полиноми (долната част на фигурата).

Интензивните експерименти доказаха, че локалните регресионни дървета значително по-точни от "глобалните", обаче изискват по-големи изчислителни ресурси и по-малко разбираеми от първите. Локалните регресионни дървета значително по-бързи от методи за локално моделиране. Интегрирането на двата подхода значително повишава точността в области със значителни разриви в непрекъснатостта. Най-добрата точност измежду възможни варианти на локални регресионни дървета се постига от интегрирането с частично линейни регресионни модели, които в повечето от случаи превъзхождат такива общи признати методи като CART (Brieman et al. 1984) и CUBIST (Quinlan 1993). Локалните регресионни модели спечелили първото място в конкурса, проведен от организационния комитет на международната конференция Knowledge Discovery in Databases 2000, постигайки най-добрите резултати върху предложените бази данни, отнасящи се за екологическото състояние на някои райони от Европа.