Лекция 4. Проблеми при обучение чрез класификационни дървета

4.1. Избягване на прекаленото нагаждане към данни

Разгледаният от нас базовия алгоритъм ID3 нараства всеки клон на дървото до такава дълбочина, докато не класифицира перфектно всички обучаващи примери. Тази, в общия случай, разумна стратегия води до сериозни проблеми при наличието на зашумени обучаващи примери или когато броят на обучаващите примери е прекалено малък, за да даде представителната извадка на истинската целева функция. И в двата случая алгоритмът създава дървета, които *прекалено се нагаждат* към обучаващите примери.

Ще казваме, че една хипотеза прекалено се нагажда към обучаващите примери (т.е. е *преспецифицирана*), ако съществуват други по-лошо нагодени към същите обучаващи примери хипотези, които имат по-добро поведение върху цялото разпределение от примери (т.е. включващо и примери извън обучаващото множество). По формално:

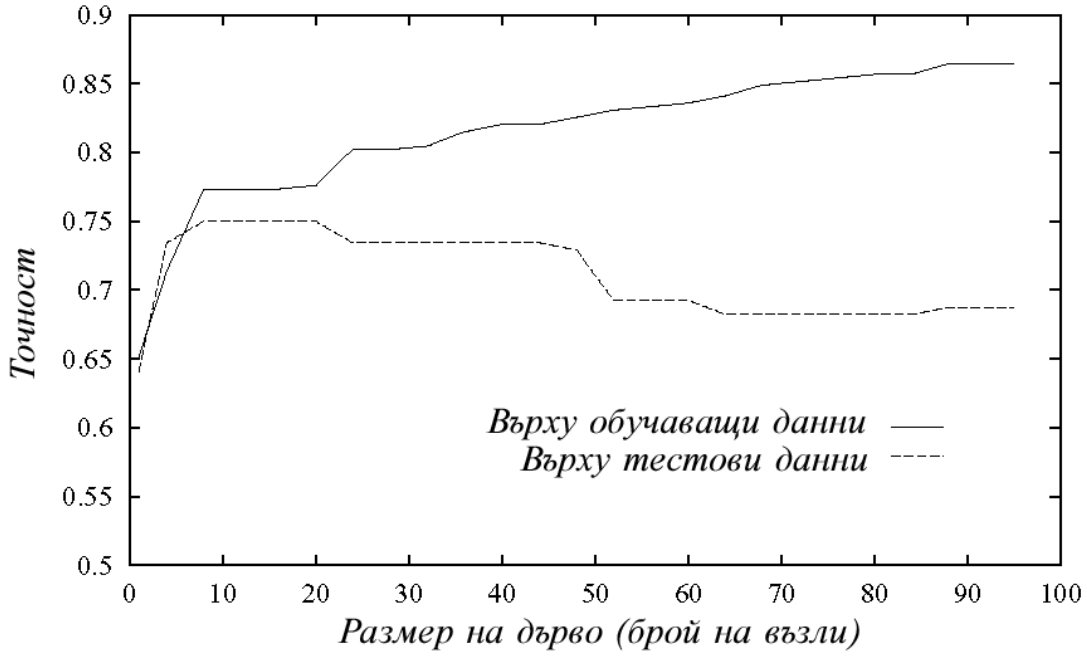

Рис. 4-1. Ефект на преспецифициране при обучение чрез класификационни дървета

Определение. При зададено множество от хипотези H, ще казваме, че хипотезата $h \in H$ е *преспецифицирана* (overfitting) към обучаващите данни, ако съществува някоя алтернативна хипотеза $h' \in H$, такава че h има по-малка грешка от h' върху

множеството от обучаващите примери, но *h'* има по-малка грешка от *h* върху цялото множество от примери.

На рисунката е показан ефектът от преспецифициране при едно от приложения на класификационни дървета (предсказване, кои пациенти имат диабет). Графикът представя функцията на точността на предсказване на едно дърво в зависимост от неговия размер (като брой на възли), който се получава в процеса на построяване на дървото. Плътната линия показва предсказването върху обучаващото множество от примери, а пунктираната – върху независимото (т.е. невключеното в обучаващото множество) множество от тестови примери. Докато точността на дървото върху обучаващите примери монотонно нараства с броя на възли в дървото, същата точност, измервана върху тестовите примери, отначало нараства, а след това (някъде след 25 възли) – започва да намалява.

Как е възможно, че едно дърво *h*, което е нагодено към обучаващите примери по-добре от *h'*, да има по-малка точност на последващите примери? Една от причините да се случи това е наличието на случайни грешки или шум в обучаващите данни. За илюстрация, да разгледаме ефекта от добавянето на следващия положителен пример, неправилно маркиран като отрицателен, към правилните обучаващи примери от Таблицата 3-2 (виж Лекция 3):

<Небе=Слънце, Температура=Горещо,Влажност=Нормална,Вятър=Силен,Игра-на-тенис=не>

Използвайки оригинални (без грешка) примери, ID3 конструира дърво, показано на Рис. 3-1. Обаче, добавката на сгрешения пример кара ID3 да построява по-сложното дърво. В частност, новият пример ще бъде сортиран във второто от ляво листо на научената дърво от Рис. 3-1 заедно с положителните примери D9 и D11. Обаче, тъй като новият пример е маркиран като отрицателен, ID3 ще продължи да търси по-нататъшното уточняване на дървото, построявайки нови клонове под този (вече не терминален) възел. Тъй като новият пример се отличава по нещо от другите примери в този възел, ID3 ще успее да намери нов класификационен атрибут (в случая *Температура*), за да отдели новия пример от другите два положителни. В резултат, ID3 ще изведе новото дърво (*h*), което е напълно нагодено към всички обучаващи данни, докато старото по-просто дърво (*h'*) ще прави грешка върху новия (сгрешен) пример. Обаче, тъй като новият класификационен възел е резултат от усилието за нагаждане към сгрешен обучаващ пример, е нормално да очакваме, че старото дърво *h* ще класифицира нови (без грешки) данни по-точно от новото дърво *h'*.

Преспецифицирането може да се получи и без наличието на сгрешени обучаващи примери. Това е възможно в случаи, когато броят на обучаващите примери, асоциирани с класификационни възли, е прекалено малък, за да може да отрази истинското разпределение на примери. В такива случаи са възможни случайни съвпадения в стойностите на атрибути, които водят до добро разпределение на примери, без да имат някаква реална връзка с действителна целева функция.

Изследванията показват, че за повечето реални задачи ефектът от преспецифицирането на класификационни дървета намалява тяхната класификационна точност с 10-25%.

Съществуват няколко подхода за избягване на преспецифицирането при научаване на класификационни дървета. Те могат да бъдат групирани към следните два класа:

- Подходи, спиращи изграждане на дървото преди то да стигне точката, където перфектно класифицира всички обучаващите примери.
- Подходи, позволяващи построяване на преспецифицираното дърво, което след това се подлага на допълнително окастряне.

Макар, че първият клас подходи изглежда по-пряк, вторият подход – допълнителното окастряне на преспецифицираните дървета – е доказано по-удачен. Причината е сложността в първия подход да прецени, кога точно трябва да бъде спряно изграждането на дървото.

Не зависимо от това, кой от подходите е избран, ключовият въпрос е *какъв критерий трябва да се използва за определяне на размера на финалното дърво*. Използваните подходи са:

- Използване за оценяване на качеството на завършващото подрязване на възли в дървото на отделното множество от примери, различаващо се от обучаващите примери.
- Използване за обучение на всички достъпни данни, но прилагане на статистическия тест за оценяване, дали разширяването (или изрязването) на конкретния възел може да доведе до подобрение в поведението на дървото върху данни извън обучаващото множество. Например, Quinlan (1986) използва чи-квадрат тест за преценяване, дали по-нататъшното развиване на възела е може да подобри поведението на дървото върху цялото разпределение на примери.
- Използване на определена мярка за кодиране на сложността на обучаващите примери и класификационното дърво, която спира разрастването на дървото, когато размерът на това кодиране е минимален. Този подход, базиран на евристиката, наречена принцип за *минималната дължина на описание* (MDL), ще бъде дискутиран по-късно при разглеждане на Бейсовия подход към самообучение.

Първият подход се използва най-често и обикновено се нарича *подходът с използване на обучаващото и потвърждаващото множества.* Основното на подхода е, че всички налични данни се разбиват на два множества – *обучаващото множество*, чиито примери се използват за научаване на хипотезата, и отделното *потвърждаващо множество,* използвано за оценка на точността на тази хипотеза върху последващи данни и, в частност, за оценяване на влияние от подрязването на хипотезата. Основният мотив за такова разбиване е следният – дори, ако алгоритмът е бил подведен от наличието на шума в обучаващите данни или от някои случайни съвпадения в това множество, потвърждаващото множество е

много по-малко вероятно да притежава същите случайни флуктуации. По тази причина очаква се, че потвърждаващото множество осигурява достатъчно надеждна проверка за преспецифицирането, породено от наличието на някои фалшиви характеристики в обучаващото множество. Ясно е, че размерът на потвърждаващото множество трябва да бъде достатъчно голям, за да осигурява статистически значима извадка от примери. Една от най-често използвани евристики е една трета от всички налични примери да се ползва за потвърждаване, а две трети – за обучение.

4.1.1. Допълнително подрязване, минимизиращо грешката

Как точно може да се използва потвърждаващото множество за избягване на преспецифицирането? Един възможен подход, наречен *подрязване, минимизиращо грешката* (Quinlan 1987), се състои в разглеждането на всеки решаващ (нетерминален) възел в дървото като възможен кандидат за изрязване. Изрязването на възела се състои в изхвърлянето от дървото на цялото поддърво с корен в дадения решаващ възел и замяната му с възел-листо, класификацията на който се определя от най-често срещана класификация измежду обучаващите примери, асоциирани с възела. Възлите се премахват само, ако полученото подрязано дърво има по-добро класификационно поведение от оригиналното (не подрязаното) дърво върху потвърждаващото множество. Ефектът от този процес е, че всеки възел-не листо, добавен към дървото в резултат от случайни съвпадения (регулярности) в обучаващото множество, има вероятност да бъде премахнат, тъй като е малко вероятно, че същите съвпадения присъстват и в потвърждаващото множество от примери. Възлите в дървото се изрязват итеративно, винаги започвайки с възела, чието отстраняване води до най-голямото увеличение на точността на класификационното дърво върху потвърждаващото множество. Подрязването на дървото се спира, ако по-нататъшното премахване на възлите води до намаляване на точността.

Малко по-подробно за този процес. Точността на един възел-листо се изчислява на базата на примери, правилно класифицирани от листото, и общия брой на примери, минали през листото (т.е. покрити от листото). По същия начин се изчислява и точността на решаващия възел, който след изрязването се заменя с листо. Точността на някой нетерминален (решаващ) възел в дървото се изчислява като претеглената сума от точностите на неговите възли-наследници, като тегловните коефициенти са просто отношение между броя на примери, минали по конкретния клон на дървото, към общия брой на примери, минали през възела, от който излиза този клон. Този начин на изчисляване на точността се разпространява от листата на дървото към корена. Възлите, за които така изчислена точност е по-малка или равна на точността, получавана след тяхното евентуално изрязване, са кандидати за реалното премахване. Премахва се този възел, който води до най-голямото увеличаване на точността.

Използването на отделното множество от примери, управляващо подрязване на дървото, представлява един ефективен подход при условие, че разполагаме с достатъчно голямо количество данни. Най-големият недостатък на този подход е, че когато броят на такива данни е ограничен, разбиването на това множество на две части още повече намалява броя на примери, използвани за обучение, което от своята страна, води до невъзможността да бъде научена целевата функция достатъчно точно.

Алтернативният подход към подрязване на дървета ще бъде разгледан в следващия раздел. Той доказа своята ефективност при решаване на множество практически задачи, използващи ограничено множество от данни.

4.1.2. Допълнително подрязване чрез правила

Един доста успешен метод за намиране на хипотези с голяма класификационна точност е чрез прилагане на техниката за така наречено *допълнително подрязване чрез правила.* Вариантът на тази техника е приложен в системата C4.5 (Quinlan 1993), която е наследник на ID3. Подрязването чрез правила съдържа следните стъпки:

1. Научаване на класификационното дърво от обучаващото множество, което е най-добре нагодено към данни (възможно и да е преспецифицирано).
2. Превръщане на дървото в еквивалентното множество от правила чрез създаване на по едно правило за всеки път в дървото – от корена до всяко от листата му.
3. Подрязване (обобщаване) на всяко едно правило чрез премахване на всяко негово предусловие, което, като резултат, води до подобряване на оценъчната точност на правилото.
4. Сортиране на подрязаните правила в съответствие с тяхната оценъчна точност и разглеждането им в същата последователност при класификация на неизвестни примери.

За илюстрация да разгледаме отново класификационното дърво за понятието *Игра-на-тенис* (Рис. 3-1). На първата стъпка, при превръщане на дървото в правила, всеки тест на атрибута по пътя от корена към конкретното листо се превръща в предусловие на правилото, а класификацията на листото – в заключение (постусловие) на правилото. Например, първите два най-леви пътища в дървото се превръщат в следните правила:

АКО (*Небе* = *Слънце*) ∧ (*Влажност* = *Висока*) *ТО* (*Игра-на-тенис = не)*
АКО (*Небе* = *Слънце*) ∧ (*Влажност* = *Нормална*) *ТО* (*Игра-на-тенис = да)*

На следващата стъпка всяко подобно правило се подрязва чрез премахване на едно от неговите предусловия, чието отстраняване не намалява оценъчната точност на правилото. В примера с първото правило, алгоритмът ще разгледа последователно

възможността за премахването на предусловия *(Небе = Слънце)* и *(Влажност = Висока)*. Ще бъде избрано обобщеното (подрязаното) правило, в което премахването на предусловие доведе до най-голямото подобряване на оценъчната точност на правило. На следващата стъпка на подрязването ще бъде разгледана възможността да бъде премахнато следващо предусловие и т.н. Подрязването не се прави, ако това води до намаляване на оценъчната точност.

Как се изчислява оценъчната точност на едно правило? По-горе ние разгледахме използването за тази цел на потвърждаващото множество от примери. В C4.5 се прилага друг метод, който се базира на използването на самото обучаващо множество. Алгоритмът изчислява една песимистична оценка за точността, имайки пред вид, че използване на обучаващото множество дава завишената оценка в полза на правилата. По-точно, C4.5 изчислява емпиричната точност на правилото върху обучаващите примери, които то покрива. След това изчислява стандартното отклонение на тази оценка, предполагайки биномното разпределение на тази случайна величина. При зададено равнище на значимост, като мярка за оценъчната точност на правилото се избира долната граница на доверителния интервал. Например, при равнището на значимост от 95%, точността на правилото се оценява песимистично като наблюдаваната емпирична точност на правилото, изчислена върху обучаващите примери, които покрива правилото, минус 1.96 умножено по оценъчното стандартно отклонение. При големи множества от обучаващи данни тази оценка е много близка до наблюдаваната точност (тъй като стандартното отклонение е много малко), докато при намаляването на размера на обучаващите данни тази разлика се увеличава. Макар, че описаният метод не е обоснован от статистическата гледна точка, той дава много добри резултати на практиката. Ще говорим по-подробно за статистическите оценки на хипотези в лекциите 18 и 19.

Защо трябва да превръщаме дървета в правила преди подрязването? Предимствата са следните:

- Превръщането в правила позволява да различаваме различни контексти, в които се използва даденият възел. Тъй като всеки различен път в дървото се превръща в отделното правило, решението за изрязване на конкретния решаващ възел (тест на атрибута) може да бъде направено по различен начин за всеки път. Точно обратното става при дървото: ако решим да изрежем някой решаващ възел, единствената алтернатива е или да премахнем цялото поддърво с корен в този възел, или да го оставим изцяло непокътнато.
- Превръщането в правила премахва разликите между проверки на различни атрибути, не зависимо от това, дали те се намират близо до листото или до корена. Това позволява да се избегне трудоемката работа по реорганизация на дървото, ако решим да премахнем някой по-горен възел, запазвайки поддървото под него.
- Превръщането в правила подобрява разбираемостта. Правилата се разбират от хората значително по-лесно от класификационните дървета.

4.2. Оценка на вероятности

Връщайки се към базовия алгоритъм ID3 виждаме, че при изчисляване на ентропията на дадения възел се използват оценки за вероятността, че някой случайно избран пример от съответстващото на възела множество от примери е положителен или отрицателен (p_+ или p_-). Описаният в алгоритъма начин за изчисляване на тези вероятности е чрез пропорцията на съответните примери, т.е. като *относителната честота на срещане*:

$$p_+ = \frac{s_+}{s_+ + s_-}; \quad p_- = \frac{s_-}{s_+ + s_-}; \quad S = s_+ + s_-$$

където s_+ и s_- са съответно брой на положителните и отрицателните примери в S.

Обаче, при създаване на едно класификационно дърво, броят на примери, преминаващи през дадения възел, много бързо намалява с всяко следващо разклоняване на дървото. По тази причина описаният по-горе начин за оценка на вероятностите става ненадежден в долните възли на дървото. За избягване на този недостатък често се използва формулата на Лаплас (наречена *Лапласова корекция*), която в случая на k класа изглежда по следния начин:

$$p_i = \frac{s_i + 1}{S + k}; \quad S = \sum_{i=1}^{k} s_i \text{ , където } s_i \text{ е броят на примери в } S \text{ от клас } i.$$

За илюстрация на оценките на вероятности със и без Лапласовата корекция, да разгледаме примера с последователните хвърляния на монета, която всеки път пада по герб или лице. В таблицата по-долу са показани вероятностите, че монетата ще падне по лице (в %), изчислени като относителната честота на срещане и като Лапласовата корекция. Очевидно е, че формулата на Лаплас дава по-реалистични резултати.

№ на хвърляне	*1*	*2*	*3*	*4*	*5*
Резултат	лице	лице	герб	герб	герб
Относителната честота	100	100	67	50	40
Лапласова корекция	67	75	60	50	43

4.3. Обработка на непрекъснати атрибути

В своя първоначален вариант ID3 е ограничен да работи само с атрибути, приемащи дискретни стойности. Първо, целевият атрибут, чиято стойност се предсказва, трябва да бъде дискретен, и второ, всички атрибути, проверявани в решаващи възли на дървото, също трябва да бъдат дискретни. Това второ ограничение може сравнително лесно да бъде премахнато, позволявайки на

атрибутите да приемат стойностите от непрекъснатия интервал. За да стане това, алгоритмът трябва да може динамично да разбива непрекъснати атрибути на интервали, които да се разглеждат като номинални стойности на някой нов дискретен атрибут. В частност, за един непрекъснат атрибут A алгоритмът може динамично да създаде нов двоичен (Булев) атрибут A_c, който приема стойност *ИСТИНА*, когато $A < c$ и *ЛЪЖА* - в противния случай. Въпросът е, как да бъде избран този праг c?

За илюстрация да предположим, че атрибутът *Температура*, описващ примери на понятието *Игра-на-тенис*, приема непрекъснати стойности. Нека, след като подредихме конкретните стойности на атрибута *Температура* на всички примери в някой от възлите на дървото, получихме следната таблица, отразяваща стойностите на този атрибут и на целевия атрибут:

Температура	5	9	15	21	25	30
Игра-на-тенис	Не	Не	Да	Да	Да	Не

Как да изберем праг c? Ясно е, че бихме искали да изберем такъв праг c, който дава най-голяма информационна печалба. След подреждането на стойностите на непрекъснатия атрибут по големина, можем да генерираме множеството от кандидати за желания праг. Доказано е (Fayyad 1991), че оптималният праг c винаги ще лежи между съседните стойности, за които съответните значения на целевия атрибут са различни. В нашия случай имаме два кандидата за прагове, съответстващи на *Температура*, при които стойността на *Игра-на-тенис* се променя: $(9 + 15)/2 = 12$ и $(25 + 30)/2 = 27.5$. Информационната печалба трябва да бъде изчислена и за двата кандидат-атрибута: $Температура_{>12}$ и $Температура_{>27.5}$. Избира се прагът (т.е. нов двоичен атрибут), носещ най-голяма информационна печалба. При избора, кой следващ атрибут трябва да бъде тестван при развитие на дървото, информационната печалба на този нов атрибут се сравнява със стойностите на информационната печалба на други атрибути и се избира най-добрият атрибут.

И така, ако имаме няколко непрекъснати атрибута, то към базовия алгоритъм на ID3, преди избора на атрибута с най-голяма информационна печалба, трябва да добавим процедурата за избор на най-добрата (в същия смисъл) бинаризация на всички непрекъснати атрибути. Обърнете внимание, че тази процедура трябва да се повтаря във всеки нов възел, тъй като се променя множеството от примери, минаващи през този възел, и, следователно, се променя и множеството от възможните стойности, които могат да приемат непрекъснатите атрибути в този възел.

(Fayyad & Irani 1993) предложили разширението на описания алгоритъм за бинаризацията, позволяващ дискретизацията на непрекъснати атрибути на повече от два интервала. Обаче обикновено той се прилага като една предварителна стъпка – за дискретизацията на цялото множество от примери, към което след това се прилага някой алгоритъм за построяване на класификационни дървета.

4.4. Алтернативни мерки за избор на атрибути

Мярката за информационна печалба има естественото предпочитание към атрибути, които имат повече стойности. В качество на един екстремен случай, да разгледаме, например, атрибут *Дата*, който има много голямо множество от възможни стойности (например, 18 октомври 2017 г). Ако добавим такъв атрибут към описанието на дни, в които предпочитаме (или не) да играем тенис (Табл. 3-2), той ще има най-голяма информационната печалба в сравнение с всички останали атрибути. Причината за това е, че атрибутът Дата *самостоятелно* абсолютно точно предсказва стойността на целевата функция при всички обучаващи примери. Следователно, той ще бъде избран от ID3 за корен на дървото, което ще бъде много широко и с дълбочина 1. Това дърво абсолютно точно ще класифицира всеки от обучаващите примери, но ще класифицира много лошо новите примери. С други думи, атрибутът има толкова много възможни стойности, че разбива обучаващите примери на множество от много малки подмножества и, като резултат, много лошо предсказва стойността на целевата функция на новите примери.

Един възможен начин за избягване на този недостатък се състои в избора на решаващите атрибути на базата на друга мярка за тяхната полезност, различна от информационната печалба. Една такава алтернативна мярка, която е била успешно използвана, е *относителната печалба* (gain ratio) (Quinlan 1986). Мярката за относителната печалба наказва атрибути с прекалено голям брой възможни стойности чрез включване в себе си на елемент, наричан *информация за разделяне* (split information), който е чувствителен към това, на колко части и доколко еднообразно атрибутът разделя данните:

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

където $S_1, \ldots, S_c$ са подмножества на примери, получени от разбиването на S чрез c стойности на атрибута A. Обърнете внимание, че на практика информацията за разделяне е ентропията на S по отношение на стойностите на атрибута A, което контрастира с нашето предишното използване на ентропията, в което разгледахме само ентропията на S по отношение на целевия атрибут.

Относителната печалба се дефинира като отношението на въведената по-рано мярка за информационната печалба към информацията за разделяне:

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

От формулата се вижда, че *SplitInformation(S, A)* наказва използването на атрибути с голям брой еднакво разпределени стойности. Например, да разгледаме n примера, които са напълно разделени по атрибута A (напр. *Дата*). В този случай стойността на *SplitInformation* е равна на $log_2 n$. Ако имаме друг атрибут B - двоичен, който разделя тези примери точно на две части, то за него *SplitInformation* е *1*. Ако

атрибутите *A* и *B* дават еднаква информационна печалба, то очевидно е, че стойността на *GainRatio* за *B* ще бъде по-висока.

Едно практическо затруднение с използването на мярката за относителната печалба за избор на атрибути е, че знаменателят може да бъде равен на нула или да приема много малки стойности, когато $| S_i | \approx | S |$ за някои от стойностите на атрибута. Това прави мярката да бъде неопределена или много голяма за атрибутите, които имат една и съща стойност почти за всички примери в *S*. За избягване от този проблем се използва следната евристика (Quinlan 1986): отначало се изчислява информационната печалба за всеки атрибут, а след това проверката за относителната печалба се прилага само към тези атрибути, за които стойността на информационната печалба е над средната.

4.5. Обработка на обучаващите примери с липсващи стойности на атрибути

В някои случаи наличните данни могат да съдържат липсващите стойности за някои от атрибути. Например, в медицинската диагностика, когато задачата е да предскажем някое заболяване на пациента на базата на различни лабораторни тестове, е възможно, че лабораторните тестове на кръвта са направени само за някои от пациенти. В такива случаи един общоприет подход е да се прави приблизителната оценка на стойността на липсващия атрибут на базата на примери, за които стойността на същия атрибут е известна.

Да разгледаме ситуацията, в която *Gain(S, A)* трябва да бъде изчислена във възела *n* на класификационното дърво, за да прецени, дали атрибутът *A* е най-добрият атрибут за тестване в този възел. Да предположим, че $<x, c(x)>$ е един от обучаващите примери в *S* и че стойността *A(x)* е неизвестна.

Една от стратегиите за работа с липсващите стойности на атрибути е тяхното заместване със стойността, която се среща най-често сред обучаващите примери от същия възел. При другия подход се използва най-често срещаната стойност на атрибута сред обучаващите примери в същия възел, които имат същата класификация *c(x)*. Попълненият по този начин пример може директно да се използва от алгоритъма за научаване на класификационните дървета.

При една по-сложна процедура вместо използване на най-често срещаната стойност, на всяка възможна стойност на атрибута се присвоява определена вероятност. Тази вероятност може отново да бъде оценена на базата на наблюдаваните честоти на срещане на различните стойности на атрибута сред примерите от същия възел. Да предположим, че за някой двоичен атрибут *A* в текущия възел се намират 6 примера с известната стойност на $A = 1$ и 4 примера с $A = 0$. Тогава за липсващата стойност *A(x)* на някой пример във възела можем да кажем, че вероятността, че $A(x) = 1$ е 0.6, а че $A(x) = 0$ е 0.4. След това 0.6 частта от примера, с неизвестна стойност на *A* се предава надолу по клона $A = 1$, а 0.4 частта

от същия пример – по клона *A = 0*. Тези "дробни" примери се използват за изчисляване на информационната печалба на атрибута *A* и могат по-нататък да бъдат разделени още в по-следващите клонове на дървото, ако трябва да се тества втория липсващ атрибут. Същото раздробяване на примери може да се ползва и след научаване на дървото, когато трябва да се класифицират нови примери с неизвестни стойности на атрибути. В този случай класификацията на примера се свежда до намиране на най-вероятната класификация, изчислена чрез сумиране на теглата на фрагментите на примера, класифицирани по-различни начини от листата на дървото. Този метод за обработка на липсващите атрибутни стойности е използван в C4.5 (Quinlan 1993).

4.6. Обработка на атрибути с различна цена

В някои задачи различните атрибути на примери могат да имат различна цена (тегло). Например, в задачата за научаване как да класифицираме определени заболявания, пациентите могат да бъдат описани чрез такива атрибути, като *Температура, Резултати-от-биопсия, Пулс, Резултати-от-теста-на-кръвта* и т.н. Тези атрибути значително се различават по своята цена, както в паричното изражение, така и по степента на неудобства, причинени на пациента. При подобни задачи би било желателно да бъдат предпочитани класификационните дървета, използващи по възможност по-евтини атрибути и ползващи по-скъпи атрибути само когато е необходимо да бъде направена много надеждна класификация.

ID3 може да бъде променен за отчитане на цената на атрибути чрез промяна на мярката за избор на най-добрия атрибут. Така Tan (1993) описва един такъв подход към задачата, в която роботът трябва да бъде научен, как да класифицира обекти на базата на това, как те могат да бъдат хванати от неговия манипулатор. В този случай атрибутите съответстват на показателите на различни датчици, получавани от подвижния сонар на робота. Цената на атрибута *Cost(A)* се измерва като количество секунди, необходими за получаване на стойността на атрибута чрез нагласяване и работата на сонара. Показано е, че по-ефективните стратегии се научават без да се жертва класификационната точност при използване на следната мярка:

$$\frac{Gain^2(S, A)}{Cost(A)}$$

При научаване на правила за медицинската диагностика Nunez (1988) приложи друг критерий за избор на атрибути:

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^m}$$

където $m \in [0,1]$ е параметърът, определящ относителната важност на цената относно информационната печалба на атрибута.