Otsustuspuudega klassifitseerimine

Kristo Käärmann
Arvutiteaduse instituut, Tartu Ülikool
kristox@ut.ee

Andmekaevandamise uurimisseminar MTAT.03.169.
Arvutiteaduse instituut, Tartu Ülikool
Detsember 2003, lk. 16–33

Kokkuvõte

Hierarhiline otsuste puu rekursiivne konstrueerimine on end tõestanud mitmete reaalelu probleemide lahendamisel. Üheks otsustuspuude poplaarsuse põhjuseks ning eeliseks teiste klassifitseerimismeetodite ees on nende intuitiivne arusaadavus ning kasutuslihtsus. Sellest hoolimata ei suuda iga puu igas kontekstis mõistlikku tulemust anda ning ka otsustuspuuga klassifitseerimisel tuleb omada piisavaid teadmisi ülesande domeenist ning ülevaadet otsustuspuude erinevatest meetoditest.

Järgnev referatiivne kirjatöö tutvustab otsustuspuude meetodit andmete klassifitseerimiseks, annab ülevaate levinud algoritmidest, mõnedest uurimissuundadest ning viiteid rakendustele.

Töö lõpuosas pakutakse välja edasimõtlemist vajavad ideed otsustuspuude kasutamiseks andmekogude kirjeldamisel (näiteks tundmatute andmete analüüsil).

1 Sissejuhatus

Üks konkreetne liik ülesandeid, milleks masinõppimist praktikas edukalt rakendatakse, on reaalmaailma objektide ja nähtuste "*ära tundmine*". Näiteks võib kardiogrammi põhjal ära tunda südamehaige, pikslite kombinatsioonis tuvastada tuttava inimese kujutise või serveri süsteemisessiooni logi põhjal avastada pahataht-

liku ründe. Objektide, nähtuste ja situatsioonide äratundmist võib üldisemalt vaadelda ka kui klassidesse jaotamist e. *klassifitseerimist*.

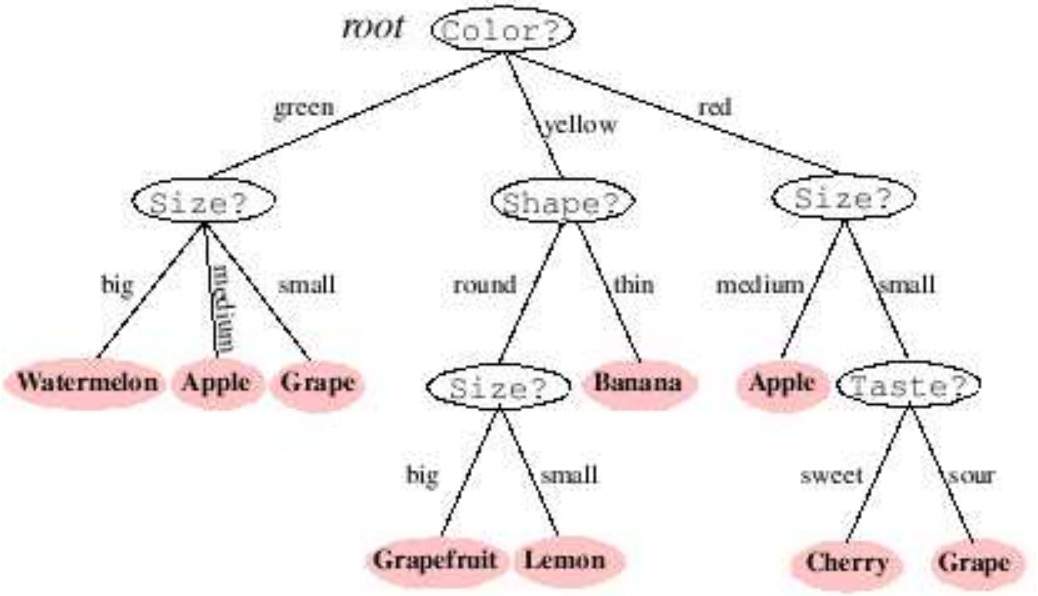
Selle asemel, et masinat programmeerida teatud nähtust või objekti ära tundma, tegeleb masinõppimine arvuti õpetamisega näidete varal. Üldjuhul *treenitakse* teatud adapteeruv algoritm (nt. neurovõrk, otsustuspuu, Bayes-i võrk) teadaolevate objektidega (e. õpiandmetega) välja ning rakendatakse sedasama väljaõpetatud algoritmi hiljem tundamatute objektide *klassifitseerimiseks*. Objekti kohta teada olevaid suurusi, mille alusel objekti üritatakse ära tunda, nimetatakse edaspidi objekti atribuutideks ning klassifi tseerimisel objektile antavat tunnust *klassitunnuseks*. Lihtsamates käsitlustes samastataksegi tundmatu objekt tema atribuutide vektoriga ning abstraktselt võib rääkida ka andmevektorite klassifi tseerimisest. Järgnevas tutvustataksegi lähemalt otsustuspuud, kui üht treenitavat algoritmi klassifi tseerimisülesande lahendamiseks.

Klassifi tseerimismeetodeid tuntakse masinõppimises teisigi: bayesi- ja neurovõrgud, reeglipõhised klassifi tseerijad. Kuna ei saa öelda, et ükski neist meetoditest oleks prevalveeriv, on otsustuspuudest rääkides oluline välja tuua nende eelised teiste meetodite ees:

- väljatreenitud otsustuspuu kasutamine klassifi tseerimisel on algoritmiliselt lihtne, tööpõhimõte intuitiivne ning ilmutatud andmestruktuuri on võimalik täiendada ekspert-teadmistega;
- otsustuspuud ei jää klassifi tseerimistäpsuselt alla teistele meetoditele ning meetodid nende konstrueerimiseks on piisavalt arvutusefektiivsed;
- enam-vähem tänapäevasel kujul võeti otsustuspuude meetod kasutusele 1986. aastal (Quinlan 1986). See on jätnud meetodile piisavalt aega, et end praktilistes rakendustes tõestada, ning uurijatele piisavalt võimalusi edasiarenduste läbikatsetamiseks.

1986. aastal püstitatud klassifi tseerimisülesannet on vahepeal laiendatud pidevate tunnuste ennustamisele (regressioon) ning välja pakutud täiendusi tuletamaks kõikvõimalike eri kuju ning omadustega klassifi tseerivaid puid. Põhjaliku ülevaate otsustustemaatikas avaldatud olulisematest tulemustest annab (Murthy 1998).

Parim autorile teadaolev allikas otsustuspuude tundmaõppimiseks on vastav peatükk Duda ja Hart-i masinõppimise klassikateosest (Duda, Hart, & Stork 1997).



Joonis 1: Lihtne näide otsustuspuust

2 Otsustuspuu õpetamine

Võib ette kujutada tundmatu objekti (e. andmevektori) äratundmist, küsides järjepanu küsimusi objekti tunnuste kohta ning olles kogunud piisavalt infot, otsustada millise objektiga on tegemist (e. objekti klassikuuluvus). Sarnast mängu mängivad vahel lapsed igavuse peletamiseks ning sedasama meetodit objektide klassifi tseerimiseks formuleeritaksegi masinõppimises otsustuspuuks (vt. joonist 1).

Kui otsustuspuu kasutamine on triviaalne, siis veidi rohkem süvenemist vajab otsustuspuu konstrueerimine. Enamlevinud lähenemine otsustuspuu konstrueerimiseks on teha seda ülalt-alla tipu poolitamise teel, alustades kõikide õpiandmetega juurtipust ning rakendades rekursiivselt järgmist üldist eeskirja:

1. Kui puu tippu kuuluvad vaid ühe klassi esindajad, siis märkida tipp leheks ning varustada vastava klassitunnusega.

2. Hinnata *headuse mõõduga* kõikvõimalikke tipu lahutusi (ik *split*), so. tippu kuuluvate õpiobjektide jagunemisi alamtippude vahel. Valida parim lahutus ning formuleerida see tipu küsimuseks (ik *test*).

3. Vastavalt valitud lahutusele luua alamtipud, jaotada nende vahel tipu objektid (so. õpiandmed) ning rakendada sama eeskirja alamtippudele.

Reaalses kasutuses ei vaadata 2. sammul läbi kõikvõimalikke lahutusi (sest selleks tuleks läbi vaadata kõik õpiandmete hulkade hulgad), vaid piirdutakse jagunemistega ainult ühe tunnuse järgi korraga. Et moodustada üldsõnalisest eeskirjast konkreetne algoritm ning saada seejuures võimalikult hea (so. väike ning hästi klassifi tseeriv) otsustuspuu, tuleb esmalt leida lahendused järgmistele küsimustele:

- Mitu vastust võib anda küsimusele igal sammul? Kas tasuks tipus lubada vaid jah/ei küsimusi, so. binaarseid lahutusi?
- Millise omaduse kohta küsida järgmisena ning kuidas hinnata lahutusi?
- Millal on piisavalt infot, et otsustada objekti klassikuuluvus, e. millal kuulutada puu tipp leheks?
- Kui puu kasvab liiga suureks ja keeruliseks, kuidas seda kärpida (ik *pruning*)?
- Kuidas toimida puuduvate andmete ja müra esinemisel?

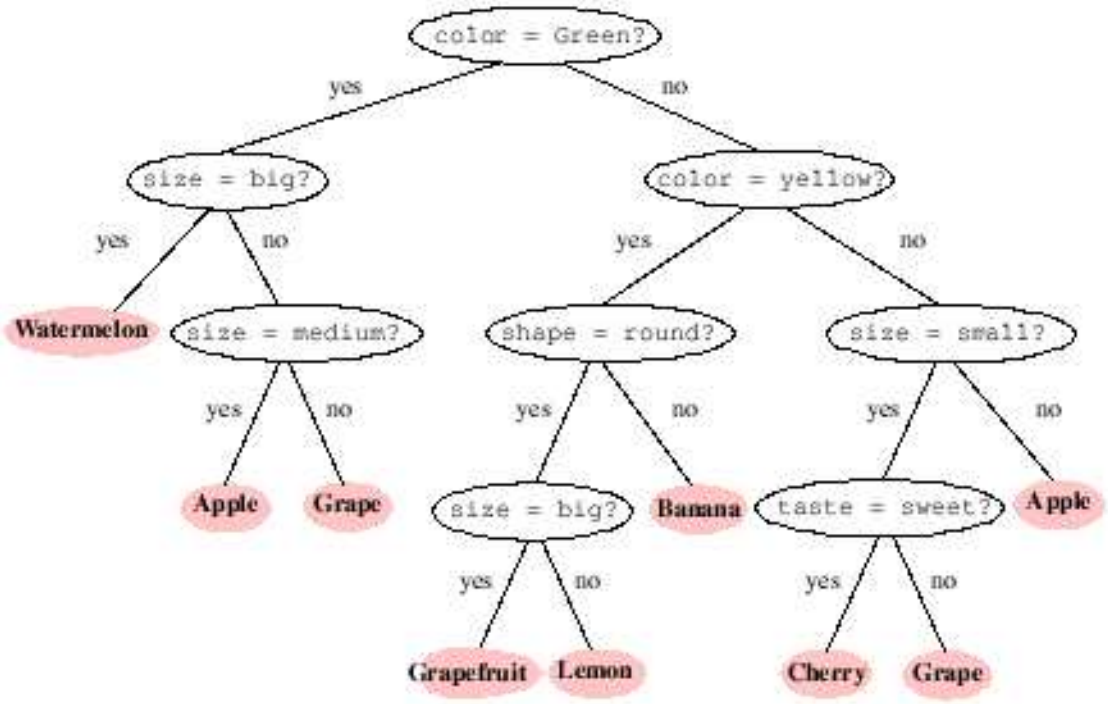
Edasises leiavad käsitlemist peamised meetodid ülalmainitud küsimustele optimaalsete lahenduste leidmiseks.

2.1 Valikute arv puu tipus

Küsimus sellest, mitmeks alamtipuks vaadeldav puu tipp peaks jagunema, sõltub peamiselt, millise kriteeriumi järgi sobivaid lahutusi otsitakse. Mitmeseid jagunemisi on võimalik alati kirjeldada kombinatsiooniga kahekaupa jagunemistest. Seetõttu kasutatakse lihtsuse huvides tihtipeale just tipu jaotamist kaheks, mille tulemusena tekib otsustuspuuna kahendpuu (vt joonist 2). Kahendpuu kasutamine muudab ka algoritmide konstrueerimise tunduvalt lihtsamaks.

2.2 Järgmise jagunemise valimine

Tüüpiliselt otsitakse puu tipus iga tunnuste jaoks parimat õpiandmete jagunemist ning lõpuks valitakse välja tunnus, mille parim jagunemine on teiste tunnustega võrreldes omakorda parim. Edasises vaatame, kuidas oleks mõistlik jagunemise *headust* hinnata.



Joonis 2: Otsustuspuu kahendpuuna

2.2.1 Infosisaldus, entroopia

Enimkasutatavad meetodid lahutuse valikul otsivad kõige informatiivsemat lahutust, so. lahutust tunnuse baasil, mis eraldaks klasse kõige paremini. Olgu meil vaadeldavas tipus õpiobjektide hulk C, kuhu kuulub $|\omega_1|$ objekti klassist ω_1 ja $|\omega_2|$ objekti klassist ω_2. Tipu ebapuhtust (ik *impurity*) saame hinnata näiteks järgmise, infosisaldust kirjeldava valemiga:

$$I(C) = -(\frac{|\omega_1|}{|C|}\log_2\frac{|\omega_1|}{|C|} + \frac{|\omega_2|}{|C|}\log_2\frac{|\omega_2|}{|C|}),$$

mida võib n klassi korral üldistada valemiks

$$I(C) = -\sum_{j=1}^{n} P(\omega_j)\log_2 P(\omega_j),$$

kus $P(\omega_j)$ tähistab klassi ω_j objektide osakaalu tipus C. Kirjeldatud entroopiafunktsiooni $I(C)$ väärtus on niisiis võrdne 0-ga, kui tipus on vaid ühe klassi objektid. $I(C)$ väärtus on seda suurem, mida võrdsemalt on hulgas esindatud erinevate klasside objektid.

Ülaltoodud meetod annab võimaluse hinnata ebapuhtust (klassifi tseerimisviga) ühes konkreetses puu tipus. Loogiline on eeldada, et mida kaugemale puu läbimisega jõuame, seda täpsemaks peab klassifi tseerimine muutuma ning andmete lahutus klassidesse peab pärast iga sammu olema eelnevast *puhtam*.

Seega paistab mõistlik valida järgmine tipu-küsimus niimoodi, et selle järgi õpiandmete jagunemine kutsuks esile võimalikult suure kasvu klassifi tseerimise täpsuses. Entroopia vähenemise läbi defi neeritud klassifi tseerimistäpsuse paranemist binaarse lahutuse puhul kirjeldab:

$$\Delta I(s) = I(C) - P(C_{vasak})I(C_{vasak}) - P(C_{parem})I(C_{parem}).$$

ehk kaalutud keskmise entroopia kahanemine liikudes puu järgmisele tasemele. Üldistatuna mitmeseks (so. B alamtipuliseks) lahutuseks kujuneb sama valem:

$$\Delta I(s) = I(C) - \sum_{i=1}^{B} P(C_i)I(C_i).$$

Tasub täheldada, et kui teha puu tipus täielik hargnemine, so. jagada iga õpiobjekt eraldi alamtippu, saame selle valemi põhjal järgmisel tasemel entroopia võrdseks nulliga ning täpsuse kasvu maksimaalse. Samas ei ole tavaliselt selline

täielik lahutus otsustuspuu juures loomulik ega kirjeldav. Kuna funktsioon soosib põhjendamatult võimalikult mitmese hargnemisega lahutusi, kasutatakse mitmese hargnemisega puudes selle modifi katsiooni:

$$\Delta I(s) = \frac{I(C) - \sum_{i=1}^{B} P(C_i)I(C_i)}{-\sum_{i=1}^{B} P(C_i)log_2P(C_i)}.$$

Maksimiseerides $\Delta I(s)$ üle kõikvõimalike tunnuste kaupa koostatud jagunemiste C-s, õnnestub valida küsimus, mis kõige paremini täpsustab tipu objektide klassikuuluvust.

Põhjalik näide entroopia vähenemise hindamiseks on toodud allikas (Quinlan 1986).

2.2.2 Gini indeks

Alternatiivina entroopia mõõtmisele puu tipus, kasutatakse ebapuhtuse hindamiseks ka klasside osakaalu paarikaupa *varieeruvust*, mida kahe klassi juhul väljendab nende koosesinemise sagedus tipus

$$Gini(C) = P(\omega_1)P(\omega_2)$$

ning mis üldistub mitme klassi juhul *Gini indeksiks*

$$Gini(C) = \sum_{i \neq j} P(\omega_i)P(\omega_j) = 1 - \sum_{i} P^2\omega_i.$$

Gini indeksi funktsioon käitub sarnaselt entroopiafunktsiooniga, kuid kasvab aeglasemalt ning võimaldab veidi paremini vahet teha halbade ning veidi vähem halbade jagunemiste vahel.

2.2.3 Muud tunnuse valiku kriteeriumid

Tunnuse valiku kriteeriumeid on pakutud hulgaliselt, lisaks entroopial ja ebapuhtusel põhinevatele mõõtudele, on kasutatud ka kaugusmõõte ning erinevaid statistikuid (nt χ^2). Praktilised tulemused (Murthy 1998) vihjavad, et pole suurt vahet, millise kriteeriumi (entroopia, Gini, χ^2) järgi puu konstrueerimisel tipu jagunemisi valida. Tunduvat suuremat rolli mängivad tingimused peatumiseks ning meetodid puu tagasilõikamiseks.

2.3 Otsustada klass või jätkata täpsustamist?

Eelpool kirjeldatud algoritmi kohaselt pole keeruline koostada täielikku puud, kus halvemal juhul igale lehele vastab üks õpiobjekt. Sellise puu miinuseks on, et kuigi õpiandmeid kirjeldab ta hästi, siis rakendatuna klassifi tseerimise eesmärgil tundmatutele objektidele osutub täielik puu liiga spetsiifi liseks. Üleõpetatud (ik *overfitted*) puu ei suuda teha üldistavaid otsuseid objektide kohta, mille tunnuste kombinatsioone õppimises ei kasutatud. Halvimal juhul käitub saadud otsustuspuu mugandatud paisktabelina.

2.3.1 Ristkontroll

Ristkontrolli meetodit (ik *cross-validation*) kasutatakse ka teiste klassifi tseerimistehnikate üldistamisvõime testimiseks. Väikesest osast õpiandmetest (näiteks 10%) kasutatakse testandmetena - neid puu konstrueerimise faasis ei kasutata. Valmis puud testitakse seejärel varem kõrvale pandud testandmete peal, et näha, kas saadud puu suudab klassifi tseerida vaid õpiobjekte või on see piisavalt korrektne ka tundmatute objektide korral.

Tihti jagataksegi õpihulk juhuslikult 10-ks ning konstrueeritakse 10 erinevat otsustuspuud, valides iga kord erineva test-kümnendiku. Edasi hinnatakse iga puu klassifi tseerimisviga oma testandmetel ning edasitöötamiseks valitakse vähima veaga puu.

2.3.2 Peatumise lävi

Teine võimalus on määrata parameetriga minimaalne lehe suurus (õpiobjektide arv lehes) enne puu konstrueerimise alustamist ja/või minimaalne informatiivsuse kasv (ebapuhtuse vähenemine) tipu jagunemisel. Esimesel juhul peatutakse puu konstrueerimisel automaatselt, kui vaadeldavas puu tipus on k või vähem õpiobjekti. Teisel juhul peatutakse siis, kui ükski vaadeldav jagunemine ei annaks etteantud lävest kõrgemat puhtuse kasvu $\Delta I(s)$. Keeruliseks teeb nende meetodite kasutamise just see, et sobiva parameetri leidmine enne puu konstrueerimist ei ole lihtne.

2.3.3 Vähim kirjelduspikkus (MDL)

Alternatiivne võimalus saadud puu hindamiseks on kasutada vähimale kirjelduspikkusele sarnanevat mõõtu:

$$mdl(T) = \alpha \cdot T_{size} + \sum_{C \in T_{leaves}} I(C)$$

Toodud funktsioon võtab korraga arvesse nii puu suurust kui ebapuhtust (entroopiat) puu lehtedes. On selge, et puu kasvatamise hinnaga saame parendada klassifi tseerimistäpsust lehtedes. Funktsiooni $mdl(T)$ järgi minimaalne puu on seega mõlemas mõttes võimalikult optimaalne puu. Kaaluga α saame märkida, kui kulukaks peame puu suurust tema klassifi tseerimistäpsuse suhtes.

2.4 Tagasilõikamine ehk kärpimine

Peatumise kõrval on teine võimalus otsustuspuu kõrguse optimeerimiseks kasutada tagasilõikamist (ik *pruning*). Peatumisega jagunemise puudus on muuhulgas see, et ei suudeta näiteks ära kasutada ülejärgmisel sammul võimalikuks saavat head jagunemist, kuna järgmisel sammul on ka kõige parem jagunemine peatumiskriteeriumite mõistes ebapiisav jätkamiseks.

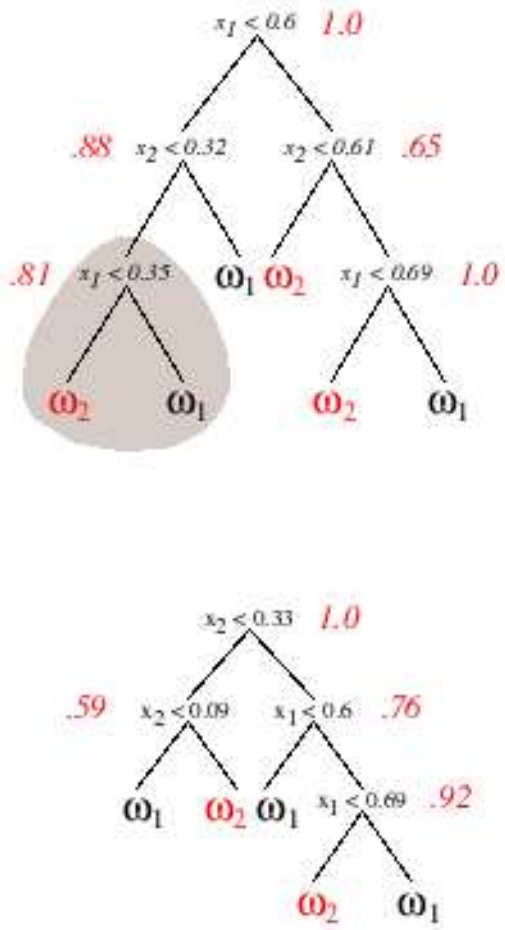
Tagasilõikamise kasutamisel konstrueeritakse esmalt täielik otsustuspuu, kus lehtedes on täpne klassijaotus (minimaalne entroopia). Seejärel hakatakse alt-üles kontrollima naaberlehti - kas nende ühendamisel tekib piisavalt väike klassifi tseerimistäpsuse kadu? Kui jah, siis tipud ühendatakse (vt. joonist 3).

Otsustuspuu optimeerimine tagasilõikamise abil on muidugi arvutuslikult tunduvalt mahukam, kui lihtsalt õigel hetkel peatumine. Samas väiksemate ülesannete puhul eelistatakse just tagasilõikamist, kuna annab paremini klassifi tseerivaid puid kui peatumisega optimeerimine (Duda, Hart, & Stork 1997).

2.4.1 Reeglitega tagasilõikamine

Otsustuspuu lehti võib vaadata kui reegleid - teel juurest vaadeldava leheni asuvate üksikute otsuste konjunktsiooni. Kirjutades iga lehe jaoks välja tema *reegli* ning ühendades need disjunktsioonide abil klasside reegliteks võib üritada reegleid lausearvutusreeglite jms. abil lihtsustada.

Sel moel on võimalik vähendada otsustuspuu kirjelduspikkust, kuid üldistusvõime parendamiseks (mis on tagasilõikamise peamine eesmärk) tasub ristkontrolli kasutades uurida, kuidas mõjutab mõne reegli eemaldamine puu klassifi t-



Joonis 3: Tagasilõikamine lehtede ühendamisega

seerimistäpsust. Reeglite abil on võimalik "välja lõigata"mõni ebaoluline tipp ka juurtipu lähedalt.

2.5 Müra ja puuduvad tunnused

Mittesüstemaatiliste vigade e. müraga õpiandmete puhul ei pruugi olla ka kõige parema tahtmise juures võimalik konstrueerida täielikult korrektset otsustuspuud. Konflikti esile kutsumiseks piisab, et õpiandmetesse satuks kaks erinevast klassist kuid täpselt samade tunnustega kirjeldatud objekti. Müra ja puuduvate tunnuste problemaatikaga on tõsiselt tegelenud ID3 ja C4.5 algoritmide autor J.R. Quinlan ning täpsemalt tasub uurida (Quinlan 1986) ning tema uuemaid artikleid.

2.5.1 Müra

Lisaks eelpool kirjeldatud peatumise kriteeriumitele ja tagasilõikamise meetodile, mis lisavad puule üldistavust, kasutatakse müraga toimetulemiseks tunnuste eelanalüüsi.

Jaotagu tunnus A tipu C alamtippudeks $C_1, \ldots, C_m$, siis klassi ω jaoks on oodatav esinemistõenäosus tipus C_i defineeritav

$$P'_i(\omega) = P(\omega) \cdot \frac{|C_i|}{|C|}.$$

Kahe klassi juhul võime uurida χ^2 põhinevat statistikut

$$\sum_{i=1}^{m} \frac{(P_i(\omega_1) - P'_i(\omega_1))^2}{P'_i(\omega_1)} + \frac{(P_i(\omega_2) - P'_i(\omega_2))^2}{P'_i(\omega_2)}$$

ning väita vastava kindlusega, et C objektide klassikuuluvus sõltub atribuudi A väärtusest. Edasi võib seada otsustuspuu konstrueerimisel läve, et näiteks ükski otsustuspuus kasutatav tunnus ei tohi olla statistiku järgi enam kui 95% χ^2-kindlusega sõltumatu klassikuuluvusest. Tunnuste eelanalüüs vältimaks müraga tunnuste kasutamist puu konstrueerimisel aitab praktikas saada otsustuspuid, mis toimivad müravabadel andmetel sarnaselt, kui olles rakendatud müraga andmetele.

Eksperimentaalsetes katsetes (Quinlan 1986) on tuvastatud, et oluline on kasutada reaalsusega sarnase müratasemega õpiandmeid. Katsed näitavad, et korrektsete (ilma mürata) õpiandmete kasutamisel klassifitseerib puu hiljem reaalseid andmeid tunduvalt halvemini, kui seda teeb otsustuspuu, mis konstrueeriti tõepärase müraga õpiandmete järgi.

2.5.2 Puuduvad tunnused

Objektidel võib olla puuduvaid tunnuseid nii õpiandmetes kui ka hiljem klassifitseeritavates andmetes. Mõlema juhu arvestamiseks tuleb puu konstrueerimise algoritmi veidi modifitseerida.

Kui õpiandmetes esineb puuduvaid tunnuseid, siis levinud strateegia on järgmise tipu-küsimuse valimise ajal täielike tunnuste korral käituda tavalisel viisil. Tunnuste puhul, mis mõnel õpiobjektil puuduvad, arvutada hinnangud aga neid objekte lihtsalt arvestamata. Kui kasulikuks osutub valida mittetäielik tunnus, võib järgmisel sammul kasutada puudunud tunnusega objekte juba tipu kõikides alamtippudes mõne teise tunnuse järgi ülejärgmise jagunemise hindamiseks.

Teine lähenemine on üritada leida puuduva väärtuse asemele kõige tõenäolisim väärtus. Huvitava lahenduse pakkus Alen Shapiro (Quinlan 1986), kes soovitas ehitada iga puuduva attribuudi selgitamiseks eraldi otsustuspuu, kus õpiandmete klassitunnused on võetud attribuutidele lisaks ning otsitav puuduv attribuut klassitunnuseks. Tunnust x_i tuvastav otsustuspuu oleks treenitud jagama puuduvate tunnustega andmevektorid nii mitmesse klassi, kuipalju on vaadeldaval tunnusel x_i võimalikke väärtusi. Kui esialne otsutuspuu on väljendatav valemiga

$$Tree(x_1, \ldots, x_i, \ldots, x_n) = \omega_x,$$

siis puuduva tunnuse x_i otsimiseks võime genereerida samadelt andmetelt

$$Tree_i(x_1, \ldots, \omega_x, \ldots, x_n) = x_i.$$

Samuti peab õpiprotseduuri veidi kohandama sellisele praktikas levinud juhule, kus ka *klassifitseeritavates* andmetes võib esineda puuduvaid tunnuseid. Puuduvate tunnuste toetamiseks kasutusfaasis genereeritakse igas tipus lisaks primaarsele tipu-küsimusele veel üks või kaks asendusküsimust teiste tunnuste järgi. Sobivate asendusjagunemiste (ik *surrogate split*) hindamisel otsitakse sellist jagunemist, mis võimalikult palju lähendaks primaarset jagunemist, so. jaotaks võimalikult palju õpiobjekte samasse alamtippu kui primaarjagunemine.

3 Levinud algoritmid

3.1 ID3

J.R. Quinlan-i ID3 algoritmi tutvustamist 1986. aastal (Quinlan 1986) peetakse siiani otsustuspuude sünniks, kuigi sarnaseid konstruktsioone oli ekspertsüsteemi-

des kasutatud varemgi. ID3 oli algselt disainitud nominaalsetele järjestuseta tunnustele ning kasutab tippudes jagunemise valikul infosisalduse vähendamise põhimõtet. Iga tunnuse jaoks määratakse eelnevalt hargnemiskordaja, so. selle tunnuse järgi lubatud hargnemiste arv suvalises tipus. Reaalarvuliste väärtustega tunnused jagatakse diskreetseteks intervallideks ning kasutatakse neid samuti nominaalsete tunnustena. ID3 algoritm ei sisaldanud algselt ei peatumist ega ka tagasilõikamist.

3.2 C4.5

C4.5 on Quinlan-i edasiarendus ID3 algoritmist, kuhu on lisatud tugi reaalarvuliste tunnuste kasutamiseks, täielike puude tagasilõikamiseks ning lausearvutusvalemite kujul reeglite genereerimiseks puult.

Lisaks tavalisele lehtede ühendamise baasil tagasilõikamisele on C4.5-s võimalik ka reeglite lihtsustamise abil puu kärpimine. C4.5 algoritm ei toeta aga puuduvate tunnuste töötlemiseks mõeldud asendusjagunemisi tippudes. Selle asemel läbitakse puuduva tunnuse korral kõik alamtipud ning otsustatakse igast tipust saadud tulemusi võrreldes sobivaim klassitunnus.

C4.5 algoritmi C realisatsioon on vabalt (kuid mitte edasilevitamiseks) saadaval Quinlan-i enda veebilehelt ning asendanud praktikas ID3-e täielikult. Algoritm on eelistatud juhtudel, kus soovitakse otsustuspuu mahtu minimeerida.

3.3 CART - Classification and Regression Trees

CART on Breiman jt. (Brieman *et al.* 1984) poolt välja pakutud üldine raamistik puu kasvatamiseks õpiandmetelt. Etteantud raamistikus võib kasutada erinevaid hinnangufunktsioone sobiva tipu-jagunemise valimiseks, valida kriteeriumeid peatumiseks ning valmis puude tagasilõikamiseks. CART-i abil saab ehitada lihtsa vaevaga erinevatel alustel mitmeid puid ning võrdluses välja tuua enda ülesande tarvis parim. Kuna pole välja kujunenud algoritmi, mis iga kasutusjuhu jaoks parima puu ette valmistab, saadakse adekvaatseimad tulemused just paljude erinevate konstruktsioonide läbi proovimisel.

3.4 Arvutuslik keerukus

Kuigi otsustuspuude konstrueerimiseks on palju erinevaid algoritme ning nende modifi katsioone, võib eeskirja tüüpilist arvutuskeerukust hinnata komponentidest: ühekordne sorteerimine iga tunnuse järgi ning igakordne infosisalduse arvutus tipu-küsimuse valimisel.

Infosisalduse arvutuse keerukus juurtipus on andmete dimensionaaluse d juures $O(n) + (n-1)O(d)$, sest läbi tuleb kontrollida kõik jagunemised d tunnuse järgi. Sarnaselt juurtipu keerukusele $O(dn \log n)$ võib avaldada k-nda taseme tipu arvutuskeerukuse $O(dn \log \frac{n}{k})$ ning summeerides üle $\log n$ taseme, saame otsustuspuu konstrueerimise kogukeerukuseks $dn \log^2 n$.

Kuna otsustuspuu kasutamine seisneb puu ühekordses läbimises, on puuga klassifi tseerimise keerukus $O(\log n)$ õpiandmete hulga (täpsemalt puu suuruse) suhtes.

On teada, et optimaalseima otsustustasandi tuvastamine õpiandmetest on üldiselt NP-täielik ülesanne. Samas tüüpiline otsustuspuu konstrueerimise algoritm töötab lineaarses- või ruutajas õpiandmete hulga suhtes ning annab seejuures ideaalilähedasi tulemusi.

4 Erikujulised puud ja eksperimentaalsed meetodid

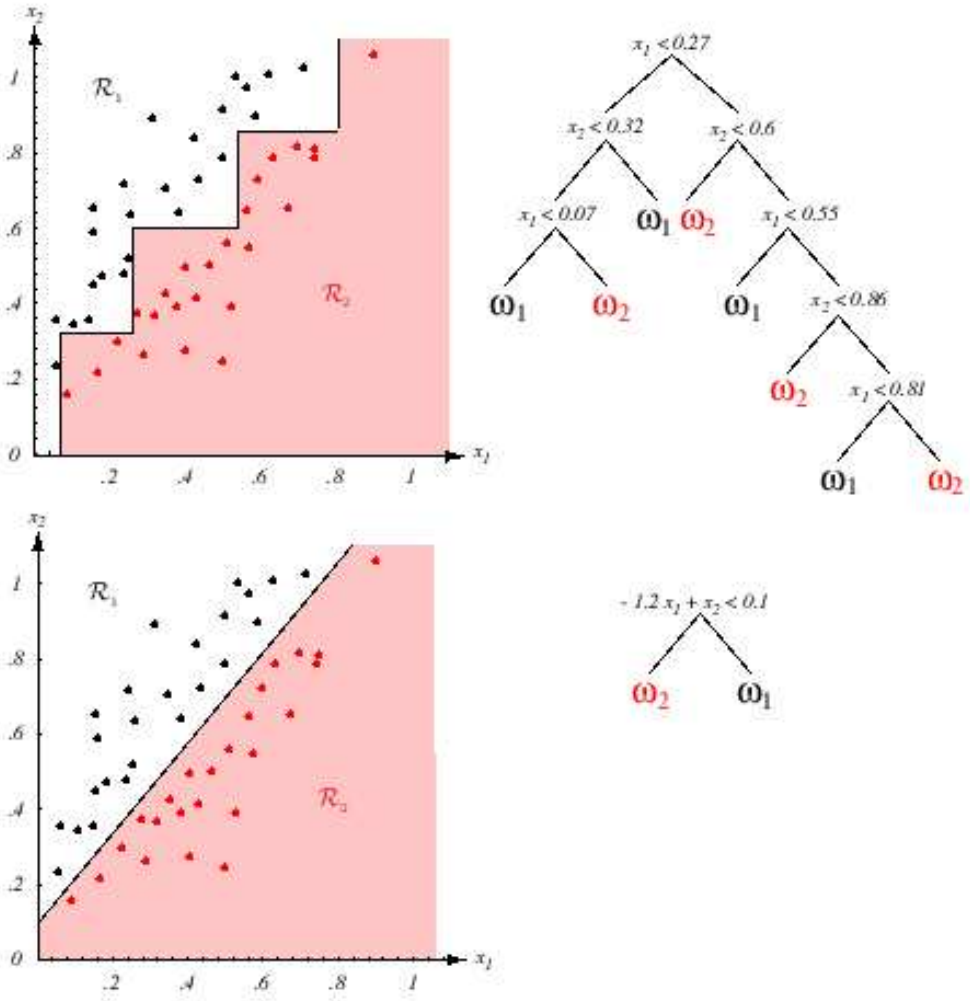
4.1 Mitmemuutuja lahutusega puu

Enamik praktikas kasutatavaid otsustuspuid valivad igal sammul vaid ühe tunnuse, mille järgi õpihulka alamtippudesse jagada. Kuigi see on arvutuslikult palju mahukam, võiks otsida igas tipus ka parimat tunnuste paari või kolmikut. Veelgi üldistatuna võib igas tipus otsuse tegemiseks kasutada kõigi tunnuste lineaarkombinatsiooni ning saada mitmemuutuja lahutusega puu (ik *multivariate tree*) (Brodley & Utgoff 1992) (vt. joonist 4). Praktikas pole mitmemuutuja lahutusega puude laialdane kasutamine siiski empiiriliselt põhjendatud, kuna nende konstrueerimise ressursinõudlikus ei too piisavat kasvu klassifi tseerimistäpsuses.

4.2 Inkrementaalne treenimine

Kui võrrelda otsustuspuude "kasvatamist"neurovõrkude õpetamisega, siis otsustuspuu tüüpiliselt kasvatatakse lõpuni välja, testitakse, muudetakse sisendparameetreid ja õpiandmete valikut ning kasvatatakse järgmine. Sellisel moel itereeritakse mõned korrad ja valitakse välja kõige paremini klassifi tseeriv puu. Neurovõrkude puhul seevastu ei ehitata võrku igal õppimise iteratsiooni sammul uuesti üles, vaid *kohandatakse* jooksvalt kaalusid neuronite vahel.

Sarnast ideed on üritatud katsetada ka otsustuspuude juures, et võimaldada puudele pidevat kohandumist juhul kui õpiandmeid tuleb pidevalt juurde. Hea näide on ilma või mõne muu nähtuse ennustamine, kus klassifi tseeritud andmeid



Joonis 4: Mitmemuutuja lahutusega puu

tuleb jooksvalt juurde ning kannavad potentsiaalselt uut teadmist. Otsustuspuude täiendamise temaatiga on tegelenud Utgoff (Utgoff, Berkman, & Clouse 1997) ning pakkunud välja vastavad meetodid.

4.3 Hägusad puud

Sügaval otsustuspuus asetsevas tipus ei pruugi olla enam piisavalt õpiandmeid, et tehtav otsustus oleks piisavalt suure *tähendusega* (ik *significance*). Sellisel juhul on võimalik teha *osaline jagunemine*, kus õpiandmeid ei jaotata eralduvatesse hulkadesse, vaid mõned õpiobjektid võivad edasi minna mitmesse alamtippu. Sarnast konstruktsiooni kasutatakse C4.5 puuduvate tunnuste probleemiga toimetulemiseks.

4.4 Puud andmete kirjeldamiseks

Käesoleva kirjatüki autori huvi otsustuspuude valdkonnas on äratanud otsustuspuude kirjeldav omadus. Otsustuspuu ehitamise käigus leitakse igale andmete klastrile (sarnase klassitunnusega lähedased objektid) andmeruumis sobivaim kirjeldus dimensioonide järgi. Ühendatuna k-naabrite klasterdusmeetodiga võib otsustuspuulaadne jagunemiste hindamine olla efektiivne viis saada andmekogu *informatiivseim* kirjeldus. Praktiline kasutus võib olla dimensionaalsete andmete põhjal kõikidele nominaalseletele atribuutidele ükshaaval sobivaima (kõige paremini kirjeldava) hierarhia ehitamine.

Samuti võiks ülalt-alla meetodit kaaluda hierarhilise klasterduse kontekstis, mis võimaldaks asendada väikeste lähedaste klastrite kahekaupa ühendamise "pikemate"ja informatiivsemate sammudega hierarhias.

Selles valdkonnas ei ole autoril viiteid tehtud töödele, vaid ainult mõned ideed otsustuspuude metoodika rakendamisek teistes praktilistes andmeanalüüsi valdkondades.

5 Rakendused ja tulemused

Hea kokkuvõtte otsustuspuude kasutusest paljudes eluvaldkondades annab allikas (Murthy 1998). Siinkohal tasub vaid mainida, et levinud kasutusalad hõlmavad valdkondi, kus ajalooliselt on populaarsed olnud ekspertsüsteemid (nt. diagnostika). Otsustuspuude eeliseks (nt. neurovõrkude ees) on asjaolu, et otsustuspuudest ning ekspertteadmistest saab moodustada hübriidsüsteeme, kus ekspertide roll on

vaid õpitulemuste valideerimine ja vajadusel täiendamine. Uuemad valdkonnad otsustuspuude kasutamiseks on molekulaarbioloogias (Salzberg *et al.* 1998) ja pilditöötluses.

Üks paljutõotav tulemus otsustuspuudele ning masinõppimisele tervikuna saadi Giplini poolt (Murthy 1998), kes võrdles CARTi, lineearset diskriminantanalüüsi ning logistilist regressiooni kolme kogenud kardioloogiga. Ülesandepüstituseks oli otsustada, kas patsient sureb aasta jooksul pärast müokardiinfarkti läbielamist või jääb elama. Katse tulemusena tõdeti, et masinate ning arstide prognoosidel polnud mingit tähelepandavat erinevust.

Masinõppimise meetoditega *hands on* tutvumiseks tasub vaadata Weka projekti, mis pakub avatud koodiga raamistiku Javas ning hulgaliselt erinevaid andmekaevanduse algoritmide implementatsioone.

6 Kokkuvõte

Hierarhiline otsuste puu rekursiivne konstrueerimine on end tõestanud mitmete reaalelu probleemide lahendamisel. Üheks otsustuspuude poplaarsuse põhjuseks ning eeliseks teiste klassifi tseerimismeetodite ees on nende intuitiivne arusaadavus ning kasutuslihtsus. Sellest hoolimata ei suuda iga puu igas kontekstis mõistlikku tulemust anda ning ka otsustuspuuga klassifi tseerimisel tuleb omada piisavaid teadmisi ülesande domeenist ning ülevaadet otsustuspuude erinevatest meetoditest.

Viited

Brieman, L.; Friedman, J.; Olshen, R.; and Stone, C. 1984. *Classification and Regression Trees*. Wadsworth and Brooks.

Brodley, C. E., and Utgoff, P. E. 1992. Multivariate decision trees. Technical Report UM-CS-1992-083.

Duda, R.; Hart, P.; and Stork. 1997. *Pattern Classification*.

Murthy, S. K. 1998. Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery* 2(4):345–389.

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

Salzberg, S.; Delcher, A. L.; Fasman, K. H.; and Henderson, J. 1998. A decision tree system for finding genes in DNA. *Journal of Computational Biology* 5(4):667–680.

Utgoff, P. E.; Berkman, N. C.; and Clouse, J. A. 1997. Decision tree induction based on efficient tree restructuring. *Machine Learning* 29(1):5–44.