ODLOČITVENA DREVESA IN SISTEMI Z VEČDIMENZIONALNIMI REŠITVAMI

Matej Šprogar, Vili Podgorelec, Peter Kokol
Fakulteta za elektrotehniko, računalništvo in informatiko
Univerza v Mariboru, Smetanova 17, 2000 Maribor
matej.sprogar@uni-mb.si

Povzetek

Pri sprejemanju odločitev je današnjemu strokovnjaku lahko v veliko pomoč sistem za podporo odločanju, ki je sposoben pregledati ogromne količine podatkov. Odločitveni sistem mora biti zanesljiv in tudi dovolj sposoben, da zadosti vsem zahtevam. Članek predstavlja razširitev koncepta odločitvenih dreves na multidimenzionalna – vektorska – drevesa, ki so sposobna dati odgovor na več različnih vprašanj hkrati, in evolucijski pristop h gradnji takšnih dreves. Vektorsko drevo pomeni zlitje več klasičnih odločitvenih dreves, ki delujejo nad isto množico atributov, a dajejo odgovore na različna vprašanja. Takšno drevo je enostavnejše za uporabo, lažje ga je analizirati in izraža možne povezave med končnimi odločitvami, ki drugače niso vidne. Predstavljeno je orodje DecRain, ki gradi vektorska odločitvena drevesa s pomočjo genetskih algoritmov.

Abstract

In the decision-making process the human expert will always appreciate help from a decision support system, which can examine huge amounts of data. Such a decision system should be reliable and should satisfy all the demands it might face. The article presents the extension of a common decision tree concept to a multidimensional – vector – decision tree. Contrary to the common decision tree the vector decision tree can answer more than just one question at a time. It has the functionality of many separate decision trees acting on a same set of data and answering different questions. Vector decision tree is simpler in its form, it is easier to use and investigate and can express some relationships between decisions not visible before. DecRain is a special tool for the vector decision tree induction. In the article both the DecRain tool and its evolutionary approach to the vector decision tree induction are presented.

■ ■ ■

Uvod

Čeprav smo že krepko v informacijski dobi, so številne naloge še vedno v domeni človeka. Ko gre za ključne odločitve, kjer ne sme biti napake, so neprecenljive prav izkušnje in osebni občutek posameznika. Informatika v takih primerih naleti na težko oviro, ki bi jo lahko premostila le s pomočjo razvoja inteligentnih programov, sposobnih učenja in razmišljanja na človeku podoben način. Inteligentni sistemi so zaenkrat še utopična želja, poznamo pa nekaj njihovih lastnosti, ki jih lahko uporabimo pri načrtovanju sistema za pomoč pri odločanju. Tak sistem bi se po eni strani naslanjal na rešitve, ki izhajajo iz klasične informacijske teorije, po drugi pa na popolnoma drugačne koncepte programiranja, ki odpirajo nove poglede na problemsko področje. S tem bi premostili določene pomanjkljivosti obstoječih informacijskih rešitev, ki pomagajo človeku v poplavi podatkov, niso pa sposobne dejansko razumeti obdelanih dejstev in izluščenih relacij in jih tudi aktivno uporabiti pri novem problemu. Klasični princip torej temelji na programsko kodiranih načelih, ki so nespremenljiva in absolutna in prav zato tudi močno omejujoča.

Da bi premostili to absolutnost, je potrebno spremeniti ne le metodologijo, ampak tudi princip programiranja. Pri velikih problemih namreč ni dovolj le hitro izvajanje ukazov, ampak predvsem način iskanja rešitev. Tako je poglavitna težava največkrat prav velikost problemskega področja in število možnih kombinacij in odvisnosti, ki jih je enostavno nemogoče zajeti v deterministični algoritem. Kot alternativa so se pojavili evolucijski sistemi, ki črpajo ustvarjalno moč iz naravnih načel reprodukcije in selekcije. Kombinacija z določenim odločitvenim modelom nam tako lahko da izredno dober in učinkovit sistem za pomoč pri odločanju.

Vloga sistema za pomoč pri odločanju je danes predvsem svetovalna, odločitveni sistem pa bi naj ob predlagani rešitvi nudil tudi razlago oz. utemeljitev odločitve, kar bi dodatno pomagalo strokovnjaku. Tako lahko človek lažje preveri smiselnost odločitve, ugotovi pomanjkljivosti ali nelogičnosti, pa tudi dobi

nova spoznanja. Področja uporabe takšnih sistemov so seveda izredno široka, predvsem so sistemi za podporo odločanju primerni tam, kjer morajo biti odločitve sprejete hitro in z določeno stopnjo zanesljivosti. Za takšne naloge so se kot zelo dobra izkazala tudi odločitvena drevesa, saj so konceptualno zelo preprost model z možnostjo avtomatskega učenja [1]. Drugače uspešen klasični način gradnje odločitvenih dreves ima nekaj pomanjkljivosti, ki pa se jih da elegantno odpraviti prav z uporabo evolucijskih tehnik in genetskih algoritmov [3].

Ker genetski pristop k izdelavi odločitvenih dreves odpira novo perspektivo in ne postavlja nobenih omejitev za iskano rešitev, je postalo zanimivo tudi hkratno odločanje o več različnih vprašanjih in primerjava takšnih rezultatov z drugimi drevesi in tudi drugimi sistemi odločanja. Klasični načini gradnje dreves so namreč dokaj omejujoči glede obsežnosti rešitve, kar je bil pogosto tudi razlog za izbiro katerega drugega odločitvenega modela, kot so npr. nevronske mreže. Z možnostjo hkratnega odločanja o več vidikih problema pa bi to težavo ne samo odpravili, ampak celo krepko presegli. V ta namen smo razvili sistem za generiranje odločitvenih dreves, ki je sposoben izdelati odločitvena drevesa z izhodom v obliki vektorja več odločitev namesto klasično ene same odločitve. Upali smo, da bo takšen sistem izkazal dodatne lastnosti, kot jih imajo npr. nevronske mreže, in imel hkrati vse prednosti odločitvenih dreves. Prvi rezultati so zelo vzpodbudni.

Odločitvena drevesa

Odločitvena drevesa spadajo med metode induktivnega učenja, torej avtomatskega učenja iz rešenih primerov. Cilj je odločitvena struktura - drevo, ki temelji na učnih primerih in je sposobno čim bolj uspešno "rešiti" še neznani primer. Sam proces učenja je zajet v oblikovanju dreves, saj odločitvena drevesa hranijo izluščeno znanje prav v svoji obliki. Samo drevo sestoji iz dveh vrst vozlišč – atributna (testna) vozlišča so testi vhodnih vzorcev, končni listi pa so kategorije testiranih vzorcev. Drevo bo vhodni vzorec klasificiralo s sprehodom od začetnega vozlišča do končnega lista s sprotnim odločanjem v vsakem vozlišču o nadaljnji smeri sprehoda. Sama drevesa se ločijo po številu testov, ki jih lahko opravimo v enem vozlišču, po številu možnih odgovorov in tudi po številu razredov, kamor lahko drevo uvrsti neki vzorec. Sami testi v notranjih vozliščih se lahko izvajajo nad numeričnimi ali diskretnimi vrednostmi in so torej ustrezne oblike, vsak list pa predstavlja možni odgovor na zastavljeno vprašanje. Glavni odliki dreves sta njihova preprostost za uporabo in izraznost rešitve ter naučenega znanja – zgradba drevesa je osnova za napoved, v kateri razred neki vzorec spada in katera vprašanja so pri tej odločitvi pomembna.

Sama oblika odločitvenih dreves je zelo preprosta in nedvoumna, vendar pa pri klasični gradnji (učenju) nastopa kar nekaj problemov. Glavni problem je izbira vrstnega reda testov, če pa so testi še nad numeričnimi vrednostmi, je potrebno določiti tudi sam test. Za učenje odločitvenih dreves je bilo predlaganih kar nekaj metod in večina od njih je osnovana ali izpeljana iz ideje Quinlanovega ID3 algortima [6][7]. Ta pri izbiri vrstnega reda testov izbere vedno tisti test, ki maksimalno reducira neko na entropiji osnovano merilo. Takšno hevristično ocenjevalno funkcijo je možno uspešno razširiti iz binarnih atributov tudi na atribute z več vrednostmi [4], problemi pa se pojavijo pri klasifikacijah s širokim naborom razredov [5].

Klasična odločitvena drevesa kot vhod sprejmejo nabor vhodnih atributnih vrednosti, kot izhod pa v listu drevesa podajo eno samo odločitev. Zamišljena razširitev pa za razliko od tega kot izhod podaja vektor več ločenih odločitev, od katerih ima vsaka svoj nabor vrednosti. Takšno drevo torej klasificira neki vzorec ne samo po enem, ampak po več kriterijih hkrati. Razlika med vektorskimi in klasičnimi drevesi je torej le v končnih listih, ki podajajo odločitve, vsa ostala vozlišča pa so funkcionalno enaka. Slika 1 podaja primer preprostega vektorskega odločitvenega drevesa z dvema odločitvama. Atributna vozlišča postavljajo vprašanja o dejstvih, v listih dreves pa imamo naučeno odločitev. Atributi so v tem trivialnem primeru podatki o vremenu, učna množica pa so podatki o preteklih izkušnjah.

Slika 1: Preprost primer vektorskega drevesa z dvema odločitvama

Evolucijski model, genetski algoritmi

Evolucijski model povzema idejo naravne evolucije. Kot se skozi evolucijo razvijajo, izumirajo in spreminjajo biološke populacije, se tudi naše rešitve razvijajo po načelih »naravne selekcije«. Torej lahko z računalniškim tekom dejansko simuliramo naravne procese selekcije, razmnoževanja, tekmovanja... Z genetskimi algoritmi lahko najdemo rešitve tudi najbolj kompleksnih problemov, saj se za razliko od klasičnih iskalnih strategij problema lotijo na popolnoma svojevrsten način.

Genetski algoritem je preslikava iz narave v računalniški svet, kjer prav tako ustvarimo populacijo bolj ali manj dobrih rešitev znotraj iskalnega prostora. Populacijo predstavljajo posamezni osebki, opisani z genetsko kodo, ki določa njihove lastnosti. Tako lahko sprememba genetske kode osebka povzroči spremembo njegovih lastnosti, obnašanja, torej napredek ali tudi nazadovanje glede na iskane lastnosti. S pomočjo genetskih operatorjev selekcije, križanja in mutacije lahko potem razvijamo genetsko sliko osebkov v populaciji. S tem se hkrati premikamo po iskalnem prostoru, dokler ne najdemo točke – osebka, ki predstavlja zadovoljivo rešitev.

Osrednjo vlogo v evoluciji imajo genetski operatorji, ki zagotavljajo spreminjanje in v splošnem napredovanje osebkov skozi čas. Selekcija je način izbora osebkov za reprodukcijo in s tem tudi preživetje genetskega materiala v produciranih potomcih. Operator križanja proizvede nov osebek iz dveh staršev s pomočjo križanja njune genetske kode. Samo križanje mora glede na obliko osebka zagotavljati tudi logično pravilno končno genetsko sliko, ki bo definirala zdrav osebek, sposoben boja za obstanek. Občasno uporabljen operator mutacij pa zagotavlja vnos potrebnih nenadzorovanih sprememb, ki dajo populaciji nov zagon. Tako kot v naravi tudi tukaj genetski operatorji ne zagotavljajo vedno pridobitve novih kvalitet in imajo lahko tudi dokaj velik destruktiven učinek, kar pa je nujni del evolucije [12].

Združitev odločitvenih dreves in evolucijskih načel

Ker za gradnjo odločitvenih dreves obstajajo močne teorije [6][7], bi lahko celo opustili nadaljnje raziskave v tej smeri. Vendar je gradnja odločitvenih dreves kompleksen problem, ki ga ni mogoče vedno uspešno opisati in rešiti s statističnimi metodami. Težave lahko nastopijo pri pomanjkljivih ali napačnih podatkih, pri določanju atributov in ustreznih razmejitvenih intervalov. In če že uspemo rešiti te probleme, pa rezultati včasih enostavno niso dovolj dobri. Razlogi so potem lahko ali na strani gradnje dreves ali pa tudi v neprimernosti samih dreves za izbrano nalogo.

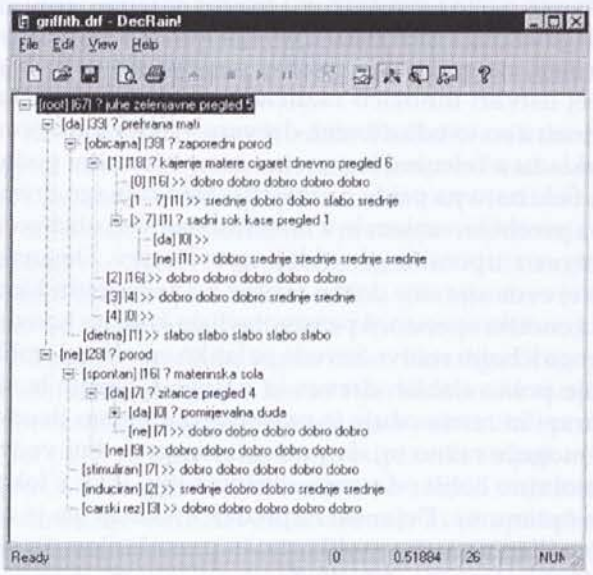
Ker pa imajo drevesa nekaj izjemnih lastnosti, se je bilo smiselno distancirati od statističnih teorij in poiskati alternativne načine za gradnjo dreves. V skladu z razmišljanji v uvodu smo zato skušali združiti dva svetova – preprost koncept odločitvenih dreves in sposobnost evolucijskega pristopa, da se loti še tako kompleksnega problema. Za razliko od klasičnega načina gradnje, ko učenje pomeni gradnjo enega samega drevesa od prvega vozlišča do zadnjega lista, pa evolucijski pristop postopa drugače. Najprej ustvari množico različnih osebkov – v našem primeru so to odločitvena drevesa – in jih nato oceni v skladu z želenimi lastnostmi in s poznanimi podatki. Šele nato pa pride na vrsto iskanje ciljnega drevesa s preoblikovanjem in z razmnoževanjem obstoječih dreves z uporabo genetskih operatorjev. Dejansko torej evolucija išče dobro rešitev na več mestih hkrati, genetski operatorji pa zagotavljajo končno konvergenco k boljši rešitvi. Seveda se lahko v procesu izoblikuje polno slabših dreves in naloga selekcije je, da ohrani in razmnožuje le najboljše. Zanimivo dejstvo je mogoče ravno to, da potomci ne smejo biti vedno absolutno boljši od svojih staršev, saj to vodi v lokalne optimume. Dejanski napredek evolucije pa je zagotovljen prav z raznolikostjo in prenašanjem ter kombiniranjem genetske kode.

Gradnja odločitvenih dreves s pomočjo genetskega pristopa je že dala odlične rezultate [3] in pokazala določene prednosti pred klasičnim pristopom. Z razširitvijo odločitvenega drevesa na večdimenzionalno – vektorsko – odločitev pa pridobimo na funkcionalnosti dreves. Takšna razširitev bi bila s klasično metodo težko izvedljiva, izvedba z genetskim pristopom pa ne daje nikakršnih omejitev v obliki in obsegu vhodnih podatkov ali izhodnih odločitev.

Orodje DecRain

Eno trenutno najbolj uporabljanih orodij za gradnjo klasičnih odločitvenih dreves je zagotovo Quinlanov C4.5 (in izboljšana verzija C5) algoritem [7], tam uporabljeni format pa nekakšna osnova za pripravo podatkov in predstavitev rezultatov. Pri izdelavi orodja DecRain smo zato posnemali preprostost C5 in poskusili narediti gradnjo vektorskih dreves čim bolj enostavno. Ker zahteva priprava podatkov največ časa in lahko količina podatkov krepko preseže pregledne sposobnosti tekstovnega urejevalnika, pa zahteva DecRain podatke iz podatkovne baze, ki podpira ODBC standard, s čimer danes razpolaga večina boljših podatkovnih baz tako pod Windows kot tudi Unix in Linux okolji. Takšna oblika hkrati zagotavlja lažjo obdelavo, spremembe, nadzor in prenosljivost podatkov. Zaradi primerljivosti pa smo naredili tudi možnost pretvorbe v format, kot ga zahtevata orodji C5 in orodje MtDecIt [4].

DecRain simulira evolucijo odločitvenih dreves, hkrati pa seveda omogoča dinamično nastavitev vseh paramentrov evolucije, popoln nadzor nad njenim potekom in prikaz rezultatov (statistični rezultati na učni in testni množici, prikaz najboljšega zgrajenega drevesa in tabelarični prikaz vseh rezultatov – Slika 2).



Slika 2: Okolje DecRain in primer zgrajenega vektorskega odločitvenega drevesa s petimi odločitvami.

DecRain uporablja načela genetskih algoritmov. Velikost izhodnega vektorja ustreza številu iskanih odločitev, če je vzorec možno kategorizirati na en sam način (velikost izhodnega vektorja je 1), pa bo končno odločitveno drevo funkcionalno enako klasično zgrajenemu drevesu. V splošnem lahko pričakujemo drevo podobnih kvalitet. Alternativa takšnemu pristopu je tudi klasična izgradnja ločenih dreves za vsako od želenih odločitev posebej in nato sestavljanje rezultatov. Medsebojna odvisnost več odločitev sicer ne bo neposredno razvidna iz samega vektorskega drevesa, lahko pa o njej sklepamo na podlagi primerjave velikosti vektorskega drevesa in ustreznega števila navadnih odločitvenih dreves. Če so odločitve med seboj dokaj povezane in korelirane, potem vektorsko drevo ne bo imelo prevelikega prirastka v številu vozlišč v primerjavi z ločenimi odločitvenimi drevesi za posamične odločitve.

Evolucija odločitvenih dreves

Ker se vektorska odločitvena drevesa razlikujejo od klasičnih samo v končnih vozliščih, je evolucija »navadnih« dreves zelo podobna evoluciji »vektorskih« dreves. Do razlike prihaja le pri ocenjevanju, saj moramo pri vektorskem drevesu oceniti njegovo uspešnost v več dimenzijah. Vendar za začetek poglejmo, kako sploh uporabljamo genetske algoritme pri razvoju odločitvenih dreves.

Potrebno začetno populacijo dreves lahko enostavno ustvarimo z naključno gradnjo dreves, kar zagotavlja potrebno genetsko raznolikost. Takšna populacija naj vsebuje dovolj veliko število dreves, od zelo dobrih pa tudi do zelo slabih. Evolucija bo namreč poskusila najti tiste dobre veje v posameznih drevesih, ki bi lahko združene sestavljale optimalno drevo. Razvoj načeloma vedno poteka po enem od dveh scenarijev glede na način polnjenja nove populacije [9], povsod pa je pomembna ciklična uporaba genetskih operatorjev selekcije, križanja in mutacije. Najbolj pomembna je ideja, da optimalnega drevesa ne gradimo s pomočjo determinističnega postopka v enem prehodu, ampak da razvoj prepustimo iterativnemu procesu.

Omenili smo že, da osebek v populaciji predstavlja enoodločitveno drevo. Vsak osebek pa mora biti predstavljen v obliki genetske kode. Najenostavnejše kodiranje bi bilo kar znakovno zaporedje, v primeru dreves pa je genotip lahko tudi drevesna struktura, kar pa je tudi končna želena oblika osebka – njegov fenotip. S fenotipom pa je določeno tudi obnašanje drevesa – pretvorba podanega vhoda v izhodni vektor odločitev s pomočjo preprostih vprašanj *if-then*. Evolucija nato preoblikuje drevesa v populaciji, genetski operatorji pa spreminjajo že spremenjene oblike. Pomemben faktor v tem postopku je ocena, kdaj je neko drevo dobro in kdaj slabo. Razlika med genetskim generiranjem navadnih, enodimenzionalnih, in vektorskih odločitvenih dreves je tako samo pri odločanju, kako dobro se neko drevo dejansko obnaša. Ker je za ocenjevanje dreves zadolžena posebna funkcija, je tako ta razlika vidna le na enem mestu in to je prav v ocenitveni funkciji drevesa. Operatorja križanja in mutacije delujeta neodvisno od ocene drevesa, le operator selekcije bo upošteval oceno drevesa pri izboru dreves za nadaljnjo reprodukcijo.

Ocenjevanje osebkov

Ocenitev novo nastalega drevesa poteka vedno s pomočjo učne množice, ocenitvena funkcija pa v oceni pove, kako dobro je neko drevo. Torej moramo drevo uporabiti na vseh učnih vzorcih in dobljene rešitve primerjati z dejanskimi pravilnimi vrednostmi. In tukaj je možnih kar nekaj scenarijev. Najenostavnejše merilo je gotovo odstotni delež pravilnih zadetkov. Pri navadnih, enoodločitvenih drevesih je pogosta praksa tudi uporaba uteži (stroškov) za določanje pomembnosti zadetkov, s čimer lahko uspešno pritisnemo na smer razvoja evolucije, saj tako umetno damo prednost določeni vrsti zadetkov. Tako lahko

npr. prisilimo drevesa, da se usmerijo proti določenim tipom rešitev. To je pomembno predvsem pri klasičnih enodimenzionalnih sistemih, ki imajo več možnih klasifikacij. S tem lahko povečamo zanesljivost oz. pravilnost sprejete klasifikacije. V primerih vektorskih dreves pa je takšen model povezan z mnogimi problemi, saj bi bila stroškovna matrika potrebna za vsako odločitev posebej in bi z njeno uporabo ogrozili ravno želeno iskanje odvisnosti odločitev.

Če označimo število odločitev, torej velikost odločitvenega vektorja, z v, potem ima lahko en vektor med 0 in v pravilnih vrednosti. Poseben primer pa so polni (*jackpot*) zadetki, ko je izhodni vektor popolnoma enak pravilnemu vektorju klasifikacij za izbrani vzorec. Če je velikost učne množice u, potem ima lahko drevo največ u polnih zadetkov. To sta hkrati tudi prvi dve merili uspešnosti drevesa. Preostali dve merili se podrobneje spustita v vsako od posameznih odločitev. Če ima i-ta odločitev v vektorju v_i možnosti, potem lahko za vsako j-to možnost preverimo, kolikokrat bi se dejansko morala zgoditi (v_{ij}) in kolikokrat se tudi je (v_{ij}'). Nato lahko izračunamo povprečje za celotno odločitev in nato še čez vse odločitve vektorja. Če npr. za vprašanje **a** velja, da ima 3 klasifikacije, potem ima lahko vsaka od klasifikacij $\mathbf{a}_1$, $\mathbf{a}_2$ in $\mathbf{a}_3$ svoj odstotek zadetkov. Povprečje se nato preračuna čez vse tri klasifikacije in nato še čez vse odločitve v vektorju. Kot zadnje merilo pa lahko uporabimo maksimalno napako (minimalni zadetek), ki jo pri prejšnjem izračunu doprinese posamezna klasifikacija odločitve k skupnemu povprečju.

Že prva dva kriterija tvorita enostavno in močno ocenitveno funkcijo, z drugima dvema pa lahko tudi krmilimo tendenco evolucije v enakomernejšo razdelitev zadetkov po odločitvah, odvisno seveda od primera do primera (Enačba 1). Ocenitvena funkcija v orodju DecRain vsebuje obteženo (w_i) povprečje opisanih ocen. Z utežmi lahko izboljšamo splošno konvergenco in prilagodimo model specifičnemu problemu.

Ker lahko pri razmnoževanju križamo različne veje na različnih globinah, imajo novo nastala drevesa tendenco hitre rasti. Veliko število vozlišč ponavadi povzroči dobre rezultate na učni množici, hkrati pa negativno vpliva na splošne generalizacijske sposobnosti drevesa. Da bi preprečili čezmerno rast, smo v oceno drevesa dodali tudi kazenski člen za *intronska* vozlišča. Introni so pomemben dejavnik v evoluciji in imajo velik vpliv na efektivno oceno (*effective fitness*) osebkov v populaciji. Efektivna ocena namreč opisuje sposobnost osebka, da zaščiti svoje potomstvo pred možnimi kvarnimi vplivi genetskih operatorjev, povečuje pa se ravno s selekcijskim pritiskom za ustvarjanje intronov [9]. Introni nastanejo v primerih podvojitve testov v isti veji drevesa. Če npr. vozlišče s testom atributa a_i že razporedi vzorce po izhodnih podvejah, je ponovitev *istega* testa v neki podveji popolnoma nekoristna. Ker pa mora biti drevo logično pravilno, mora takšno vozlišče vsebovati tudi eno ali več nekoristnih podvej. Razmerje med velikostjo drevesa in številom intronov lahko uporabimo kot kriterij dodelitve kazenskih točk k skupni oceni drevesa. Pričakujemo namreč lahko, da bodo večja drevesa praviloma vsebovala več intronov kot kompaktnejša, manjša drevesa. Možnih je še nekaj drugih rešitev za omejevanje čezmerne rasti dreves, npr. razmerje med aktivnim številom listov in številom vseh listov v drevesu, pa tudi med številom aktivnih listov in številom možnih vektorjev. Predlagani kriterij lahko s podobnimi nastavitvami uporabimo za gradnjo odločitvenih dreves pri različnih problemih, torej brez okvirne ocene, kako veliko bo dejansko drevo.

Selekcija, križanje in mutacija dreves

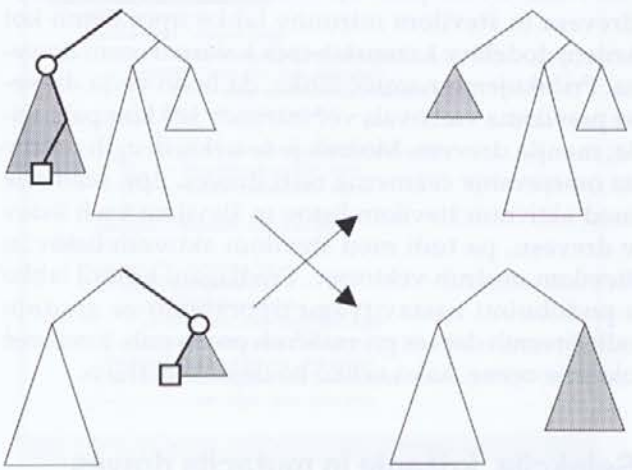
Operator selekcije ima nalogo odločiti, ali ima nek osebek potencial za nadaljnji razvoj ali pa ga je potrebno zavreči in nadomestiti z drugim osebkom. V našem orodju smo uporabili načelo linearnega razvrščanja (*linear ranking*) [9]. Ta selekcijska shema najprej na podlagi ocene osebka določi njegov položaj v populaciji, vsakemu položaju pa je dodeljena točno določena verjetnost izbora za nadaljnje razmnoževanje. Linearno razvrščanje pogosteje izbira boljša drevesa, nemalokrat pa tudi slabše osebke iz ozadja. Tak pristop ohranja genetsko raznovrstnost in hkrati omogoča kvalitativni napredek, kar je pogoj za uspešen razvoj brez prehitre konvergence k lokalni rešitvi. Ker so vsi parametri dinamično nastavljivi, lahko spreminjamo velikost izbora glede na velikost populacije in s tem spreminjamo selekcijski pritisk.

Pri križanju dveh dreves lahko uporabimo več strategij. Najbolj splošna bi bila naključna izbira dveh vozlišč in nato izmenjava z njima določenih poddreves. Takšna rešitev je popolnoma naključna in teži k destruktivnemu delovanju. Tako kot v naravi bi si namreč želeli kriterij, ki bi omogočil križanje dveh dreves samo na določenih točkah. Preprosto pravilo bi

$$fitness = F(w_1 \cdot \frac{hits}{v \cdot u}, w_2 \cdot \frac{jackpots}{u}, w_3 \cdot \underset{i,j}{E}(\frac{v_{ij}'}{v_{ij}}), w_4 \cdot \underset{i,j}{Min}(\frac{v_{ij}'}{v_{ij}})) + w_5 \cdot penalty(\text{introns})$$

Enačba 1. Ocenitvena funkcija v DecRainu

bilo tudi križanje v točkah z istim atributom, kar pa ne pomeni nujno dveh točk, ki bi odločali o istih vzorcih. Zato smo za izbiro točk križanja izbrali princip prehojene poti – naključno izbran vzorec iz učne množice spustimo skozi obe drevesi in nato na obeh prehojenih poteh izberemo po eno vozlišče. S križanjem v tako izbranih točkah tudi dejansko zamenjamo dve veji, ki z dosti večjo verjetnostjo sprejemata odločitve o istih vzorcih (Slika 3).



Slika 3: Križanje dveh dreves na prehojeni poti istega naključnega vzorca.

Genetski operator mutacij vnaša v genetsko kodo – strukturo drevesa – naključne spremembe. Nenadzorovana sprememba v arhitekturi drevesa je ponavadi destruktivna, saj zelo verjetno poruši ravnotežje, ki se je ustvarilo skozi evolucijski razvoj. Hkrati pa mutacije zagotavljajo nenadne preobrate, potrebne za iskanje globalnih rešitev, in vnašajo sveže spremembe v starejši genetski material osebka [11]. Mutacija v DecRain lahko nastopi nad katerimkoli vozliščem v drevesu in mu lahko spremeni tako tip kot tudi vsebino. Tako lahko mutacija spremeni list drevesa v notranje vozlišče ali pa enostavno spremeni končno odločitev v listu. Lahko pa deluje nad notranjim vozliščem drevesa, kjer lahko zamenja posamezne veje, določi novo poljubno vprašanje ali spremeni vozlišče v list z odločitvijo. DecRain omogoča dinamično nastavitev verjetnosti posameznih mutacij.

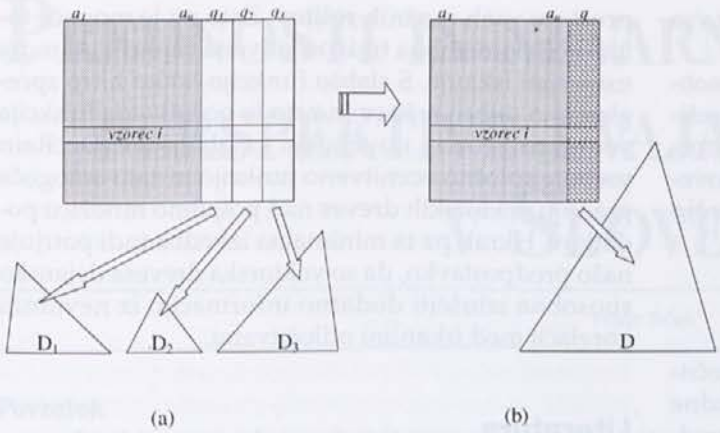
Rezultati

Vektorska drevesa lahko načeloma uporabimo tudi na klasičnih klasifikacijskih problemih z enim vprašanjem, sam vektorski koncept pa je seveda najprimernejši za naloge, kjer iščemo odgovore na več vprašanj hkrati. Za predstavitev ideje in orodja smo izbrali medicinski problem, ki je že bil deloma obdelan s pomočjo klasičnih dreves [3]. Ker je v medicini zelo pomemben dejavnik predvsem preglednost rezultatov in faktorjev, ki so pripeljali do določenih odločitev, smo se omejili izključno na odločitvena drevesa.

Sama priprava podatkov ni toliko zahtevna, kot so zahtevne priprave alternativnih metod reševanja. Da bi lahko dobili vse iskane odgovore s pomočjo klasičnih dreves, moramo postopati drugače, kot smo vajeni. Pri iskanju odgovorov na več vprašanj imamo dve možnosti. Prva je intuitivna in zahteva izgradnjo več samostojnih dreves za vsako zastavljeno vprašanje posebej. Osnovno idejo predstavlja slika 4.a, kjer so za tri postavljena vprašanja (Q_1, Q_2 in Q_3) zgrajena tri neodvisna drevesa.

Če označimo en vzorec kot množico n opisnih atributov (a) in treh pripadajočih rešitev $\{a_1, ..., a_n, q_1, q_2, q_3\}$, potem je s klasičnimi orodji mogoče enostavno zgraditi drevo za vsako vprašanje s pomočjo podmnožice $\{a_1, ..., a_n, q_i\}$. Idejo je mogoče razširiti tudi na gradnjo odvisnih (povezanih) dreves, kjer bi najprej zgradili npr. D_1 in nato še D_2, ki bi odgovor q_1 jemalo kot vhodni atribut a_{n+1} v svoji učni množici. Podobno velja za zadnje - tretje odločitveno drevo. Tak pristop je možen, če vprašanja niso ciklično odvisna in lahko določimo vrstni red gradnje dreves; to pa je redko, saj ponavadi ne poznamo niti vseh odvisnosti posameznih odgovorov od vhodnih atributov. Z vsemi tremi drevesi lahko potem enostavno dobimo vse tri odgovore za vsak vzorec. Problem tega pristopa je v ločenosti odgovorov, saj drevo D_2 nima nobene vednosti o zgradbi in znanju drevesa D_1. V primeru povezanih dreves pa so rezultati občutljivi na kopičenje napake, saj napačna klasifikacija v prvem drevesu verjetno pomeni tudi napako v vseh naslednjih drevesih.

Drugi pristop s pomočjo klasičnih dreves predvideva predhodno združitev vseh treh klasifikacij v eno samo klasifikacijo - Slika 4.b. Torej združitev vprašanj Q_1, Q_2 in Q_3 v novo vprašanje Q. Če je imelo prvo vprašanje 3 možne odgovore, drugo 2 in tretje 4, potem ima lahko Q celo 24 možnih odgovorov (toliko je tudi možnih različnih vektorjev v vektorskem drevesu). Slabost tega pristopa je skrita v logiki večine klasičnih algoritmov za gradnjo odločitvenih dreves. Ti predvidevajo, da so vsi odgovori medsebojno ločeni in neodvisni, kar pa tukaj seveda ni res. Algoritem pozna samo pravilne in napačne odgovore, pojma *bolj* ali *manj* pravilen nista smiselna. Sta pa pomembna pri vektorskih rešitvah, saj so si posamezne rešitve lahko bolj ali manj podobne. Če bo torej algoritem prisiljen izbirati iz množice napačnih odločitev, se lahko zgodi, da bo izbral popolnoma napačno rešitev, čeprav je imel na voljo optimalni zelo podobno rešitev. Ker pa



Slika 4: Rešitev s klasičnimi drevesi. (a) tri ločena drevesa (b) skupno pseudo vektorsko drevo

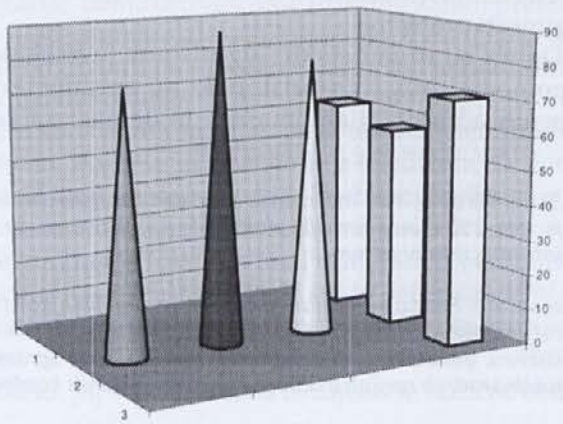
sta obe rešitvi napačni, tega seveda ni sposoben opaziti. Ker je rezultat tega drevesa po svoje tudi vektor odločitev, bomo to drevo poimenovali kar "pseudo vektorsko" drevo. Za vektorska drevesa zato lahko pričakujemo, da se bodo obnašala vsaj tako dobro kot oba predstavljena modela.

Gradnja vektorskih dreves z orodjem DecRain

DecRain smo uspešno uporabili na več primerih. Ker je za primerjave z drugimi modeli potrebnega veliko dela, smo realno primerjavo z nevronskimi mrežami lahko izvedli le na manjšem problemu, kjer so bili rezultati skoraj identični. Tukaj bi samo okvirno podali rezultate na večjem medicinskem problemu z več kot 100 atributi in več kot 600 vzorci. Sistem je imel 3 odločitve, reševali pa smo ga na oba prej opisana načina in seveda s pomočjo vektorskih odločitvenih dreves. Klasična drevesa so bila grajena s pomočjo orodja MtDecIt 1.0 [4], ki dosega rezultate enakovredne rezultatom C5 algoritma. Da bi zagotovili popolno skladnost rezultatov, smo klasična drevesa gradili z istimi delitvami intervalov kot vektorska drevesa.

Gradnja vektorskega drevesa s pomočjo orodja je preprosta. Vendar pa se ne smemo zadovoljiti že s prvo rešitvijo, ampak je potrebno preizkusiti različne nastavitve in možnosti. Tako smo vektorska drevesa za naš primer gradili kar nekajkrat. Sproti smo opazovali obnašanje dreves na učni in testni množici in prilagajali predvsem ocenitveno funkcijo, velikost selekcijskega izbora in velikost populacije. Največkrat smo začeli s populacijo 200 dreves, od katerih smo za razmnoževanje izbirali od 5-30 dreves (linearno razvrščanje). Z naključno določenimi pari teh dreves potem DecRain ustvari novo populacijo dreves s pomočjo križanja in mutacij. Za samo selekcijsko izbiro je pomembna ocena drevesa in s spremembo razmerja intronov v drevesu je moč hitro dobiti populacijo ogromnih ali minimalnih dreves – želimo pa si predvsem krhkega optimalnega ravnovesja med količino intronov aktivnih vozlišč. Spremembe v utežeh posameznih kriterijev lahko hitro spremenijo tendenco dreves k iskanju polnih zadetkov ali k splošnemu večanju natančnosti. Mutacije so bile pomembnejše predvsem v drugi fazi, ko se v populaciji že vzpostavi določena kvaliteta, ki pa teži k lokalnemu optimumu. V vseh primerih pa smo verjetnost mutacij zadržali na dokaj nizki ravni, pod 1%, in prepustili kreativnost predvsem operatorju križanja. Spremembe v velikosti populacije najbolj vplivajo na kvaliteto dobljenih dreves. Manjše začetne populacije lahko hitro izgubijo genetsko raznovrstnost in težko najdejo dobro rešitev, prevelike populacije pa konvergirajo dosti počasneje. Kot končno vektorsko drevo smo potem izmed vseh zgrajenih izbrali tisto drevo, ki se je v povprečju najbolje obnašalo na testni in tudi na učni množici. Dejstvo pa je, da isti nabor nastavitev v ločenih primerih ne zagotavlja nastanka dreves podobnih kvalitet.

Na testni množici je klasično pseudo drevo kombiniranih odločitev (Slika 4.b) doseglo največ 62% zanesljivost, merjeno pri različnih stopnjah tolerance gradnje drevesa. Tri ločena odločitvena drevesa pa so posamično imela 76% / 90% oz. 80% zanesljivost. Ko pa smo vse tri odločitve sestavili, je bilo *polnih* zadetkov le še okoli 58%. Torej se je klasični algoritem bolje odrezal v prvem načinu. Z uporabo orodja DecRain zgrajeno vektorsko odločitveno drevo je na testni množici doseglo 71% polnih zadetkov, kar je za 9% oz. 12% več kot pri klasično dobljenih rezultatih (Slika 5).



Slika 5: Natančnost za polne zadetke (1 – pseudo vektorsko drevo, 2 – tri ločena drevesa z delnimi rezultati, 3 – vektorsko drevo).

Prav tako je bilo vektorsko drevo celo manjše kot vsa klasično grajena drevesa!

V tem primeru (Slika 5.) tudi lepo vidimo sposobnost klasičnih odločitvenih dreves, da zelo dobro rešijo navadne enodimenzionalne probleme (delni rezultati ločenih dreves), vendar pa so te rešitve nepovezane. Vektorsko drevo nasprotno dosega večjo natančnost v vseh dimenzijah hkrati.

Zaključek

Predstavljeno orodje za gradnjo vektorskih odločitvenih dreves je dalo že sedaj zanimive in vzpodbudne rezultate, zato bo deležno še določenih izboljšav predvsem pri dinamični delitvi numeričnih atributov. Test takšnega atributa mora razmejiti vzorce v dve ali več kategorij in s tem seveda izredno vpliva na kvaliteto drevesa. Določitev tega testa pa je možno prepustiti evoluciji hkrati s samim razvojem samih odločitvenih dreves. Razvoj bo šel tudi v smeri samoprilagodljivih genetskih parametrov, kar pomeni evolucijski razvoj ne samo dreves, ampak tudi nastavitev za preoblikovanje odločitvenih dreves. Glavna težava vseh genetskih algoritmov je vsekakor časovna zahtevnost – čeprav je moč rešitve za določene probleme najti relativno hitro, pa bo tudi najhitrejši genetski algoritem prej ali slej klonil pred dovolj velikim problemom. Ob tem nam še vedno ostane vprašanje kvalitete rezultatov – če dobljenih rezultatov ne moremo primerjati s kako drugače zgrajenimi rešitvami, nam ne preostane drugega kot ponovna evolucijska gradnja večjega števila rešitev. Najboljši rezultat lahko nato proglasimo kot dokončno rešitev problema. Kar pa je seveda spet zamudno in zahteva tudi precej dela. Pri sprotnem opazovanju dobljenih dreves se lahko tudi mnogo naučimo o samem problemu, ki ga rešujemo. Glede na potrebno velikost začetne populacije in drugih nastavitev in ob upoštevanju delnih rezultatov ter hitrosti konvergence lahko posredno sklepamo tudi o težavnosti problema.

Dobra stran genetskih algoritmov je vsekakor popolnoma drugačen pristop, ki ne predpostavlja nobenih odvisnosti ali omejitev in dejansko išče v prostoru vseh možnih rešitev. Zato pa je mogoče toliko bolj pomembna tudi ocenitvena funkcija, ki mora usmerjati iskanje. S slabšo funkcijo lahko hitro spregledamo dobro rešitev in zato je ocenitvena funkcija vedno prostor za nove ideje in izboljšave. DecRain vsebuje splošno ocenitveno funkcijo in zato omogoča gradnjo vektorskih dreves nad poljubno množico podatkov. Hkrati pa ta minimalna izvedba tudi potrjuje našo predpostavko, da so vektorska drevesa dejansko sposobna izluščiti dodatno informacijo iz nevidnih korelacij med iskanimi odločitvami.

Literatura

[1] Kokol P., et al: Decision Trees and Automatic Learning and Their Use in Cardiology, Journal of Medical Systems 19(4), 1994.

[2] Kokol P., et al: Participative Design, Decision Trees, Automatic Learning And Medical Decision Making, Medical Informatics Europe '96, Studies in Health Technology and Informatics, Vol. 34, pp. A 501 – 505, Amsterdam 1996.

[3] Vili Podgorelec: Samoprilagodljiv evolucijski model za pomoč pri odločanju, Zbornik 7. elektrotehniške in računalniške konference ERK 98, pp. 471-472, 1998.

[4] Kokol P., et al: The Limitations of Decision Trees and Automated Learning in Real World Medical Decision Making, MEDINFO ž98, Vol. 52, pp. 529-533, Amsterdam 1998.

[5] Nilsson N. J., Introduction to Machine Learning, Stanford University, 1996.

[6] Quinlan J. R.: Induction of Decision Trees, Machine Learning, No. 1, pp 81-106, 1986.

[7] Quinlan J. R.: Decision Trees and Instance Based Classifiers, Artificial Intelligence and Robotics, pp. 521- 535, 1997.

[8] Quinlan J. R.: C4.5: Programs for Machine Learning, Morgan Kaufmann publishers, San Mateo, CA, 1993.

[9] Banzhaf W., Nordin P., Keller R. E., Francone F. D.: Genetic Programming – An Introductionary Approach, Morgan Kaufmann Publishers Inc., 1998.

[10] Bäck T.: Evolutionary Algorithms in Theory and Practice, Oxford University Press, Inc., 1996.

[11] Goldberg D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, Reading MA, 1989.

[12] Koza J. R.: Genetic Programming: On the Programming of Computers by Natural Selection, MIT Press, 1992.

Matej Šprogar, dipl. inž. rač., je zaposlen kot raziskovalec na Fakulteti za računalništvo in informatiko v Mariboru. Diplomiral je leta 1996 in pripravlja doktorat. Raziskovalno se ukvarja z inteligentnimi sistemi, evolucijskim modeliranjem ter z informacijskimi sistemi.

◆

Mag. Vili Podgorelec, dipl. inž. rač., je zaposlen kot raziskovalec na Univerzi v mariboru, na Fakulteti za elektrotehniko, računalništvo in informatiko. Magistriral je leta 1999 in trenutno pripravlja doktorat. Raziskovalno se ukvarja z inteligentnimi sistemi, genetskimi algoritmi in avtomatskim programiranjem ter z analizo programov. Je avtor več člankov, objavljenih v mednarodnih revijah in zbornikih mednarodnih konferenc ter je član IEEE in ACM.

◆

Dr. Peter Kokol je zaposlen kot izredni profesor s področja računalništva na Univerzi v Mariboru, na Fakulteti za elektrotehniko, računalništvo in informatiko. Doktoriral je leta 1992. Raziskovalno se ukvarja s programskimi jeziki, analizo programov, razvojem informacijskih sistemov in teorijo sistemov. Je avtor mnogih člankov, objavljenih v zbornikih mednarodnih konferenc in v mednarodnih revijah ter je član IEEE, ACM, ISCA in Slovenskega društva INFORMATIKA.