Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Marko Robnik Šikonja

Lastnosti in uporaba hevristične funkcije Relief v strojnem učenju

DOKTORSKA DISERTACIJA

Mentor: prof.dr. Igor Kononenko

Ljubljana, 2001

Povzetek

Pri strojnem učenju se je za enega ključnih elementov izkazala hevristična ocena kakovosti atributov. Ocenjevanje kakovosti atributov bistveno vpliva na več podpodročij strojnega učenja: izbiro podmnožice pomembnih atributov, konstruktivno indukcijo, izgradnjo odločitvenih in regresijskih dreves, itd. V literaturi so opisani številni postopki za ocenjevanje kakovosti atributov v regresiji in klasifikaciji. Večina jih predpostavlja, da so atributi pogojno neodvisni pri danem razredu in so zato neprimerni za uporabo v problemih, kjer morda obstajajo močne pogojne odvisnosti med atributi.

Skupina algoritmov Relief (Relief, ReliefF in RReliefF) ne predpostavlja neodvisnosti atributov. Ti algoritmi se zavedajo konteksta, so učinkoviti in zanesljivo ocenjujejo kakovost atributov tudi v problemih z močnimi pogojnimi odvisnostmi med atributi. Uspešno so bili uporabljeni za številne naloge: za izbiro podmnožice pomembnih atributov in uteževanje atributov, pri gradnji odločitvenih in regresijskih dreves, pri klasifikaciji s pomočjo povezovalnih pravil, v konstruktivni indukciji, za diskretizacijo atributov in v induktivnem logičnem programiranju. Široko področje uspešne uporabe je motivacija za skrbno analizo in sistematično raziskavo različnih lastnosti teh algoritmov, s čemer se ukvarja ta disertacija.

Najprej smo opisali družino algoritmov Relief ter preučili podobnosti in razlike med posameznimi algoritmi. Zaradi razumljivosti smo predstavili osnovni algoritem Relief, ki je omejen na dvorazredne klasifikacijske probleme. Pojasnili smo njegovo razširitev ReliefF, ki zna ocenjevati atribute v večrazrednih problemih, je bolj robusten in lahko obravnava šumne in manjkajoče vrednosti. Predstavili smo verjetnostno interpretacijo ocen kakovosti in pojasnili kaj predstavljajo ocene; preko te analize smo prišli do prilagoditve algoritma za regresijske probleme, ki se imenuje RReliefF.

Teoretično smo analizirali računsko zahtevnost algoritmov Relief ter pojasnili vlogo, ki jo imajo pri njej k-d drevesa in vzorčenje. Vzpostavili smo relacijo med ocenami algoritmov Relief in funkcijami nečistoče. Predstavljamo splošen okvir, v katerega lahko postavimo algoritme za ocenjevanje kakovosti atributov, ki temeljijo na kontekstni podobnosti/različnosti med vrednostmi atributa in rešitvijo učnega primera. Pojasnili smo poseben primer tega okvira, ki predstavlja algoritme Relief, in jih nato na tej podlagi primerjali s sorodnimi algoritmi.

Pokazali smo, da lahko oceno algoritma Relief v limiti, ko gre število primerov v neskončnost, interpretiramo kot delež sprememb koncepta, ki jih ta atribut razloži. Ta interpretacija nam je omogočila, da smo razložili obnašanje ocen kakovosti pri naraščanju števila pomembnih atributov in pri odvečnih atributih.

Teoretično in eksperimentalno smo analizirali različne parametre algoritmov Relief ter podali nekaj napotkov za njihovo določanje. Prikazali smo neobčut-

ljivost algoritmov glede na uporabo evklidske ali manhattanske razdalje ter občutljivost pri obravnavi razdalje v problemskem prostoru. Pojasnili smo povezavo med globalnostjo ocen, kontekstno občutljivostjo ter številom in uteženostjo bližnjih sosedov. Predstavili smo posebnosti pri obravnavi številskih atributov in razložili potrebno velikost vzorca in število iteracij.

Pri analizi in razumevanju problemov ter v meta učenju so pomembne mere, ki opisujejo težavnost koncepta. Spremenili smo definicijo spremenljivosti koncepta, ki je taka mera. Vpeljana mera združuje dobre lastnosti dveh že obstoječih. Na njeni podlagi smo skušali razložiti razlike v ocenah kakovosti atributov v različnih problemih.

Empirično smo ovrednotili delovanje algoritmov ReliefF in RReliefF. Preverjali smo, katere odvisnosti zaznavata algoritma. V ta namen smo ju preskusili na problemih Modulo-p-I, ki so celoštevilčna posplošitev koncepta parnosti, na znanih problemih MONK, za regresijsko inačico pa smo definirali nekaj linearnih in nelinearnih odvisnosti. Vpeljali smo meri ločljivosti in uporabnosti ocen atributov. Ločljivost ocen smo definirali kot razliko v oceni kakovosti najslabše ocenjenega pomembnega atributa in najboljše ocenjenega nepomembnega atributa, uporabnost pa kot razliko v oceni kakovosti najboljše ocenjenega pomembnega atributa in najboljše ocenjenega nepomembnega atributa. Na vrsti problemov smo preverjali, najmanj koliko učnih primerov potrebujemo, da bodo ocene kakovosti z veliko verjetnostjo ločljive in uporabne. Glede na veliko verjetnost ločljivosti in uporabnosti smo opazovali tudi število potrebnih iteracij, število nepomembnih atributov v problemu in stopnjo šumnosti napovedane vrednosti. Predstavili smo delovanje glede na število pomembnih atributov in ocene odvečnih atributov. Kjer je bilo smiselno, smo obnašanje primerjali s hevristikama za ocenjevanje atributov Gain ratio ali MSE.

Podali smo krajši pregled uporabe algoritmov Relief. Opisali smo izbiro podmnožice pomembnih atributov, uteževanje atributov, usmerjanje konstruktivne indukcije, diskretizacijo številskih atributov, uporabo v induktivnem logičnem programiranju in pri klasifikaciji s pomočjo povezovalnih pravil. Pri opisu uporabe v drevesnih modelih smo opisali razvito metodo za prehod med nekratkovidno in kratkovidno hevristiko pri gradnji drevesa, ki temelji na korelaciji ocen algoritma Relief s funkcijami nečistoče in na lokalnosti.

Features of the heuristic function Relief and its use in machine learning

Abstract

The problem of estimating the quality of attributes (features) is an important issue in machine learning. There are several important tasks in the process of machine learning e.g., feature selection, constructive induction, decision and regression tree building, which contain an attribute estimation procedure as their (crucial) ingredient. The problem of feature (attribute) estimation has received much attention in the literature. There are several measures for estimating the attribute's quality for a discrete target concept (classification problem) and for a real valued function (continuous class or regression problem). Most of the heuristic measures for estimating the quality of the attributes assume the independence of the attributes given the predicted value and are therefore less appropriate in problems which possibly involve much feature interaction. Relief algorithms (Relief, ReliefF and RReliefF) do not make this assumption. They are efficient, aware of the contextual information, and can correctly estimate the quality of attributes in problems with strong dependencies between the attributes.

Relief algorithms are general and successful attribute estimators and are especially good in detecting conditional dependencies. They provide a unified view on attribute estimation in regression and classification and their quality estimates have a natural interpretation. While they have commonly been viewed as feature subset selection methods that are applied in a preprocessing step before the model is learned, they have actually been used successfully in a variety of settings: to select splits or to guide constructive induction in the building phase of decision or regression tree learning, as an attribute weighting method, in discretization of numerical attributes, in inductive logic programming, and with association rules based classifier. Such a broad spectrum of successful use calls for an especially careful investigation of various features the Relief algorithms have, which is a topic of this dissertation.

We describe the algorithms Relief and study their differences and similarities. First we present the original Relief algorithm which was limited to classification problems with two classes. We give account on how and why it works. We discuss its extension ReliefF which can deal with multiclass problems. The improved algorithm is more robust and also able to deal with incomplete and noisy data. Then we show how ReliefF was adapted for regression problems and describe the resulting RReliefF algorithm. After the presentation of the algorithms we tackle some theoretical issues about what Relief output actually is.

We analyze theoretical properties of the Relief algorithms. First we derive

computational complexity of the algorithms and describe the role of k-d trees and sampling. We show the relation of estimates with the impurity functions. Algorithms for context sensitive attribute estimations can be put into a general framework. We present such a framework based on the similarity between attribute values and predicted values. Relief algorithms are put into this framework and analysed on its basis. This analysis enables the comparison with related algorithms.

Another view on Relief's estimate of an attribute is possible through the proportion of class labels that the given attribute helps to determine. We claim that in the classification problem as the number of examples goes to the infinity the Relief's weights for each attribute converge to the proportion of class labels the attribute determines. This interpretation enables the explanation of estimates when we increase the number of important attributes and with the presence of redundant attributes.

We address different parameters of the Relief algorithms and give some advice concerning their settings. The use either of the Euclidian or Manhattan distance is shown not to have significant impact on estimates, however the way the distance is taken into account might crucially change the estimates. Special care has to be taken when estimating numerical attributes, we explain why and how. The number and weighting of nearest neighbors is related with the context and the locality of view. We investigate these parameters as well as sample size and the number of iterations.

When analyzing and trying to understand a new problem it is useful to look at its various features. Concept variation is a measure providing insight into concept difficulty. We redefined concept variation by merging favorable properties of previous definitions. The new measure can be used on real world datasets and is suitable for using in meta learning. With its help we explain some of the differences in quality estimates over various problems.

The performance of ReliefF and RReliefF is empirically evaluated. We investigate what kind of dependencies they detect and test them on some of Modulo-p-I problems, which are integer generalizations of the parity problems, and on the well known MONK's datasets. For RReliefF we introduce problems with linear and non-linear dependencies. We introduce two measures for the evaluation of attribute estimations. The *separability* is the difference between the quality estimates of the worst estimated important attribute and the best estimated random attribute, while the *usability* is the difference between the quality estimates of the best of the important attributes and the best of the random attributes. On a number of problems we measure the number of examples needed for high probability of positive separability and usability. With the same criteria we measured also the number of iterations, the number of random attributes and the amount of noise. We also raise a question of the number of important and redundant attributes. Wher-

ever appropriate we compare the behavior of the Relief algorithms with attribute estimation measures Gain ratio or MSE.

Finally, we present a short overview of different applications of the Relief algorithms. We present the use in feature subset selection, feature weighting, guidance of the constructive induction, discretization of numerical attributes, in inductive logic programming, and with the association rules based classifier. When describing the use in tree based models we introduce the method for switching from the Relief algorithms to a myopic estimator (error minimization) in the top-down building of tree based models. The switching point is determined on the basis of locality and correlation between the Relief's estimate and the impurity based estimate.

Ključne besede

Keywords

umetna inteligenca	artificial intelligence
strojno učenje	machine learning
ocenjevanje atributov	attribute estimation
algoritem Relief	Relief algorithm
induktivno učenje	inductive learning
klasifikacija	classification
regresija	regression
drevesni modeli	tree based models
odločitvena drevesa	decision trees
regresijska drevesa	regression trees

Zahvala

Najprej bi se rad zahvalil svojemu mentorju prof. dr. Igorju Kononenku, ki me je v teku podiplomskega in doktorskega študija vsestransko spodbujal, usmerjal in podpiral. Naučil me je strogosti znanstvenega načina razmišljanja ter pokazal, da v znanosti in življenju ni prostora za dogme.

Zahvala gre predstojniku laboratorija in vodji naše raziskovalne skupine, prof. dr. Ivanu Bratku, ki me je sprejel v svoje dobro utečeno in spodbudno raziskovalno in delovno okolje, v katerem sem se lahko naučil raziskovalnega dela.

Hvaležen sem sodelavcem obeh ljubljanskih laboratorijev za umetno inteligenco, ki so prispevali ustvarjalno vzdušje ter odgovorili na mnoga moja vprašanja znanstvene in strokovne narave. Še posebej bi izpostavil Matjaža Kukarja, Uroša Pompeta, Janeza Demšarja, in Doriana Šuca. Njim sem hvaležen tudi za prijetno in prijateljsko vzdušje, ki je predpogoj za koristno usmerjanje energije.

Moje raziskovanje in potovanja na konference sta finančno podprla Ministrstvo za znanost, šolstvo in šport s projektom Avtomatizirana sinteza znanja in Fakulteta za računalništvo in informatiko, za kar sem jima hvaležen.

Zahvaljujem se vsem bližnjim, ki so me podpirali in tolerirali.

Moje žena Žo ter sinova Jakob in Matic mi dajejo motivacijo, energijo in smisel za moje delo in bivanje. Njim in življenju hvala za vso ljubezen in svetlobo.

Kazalo

Slike

Tabele

1

Uvod

Živi za sedanjost, sanjaj o prihodnosti, uči se iz preteklosti.

anonim

Učenje je gotovo ena od dejavnosti, ki najmočneje oblikujejo človeka, morda tudi naravo. Proučujejo ga na številnih znanstvenih področjih. V tem delu gledamo na učenje z vidika umetne inteligence, oziroma natančneje z vidika strojnega učenja.

V informacijski družbi je eden ključnih problemov obvladovanje in analiza velikih količin podatkov, ki vsakodnevno nastajajo in luščenje relevantnih informacij iz njih, zato ima avtomatizacija tega procesa velik pomen.

Strojno učenje (Mitchell, 1997; Dietterich in Shavlik, 1990; Kononenko, 1997) se ukvarja z računalniško podprtim generiranjem znanja iz zbranih učnih primerov in predznanja ter s predstavitvijo naučenega v simbolni ali številski obliki. Generirano znanje lahko uporabimo bodisi za boljše razumevanje problema, ki ga obdelujemo, bodisi za reševanje novih, še neznanih primerov. Če so učni primeri rešeni in opisani v atributnem jeziku, imenujemo tako učenje nadzorovano atributno učenje.

1.1 Ocenjevanje kakovosti atributov

Ocenjevanje kakovosti atributov je pomembno vprašanje strojnega učenja, ki bistveno vpliva na več podpodročij strojnega učenja: izbiro podmnožice pomembnih atributov (Blum in Langley, 1997), konstruktivno indukcijo (Liu in Motoda, 1998), izgradnjo odločitvenih in regresijskih dreves (Breiman in sod., 1984), itd.

Mnogi učni problemi so opisani z več sto atributi. Večina učnih metod ni primernih za take probleme, saj primeri z mnogo nepomembnimi, šumnimi atributi vsebujejo le malo informacije. Postopki za izbiro podmnožice pomemb-

nih atributov nam v idealnem primeru vrnejo majhno, toda potrebno in zadostno množico atributov za opis danega koncepta (Blum in Langley, 1997; Kira in Rendell, 1992a). Za odločanje o tem, katere atribute izbrati v tako množico, potrebujemo zanesljivo in praktično učinkovito metodo za ocenjevanje kakovosti atributov v povezavi s ciljnim konceptom.

Na podoben problem naletimo v konstruktivni indukciji, kjer zato, da bi povečali izrazno moč opisnega jezika in skonstruirali novo znanje, uvajamo (številne) konstrukte (Brodley in Utgoff, 1995). Postopek za ocenjevanje njihove kakovosti v povezavi s ciljnim konceptom je gotovo ključnega pomena za uspeh konstruktivne indukcije.

Odločitvena in regresijska drevesa so v strojnem učenju priljubljen opisni jezik (Breiman in sod., 1984; Quinlan, 1993a). Med njihovo gradnjo se učni algoritem v vsakem notranjem vozlišču odloča za pravilo, ki razbije problemski prostor. To pravilo je lahko en sam atribut ali pa izraz sestavljen iz več atributov. Postopek za ocenjevanje kakovosti teh pravil pomembno vpliva na uspešnost učenja (Breiman in sod., 1984; Kononenko in sod., 1997), saj določa strukturo drevesa.

V literaturi so opisani številni postopki za ocenjevanje kakovosti atributov. Če je ciljni koncept predstavljen z diskretno spremenljivko (klasifikacijski problem) so to npr. informacijski prispevek (Hunt in sod., 1966), indeks Gini (Breiman in sod., 1984), mera razdalje (Mantaras, 1989), j-ocena (Smyth in Goodman, 1990), Relief (Kira in Rendell, 1992b), ReliefF (Kononenko, 1994), MDL (Kononenko, 1995), uporabljata pa se tudi statistiki χ^2 in G. Kadar je ciljni koncept predstavljen kot realna funkcija (regresijski problem), so to npr. srednja kvadratna in srednja absolutna napaka (Breiman in sod., 1984) ter RReliefF (Robnik Šikonja in Kononenko, 1997).

Zgornje mere za oceno kakovosti atributov, razen algoritmov Relief, predpostavljajo, da so atributi pogojno neodvisni od razreda in so zato neprimerne za uporabo v problemih, kjer morda obstajajo močne pogojne odvisnosti med atributi.

1.2 Odvisnosti med atributi

Ko v tem delu govorimo o odvisnostih med atributi, imamo v mislih pogojne odvisnosti glede na napovedano vrednost. Ponazorimo te odvisnosti na treh primerih z logičnimi funkcijami.

1. Naj bo razred definiran kot parnost dveh atributov (XOR): $C = A_1 \oplus A_2$. Če poznamo vrednost razreda in enega od atributov,. imamo vso informacijo tudi o drugem atributu, medtem ko poznavanje samo vrednosti razreda ne vsebuje nobene informacije o vrednostih atributov. Govorimo o močni (maksimalni) pogojni odvisnosti A_1 in A_2 (glede na razred).

2. Naj bo razred konjunkcija dveh atributov: $C = A_1 \wedge A_2$. Poznavanje vrednosti razreda in enega od atributov je včasih zadosti, da določimo vrednost drugega atributa (npr., če $C = 0$ in $A_1 = 1$, mora biti vrednost $A_2 = 0$), ne pa vedno (npr., če $C = 0$ in $A_1 = 0$, ostaja vrednost A_2 nedoločena). S poznavanjem samo vrednosti razreda imamo na voljo še manj informacije (npr., če $C = 0$, ne moremo določiti niti vrednosti A_1 niti vrednosti A_2). Rečemo, da sta A_1 in A_2 (nekoliko) pogojno odvisna.

3. Naj bo razred poljubna nekonstantna stohastična logična funkcija, vrednosti atributov naj bodo enako verjetne $P(A_1 = 1) = P(A_2 = 1) = 0.5$ in atributi naj bodo definirani kot $P(A_1 = 1|C = 1) = p$, $P(A_1 = 1|C = 0) = 1 - p$ in $P(A_2 = 1|C = 1) = q$, $P(A_2 = 1|C = 0) = 1 - q$. Vsak atribut vsebuje nekaj informacije o vrednosti razreda, vendar vsebujeta to informacijo neodvisno drug od drugega. S poznavanjem vrednosti razreda imamo nekaj informacije o vrednosti obeh atributov, vendar informacija o vrednosti enega atributa ne naraste, če zvemo za vrednost drugega atributa. V tem primeru pravimo, da sta A_1 in A_2 pogojno neodvisna. Drug primer pogojne neodvisnosti so naključni atributi.

1.3 Relief

Družina algoritmov Relief (Relief, ReliefF in RReliefF) ne predpostavlja neodvisnosti atributov. Ti algoritmi se zavedajo konteksta, so učinkoviti in zanesljivo ocenjujejo kakovost atributov tudi v problemih z močnimi pogojnimi odvisnostmi med atributi. Originalno je algoritem Relief (Kira in Rendell, 1992b;a) deloval le na dvorazrednih klasifikacijskih problemih. Algoritem ReliefF (Kononenko, 1994) je robustna razširitev originalnega algoritma na večrazredne probleme, ki rešuje tudi problem šumnih podatkov ter manjkajočih vrednosti atributov. Prilagoditev tega algoritma za regresijske probleme imenujemo RReliefF (Robnik Šikonja in Kononenko, 1997).

V literaturi se algoritmi Relief omenjajo večinoma v povezavi z izbiro podmnožice pomembnih atributov, kot del predprocesiranja podatkov (Kira in Rendell, 1992b) in so v tej vlogi med najuspešnejšimi algoritmi (Dietterich, 1997). Uspešno so bili uporabljeni tudi v številnih drugih vlogah: pri gradnji odločitvenih in regresijskih dreves (Kononenko in sod., 1997; Robnik Šikonja in Kononenko, 1997), pri klasifikaciji s pomočjo povezovalnih pravil (Jovanoski in Lavrač, 1999), v konstruktivni indukciji (Robnik Šikonja, 1997), kot metoda za uteževanje atributov (Wettschereck in sod., 1997), za diskretizacijo atributov (Robnik Šikonja in Kononenko, 1995), pa tudi v induktivnem logičnem programiranju (Pompe in Kononenko, 1995).

Algoritma ReliefF in RReliefF sta bila dobro sprejeta v raziskovalni skupnosti strojnega učenja in sta danes vključena v več programskih paketov za strojno učenje in analizo velikih količin podatkov, npr. Orange (Demšar in Zupan, 2001) in Weka (Witten in Frank, 1999).

Širok spekter uspešne rabe kliče po skrbni analizi in sistematični raziskavi različnih lastnosti teh algoritmov. Poskušali bomo odgovoriti na praktična vprašanja, ki zadevajo njihovo uporabo: kako in zakaj delujejo, kakšne odvisnosti razpoznavajo, kako interpretirati njihove ocene kakovosti atributov, kako se obnašajo glede na število primerov, glede na število pomembnih in odvečnih atributov ter kako prenašajo šum v podatkih. Poleg teh praktičnih vprašanj bomo analizirali tudi njihovo računsko zahtevnost, njihove parametre ter se ukvarjali s splošnimi vprašanji ocenjevanja atributov.

1.4 Pregled vsebine

V disertaciji predstavimo lastnosti in uporabo algoritmov Relief.

V drugem poglavju definiramo atributni opis problemov ter nato opišemo različne vloge, ki jih ima ocenjevanje atributov v strojnem učenju, in kriterije uspešnosti algoritmov za ocenjevanje kvalitete atributov. Poglavje končamo s kratkim pregledom najvažnejših postopkov za ocenjevanje atributov.

V tretjem poglavju opišemo algoritme Relief, ReliefF in RReliefF, njihove medsebojne podobnosti in razlike. Predstavimo tudi verjetnostno interpretacijo ocen kakovosti, ki jo izračunavajo.

V naslednjem poglavju se lotimo teoretične analize. Začnemo z računsko zahtevnostjo ter pojasnimo vlogo, ki jo imajo pri njej k-d drevesa in vzorčenje. Predstavimo splošen okvir, v katerega lahko postavimo algoritme za ocenjevanje kakovosti atributov, ki temeljijo na kontekstni podobnosti/različnosti med vrednostmi atributa in napovedano vrednostjo. V ta okvir umestimo algoritme Relief in na tej podlagi ReliefF primerjamo s sorodnim algoritmom Contextual Merit (Hong, 1997). Predstavimo relacijo s funkcijami nečistoče ter končamo z asimptotično interpretacijo ocen kakovosti atributov, ki jih vrne Relief.

S parametri algoritmov ReliefF in RReliefF se ukvarjamo v petem poglavju. Preučimo vpliv različnih metrik in načine upoštevanja razdalje. Pogledamo si posebnosti pri uporabi številskih atributov in razložimo pomen števila bližnjih sosedov, velikosti vzorca in števila iteracij.

V šestem poglavju se posvetimo empirični analizi in se ukvarjamo s praktičnimi aspekti delovanja algoritmov ReliefF in RReliefF. Začnemo z definicijami nekaterih koristnih mer in konceptov, ki jih v nadaljevanju uporabimo pri analizi in razlagi delovanja. S poskusi preverjamo, katere odvisnosti zaznavata algoritma, kako se obnašata pri velikem številu primerov in atributov, koliko iteracij je po-

trebnih za zanesljive ocene, kako robustna sta glede šuma ter vpliva nepomembnih in odvečnih atributov.

V sedmem poglavju podamo krajši pregled uporabe. Začnemo z izbiro podmnožice pomembnih atributov in z uteževanjem atributov. Pri opisu uporabe v drevesnih modelih predstavimo metodo za prehod med nekratkovidno in kratkovidno hevristiko pri gradnji drevesa ter usmerjanje konstruktivne indukcije. Omenimo razumljivost naučenih dreves in končamo z uporabo pri diskretizaciji številskih atributov in v induktivnem logičnem programiranju.

V zadnjem poglavju navajamo naše zaključke ter smernice za nadaljnje delo.

V prvem dodatku opišemo značilnosti uporabljenih problemov v drugem pa učni sistem CORE v katerem so sprogramirani vsi algoritmi opisani v tem delu.

1.5 Prispevki k znanosti

Disertacija vsebuje naslednje izvirne prispevke k znanosti:

1. Algoritem Relief smo prilagodili za ocenjevanje atributov v regresijskih problemih. Ta različica, ki smo jo poimenovali RReliefF, daje v praksi zelo dobre rezultate in je bila dobro sprejeta v raziskovalni skupnosti strojnega učenja.

2. Razvit je bil splošen okvir za analizo algoritmov, ki ocenjujejo kakovost atributov glede na kontekstno podobnost/različnost med vrednostmi atributa in označbo učnega primera. To nam je omogočilo posplošitev algoritmov Relief na uporabo poljubne mere podobnosti. Na tej osnovi smo razložili razlike v delovanju v primerjavi s sorodnim algoritmom CM.

3. Pokazali smo, da lahko oceno algoritma Relief v limiti, ko gre število primerov v neskončnost, interpretiramo kot delež sprememb koncepta, ki jih ta atribut razloži. Ta interpretacija je intuitivno razumljiva in nam je omogočila, da smo razložili tudi obnašanje ocen kakovosti pri naraščanju števila pomembnih atributov in pri odvečnih atributih.

4. Teoretično in eksperimentalno smo analizirali različne parametre algoritmov Relief: uporabo metrik, obravnavo razdalje, število in uteženost bližnjih sosedov, velikost vzorca in število iteracij. Podali smo nekaj napotkov za njihovo določanje.

5. Teoretično in eksperimentalno smo analizirali delovanje algoritmov ReliefF in RReliefF: katere odvisnosti razpoznavata in kako se pri njih obnašata glede na število učnih primerov, število iteracij, vsebovanost šuma in nepomembnih atributov. Te vrednosti smo izmerili glede na veliko verjetnost

ločljivosti in uporabnosti med ocenami pomembnih in nepomembnih atributov.

6. Redefinirali smo spremenljivost koncepta kot mero težavnosti koncepta. Vpeljana mera združuje dobre lastnosti dveh že obstoječih mer.

7. Razvili smo metodo za preklop na minimizacijo napake med gradnjo drevesnega modela, ki temelji na korelaciji Reliefovih ocen z ocenami kakovosti atributov funkcij nečistoče.

8. Prilagodili smo algoritme Relief za uporabo v analizi velikih količin podatkov. V algoritmih Relief smo uporabili k-d drevesa in jih nato analizirali glede računske zahtevnosti. V tem kontekstu smo obravnavali tudi možnosti za paralelizacijo ter vpliv velikosti vzorca in potrebnega števila iteracij na zanesljivost ocen.

9. Razvite metode smo implementirali v učni sistem CORE za ocenjevanje atributov in induktivno učenje odločitvenih in regresijskih dreves, v katerega smo vgradili različne mere za ocenjevanje atributov, več operatorjev za generiranje konstruktov v notranjih vozliščih dreves ter različne modele za napovedovanje vrednosti primerov v listih dreves.

10. Sistem CORE smo empirično ovrednotili na več umetnih in realnih podatkovnih bazah ter uspešno uporabili na realnih problemih iz ekologije in medicine.

2

Ocenjevanje kakovosti atributov

Napovedovati je zelo težko. Še posebno prihodnost.

Niels Bohr

V tem delu se bomo ukvarjali z ocenjevanjem kakovosti atributov v nadzorovanem atributnem učenju, kjer so učni primeri opisani v atributnem jeziku in označeni z neko vrednostjo. Namen te vrste učenja je iz podanih primerov in njihovih označb določiti model, na podlagi katerega bo znal nek postopek novim primerom iste vrste kar najbolj pravilno določiti označbo. Včasih želimo model, ki bo razumljiv ljudem in jim bo pomagal razumeti problem. Ocenjevanje kvalitete atributov je pomembna sestavina različnih učnih postopkov, včasih pa so ocene kvalitete atributov že same po sebi zanimive tudi za boljše razumevanje problema.

V tem poglavju najprej definiramo atributni opis problemov, nato opišemo različne vloge, ki jih ima ocenjevanje atributov v strojnem učenju, in kriterije uspešnosti algoritmov za ocenjevanje kvalitete atributov. Končamo s kratkim pregledom postopkov za ocenjevanje atributov.

2.1 Atributni opis problemov

Predpostavili bomo, da so učni primeri $I_1, I_2, ..., I_n \in \mathcal{I}$ opisani z atributi (lastnostmi) $A_1, A_2, ..., A_a \in \mathcal{A}$ ter označeni z vrednostjo τ_i. Podanih je torej n točk v a razsežnem prostoru. Če je označba (imenujemo jo tudi napovedana vrednost, predikcija ali funkcijska vrednost) imenska vrednost, imenujemo problem klasifikacijski in označbo razred, če pa je številska, govorimo o regresijskem problemu ter o številskem razredu ali številski funkcijski vrednosti. Glede na kontekst označujemo napovedano vrednost tudi s C ali f. Pri klasifikacijskih problemih označimo število razredov s c, vrednost j-tega razreda pa s c_j.

Definirajmo tudi pojma učna in testna množica primerov. Prvi predstavlja učne primere s pomočjo katerih učni algoritem sestavi model problema, drugi pa učne primere s katerimi vrednotimo naučeni model.

2.2 Nameni ocenjevanja atributov

Postopki za ocenjevanje kvalitete atributov dobijo kot svoj vhod opis problema v atributni obliki, vrnejo pa za vsak atribut njegovo oceno kvalitete. To je ponavadi številska vrednost, kar omogoči ureditev atributov glede na oceno. Oceno in vrstni red atributov interpretiramo glede na postopek ocenjevanja in glede na vlogo, ki jo ima ocenjevanje kvalitete atributov pri učenju.

Ocenjevanje kakovosti atributov je sestavni del več pomembnih podpodročij strojnega učenja: izbire podmnožice pomembnih atributov, uteževanja atributov, konstruktivne indukcije, izgradnje odločitvenih in regresijskih dreves, itd.

Številni učni problemi so lahko opisani z več sto atributi. Večina učnih metod ni primernih za take probleme, saj primeri z mnogo nepomembnimi, šumnimi atributi vsebujejo le malo informacije pa tudi računska zahtevnost je prevelika. Postopki za izbiro podmnožice pomembnih atributov nam v idealnem primeru vrnejo majhno, toda potrebno in zadostno množico atributov za opis danega koncepta (Blum in Langley, 1997; Kira in Rendell, 1992a). Za odločanje o tem, katere atribute izbrati v tako množico, potrebujemo zanesljivo in praktično učinkovito metodo za ocenjevanje kakovosti atributov v povezavi s ciljnim konceptom. Uspešnost teh metod sodimo glede na njihovo računsko učinkovitost ter glede na uspešnost metode, ki uporablja izbrane atribute. Če izbiramo na primer podmnožico pomembnih atributov za nadaljnjo obdelavo z nevronsko mrežo, je uspešnost postopka za ocenjevanje atributov definirana z uspešnostjo nevronske mreže, za katero je ponavadi kriterij uspešnosti kar najmanjša napaka napovedi na testni množici primerov.

Na podoben problem naletimo v konstruktivni indukciji, kjer zato, da bi povečali izrazno moč opisnega jezika in skonstruirali novo znanje, uvajamo (številne) konstrukte (Brodley in Utgoff, 1995). Ti so sestavljeni iz več atributov, vrednosti atributov ali intervalov vrednosti atributov ter povezani z različnimi operatorji (npr. kartezični produkt vrednosti, konjunkcija vrednosti in intervalov vrednosti, vsota atributov, ...). Za izbiro atributov, ki jih je smiselno združevati, in za vrednotenje konstruktov potrebujemo postopek za ocenjevanje njihove kakovosti v povezavi s ciljnim konceptom. Uspešnost ocenjevanja sodimo glede na uspeh postopka konstruktivne indukcije: ali je učni postopek, ki uporablja konstruktivno indukcijo uspešen (napaka na učni množici, razumljivost in kompleksnost naučenega modela, računska učinkovitost), ali smo dobili smiselne in ljudem razumljive konstrukte, ali ti definirajo nek pomenski podkoncept v našem problemu,

itd.

Odločitvena in regresijska drevesa so v strojnem učenju priljubljen opisni jezik (Breiman in sod., 1984; Quinlan, 1993a). Med njihovo gradnjo se učni algoritem v vsakem notranjem vozlišču odloča za pravilo, ki razbije problemski prostor. To pravilo je lahko en sam atribut ali pa izraz sestavljen iz več atributov. Postopek za ocenjevanje kakovosti teh pravil pomembno vpliva na uspešnost učenja (Breiman in sod., 1984; Kononenko in sod., 1997). Merilo uspešnosti postopka za ocenjevanje kakovosti teh pravil je spet uspešnost učenja, ki je pri drevesnih modelih, poleg kar najmanjše napake napovedi na testni množici primerov dostikrat tudi struktura, razumljivost in zapletenost drevesa, ki jo ocenijo strokovnjaki z danega področja.

Tudi pri drugih uporabah postopkov za ocenjevanje atributov v strojnem učenju sodimo njihovo uspešnost glede na kriterije uspešnosti učenja.

Za ugotavljanje pravilnosti postopkov za ocenjevanje atributov bi želeli imeti jasno definirana merila, da bi lahko primerjali različne ocene atributov. Žal lahko pri realnih problemih ocene kvalitete primerjamo le posredno preko uspešnosti učenja. Na umetnih problemih, za katere poznamo pomen in vrsto atributov, pa lahko neposredno ugotavljamo različne lastnosti ocen kvalitete in jih primerjamo med seboj. Tovrstno analizo za algoritme Relief predstavljamo v 6. poglavju.

2.3 Postopki za ocenjevanje kakovosti atributov

V literaturi so opisani številni postopki za ocenjevanje kakovosti atributov. V tem razdelku predstavljamo nekaj najpogosteje uporabljanih. Ker jih večina temelji na entropiji, najprej definiramo entropijo vrednosti razreda H_C, entropijo vrednosti atributa H_A, entropijo sopojavitve vrednosti razreda in vrednosti atributa H_{CA} in entropijo vrednosti razreda pri podani vrednosti atributa $H_{C|A}$.

$$H_C = -\sum_i p_{c_i} \log p_{c_i} \tag{2.1}$$

$$H_A = -\sum_j p_{A_j} \log p_{A_j} \tag{2.2}$$

$$H_{CA} = -\sum_i \sum_j p(c_i a_j) log p_{c_i a_j} \tag{2.3}$$

$$H_{C|A} = -\sum_i \sum_j p_{c_i|a_j} \log p_{c_i|a_j}, \tag{2.4}$$

kjer so vsi logaritmi dvojiški. Verjetnost p_{c_i} označuje verjetnost i-te vrednosti razreda, p_{a_j} predstavlja verjetnost j-te vrednosti atributa, $p(c_i a_j)$ je verjetnost sopojavitve i-te vrednosti razreda in j-te vrednosti atributa in $p_{c_i|a_j}$ pomeni verjetnost i-te vrednosti razreda pri realizaciji j-te vrednosti atributa. Te verjetnosti

ponavadi ocenjujemo na učni množici primerov s pomočjo relativnih frekvenc. Številske atribute je potrebno za uporabo v teh merah najprej diskretizirati.

Informacijski prispevek (Hunt in sod., 1966) je definiran na klasifikacijskih problemih:

$$gain(A) = H_C - H_{C|A} \tag{2.5}$$

Informacijski prispevek si razložimo kot zmanjšanje entropije (pribitek informacije) vrednosti razreda, ko izvemo za vrednost atributa. Ta mera precenjuje večvrednostne atribute. To slabost odpravlja razmerje informacijskega prispevka.

Razmerje informacijskega prispevka (Quinlan, 1993a) izhaja iz informacijskega prispevka in je definirano kot

$$gainRatio(A) = \frac{gain(A)}{entropy(A)} = \frac{H_C - H_{C|A}}{H_A}, \tag{2.6}$$

Razmerje informacijskega prispevka si razložimo na enak način kot informacijski prispevek, le da uvedemo še normalizacijo z entropijo atributa, ki odpravi precenjevanje večvrednostnih atributov.

Mero razdalje (Mantaras, 1989) lahko zapišemo kot

$$1 - D = \frac{gain(A)}{H_{CA}} \tag{2.7}$$

in je podobna razmerju informacijskega prispevka, le da informacijski prispevek normaliziramo z entropijo sopojavitve vrednosti razreda in vrednosti atributa.

Indeks Gini (Breiman in sod., 1984) je definiran na klasifikacijskih problemih kot

$$Gini = \sum_j p(a_j) \sum_i p^2_{c_i|a_j} - \sum_i p^2_{c_i} \tag{2.8}$$

in poskuša ovrednotiti čistost distribucije vrednosti razredov po razdelitvi primerov v podmnožice glede na vrednosti atributov.

J-ocena (Smyth in Goodman, 1990) je definirana na klasifikacijskih problemih in skuša oceniti vsoto informacij posameznih vrednosti atributa:

$$J = \sum_j p(a_j) \sum_i p_{c_i|a_j} \log \frac{p_{c_i|a_j}}{p(c_i)} \tag{2.9}$$

Srednja kvadratna napaka (MSE) kot ocena kakovosti atributa A je definirana za regresijske probleme in je definirana kot po vseh možnih dvojiških razbitjih atributa A minimizirana utežena srednja kvadratna napaka prediktorja ϕ. Za ocenjevanje atributov uporabljamo prediktor ϕ, ki vrne povprečje napovedanih vrednosti primerov (Breiman in sod., 1984).

$$MSE(\phi, A) = \min_{razbitje\ A} p_L \cdot R_{t_L}(\phi) + p_R \cdot R_{t_R}(\phi), \tag{2.10}$$

kjer sta t_L in t_R podmnožici primerov, ki ju definira razbitje glede na atribut A, p_L in p_R pa deleža primerov, ki ustrezata temu razbitju. $R_t(\phi)$ je srednja kvadratna napaka prediktorja ϕ v primerjavi z dejanskimi napovedanimi vrednostmi τ_i primerov I_i na podmnožici primerov t:

$$R_t(\phi) = \frac{1}{n_t} \sum_{i=1}^{n_t} (\tau_i - \phi(I_i))^2 \tag{2.11}$$

$\phi(I_i)$ je vrednost, ki jo vrne ϕ. MSE razlagamo z ločevanjem manjših od večjih napovedanih vrednosti tako, da minimiziramo odstopanja od povprečnih vrednosti razbitij.

Srednja absolutna napaka (MAE) (Breiman in sod., 1984) deluje na regresijskih problemih in je definirana podobno kot MSE, le da namesto kvadrata odstopanj vzamemo absolutno vrednost odstopanj, namesto povprečja napovedanih vrednosti pa mediano. Interpretiramo jo na enak način kot MSE.

Predstavljene mere za oceno kakovosti atributov, predpostavljajo, da so atributi pogojno neodvisni od razreda in so zato neprimerne za uporabo v problemih, kjer morda obstajajo močne pogojne odvisnosti med atributi.

Družina algoritmov Relief ne predpostavlja neodvisnosti atributov. Ti algoritmi se zavedajo konteksta drugih atributov, so učinkoviti in zanesljivo ocenjujejo kakovost atributov tudi v problemih z močnimi pogojnimi odvisnostmi med atributi.

Kot smo že omenili, so algoritmi Relief med najuspešnejšimi algoritmi za ocenjevanje atributov in se uspešno uporabljajo v številnih nalogah strojnega učenja. V naslednjem poglavju jih bomo natančno predstavili.

3

Algoritmi Relief

Resnično raziskovalno popotovanje ni v odkrivanju novih dežel ampak v gledanju z novimi očmi.

Marcel Proust

Algoritmi Relief poskušajo oceniti kakovost atributov glede na to, kako dobro vrednosti atributov ločijo med primeri, ki so si podobni.

Kot vhod dobijo algoritmi opis učnih primerov ter različne parametre, vrnejo pa za vsak atribut oceno kakovosti $W[A]$, ki je vrednost na intervalu $[-1, 1]$. Pomembnejši atributi dobijo večjo vrednost ocene. Nepomembni atributi dobijo v praksi vrednosti, ki so manjše ali enake 0. Uteži $W[A]$ definirajo razvrstitev atributov glede na oceno kvalitete. Uteži in razvrstitev uporabimo glede na vlogo ocenjevanja atributov pri učenju, kot smo to opisali v prejšnjem poglavju.

Algoritmi Relief obravnavajo imenske in številske atribute. Urejenostne atribute pri katerih so vrednosti linearno urejene, a njih razlika nima pomena (npr. mrzlo, hladno, mlačno, toplo, vroče) obravnavajo kot imenske atribute. Pri tem zaradi načina ocenjevanja ne izgubimo nobene informacije, saj lahko linearno urejenost upoštevamo šele pri nadaljnjih postopkih učenja (npr. izbiranju delitvene vrednosti pri izgradnji odločitvenega drevesa).

V tem poglavju opisujemo družino algoritmov Relief ter preučujemo podobnosti in razlike med posameznimi algoritmi. Zaradi lažjega razumevanja si najprej pogledamo osnovni algoritem Relief (Kira in Rendell, 1992b), ki je omejen na dvorazredne klasifikacijske probleme. Pojasnimo, kako in zakaj deluje. Preko te analize pridemo do njegove razširitve ReliefF (Kononenko, 1994), ki je bolj robusten, zna ocenjevati tudi večrazredne probleme in lahko obravnava šumne in manjkajoče vrednosti. Nazadnje predstavimo prilagoditev tega algoritma za regresijske probleme, ki se imenuje RReliefF (Robnik Šikonja in Kononenko, 1997).

3.1 Relief - osnovne ideje

Algoritem Relief (Kira in Rendell, 1992b) je omejen na dvorazredne klasifikacijske probleme in je opisan na sliki 3.1.

Algoritem Relief
Vhod: za vsak učni primer vektor vrednosti atributov in razreda, parameter m
Izhod: vektor ocen kakovosti atributov W

1. postavi vse uteži $W[A] := 0.0$;
2. **for** i := 1 **to** m **do begin**
3. naključno izberi učni primer R_i;
4. poišči najbližji zadetek H in najbližji pogrešek M;
5. **for** A := 1 **to** a **do**
6. $W[A] := W[A] - \mathrm{diff}(A, R_i, H)/m + \mathrm{diff}(A, R_i, M)/m$;
7. **end**;

Slika 3.1: Psevdo koda osnovnega algoritma Relief.

Algoritem najprej naključno izbere učni primer R_i (3. vrstica) in poišče dva, njemu najbližja soseda: enega iz istega razreda, ki ga imenujemo bližnji zadetek H, in drugega iz različnega razreda, ki ga imenujemo bližnji pogrešek M (4. vrstica). Glede na vrednosti atributov pri primerih R_i, H in M (5. in 6. vrstica) algoritem popravi ocene kakovosti atributov v vektorju W. Če imata primera R_i in H različne vrednosti atributa A, potem atribut A ločuje primera iz istega razreda, kar ni dobro in zato zmanjšamo oceno kakovosti $W[A]$. Če pa imata primera R_i in M različne vrednosti atributa A, potem atribut A ločuje primera iz različnih razredov, kar je lastnost, ki jo želimo pri dobrih atributih, in torej povečamo oceno kakovosti $W[A]$. Ves proces se ponovi m krat, pri čemer vrednost m določi uporabnik.

Funkcija $\mathrm{diff}(A, I_1, I_2)$ izračuna razliko vrednosti atributa A med dvema primeroma I_1 in I_2. Za imenske atribute je bila originalno definirana kot

$$\mathrm{diff}(A, I_1, I_2) = \left\{ \begin{array}{l} 0 \;\; ; vrednost(A, I_1) = vrednost(A, I_2) \\ 1 \;\; ; sicer \end{array} \right\} \qquad (3.1)$$

in za številske kot:

$$\mathrm{diff}(A, I_1, I_2) = \frac{|vrednost(A, I_1) - vrednost(A, I_2)|}{max(A) - min(A)} \qquad (3.2)$$

Funkcijo diff uporabljamo tudi za izračun razdalje dveh primerov, ko iščemo najbližje primere. Razdalja med dvema primeroma je definirana kot vsota razdalj

po vseh atributih.

$$\delta(I_1, I_2) = \sum_{i=1}^{a} \text{diff}(A_i, I_1, I_2) \tag{3.3}$$

Naključno izbiranje primerov (3. vrstica na sliki 3.1) je potrebno zaradi obvladovanja računske zahtevnosti. Če imamo na voljo malo učnih primerov, lahko upoštevamo tudi vse, vendar dobimo ponavadi uporabne rezultate že s precej manj iteracijami. Podrobneje se s tem vprašanjem ukvarjamo v razdelkih 4.1, 5.5 in 6.5.

Uteži atributov, ki jih vrnejo algoritmi Relief, se zaradi naključne izbire primerov pri večkratni ponovitvi razlikujejo. Razlike med ponovitvami se manjšajo z večanjem števila iteracij m. Isti učinek dosežemo s povprečenjem več ponovitev.

Z originalnim algoritmom Relief smo omejeni le na probleme z dvema razredoma in brez neznanih vrednosti. Razširitev algoritma, ki rešuje te probleme in je tudi bolj robustna, imenujemo ReliefF.

3.2 ReliefF - razširitev

Kononenko (1994) je razširil originalni Relief tako, da deluje na nepopolnih podatkih in na večrazrednih problemih, bistveno manj pa je občutljiv tudi na šumne podatke. Razširitev, ki jo je poimenoval ReliefF (Relief-F), je opisana na sliki 3.2.

Algoritem ReliefF
Vhod: za vsak učni primer vektor vrednosti atributov in razreda, parametra m, k
Izhod: vektor ocen kakovosti atributov W

1. postavi vse uteži $W[A] := 0.0$;
2. **for** i := 1 **to** m **do begin**
3. naključno izberi učni primer R_i;
4. poišči k najbližjih zadetkov H_j;
5. **for** vsak razred $C \neq \tau(R_i)$ **do**
6. iz razreda C poišči k najbližjih pogreškov $M_j(C)$;
7. **for** A := 1 **to** a **do**
8. $W[A] := W[A] - \sum_{j=1}^{k} \text{diff}(A, R_i, H_j)/(m \cdot k) +$
9. $\sum_{C \neq \tau(R_i)} [\frac{P(C)}{1-P(\tau(R_i))} \sum_{j=1}^{k} \text{diff}(A, R_i, M_j(C))]/(m \cdot k)$;
10. **end**;

Slika 3.2: Psevdo koda algoritma ReliefF.

Podobno kot Relief tudi ReliefF naključno izbere primer R_i (3. vrstica), vendar potem poišče k najbližjih sosedov iz istega razreda, imenovanih bližnji zadetki H_j (4. vrstica), in k najbližjih primerov iz vsakega od različnih razredov, imenovanih bližnji pogreški $M_j(C)$ (5. in 6. vrstica). Ocena kakovosti $W[A]$ za vse atribute A se obnovi glede na njihove vrednosti pri primerih R_i, zadetkih H_j in pogreških $M_j(C)$ (7., 8. in 9. vrstica). Obrazec za popravek ocene kakovosti izhaja iz tistega pri algoritmu Relief (5. in 6. vrstica na sliki 3.1), le da povpreči prispevke vseh zadetkov in pogreškov. Prispevek vsakega razreda pri pogreških je utežen z apriorno verjetnostjo tega razreda $P(C)$, izračunano na učni množici. Ker želimo, da so prispevki zadetkov in pogreškov normalizirani na $[0, 1]$ in simetrični (razloge za to navajamo v nadaljevanju), moramo zagotoviti, da bo vsota verjetnosti pogreškov enaka 1. Ker v vsoti manjka verjetnost razreda zadetkov, moramo deliti utež vsakega pogreška s faktorjem $1 - P(\tau(R_i))$, kar predstavlja vsoto verjetnosti vseh pogreškov. Ves proces se m krat ponovi.

Najpomembnejša razširitev glede na originalni Relief je upoštevanje k bližnjih zadetkov in pogreškov namesto enega. Parameter k lahko določi uporabnik in z njim nadzoruje lokalnost ocen. Za večino namenov ga lahko postavimo na 10 (glej (Kononenko, 1994) in diskusijo v nadaljevanju). Ta sprememba prispeva tudi k robustnosti algoritma in njegovi neobčutljivosti za šum. Razširitev na več razredov je dosežena z uteženo vsoto prispevkov pogreškov iz vseh razredov (9.vrstica).

Manjkajoče vrednosti upoštevamo glede na njihovo verjetnost in sicer spremenimo definicijo funkcije diff. Izračunamo pogojno verjetnost (glede na razred), da imata dva primera različni vrednosti atributa:

- neznano vrednost imenskega atributa ima en primer (npr. I_1):

$$\mathrm{diff}(A, I_1, I_2) = 1 - P(vrednost(A, I_2)|\tau(I_1)) \tag{3.4}$$

- neznani sta obe vrednosti imenskega atributa

$$\mathrm{diff}(A, I_1, I_2) = 1 - \sum_V (P(V|\tau(I_1)) \times P(V|\tau(I_2))) \tag{3.5}$$

  kjer V predstavlja neko vrednost atributa A, pogojne verjetnosti pa so ocenjene z njihovo relativno frekvenco na učni množici.

Manjkajoče vrednosti številskih atributov obravnavamo zelo podobno. Namesto verjetnosti upoštevamo neko aproksimacijo gostote verjetnosti, npr. jedrne funkcije (Smyth in Mellstrom, 1992; Redner in Walker, 1984).

3.3 RReliefF - v regresiji

Opis družine algoritmov Relief končajmo z algoritmom RReliefF (Regresijski ReliefF) (Robnik Šikonja in Kononenko, 1997). Poglejmo najprej, kaj pravzaprav izračunava algoritem Relief.

Ocena kakovosti atributa $W[A]$, ki jo izračuna Relief, je približek razlike naslednjih verjetnosti (Kononenko, 1994):

$$\begin{aligned} W[A] &= P(\text{različna vrednost A}|\text{najbližji primer iz različnega razreda}) \\ &- P(\text{različna vrednost A}|\text{najbližji primer iz istega razreda}) \end{aligned} \quad (3.6)$$

Pozitivni popravki uteži (6. vrstica na sliki 3.1 in 9. vrstica na sliki 3.2) tvorijo pravzaprav oceno verjetnosti, da atribut loči med primeri iz različnega razreda, negativni popravki (6. vrstica na sliki 3.1 in 8. vrstica na sliki 3.2) pa oceno verjetnosti, da atribut loči med primeri iz istega razreda. V regresijskih problemih je napovedana vrednost τ številska, zato ne moremo uporabiti zadetkov in pogreškov. Namesto, da bi enolično določili pripadnost razredu, uvedemo neke vrste verjetnost, da sta si dva primera različna. To verjetnost modeliramo z relativno razdaljo med napovedanima vrednostma obeh primerov.

Za oceno $W[A]$ v enačbi (3.6) potrebujemo še predznaka obeh členov. V sledeči izpeljavi bomo preoblikovali enačbo (3.6) tako, da jo bomo lahko ovrednotili z uporabo verjetnosti, da sta dva primera različna. Če zapišemo

$$P_{diffA} = P(\text{različna vrednost A}|\text{bližnji primeri}) \quad (3.7)$$

$$P_{diffC} = P(\text{različen razred}|\text{bližnji primeri}) \quad (3.8)$$

in

$$P_{diffC|diffA} = P(\text{različen razred}|\text{različna vrednost A in bližnji primeri}) \quad (3.9)$$

dobimo z uporabo Bayesovega pravila iz (3.6) naslednjo enačbo:

$$W[A] = \frac{P_{diffC|diffA} P_{diffA}}{P_{diffC}} - \frac{(1 - P_{diffC|diffA}) P_{diffA}}{1 - P_{diffC}} \quad (3.10)$$

Torej lahko ocenimo $W[A]$ tako, da aproksimiramo izraze (3.7), (3.8) in (3.9). To naredi algoritem na sliki 3.3.

Podobno kot pri algoritmu ReliefF naključno izberemo primer R_i (3. vrstica) in k njemu najbližjih primerov I_j (4. vrstica). Uteži za različen razred, različen atribut ter za različna razred in atribut zbiramo v N_{dC}, $N_{dA}[A]$ in $N_{dC\&dA}[A]$. Oceno vsakega atributa $W[A]$ (po enačbi (3.10)) izračunamo v 14. in 15. vrstici.

Algoritem RReliefF
Vhod: za vsak učni primer vektor vrednosti atributov in napovedane vrednosti, parametra m in k
Izhod: vektor ocen kakovosti atributov W

1. postavi vse N_{dC}, $N_{dA}[A]$, $N_{dC\&dA}[A]$, $W[A]$ na 0;
2. **for** i := 1 **to** m **do begin**
3. naključno izberi primer R_i;
4. izberi **k** primerov I_j, ki so najbližji R_i;
5. **for** j := 1 **to** k **do begin**
6. $N_{dC} := N_{dC} + \text{diff}(\tau, R_i, I_j) \cdot d(i,j)$;
7. **for** A := 1 **to** a **do begin**
8. $N_{dA}[A] := N_{dA}[A] + \text{diff}(A, R_i, I_j) \cdot d(i,j)$;
9. $N_{dC\&dA}[A] := N_{dC\&dA}[A] + \text{diff}(\tau, R_i, I_j) \cdot$
10. $\text{diff}(A, R_i, I_j) \cdot d(i,j)$;
11. **end**;
12. **end**;
13. **end**;
14. **for** A := 1 **to** a **do**
15. $W[A] := N_{dC\&dA}[A]/N_{dC}$ - $(N_{dA}[A] - N_{dC\&dA}[A])/(m - N_{dC})$;

Slika 3.3: Psevdo koda algoritma RReliefF.

Z uporabo člena $d(i,j)$ na sliki 3.3 (vrstice 6, 8 in 10) upoštevamo razdaljo med dvema primeroma R_i in I_j. Bližnji primeri naj bi imeli večji vpliv, zato z naraščajočo razdaljo od primera R_i eksponentno manjšamo vpliv primera I_j:

$$d(i,j) = \frac{d_1(i,j)}{\sum_{q=1}^{k} d_1(i,q)} \quad \text{in} \tag{3.11}$$

$$d_1(i,j) = e^{-\left(\frac{\text{rank}(R_i, I_j)}{\sigma}\right)^2} \tag{3.12}$$

kjer funkcija $\text{rank}(R_i, I_j)$ vrne rang (vrstni red) primera I_j v padajoče urejenem zaporedju razdalj primerov od primera R_i, parameter σ pa uravnava hitrost padanja vpliva z razdaljo in ga določi uporabnik.

Ker želimo ostati pri verjetnostni interpretaciji rezultatov (glede na izraz (3.6)), normaliziramo prispevek vsakega od k bližnjih primerov tako, da ga delimo z vsoto vseh k prispevkov. Rang uporabljamo namesto dejanskih razdalj zato, ker so dejanske razdalje odvisne od konkretnega problema, uporaba uteževanja po vrstnem redu pa zagotavlja, da imajo najbližji primeri vedno enak vpliv na utež.

ReliefF je uporabljal konstanten vpliv k bližnjih primerov. Pri algoritmu RReliefF lahko dosežemo to tako, da postavimo

$$d_1(i,j) = 1/k \tag{3.13}$$

Podrobneje se s temi vprašanji ukvarjamo v nadaljevanju.

4

Teoretična analiza

Hipoteza ali teorija je lahko jasna, prepričljiva in nedvoumna, toda nihče razen avtorja ne verjame vanjo, medtem ko so eksperimentalna odkritja lahko neurejena in nenatančna, pa jim zaupajo vsi razen avtorja.

Harlow Shapley

V prejšnjem poglavju predstavljamo vse tri algoritme, v tem pa se lotimo njihove teoretične analize. Začnemo z računsko zahtevnostjo ter vzpostavimo relacijo med ocenami algoritmov Relief in funkcijami nečistoče. Predstavimo splošen okvir, v katerega lahko postavimo algoritme za ocenjevanje kakovosti atributov, ki temeljijo na kontekstni podobnosti/različnosti med vrednostmi atributa in napovedano vrednostjo. V ta okvir umestimo algoritme Relief in jih primerjamo s sorodnimi algoritmi. Ogledamo si relacijo s funkcijami nečistoče ter končamo z asimptotično interpretacijo ocen, ki jih vrne Relief.

4.1 Računska zahtevnost

Za n učnih primerov in a atributov naredi Relief (slika 3.1) $O(m \cdot n \cdot a)$ korakov. Najbolj zahtevna operacija je izbira bližnjega zadetka in pogreška, za kar moramo izračunati razdaljo med naključno izbranim primerom R in vsemi drugimi primeri, kar zahteva $O(n \cdot a)$ primerjav.

Čeprav se zdita ReliefF (slika 3.2) in RReliefF (slika 3.3) bolj zapletena, je njuna asimptotična časovna zahtevnost enaka kot pri algoritmu Relief, torej $O(m \cdot n \cdot a)$. Najzahtevnejša operacija znotraj glavne zanke *for* je izbira k najbližjih primerov, za kar moramo izračunati razdalje vseh primerov do R, kar lahko za n primerov storimo v $O(n \cdot a)$ korakih. Ostale operacije zahtevajo manj korakov. Kopico lahko zgradimo v $O(n)$ korakih, k primerov pa z nje odstranimo v $O(k \log n)$ korakih, to pa je manj kot $O(n \cdot a)$.

Podatkovna struktura k-d drevo (Bentley, 1975; Weiss, 1995) je posplošitev dvojiškega drevesa, ki namesto enega ključa uporablja k ključev (dimenzij) za razvrščanje elementov. Vsako notranje vozlišče k-d drevesa ima dva naslednika in razbije elemente glede na eno od k dimenzij v dve skupini. Koren drevesa vsebuje vse elemente, ki jih rekurzivno delimo. Rekurzija se ustavi, ko v vozlišču ostane manj kot vnaprej določeno število elementov. Ta struktura nam omogoča, da učinkovito najdemo nekemu elementu najbližje sosede (Friedman in sod., 1975; Broder, 1990), zagotoviti pa moramo, da izbrana razsežnost maksimizira varianco vrednosti v njej in da elemente delimo v približno enako velike skupine. Drevo, ki zagotavlja te lastnosti, lahko za n elementov zgradimo v $O(k \cdot n \cdot \log n)$ korakih. Algoritem za iskanje bližnjih sosedov (Friedman in sod., 1975), ki poišče t najbližjih sosedov danega element lahko opišemo rekurzivno. Parameter algoritma je vozlišče drevesa. Začnemo v korenu. Če je dano vozlišče list, poskusimo vse elemente uvrstiti v prioritetno vrsto t najbližjih elementov. Če smo v notranjem vozlišču, rekurzivno nadaljujemo iskanje v poddrevesu, ki vsebuje elemente na isti strani delitvene točke izbrane razsežnosti kot referenčni element. Ko se vračamo iz rekurzije, preverimo ali je potrebno preiskati tudi elemente na drugi strani referenčne točke, ali pa so morda ti že preveč oddaljeni od trenutno t-tega najbolj oddaljenega elementa. Pričakovana časovna zahtevnost za t najbližjih sosedov je $O(\log n)$.

Z uporabo k-d dreves za iskanje bližnjih primerov lahko zmanjšamo zahtevnost vseh treh algoritmov Relief na $O(a \cdot n \cdot \log n)$ (Robnik Šikonja, 1998). Za algoritem Relief najprej zunaj glavne zanke zgradimo optimizirano k-d drevo v $O(a \cdot n \cdot \log n)$ korakih, tako da v zanki potrebujemo le še $O(m \cdot a)$ korakov in zahtevnost celotnega algoritma je zdaj enaka zahtevnosti predprocesiranja, ki je $O(a \cdot n \cdot \log n)$. Zahtevana velikost vzorca m je povezana s zahtevnostjo problema (in ne s številom primerov) in je tipično dosti večja kot $\log n$, torej smo asimptotično zmanjšali zahtevnost algoritma.

Računska zahtevnost algoritmov ReliefF in RReliefF je tudi z uporabo k-d dreves enaka kot pri algoritmu Relief. Potrebujeta $O(a \cdot n \cdot \log n)$ korakov za izgradnjo k-d drevesa in v glavni zanki izbereta t bližnjih primerov v $\log n$ korakih, popravita uteži v $O(t \cdot a)$ korakih, toda $O(m(t \cdot a + \log n))$ je asimptotično manj kot predprocesiranje, kar pomeni, da se je zahtevnost zmanjšala na $O(a \cdot n \cdot \log n)$. Zaključimo lahko, da je asimptotična zahtevnost algoritmov Relief enaka zahtevnosti algoritmov za urejanje z več ključi.

Uporaba k-d dreves postane neučinkovita z naraščanjem števila atributov (Friedman in sod., 1975; Deng in Moore, 1995; Moore in sod., 1997) in to smo potrdili tudi za algoritme Relief (Robnik Šikonja, 1998).

Kira in Rendell (1992b) upoštevata m, kot da je poljubno izbrana konstanta

in trdita, da je zahtevnost algoritma Relief $O(a \cdot n)$. Če sprejmemo njuno interpretacijo, je zahtevnost algoritmov ReliefF in RReliefF prav tako $O(a \cdot n)$ in je analiza z uporabo k-d dreves odveč. Vendar, če želimo dobiti z algoritmi Relief smiselne in zanesljive ocene kakovosti, je potrebna velikost vzorca m povezana s zahtevnostjo problema in ni konstantna, kar bomo v naslednjih poglavjih večkrat pokazali.

4.2 Posplošitev algoritmov Relief

Algoritme Relief lahko postavimo tudi v širši okvir in nanje gledamo kot na poseben primer algoritmov za ocenjevanje kakovosti atributov, temelječih na kontekstni podobnosti/različnosti med vrednostmi atributa in napovedano vrednostjo. V tem razdelku predstavljamo takšno posplošitev in vanjo umestimo algoritme Relief.

Prepišimo enačbo (3.6) v obliko, primerno tudi za regresijo

$$\begin{aligned} W[A] &= P(\text{različna vrednost A}|\text{bližnji primeri z različno predikcijo}) \\ &- P(\text{različna vrednost A}|\text{bližnji primer z isto predikcijo}) \end{aligned} \tag{4.1}$$

Vidimo, da imamo opravek s podobnostjo (različnostjo) vrednosti atributov in napovedanih vrednosti (bližnjih primerov). Posplošitev algoritmov Relief bi upoštevala podobnost atributov A in napovedanih vrednosti τ ter ju združila v posplošeno utež:

$$W_G[A] = \sum_{I_1, I_2 \in \mathcal{I}} \text{similarity}(\tau, I_1, I_2) \cdot \text{similarity}(A, I_1, I_2) \tag{4.2}$$

kjer sta I_1 in I_2 primerno vzorčena iz populacije primerov $\mathcal{I}$. Če uporabimo mero podobnosti normalizirano na $[0, 1]$ (kot je npr. diff), lahko s temi utežmi modeliramo naslednje verjetnosti:

- $P(\text{podoben } A|\text{podoben } \tau)$, $P(\text{različen } A|\text{podoben } \tau)$ in
- $P(\text{podoben } A|\text{različen } \tau)$, $P(\text{različen } A|\text{različen } \tau)$.

Ker gre za verjetnosti, velja:

$$P(\text{podoben } A|\text{podoben } \tau) + P(\text{različen } A|\text{podoben } \tau) = 1 \tag{4.3}$$

$$P(\text{podoben } A|\text{različen } \tau) + P(\text{različen } A|\text{različen } \tau) = 1 \tag{4.4}$$

in zato že z izračunom po ene verjetnost iz vsakega para dobimo vso informacijo.

Pomislimo, kaj pričakujemo od dobrega algoritma za ocenjevanje kakovosti atributov. Kot dober naj bo ocenjen atribut, ki ločuje primere z različno napovedano vrednostjo in ne ločuje primerov s podobno napovedano vrednostjo. Tema pogojema ustrežemo, če vzamemo po eno iz vsake od zgornjih skupin verjetnosti in ju smiselno združimo. Prepišimo izraze za oceno atributov algoritma Relief iz enačbe (4.1) v naslednjo obliko:

$$\begin{aligned} W[A] &= 1 - P(\text{podoben } A|\text{različen } \tau) - 1 + P(\text{podoben } A|\text{podoben } \tau) \\ &= P(\text{podoben } A|\text{podoben } \tau) - P(\text{podoben } A|\text{različen } \tau) \end{aligned} \tag{4.5}$$

Vidimo, da je učinek algoritmov Relief tak, da sta združeni po ena verjetnost iz vsake skupine z operacijo odštevanja in torej nagrajujejo atribute, ki ne ločujejo podobnih napovedanih vrednosti in kaznujejo atribute, ki ne ločujejo različnih napovedanih vrednosti. Mera podobnosti, ki jo uporablja Relief je:

$$\text{similarity}(A, I_1, I_2) = -\text{diff}(A, I_1, I_2), \tag{4.6}$$

kar omogoča intuitivno interpretacijo rezultatov, temelječo na verjetnostih.

Iz posplošenega algoritma Relief (enačba (4.2)) lahko dobimo celo družino algoritmov za ocenjevanje kakovosti atributov tako, da uporabimo različne mere podobnosti in z njihovim kombiniranjem na različne načine.

Kot primer poglejmo algoritem Contextual Merit (CM) (Hong, 1997), ki uporablja le primere z različno predikcijo in torej upošteva le prvi člen v enačbi (4.1). CM nagrajuje atribute, ki ločujejo različne napovedane vrednosti, ne upošteva pa informacije, vsebovane v podobnih predikcijah. Posledica tega je manjša občutljivost v primerjavi z algoritmom ReliefF, npr. v problemih parnosti loči CM pomembne atribute od naključnih le za faktor 2 do 4, medtem ko je pri enakih pogojih z algoritmom ReliefF ta razlika nekaj velikostnih razredov. Del težav s številskimi atributi pri algoritmu CM prav tako izvira iz neupoštevanja drugega člena enačbe (4.1), ki kaznuje atribute, ker ločujejo primere s podobno predikcijo. Ker je to dokaj pogosto pri nepomembnih številskih atributih, mora CM težavo reševati drugače, kar pa zapleta algoritem.

Prilagoditvi algoritma ReliefF za uporabo v induktivnem logičnem programiranju (Pompe in Kononenko, 1995) in v klasifikatorju temelječem na povezovalnih pravilih (Jovanoski in Lavrač, 1999) je prav tako mogoče umestiti v zgornji okvir. Obakrat je uporabljena asimetrična mera podobnosti.

4.3 Relief in funkcije nečistoče

Ocene, ki jih vračajo algoritmi Relief, so močno povezane s funkcijami nečistoče. Naslednjo izpeljavo povzemamo po (Kononenko, 1994).

Če povečamo število bližnjih sosedov oziroma izpustimo zahtevo po bližnjih sosedih, preoblikujemo enačbo (3.6) v:

$$\begin{aligned} W'[A] &= P(\text{različna vrednost A}|\text{različen razred}) \\ &- P(\text{različna vrednost A}|\text{isti razred}) \end{aligned} \tag{4.7}$$

zdaj upoštevajmo enakosti (4.3) in (4.4) in dobimo:

$$\begin{aligned} W'[A] &= P(\text{enaka vrednost A}|\text{enak razred}) \\ &- P(\text{enaka vrednost A}|\text{različen razred}) \end{aligned} \tag{4.8}$$

Zapišimo

$$P_{eqval} = P(\text{enaka vrednost A}) = 1 - P(\text{različna vrednost A}) \tag{4.9}$$

$$P_{samecl} = P(\text{enak razred}) = 1 - P(\text{različen razred}) \tag{4.10}$$

$$\begin{aligned} P_{samecl|eqval} &= P(\text{enak razred}|\text{enaka vrednost A}) \\ &= 1 - P(\text{različen razred}|\text{enaka vrednost A}) \end{aligned} \tag{4.11}$$

in z uporabo Bayesovega pravila dobimo iz (4.8):

$$W'[A] = \frac{P_{samecl|eqval}P_{eqval}}{P_{samecl}} - \frac{(1 - P_{samecl|eqval})P_{eqval}}{1 - P_{samecl}} \tag{4.12}$$

Označimo s C razred in z V vrednosti atributa A. Za izbiranje z vračanjem v strogem smislu veljajo naslednje enakosti:

$$P_{samecl} = \sum_C P(C)^2 \tag{4.13}$$

$$P_{samecl|eqval} = \sum_V \left(\frac{P(V)^2}{\sum_V P(V)^2} \cdot \sum_C P(C|V)^2 \right) \tag{4.14}$$

Z uporabo zgornjih enakosti dobimo

$$W'[A] = \frac{P_{eqval} \cdot Ginigain'(A)}{P_{samecl}(1 - P_{samecl})} \tag{4.15}$$

kjer je

$$Ginigain'(A) = \sum_V \left(\frac{P(V)^2}{\sum_V P(V)^2} \cdot \sum_C P(C|V)^2 \right) - \sum_C P(C)^2 \tag{4.16}$$

močno koreliran s prispevkom indeksa Gini (Breiman in sod., 1984). Razlika je, da namesto faktorja $\frac{P(V)^2}{\sum_V P(V)^2}$ prispevek indeksa Gini uporablja $\frac{P(V)}{\sum_V P(V)} = P(V)$.

Enačbo (4.15) imenujemo kratkovidni ReliefF in kaže močno korelacijo med ocenami kakovosti algoritma Relief ter prispevka indeksa Gini. Verjetnost, da imata dva primera isto vrednost atributa A v enačbi (4.15) $P_{equal} = \sum_V P(V)^2$, je neke vrste normalizacijski faktor za atribute z več vrednostmi. Funkcije nečistoče težijo k precenjevanju kakovosti večvrednostnih atributov in za izničenje te anomalije se uporabljajo različne hevristike, npr. Gain ratio (Quinlan, 1986), mera razdalje (Mantaras, 1989) ali binarizacija atributov (Cestnik in sod., 1987). Enačba (4.15) priča, da Relief implicitno vsebuje to normalizacijo. Druga slabost prispevka indeksa Gini je, da se njegova vrednost zmanjšuje s povečevanjem števila razredov. Imenovalec $P_{samecl}(1 - P_{samecl})$ v enačbi (4.15), ki je konstanten pri ocenjevanju atributov, služi kot vrsta normalizacije in zato ocene kakovosti algoritmov Relief ne kažejo te anomalije. Ti normalizacijski učinki ostanejo v veljavi celo, če uporabimo enačbo (4.15) za (kratkovidno) ocenjevanje atributov (Kononenko, 1995).

V izpeljavi enačbe (4.15) iz enačbe (3.6) smo iz verjetnosti izpustili pogoj, da so primeri najbližji. Če ta pogoj vrnemo, lahko oceno, ki jo računa Relief, interpretiramo kot povprečje lokalnih ocen v različnih podprostorih celotnega problemskega prostora. Ta lokalnost omogoča algoritmu Relief, da upošteva kontekst drugih atributov kot pogojnih verjetnosti med atributi pri podani vrednosti razreda, kar lahko zaznamo le lokalno. Globalno gledano so te odvisnosti skrite zaradi povprečenja čez vse učne primere in ravno to povzroči kratkovidnost funkcij nečistoče. Funkcije nečistoče upoštevajo korelacijo med vrednostmi atributa in razreda ne glede na kontekst ostalih atributov, to pa je isto, kot če gledamo globalno in zanemarimo lokalne posebnosti.

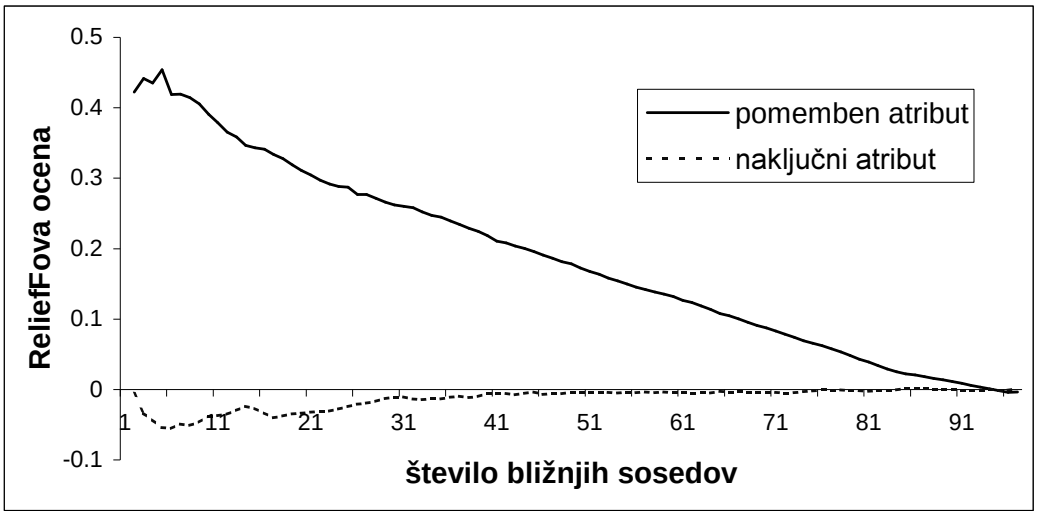
Ta pojav ponazarjamo na sliki 4.1, ki kaže odvisnost ocen algoritma ReliefF od števila upoštevanih bližnjih sosedov na problemu parnosti z dvema pomembnima, desetimi naključnimi atributi ter 200 primeri. Polna črta prikazuje, kako se ocena enega od pomembnih atributov zmanjšuje, postaja vse bolj kratkovidna in je ne moremo več ločiti od naključnih atributov.

Moč algoritmov Relief je v njihovi zmožnosti, da izkoriščajo informacijo lokalno, z upoštevanjem konteksta, kljub temu pa dajejo globalno sliko.

4.4 Konvergenca k deležu razlage koncepta

Ocene kakovosti, ki jih izračunavajo algoritmi Relief, lahko interpretiramo tudi kot delež sprememb napovedanih vrednosti, za katere je soodgovoren atribut. Natančni zapisi te lastnosti se med algoritmi Relief, ReliefF in RReliefF razlikujejo, zato jih obravnavamo vsakega posebej. Začnimo z algoritmom Relief.

Trdimo, da v dvorazrednih klasifikacijskih problemih, ko narašča število primerov v neskončnost, ocena kakovosti atributa konvergira k deležu sprememb



Slika 4.1: Ocene algoritma ReliefF za enega od pomembnih atributov v problemu parnosti pri povečevanju števila bližnjih sosedov.

vrednosti razreda, ki jih atribut določa. Če je neko spremembo vrednosti mogoče pojasniti na več načinov, si načini delijo prispevek k oceni kvalitete. Če je za nek način spremembe vrednosti razreda potrebnih več atributov, dobijo vsi enak prispevek k oceni kakovosti za ta primer.

Zapišimo to lastnost bolj formalno.

Definicija 4.1 *Označimo z $B(I)$ množico primerov iz $\mathcal{I}$, ki so najbližji primeru $I \in \mathcal{I}$, in se njihove napovedane vrednosti razlikujejo od napovedane vrednosti primera I:*

$$B(I) = \{Y \in \mathcal{I}; \mathrm{diff}(\tau, I, Y) > 0 \wedge Y = \arg\min_{Y \in \mathcal{I}} \delta(I, Y)\} \tag{4.17}$$

Označimo z $b(I)$ posamezen primer iz množice primerov $B(I)$, ter s $p(b(I))$ verjetnost njegove naključne izbire iz množice $B(I)$. Označimo z $A(I, b(I))$ množico atributov, ki imajo različno vrednost pri I in primeru $b(I)$.

$$A(I, b(I)) = \{A \in \mathcal{A}; b(I) \in B(I) \wedge \mathrm{diff}(A, I, b(I)) > 0\} \tag{4.18}$$

Rečemo, da so atributi $A \in A(I, b(I))$ odgovorni za spremembo napovedane vrednosti primera I na napovedano vrednost $b(I)$, saj je sprememba njihovih vrednosti najmanjša potrebna sprememba vrednosti atributov primera I, da se njegova napovedana vrednost spremeni v napovedano vrednost primera $b(I)$. Če so

množice $A(I, b(I))$ med seboj različne, potem pravimo, da obstajajo različne razlage spremembe napovedane vrednosti I na napovedano vrednost $b(I)$. Verjetnost posamezne razlage je enaka verjetnosti izbora primera $b(I)$.

Označimo z $A(I)$ unijo množic $A(I, b(I))$:

$$A(I) = \bigcup_{b(I) \in B(I)} A(I, b(I)) \tag{4.19}$$

Rečemo, da so atributi $A \in A(I)$ odgovorni za spremembo napovedane vrednosti primera I, saj je sprememba njihovih vrednosti najmanjša potrebna sprememba vrednosti atributov primera I, potrebnih, da se spremeni njegova napovedana vrednost.

Velikost te odgovornosti naj definira izraz, ki upošteva sočasno spremembo napovedane vrednosti in atributa:

$$r_A(I, b(I)) = p(b(I)) \cdot \mathrm{diff}(\tau, I, b(I)) \cdot \mathrm{diff}(A, I, b(I)) \tag{4.20}$$

Delež odgovornosti atributa A za določanje napovedane vrednosti množice primerov $\mathcal{S}$, moči m, je tako vsota posameznih odgovornosti:

$$R_A = \frac{1}{m} \sum_{I \in \mathcal{S}} r_A(I, b(I)) \tag{4.21}$$

Zapišimo zdaj limitno lastnost Reliefovih ocen kvalitete.

Lastnost 4.1 *Naj bo dani koncept opisan z atributi $A \in \mathcal{A}$ in podan z n nešumnimi primeri $I \in \mathcal{I}$; naj bo $\mathcal{S} \subseteq \mathcal{I}$ množica izmed njih naključno izbranih primerov, ki jih algoritem Relief po vrsti izbira (3. vrstica na sliki 3.1) ter m moč te množice. Če algoritem Relief naključno izbira najbližje primere izmed vseh mogočih najbližjih primerov, potem za njegove ocene kakovosti atributov $W[A]$ velja:*

$$\lim_{n \to \infty} W[A] = R_A \tag{4.22}$$

Oceno atributa lahko torej razložimo kot delež sprememb napovedane vrednosti, za katere je (so)odgovoren atribut.

Dokaz. Lastnost (4.22) lahko razložimo z vpogledom v prostor atributov. Algoritem Relief vzame primer R iz množice $\mathcal{S}$ ter primerja vrednost atributa in napovedane vrednosti njemu najbližjih primerov, ki jih izbere iz množice vseh primerov $\mathcal{I}$ (6. vrstica na sliki 3.1), in potem ustrezno obnovi uteži glede na te vrednosti. Pri originalnem algoritmu Relief to pomeni: $W[A] := W[A] + \mathrm{diff}(A, R, M)/m - \mathrm{diff}(A, R, H)/m$, kjer je R izbrani primer, M njemu najbližji primer iz različnega razreda in H najbližji primer iz istega razreda.

Ker imamo na voljo zadostno številom primerov $(n \to \infty)$, je H zagotovo iz istega karakterističnega razreda kot R in gredo njegove vrednosti atributov v limiti proti vrednostim primera R. Prispevek člena $\mathrm{diff}(A, R, H)$ je v limiti torej enak 0.

K oceni prispevajo le členi $\mathrm{diff}(A, R, M)$. Primer M je naključno izbran najbližji primer iz različnega razreda kot R, zato mora obstajati vsaj neka razlika v vrednostih atributov (v nešumnih problemih), M je torej primer $b(R)$ izbran z verjetnostjo $p(M)$. Ker je M iz različnega razreda kot R, zanj velja $\mathrm{diff}(\tau, R, M) = 1$. Atributi, ki imajo različne vrednosti pri primerih R in M, spadajo v množico $A(R, M)$. Prispevek primera M k uteži $W[A]$ za atribute iz $A(R, M)$ je enak
$\mathrm{diff}(A, R, M)/m = \mathrm{diff}(\tau, R, M)) \cdot \mathrm{diff}(A, R, M))/m$ z verjetnostjo $p(M))$.

Algoritem Relief obdela m naključnih primerov I, pri vsakem naključno izbere bližnji pogrešek $b(I)$ z verjetnostjo $p(b(I))$. Popravki $W[A]$ za vsakega od atributov torej znašajo $\sum_{I \in \mathcal{S}} p(b(I))\mathrm{diff}(\tau, R, b(I))\mathrm{diff}(A, R, b(I))/m = R_A$, kar je bilo treba pokazati. ■

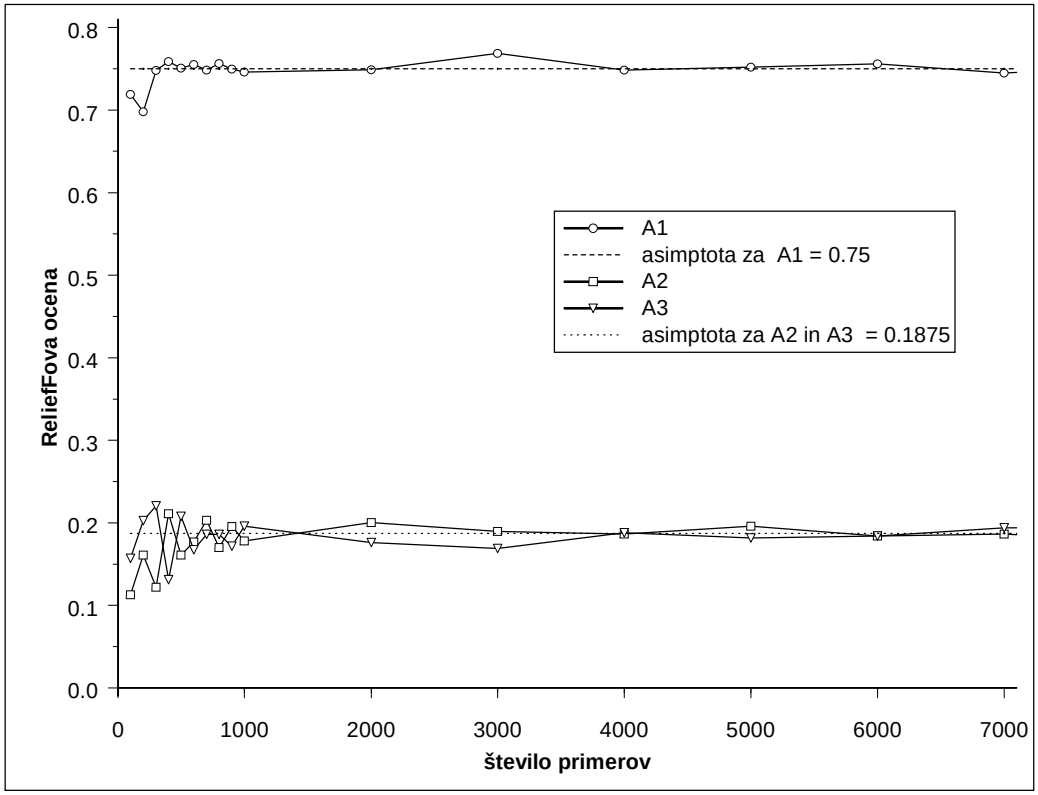
Primer. Imejmo učni problem, ki ga opisuje logična funkcija $C = (A_1 \wedge A_2) \vee (A_1 \wedge A_3)$. Tabela 4.1 predstavlja problem. Skrajno desni stolpec prikazuje odgovornost atributov za vrednost razreda.

Tabela 4.1: Tabelarični opis problema $C = (A_1 \wedge A_2) \vee (A_1 \wedge A_3)$ in porazdelitev odgovornosti med atributi.

vrstica	A_1	A_2	A_3	C	odgovornost
1	1	1	1	1	A_1
2	1	1	0	1	A_1 ali A_2
3	1	0	1	1	A_1 ali A_3
4	1	0	0	0	A_2 ali A_3
5	0	1	1	0	A_1
6	0	1	0	0	A_1
7	0	0	1	0	A_1
8	0	0	0	0	(A_1 in A_2) ali (A_1 in A_3)

V 1. vrstici je za določitev vrednosti C odgovoren atribut A_1, kajti če spremenimo njegovo vrednost na 0, bi se spremenila tudi vrednost C, medtem ko sprememba pri le enem od atributov A_2 ali A_3 nima vpliva na C. V 2. vrstici bi se vrednost C spremenila pri spremembi A_1 ali A_2, kar predstavlja dva načina za spremembo C, in ta dva načina si delita odgovornost za to spremembo. Podobno

lahko razložimo vrstice 3 do 7, medtem ko v 8. vrstici sprememba pri enem samem atributu ni dovolj za spremembo C. Potrebno je spremeniti bodisi A_1 in A_2 ali A_1 in A_3. Minimalno število potrebnih sprememb atributov je torej 2 in odgovornost za to (ter tudi pozitivni prispevki v algoritmu Relief) gre A_1 in A_2 ali A_1 in A_3. Spet imamo dva načina za spremembo in dva atributa prispevata k spremembi v vsakem od načinov. V problemu imamo 8 karakterističnih področij, ki so vsa enako verjetna pri naključnem izbiranju primerov. Atribut A_1 dobi oceno $\frac{4\cdot 1+2\cdot\frac{1}{2}+2\cdot\frac{1}{2}}{8} = \frac{3}{4} = 0.75$ (samostojno je odgovoren za 1., 5., 6. in 7. vrstico, je eden od dveh enako verjetnih načinov v 2. in 3. vrstici ter je udeležen pri obeh enako verjetnih načinih v 8. vrstici). Atribut A_2 (in podobno A_3) dobi oceno $\frac{2\cdot\frac{1}{2}+\frac{1}{2}}{8} = \frac{3}{16} = 0.1875$ (je eden od dveh enako verjetnih načinov za spremembo v 2. in 4. vrstici ter je udeležen pri enem od dveh enako verjetnih načinov za spremembo v 8. vrstici).



Slika 4.2: Ocene algoritmov Relief in ReliefF konvergirajo k deležu sprememb napovedanih vrednosti, za katere je odgovoren atribut.

Da bi razpršili koncept po prostoru smo poleg treh pomembnih v opis problema dodali še pet naključnih binarnih atributov. Slika 4.2 prikazuje kako z naraščanjem števila primerov ocena atributa A_1 konvergira k 0.75, oceni atributov A_2 in A_3 pa k 0.1875, kot smo pričakovali. ■

Obnašanje algoritma ReliefF v limiti je nekoliko drugačno. Spomnimo se, da pri tem algoritmu iščemo najbližje pogreške iz vsakega od razredov (5. in 6. vrstica na sliki 3.2), in obtežimo njihove prispevke z apriornimi verjetnostmi razredov (9. vrstica na sliki 3.2). Definirajmo torej odgovornost za spremembo razreda c_i na c_j.

Definicija 4.2 *Označimo z $B_j(I)$ množico primerov iz $\mathcal{I}$, ki so najbližji primeru $I \in \mathcal{I}, \tau(I) = c_i$, in pripadajo razredu $c_j \neq c_i$:*

$$B_j(I) = \{Y \in \mathcal{I}; \tau(Y) = c_j \wedge Y = \arg\min_{Y \in \mathcal{I}} \delta(I, Y)\} \tag{4.23}$$

Označimo z $b_j(I)$ posamezen primer iz množice primerov $B_j(I)$, ter s $p(b_j(I))$ verjetnost njegove naključne izbire iz množice $B_j(I)$. Označimo z $A(I, b_j(I))$ množico atributov, ki imajo različno vrednost pri I in $b_j(I)$.

$$A(I, b_j(I)) = \{A \in \mathcal{A}; b_j(I) \in B_j(I) \wedge \mathrm{diff}(A, I, b_j(I)) > 0\} \tag{4.24}$$

Rečemo, da so atributi $A \in A(I, b_j(I))$ odgovorni za spremembo razreda primera I na razred $b_j(I)$, saj je sprememba njihovih vrednosti najmanjša potrebna sprememba vrednosti atributov primera I, da se njegov razred spremeni v razred primera $b_j(I)$. Če so množice $A(I, b_j(I))$ med seboj različne, obstajajo različne razlage spremembe razreda $\tau(I)$ na razred $b_j(I)$. Verjetnost posamezne razlage je enaka verjetnosti izbora primera $b_j(I)$.

Označimo z $A_j(I)$ unijo množic $A(I, b_j(I))$:

$$A_j(I) = \bigcup_{b_j(I) \in B_j(I)} A(I, b_j(I)) \tag{4.25}$$

Rečemo, da so atributi $A \in A_j(I)$ odgovorni za spremembo razreda primera I na razred $j \neq i$, saj je sprememba njihovih vrednosti najmanjša potrebna sprememba vrednosti atributov primera I, potrebnih, da se njegov razred spremeni v razred j.

Velikost te odgovornosti definira izraz, ki upošteva sočasno spremembo napovedane vrednosti in atributa:

$$r_{A_j}(I, b(I)) = p(b_j(I)) \cdot \mathrm{diff}(\tau, I, b_j(I)) \cdot \mathrm{diff}(A, I, b_j(I)) \tag{4.26}$$

Delež odgovornost atributa A za spremembo razreda primerov z razredom i na razred j v množici primerov $\mathcal{S}$ moči m je vsota posameznih odgovornosti:

$$R_A(i,j) = \frac{1}{m} \sum_{I \in \mathcal{S}} r_{A_j}(I, b_j(I)) \tag{4.27}$$

Lastnost 4.2 *Naj bo dani koncept opisan z atributi $A \in \mathcal{A}$ in podan z n nešumnimi primeri $I \in \mathcal{I}$; naj bo $\mathcal{S} \subseteq \mathcal{I}$ množica izmed njih naključno izbranih primerov, ki jih algoritem ReliefF po vrsti izbira (3. vrstica na sliki 3.2) ter m moč te množice. Naj $p(c_i)$ označuje apriorno verjetnost razreda c_i. Če algoritem ReliefF naključno izbira najbližje primere izmed vseh mogočih najbližjih primerov, potem za njegove ocene kakovosti atributov $W[A]$ velja:*

$$\lim_{n\to\infty} W[A] = \sum_{i=1}^{c} \sum_{\substack{j=1 \\ j\neq i}}^{c} \frac{p(c_i)p(c_j)}{1-p(c_i)} R_A(i,j) \tag{4.28}$$

Oceno atributa lahko torej razložimo kot z apriornimi verjetnostmi razredov utežen delež sprememb razredov, za katere je (so)odgovoren atribut.

Dokaz je podoben dokazu lastnosti 4.1. Algoritem Relieff vzame primer R iz množice $\mathcal{S}$ (3. vrstica na sliki 3.2). Verjetnost, da ima primer razred c_i je enaka apriorni verjetnosti razreda $p(c_i)$. Nato iz množice vseh primerov $\mathcal{I}$ poišče k najbližjih primerov iz istega razreda in po k najbližjih primerov iz vseh drugih razredov (6. vrstica na sliki 3.1) ter primerja vrednosti atributa in razreda bližnjih primerov, ter glede na njih ustrezno obnovi uteži $W[A]$ (8. in 9. vrstica na sliki 3.2).

Ker imamo na voljo zadostno številom primerov $(n \to \infty)$, so primeri iz istega razreda H_j iz istega karakterističnega območja in gredo njihove vrednosti atributov v limiti proti vrednostim primera R. Tako gre skupni prispevek člena z bližnjimi zadetki v limiti proti 0.

K oceni prispevajo le bližnji pogreški iz različnih razredov. Ker imamo zadosti primerov $(n \to \infty)$ in so primeri M_j naključno izbrani najbližji primeri iz različnega razreda kot R, mora obstajati v nešumnih problemih vsaj neka razlika v vrednostih atributov, in so vsi primeri M_j torej primeri $b_j(R)$ izbrani z verjetnostjo $p_j(M_j)$. Prispevki k primerov se združijo glede na uteženo vsoto verjetnosti $p(b_j(R))$. Ker so M_j iz različnega razreda kot R, zanje velja $\mathrm{diff}(\tau, R, M_j) = 1$. Atributi, ki imajo različne vrednosti pri primerih R in M_j, spadajo v množico $A_j(R, b_j(R))$. Prispevek primerov M_j k uteži $W[A]$ za atribute iz $A_j(R, b_j(R))$ je enak $\sum_{\substack{j=1 \\ j\neq i}}^{c} \frac{p(c_j)}{1-p(c_i)} p(b_j(R))\mathrm{diff}(A, R_i, b_j(R))/m$.

Algoritem ReliefF obdela m naključnih primerov I, od katerih jih je $p(c_i) \cdot m$ iz razreda c_i. Pri vsakem naključno izbere bližnje pogreške $b_j(I)$, $j \neq i$ z

verjetnostmi $p(b_j(I))$. Popravki $W[A]$ za vsakega od atributov znašajo:

$$\begin{aligned} W[A] &= \sum_{I\in\mathcal{S}}\sum_{\substack{j=1\\ j\neq i}}^{c}\frac{p(c_j)}{1-p(c_i)}p(b_j(I))\text{diff}(C,I,b_j(I))\text{diff}(A,I,b_j(I))/m \\ &= \sum_{I\in\mathcal{S}}\sum_{\substack{j=1\\ j\neq i}}^{c}\frac{p(c_j)}{1-p(c_i)}r_{A_j}(I,b(I))/m, \end{aligned}$$

kar lahko zapišemo kot vsoto po razredih, saj k $R_A(i,j)$ prispevajo le primeri z razredom c_i.

$$\begin{aligned} &= \sum_{i=1}^{c}p(c_i)\sum_{\substack{j=1\\ j\neq i}}^{c}\frac{p(c_j)}{1-p(c_i)}R_A(i,j) \\ &= \sum_{i=1}^{c}\sum_{\substack{j=1\\ j\neq i}}^{c}\frac{p(c_i)p(c_j)}{1-p(c_i)}R_A(i,j), \end{aligned}$$

kot je bilo treba pokazati. ■

Posledica 4.3 *V dvorazrednih problemih, kjer je funkcija* diff *simetrična:* $\text{diff}(A,I_1,I_2)=\text{diff}(A,I_2,I_1)$*, je lastnost 4.2 enakovredna lastnosti 4.1.*

Dokaz. Ker je diff simetrična funkcija, je simetrična tudi odgovornost $R_A(i,j) = R_A(j,i)$. Zapišemo $p(c_1) = p$ in $p(c_2) = 1-p$ ter vstavimo v izraz (4.28). Z upoštevanjem dvorazrednosti dobimo $\lim_{n\to\infty} W[A] = R_A(1,2) = R_A$, ■

Na brezšumnih logičnih funkcijah, kot je ta v zgornjem primeru, dosežemo najhitrejšo konvergenco k asimptotičnim vrednostim z izbiro enega bližnjega soseda. Če uporabimo več sosedov, potrebujemo več primerov, da opazimo konvergenco, kajti utežena večina sosedov mora prihajati iz istega karakterističnega področja. Opozarjamo tudi, da je vsota izrazov (4.22) in (4.28) po vseh atributih ponavadi večja kot 1, kajti v nekaterih karakterističnih področjih je za določitev napovedane vrednosti odgovoren več kot en atribut (minimalno potrebno število sprememb je večje od 1). Primer takšnega področja je 8. vrstica iz tabele 4.1.

Če se odpovemo predpostavki o zadostnem številu atributov, so lahko ocene atributov tudi večje od njihovih asimptotičnih vrednosti, kajti atribut lahko dobi pozitivne prispevke tudi, ko je odgovoren za spremembe, večje od minimalnega potrebnega števila sprememb.

Regresijski RReliefF izračunava izraz (3.10), ki ga dobimo iz izraza (3.6), ki velja tudi za algoritem Relief. Ker k popravkom ocene kvalitete tudi v algoritmu RReliefF prispevajo le najbližji primeri z enakim vplivom kot vrednosti $r_A(I, b(I))$ pri algoritmu Relief, sklepamo, da lahko tudi za RReliefF podobno interpretiramo ocene kvalitete, namreč kot delež razloženih sprememb v konceptu. Za dokaz bi bilo smiselno preučiti parcialne odvode napovedanih vrednosti po atributih. Točen izraz in njegov dokaz pa puščamo za nadaljnje delo.

Interpretacija ocen kvalitete algoritmov Relief kot delež sprememb koncepta je pomembna za praktično uporabo. Še posebej pri reševanju problemov, kjer je potrebno sodelovanje s strokovnjaki s problemskega področja, je nujna čimbolj razumljiva interpretacija rezultatov, da bi si lahko ti ustvarili jasno miselno predstavo. Naše izkušnje na dveh praktičnih problemih iz ekologije in medicine kažejo, da je razlaga z izrazom (4.22) bolj razumljiva kot razlaga z izrazom (3.6). Podrobneje to temo obravnava 7. poglavje.

5

Parametri

Znanost? Kaj je to, če ne predvsem dolga in sistematična radovednost.
André Maurois

V tem poglavju se ukvarjamo z različnimi parametri algoritmov ReliefF in RReliefF. Preučujemo vpliv različnih metrik, uporabo številskih atributov, načine upoštevanja razdalje, število bližnjih sosedov in število iteracij.

Problemi, ki jih uporabljamo za ponazoritve in testiranje in niso definirani v tekstu, so kratko predstavljeni v dodatku A.

5.1 Metrike

Funkcija $\mathrm{diff}(A_i, I_1, I_2)$ izračunava razliko vrednosti atributa A_i za primera I_1 in I_2. Vsoto razlik po vseh atributih uporabljamo za določanje razdalje med primeri pri iskanju bližnjih primerov (glej enačbo (3.3)).

Čeprav izgleda enačba za razdaljo primerov preprosta in brez parametrov, lahko po zgledu učenja po metodi najbližjih sosedov uporabimo različne uteževalne sheme, ki posameznim atributom priredijo različne uteži:

$$\delta(I_1, I_2) = \sum_{i=1}^{a} w(A_i)\mathrm{diff}(A_i, I_1, I_2) \tag{5.1}$$

Naj omenimo, da so bile ocene atributov algoritma ReliefF uspešno uporabljene za takšno uteževanje (Wettschereck in sod., 1997).

Različne so lahko tudi uporabljene metrike razdalje, npr. razdalja Minkovskega:

$$\delta(I_1, I_2) = (\sum_{i=1}^{a} \mathrm{diff}(A_i, I_2, I_2)^p)^{\frac{1}{p}}, \tag{5.2}$$

ki nam za $p = 1$ da manhattansko razdaljo, za $p = 2$ pa evklidsko razdaljo. Pri naši uporabi algoritmov Relief nismo opazili posebnih razlik pri uporabi teh dveh metrik. Na nekaj regresijskih problemih iz zbirke UCI (Murphy in Aha, 1995) smo ocenili atribute z obema merama razdalje in vsakič izračunali (linearni, Pearsonov) korelacijski koeficient ρ ter Spearmanov (rangovni) korelacijski koeficient r. Rezultate podajamo v tabeli 5.1.

Tabela 5.1: Korelacija med ocenami atributov algoritma RReliefF pri uporabi manhattanske ali evklidske razdalje.

Problem	ρ	r
Abalone	0.993	0.952
Auto-mpg	0.998	1.0
Autoprice	0.999	0.998
CPU	0.999	1.0
Housing	0.9996	0.989
Servo	1.0	1.0
Wisconsin	0.994	0.984
Povprečje	0.998	0.989

Uporabljamo lahko tudi druge metrike, vendar smo se v tem delu omejili na manhattansko in evklidsko razdaljo, ker sta najpogosteje uporabljani v praksi. Sta enostavni, uporabni tako za imenske kot za številske atribute in ju je lahko prilagoditi za uporabo v problemih z manjkajočimi vrednostmi atributov.

5.2 Številski atributi

Uporaba funkcije diff, kot je definirana z enačbama (3.1) in (3.2), povzroči, da so številski atributi podcenjeni. Za primer vzemimo dva učna primera, ki imata pri atributu A vrednost 2 oziroma 5. Če je A imenski atribut, velja $\mathrm{diff}(A, 2, 5) = 1$, kajti imenski vrednosti sta različni. V primeru, da pa je A številski atribut, velja $\mathrm{diff}(A, 2, 5) = \frac{|2-5|}{7} \approx 0.43$. Algoritmi Relief uporabljajo vrednosti, ki jih vrača funkcija diff za obnovitev ocen kakovosti in so torej številski atributi pri uporabi te oblike funkcije diff podcenjeni.

Na levi strani tabele 5.2 v stolpcu, označenem z „brez praga”, podajamo ocene atributov algoritma RReliefF pri problemu Modulo-8-2 (glej definicijo problema z enačbo (6.7)), ki ponazarjajo to težavo. Pri tem problemu so vrednosti vsakega od 10 atributov cela števila med 0 in 7. Polovico atributov obravnavamo kot imenske, drugo polovico pa kot številske. Vsak imenski atribut je natančna slika

enega od številskih atributov. Napovedana vrednost je definirana kot vsota dveh pomembnih številskih atributov po modulo 8: $\tau = (I_1 + I_2) \bmod 8$. V tabeli vidimo, da dobijo imenski atributi skoraj dvakrat višjo oceno kakovosti kot njihove številske slike. To ne pomeni le, da so podcenjeni pomembni številski atributi, ampak so precenjeni tudi naključni številski atributi, kar zmanjša ločljivost med pomembnimi in naključnimi atributi.

Tabela 5.2: Ocene atributov algoritma RReliefF na problemu Modulo-8-2 brez in z uporabo pragovne funkcije.

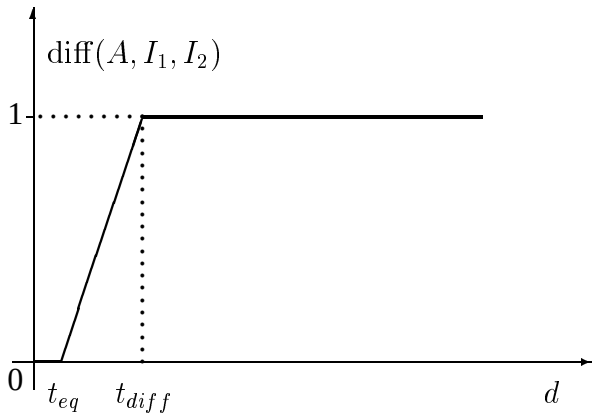
atribut	brez praga	s pragom
Pomemben-1, imenski	0.193	0.436
Pomemben-2, imenski	0.196	0.430
Naključen-1, imenski	-0.100	-0.200
Naključen-2, imenski	-0.105	-0.207
Naključen-3, imenski	-0.106	-0.198
Pomemben-1, številski	0.096	0.436
Pomemben-2, številski	0.094	0.430
Naključen-1, številski	-0.042	-0.200
Naključen-2, številski	-0.044	-0.207
Naključen-3, številski	-0.043	-0.198

Ta problem lahko obidemo s pragovno funkcijo, kot jo predlaga (Hong, 1994; 1997). Definicijo funkcije diff za številske atribute posplošimo, kot to prikazuje slika 5.1 in zapišemo z izrazom:

$$\mathrm{diff}(A, I_1, I_2) = \left\{ \begin{array}{ll} 0 & ; d \leq t_{eq} \\ 1 & ; d > t_{diff} \\ \frac{d - t_{eq}}{t_{diff} - t_{eq}} & ; t_{eq} < d \leq t_{diff} \end{array} \right\} \tag{5.3}$$

kjer $d = |vrednost(A, I_1) - vrednost(A, I_2)|$ pomeni razdaljo med vrednostma atributa dveh primerov, t_{eq} in t_{diff} pa sta pragova, ki ju definira uporabnik. t_{eq} predstavlja maksimalno razdaljo, ko dve vrednosti atributa še vedno upoštevamo kot enaki, t_{diff} pa minimalno razdaljo, da jemljemo vrednosti za različni. Če postavimo $t_{eq} = 0$ in $t_{diff} = \max(A) - \min(A)$, dobimo zopet (3.2).

Ocene atributov na problemu Modulo-8-2 pri uporabi algoritma RReliefF s pragovno funkcijo podajamo v desnem stolpcu tabele 5.2. Praga sta postavljena na njuni privzeti vrednosti in sicer t_{eq} na 5%, t_{diff} pa na 10% dolžine intervala vrednosti atributa. Vidimo, da dobijo številski atributi in njihove imenske slike zdaj enake ocene.



Slika 5.1: Vrednost funkcije diff glede na razdaljo med vrednostma atributa.

Vrednosti pragov lahko postavi uporabnik za vsak atribut posebej, kar je še posebej smiselno pri merjenih podatkih, lahko se jih naučimo vnaprej upoštevajoč kontekst (Ricci in Avesani, 1995) ali pa jim avtomatsko določimo smiselne vrednosti (Domingos, 1997). Pogled na sliko 5.1 nam porodi misel, da bi bila uporaba sigmoidne funkcije morda še splošnejša, vendar njenih parametrov ne znamo interpretirati na tako preprost način. V primeru, da ima uporabnik neko dodatno informacijo o naravi nekega atributa, lahko to informacijo izkoristi z ustrezno definicijo funkcije diff.

Pri vseh rezultatih v tem delu podajamo rezultate z uporabo pragovne funkcije.

5.3 Upoštevanje razdalje

Pri učenju po metodi najbližjih sosedov se pogosto uporablja uteževanje vpliva bližnjih primerov glede na razdaljo od primera, ki ga želimo klasificirati. Bližjim primerom dodelimo večji vpliv, bolj oddaljenim pa manjši.

Algoritem RReliefF upošteva razdaljo v izrazih (3.11) in (3.12). Na podlagi izkušenj z uporabo na več umetnih in realnih problemih, kot privzeto vrednost uporabljamo 70 bližnjih primerov in eksponentno zmanjšujemo njihov vpliv ($\sigma = 20$ v izrazu (3.12)). ReliefF je originalno uporabljal konstanten vpliv k bližnjih sosedov (s privzeto vrednostjo $k = 10$). Z uporabo več bližnjih primerov se izognemo težavi, ki smo jo opazili na primer pri uporabi na realnih podatkih (Dalaka in sod., 2000).

Imamo veliko število primerov opisanih z nekaj imenskimi in večino številskih atributov. Pri nekem izbranem primeru se lahko zgodi, da je večina bližnjih primerov na razdalji manjši od 1, kar pomeni, da nimamo bližnjih primerov z razliko v vrednostih nekaterih imenskih atributov. Če se to zgodi v velikem delu problem-

skega prostora, dobijo taki imenski atributi majhno in nezanesljivo utež. Težavo lahko rešimo s povečanjem števila ustrezno uteženih bližnjih primerov.

Algoritem ReliefF prilagodimo za upoštevanje razdalje tako, da spremenimo način obnavljanja uteži (8. in 9. vrstica na sliki 3.2):

$$W[A] := W[A] - \frac{1}{m}\sum_{j=1}^{k} \mathrm{diff}(A, R_i, H_j) d(R_i, H_j)$$

$$+ \frac{1}{m} \sum_{C \neq \tau(R_i)} \frac{P(C)}{1 - P(\tau(R_i))} \sum_{j=1}^{k} \mathrm{diff}(A, R_i, M_j(C)) d(R_i, M_j(C)) \qquad (5.4)$$

kjer je faktor razdalje dveh primerov $d(I_1, I_2)$ definiran z izrazoma (3.11) in (3.12).

Dejanski vpliv bližnjih primerov je normaliziran. Ker želimo verjetnostno interpretacijo rezultatov, mora vsak naključni primer enako prispevati k obnovitvi ocen kakovosti. Normalizacijo prispevkov v izrazu (3.11) dosežemo tako, da delimo prispevek vsakega od bližnjih primerov z vsoto prispevkov vseh k bližnjih primerov. V naših poskusih uporabljamo tudi algoritem ReliefF s 70 bližnjimi primeri in eksponentnim zmanjševanjem njihovega vpliva glede na vrstni red po padajoče urejeni razdalji od izbranega primera ($\sigma = 20$).

Določanje vpliva bližnjih primerov na podlagi vrstnega reda po padajoče urejeni razdalji zagotavlja, da imajo najbližji primeri vedno enak vpliv, vendar se s tem odpovemo morebitni lastni normalizaciji, ki jo problem vsebuje v razdaljah med primeri. Če želimo to izkoristiti in uporabljati dejanske razdalje, spremenimo izraz (3.12) v:

$$d_1(i, j) = \frac{1}{\sum_{r=1}^{a} \mathrm{diff}(A_r, R_i, I_j)} \qquad (5.5)$$

Zahtevnost algoritma se ne spremeni. Uporabimo lahko tudi katero drugo padajočo funkcijo razdalje, npr. če želimo še poudariti vpliv razdalje, lahko uporabimo obratno vrednost kvadrata razdalje:

$$d_1(i, j) = \frac{1}{(\sum_{r=1}^{a} \mathrm{diff}(A_r, R_i, I_j))^2} \qquad (5.6)$$

Dobimo lahko občutne razlike v ocenah kakovosti, vendar so te v povprečju še vedno močno korelirane, kot lahko vidimo v tabeli 5.3, ki podaja linearni (ρ) in Spearmanov (r) korelacijski koeficient med ocenami algoritma RReliefF na problemih iz zbirke UCI pri upoštevanju razdalje z enačbami (3.12), (5.5) ali (5.6).

Vzrok za precejšnje odstopanje od drugih vrednosti pri problemu Auto-mpg je občutljivost algoritma pri uporabe dejanske razdalje na število bližnjih sosedov. Medtem, ko izraz (3.12) eksponentno zmanjšuje vpliv glede na število bližjih primerov, uporabljata izraza (5.5) in (5.6) inverzno funkcijo razdalje in imajo lahko

Tabela 5.3: Korelacija med ocenami atributov algoritma RReliefF pri različnem upoštevanju razdalje: z enačbami (3.12), (5.5) ali (5.6).

Problem	(3.12) in (5.5)		(3.12) in (5.6)		(5.5) in (5.6)	
	ρ	r	ρ	r	ρ	r
Abalone	0.969	0.974	0.991	0.881	0.929	0.952
Auto-mpg	-0.486	0.174	0.389	-0.321	0.143	0.357
Autoprice	0.844	0.775	0.933	0.819	0.749	0.945
CPU	0.999	0.990	0.990	0.943	1.000	0.943
Housing	0.959	0.830	0.937	0.341	0.181	0.769
Servo	0.988	0.999	0.985	0.800	1.000	0.800
Wisconsin	0.778	0.842	0.987	0.645	0.743	0.961
Povprečje	0.721	0.798	0.888	0.587	0.678	0.818

tako tudi bolj oddaljeni primeri precejšen vpliv. S prevelikim številom bližnjih sosedov dobimo tako že kratkovidno oceno, ki ni korelirana z nekratkovidno oceno. Pri uporabi dejanske razdalje je zato potrebno uporabiti manjše število bližnjih sosedov ali pa poskusiti z več različnimi nastavitvami te spremenljivke.

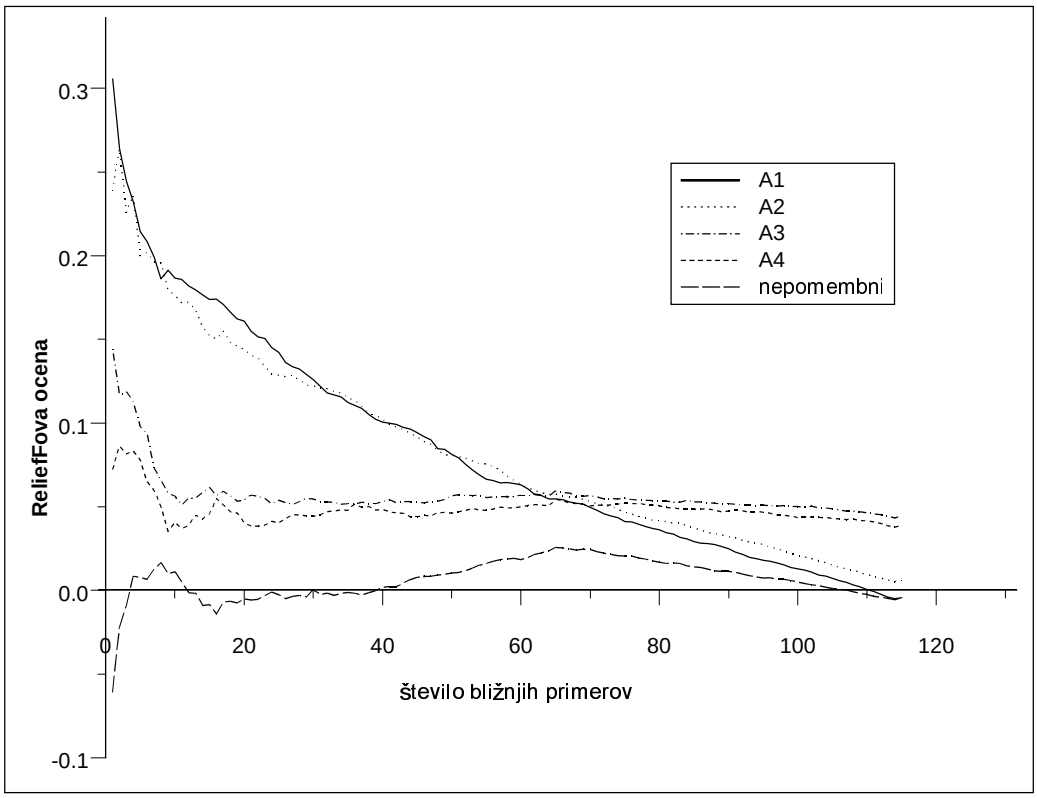
5.4 Število bližnjih sosedov

V prejšnjem razdelku smo razložili, kako je število bližnjih sosedov povezano z razdaljo, v tem pa se bomo ukvarjali z občutljivostjo algoritmov Relief na število uporabljenih bližnjih sosedov (4., 5., in 6. vrstica na sliki 3.2 ter 4. vrstica na sliki 3.3). Optimalno število bližnjih primerov je odvisno od problema in števila primerov. Slika 5.2 prikazuje ocene algoritma ReliefF za štiri pomembne in enega od naključnih atributov v logičnem problemu definiranem:

$$\mathrm{Bool-Simple:} \qquad \mathrm{C} = (\mathrm{A}_1 \oplus \mathrm{A}_2) \vee (\mathrm{A}_3 \wedge \mathrm{A}_4). \tag{5.7}$$

V učnem problemu, ki ga opisuje 200 primerov, imamo poleg štirih pomembnih še 10 naključnih atributov. Vemo, da sta atributa A_1 in A_2 pomembnejša pri določanju vrednosti C kot atributa A_3 in A_4. ReliefF to razpozna pri uporabi do 60 bližnjih sosedov. Z več kot 80 bližnjimi sosedi prevlada globalna slika in močna pogojna odvisnost med A_1 in A_2 ni več razvidna (glej sliko 5.2). Če povečamo število primerov z 200 na 900 dobimo graf podoben sliki 5.2, le da se točka zameglitve premakne od 60 na 250 bližnjih sosedov.

Predstavljeni primer dobro pokaže, da je algoritem ReliefF robusten glede števila bližnjih sosedov, dokler je to relativno majhno. Če je premajhno, se lahko



Slika 5.2: Ocene algoritma ReliefF v odvisnosti od števila bližnjih sosedov na problemu Bool-Simple z 200 učnimi primeri.

zgodi, da ni dovolj robusten, še posebej pri zahtevnih ali šumnih problemih. Če je vpliv vseh bližnjih sosedov enak, ne glede na njihovo razdaljo od izbranega primera, uporabljamo običajno 10 bližnjih primerov (Kononenko, 1994). Z upoštevanjem razdalje smo na podlagi praktičnih izkušenj določili privzeto vrednost 70 bližnjih primerov in eksponentno zmanjševanje njihovega vpliva ($\sigma = 20$ v izrazu (3.12)). Pri podobni izbiri za algoritem CM (Hong, 1997) je predlagana uporaba $\log n$ najbližjih primerov, ki daje v praksi zadovoljive rezultate. Opozoriti pa moramo, da je izbira števila bližnjih sosedov povezana s zahtevnostjo problema, količino šuma v podatkih in številom učnih primerov.

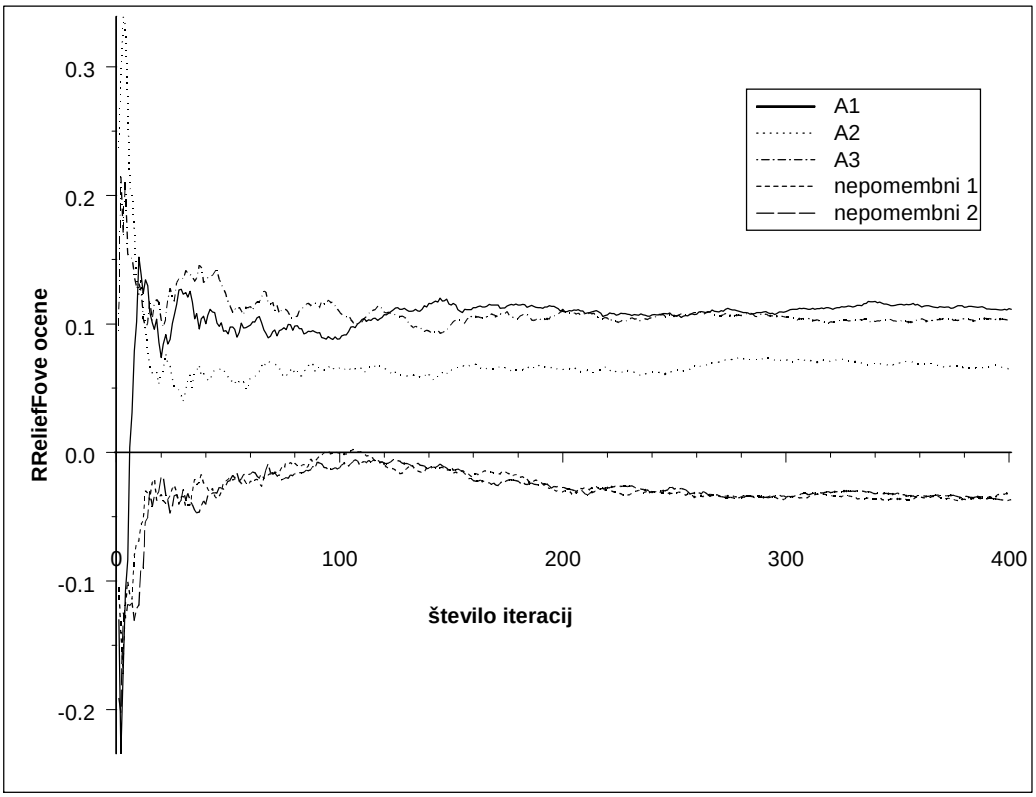
Druga možnost, s katero se izognemo številu bližnjih primerov kot parametru algoritmoc Relief, je izračun ocen kakovosti za vsa smiselna števila bližnjih sosedov. Kot končni rezultat vzamemo za vsak atribut najvišjo oceno. Na ta način se izognemo nevarnosti, da bi slučajno zgrešili pomemben atribut, vendar tvegamo, da bodo razlike med atributi nekoliko zamegljene, povečamo pa tudi računsko

zahtevnost algoritma. Nevarnosti zameglitve razlik se lahko izognemo tako, da si pripravimo graf, podoben sliki 5.2, ki prikazuje odvisnost ocen algoritma ReliefF od števila bližnjih primerov. Računska zahtevnost naraste od $O(m \cdot n \cdot a)$ na $O(m \cdot n \cdot (a + \log n))$, kajti zdaj moramo primere urediti po padajoči razdalji. V algoritem je potrebno dodati tudi nekaj več knjigovodstva, tako moramo npr. hraniti oceno vsakega atributa za vsa možna števila bližnjih sosedov.

5.5 Velikost vzorca in število iteracij

Ocene kakovosti, ki jih izračunavajo algoritmi Relief, so pravzaprav statistične ocene, kajti algoritem zbira primere (ne)razločevanja med podobnimi primeri po celotnem problemskem prostoru. Da bi dobili zanesljive ocene kakovosti, moramo zagotoviti primerno pokritost problemskega prostora. Uporabljeni vzorec mora vsebovati zadosti karakterističnih ločnic med napovedanimi vrednostmi. Odločiti se moramo med uporabo več primerov in učinkovitostjo računanja. Kadarkoli naletimo na problem z dosti primeri, je vzorčenje ena od možnosti za njegovo obvladovanje. Za probleme s precejšnjim številom primerov, kjer želimo pohitriti računanje, na podlagi praktičnih izkušenj predlagamo uporabo vseh razpoložljivih primerov (n) in nadzor nad zahtevnostjo z omejevanjem števila iteracij s parametrom m (2. vrstica na slikah 3.2 in 3.3). Ker izbira reprezentativnega vzorca, ki bi primerno pokrival ves problemski prostor nekega še neznanega problema ni enostavna, daje naš predlog prednost (morda) redkemu pokritju bolj reprezentativnega vzorca pred gostim pokritjem (morda) nereprezentativnega vzorca.

Slika 5.3 ponazarja odvisnost ocen algoritma RReliefF od števila iteracij pri problemu Cosinus-Lin, ki je definirana kot $\tau = \cos(4\pi A_1) \cdot (-2A_2 + 3A_3)$. Opisan je s 1000 učnimi primeri, tremi pomembnimi in desetimi nepomembnimi atributi. Po začetnem nihanju se pri 20 iteracijah ocene kakovosti stabilizirajo z izjemo težko ugotovljivih razlik. Za stabilno razliko v oceni med A_1 in A_3 potrebujemo okoli 300 iteracij (A_1 kot argument kosinusne funkcije določa predznak rezultata, medtem ko A_3 s koeficientom 3 odločilno določa amplitudo funkcije). Povejmo, da je to precej tipično obnašanje algoritma, saj ponavadi dobimo uporabne ocene po 20-50 iteracijah. Za boljšo ločljivost potrebujemo več iteracij, njihovo število pa je odvisno od problema. Empirično poskušamo osvetliti ta problem v razdelku 6.5.



Slika 5.3: Ocene algoritma RReliefF v odvisnosti od števila iteracij m na problemu Cosinus-Lin.

5.6 Povzetek parametrov

Povzemimo zdaj ugotovitve tega poglavja in poskušajmo sestaviti nekaj napotkov za nastavitve parametrov pri praktičnem delu z algoritmi Relief.

Ocene algoritmov ReliefF in RReliefF so se izkazale kot robustne glede na uporabo evklidske ali manhattanske razdalje. Zaradi enostavnosti računanja smo kot privzeto vrednost določili uporabo manhattanske razdalje.

Številski atributi so podcenjeni v primerjavi z imenskimi, če za njih uporabljamo linearno funkcijo razdalje, zato je za njihovo enakovredno obravnavo predlagana pragovna funkcija (5.3). Kadar na podlagi pomena atributa ne znamo nastaviti vrednosti pragov, nastavimo prag na privzete vrednosti t_{eq} na 5%, t_{diff} pa na 10% dolžine intervala vrednosti atributa.

Na podlagi praktičnih izkušenj smo za privzeto vrednost izbrali določitev vpliva bližnjih primerov na podlagi njihovega vrstnega reda po padajoče urejeni razda-

lji (izraza (3.11) in (3.12)) ter uporabo 70 bližnjih primerov in $\sigma = 20$. Algoritem ReliefF na sliki 3.2 je potrebno prilagoditi za upoštevanje razdalje tako, da spremenimo način obnavljanja uteži; 8. in 9. vrstico nadomestimo z izrazom (5.4). Pri uporabi dejanske razdalje, npr. z izrazom (5.5) ali (5.6), je potrebno manjše število bližnjih sosedov, da ne izgubimo lokalnosti konteksta.

Čeprav je izbira števila bližnjih sosedov odvisna od zahtevnosti problema, količine šuma v podatkih in števila učnih primerov, smo na podlagi praktičnih izkušenj kot privzete vrednosti določili 10 bližnjih primerov, kadar imajo vsi primeri enak vpliv ter 70 bližnjih primerov, kadar se vpliv bližnjih primerov eksponentno zmanjšuje. Število bližnjih sosedov je mogoče izločiti kot parameter tako, da izračunamo oceno kakovosti za vsa smiselna števila bližnjih sosedov ter kot končni rezultat upoštevamo za vsak atribut najvišjo oceno. Računska zahtevnost v tem primeru naraste od $O(m \cdot n \cdot a)$ na $O(m \cdot n \cdot (a + \log n))$.

Pri problemih z dosti primeri, je vzorčenje ena od možnosti za njegovo obvladovanje. Kjer želimo pohitriti računanje, predlagamo uporabo vseh razpoložljivih primerov in nadzor nad zahtevnostjo računanja z nastavitvijo števila iteracij. V praksi dobimo uporabne ocene že po okoli 50 iteracijah. Za boljšo ločljivost potrebujemo več iteracij, njihovo število pa je odvisno od problema. Pri manjših problemih je smiselno uporabiti vso razpoložljivo informacijo ter določiti število iteracij tako, da je enako številu primerov.

Vrednosti parametrov, ki smo jih priporočili v tem razdelku, uporabljamo pri vseh poskusih v tem delu, razen tam, kjer je to posebej izpostavljeno. To so privzete vrednosti za učni sistem CORE, kjer so implemetirani vsi opisani algoritmi. Sistem je kratko predstavljen v dodatku B. Privzete vrednosti so določene na podlagi lastnosti algoritmov Relief in na podlagi praktičnih izkušenj pri njihovi uporabi.

6

Empirična analiza delovanja

Karkoli je materialnega, preteklega, prihodnjega, sedanjega, subjektivnega ali objektivnega, grobega ali nežnega, slabega ali odličnega, oddaljenega ali bližnjega - vso materialnost naj popolna intuitivna modrost vidi tako, kot v resnici je: "To ni moje, to nisem jaz, to ni moj jaz."

Sidharta Gautama

V tem poglavju se ukvarjamo s praktičnimi aspekti delovanja algoritmov ReliefF in RReliefF: katere odvisnosti zaznavata, kako se obnašata pri velikem številu primerov in atributov, koliko iteracij je potrebno za zanesljive ocene, kako robustna sta glede šuma ter vpliva nepomembnih in odvečnih atributov. V bolj omejenem obsegu so bila nekatera od teh vprašanj za ReliefF zastavljena v (Kononenko, 1994; Kononenko in sod., 1997), za RReliefF pa v (Robnik Šikonja in Kononenko, 1997). Preden po vrsti analiziramo ta vprašanja, definirajmo nekaj koristnih mer in konceptov, ki jih uporabljamo pri analizi in razlagi delovanja.

6.1 Uporabne mere

Najprej prestavimo mero težavnosti problemov spremenljivost koncepta nato pa dve meri za uspešnost ocenjevanja atributov: ločljivost in uporabnost. Opišemo tudi cenilki kakovosti atributov, ki smo ju primerjali z algoritmoma ReliefF in RReliefF.

6.1.1 Spremenljivost koncepta

Algoritmi Relief temeljijo na paradigmi bližnjih sosedov in pravilno delujejo tam, kjer velja, da lahko iz podobnosti primerov sklepamo na podobnost napovedane

vrednosti. Poglejmo si mero težavnosti koncepta, imenovano spremenljivost koncepta, ki temelji na principu bližnjih sosedov.

Spremenljivost koncepta (Rendell in Seshu, 1990) je mera, ki opisuje težavnost problema v povezavi z veljavnostjo principa bližnje soseščine. Če v nekem problemu mnogo parov bližnjih sosedov ne pripada istemu razredu, imamo opraviti z veliko spremenljivostjo koncepta in problem je težak za induktivno učenje in algoritme strojnega učenja (Perèz in Rendell, 1996; Vilalta, 1999). Originalno je bila spremenljivost koncepta definirana na polnem a-razsežnem prostoru logičnih vrednosti:

$$V_a = \frac{1}{a2^a} \sum_{i=1}^{a} \sum_{\text{neigh}(X,Y,i)} \text{diff}(C, X, Y) \tag{6.1}$$

kjer notranja vsota teče po vseh 2^a parih bližnjih sosedov X in Y, ki se razlikujeta le pri i-tem atributu. Deljenje z $a2^a$ povpreči dvojno vsoto in normalizira spremenljivost na interval $[0, 1]$. Obema konstantnima konceptoma (vse vrednosti razreda so 0 oziroma 1) pripada spremenljivost 0, koncepta parnosti reda a pa imata spremenljivost 1. Naključni koncepti imajo spremenljivost okoli 0.5, kajti sosedje nekega primera so približno enakomerno porazdeljeni v oba razreda. Spremenljivost okoli 0.5 je torej visoka in problem težek.

Zgornja definicija spremenljivosti koncepta je omejena na probleme v prostoru logičnih funkcij in zahteva opis celotnega problemskega prostora, kar je za praktične probleme nemogoče. Spremenjena definicija (Vilalta, 1999) temelji na isti ideji, vendar so upoštevani tudi številski in večvrednostni imenski atributi. Prispevek para primerov I_i in I_j k spremenljivosti koncepta definira kot:

$$w(I_i, I_j) = 2^{-\alpha \cdot \frac{\delta(I_i, I_j)}{\sqrt{a} - \delta(I_i, I_j)}} \tag{6.2}$$

kjer $\delta(I_i, I_j)$ predstavlja evklidsko razdaljo med primeroma I_i in I_j, α pa definira uporabnik in uravnava vpliv razdalje. Privzeta vrednost je $\alpha = 2$, ki smo jo uporabili tudi pri primeru v tem razdelku. Člen desno od α je konstruiran tako, da doseže prispevek $w(I_i, I_j)$ zgornjo mejo 1, če je razdalja med I_i in I_j enaka 0; spodnjo mejo prispevka 0 dosežemo, ko je razdalja med primeroma enaka maksimalni razdalji v a razsežnem prostoru ($\sqrt{a}$ za evklidsko razdaljo). Spremenljivost koncepta na m primerih je tako definirana kot:

$$\bigtriangledown = \frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{m} w(I_i, I_j) \cdot \text{diff}(\tau, I_i, I_j)}{\sum_{j=1}^{m} w(I_i, I_j)} \tag{6.3}$$

Vidimo, da se uporablja vzorec primerov, zato jo je mogoče izračunati tudi za realne probleme. Težava s to definicijo je, da upošteva prispevke vseh primerov

in s tem zamegljuje lokalno informacijo. Predlagamo novo definicijo, ki združuje dobre lastnosti obeh pravkar opisanih:

$$V = \frac{1}{m \cdot a} \sum_{i=1}^{m} \sum_{j=1}^{a} \mathrm{diff}(C, \mathrm{neigh}(X_i, Y, j)) \tag{6.4}$$

kjer mora biti Y najbližji primer, ki se od primera X_i razlikuje v atributu j. Podobno kot pri originalni definiciji seštevamo razlike po vseh razsežnostih, vendar le na vzorcu m primerov. Uporabljamo funkcijo diff, ki deluje tudi na večvrednostnih in številskih atributih. V primeru, da imamo a razsežnosten logični koncept in na voljo vseh 2^a primerov, je definicija (6.4) enakovredna definiciji (6.1).

Obnašanje vseh treh definicij ponazorimo na sliki 6.1. Merili smo spremenljivost koncepta za probleme parnosti reda 2 do 9, opisanih z 9 atributi in 512 naključno generiranimi primeri (razen za definicijo 6.1, kjer smo uporabili vseh 512 možnih primerov). Po originalni definiciji (Rendell in Seshu, 1990) narašča spremenljivost linearno z redom parnosti, kar se nam zdi pravilno obnašanje, glede na težavnost učenja in paradigmo bližnjih sosedov. Pri modificirani definiciji (Vilalta, 1999) ostaja spremenljivost skoraj konstantna (0.5), medtem ko se definicija (6.4), označena z "V, 1 sosed", obnaša podobno kot originalna. Če primerov ne vzorčimo, ampak uporabimo ves problemski prostor (kot pri originalni definiciji) dobimo z našo definicijo natančno enako obnašanje kot originalni definiciji. Da je lokalnost res bistvena za definicijo spremenljivosti koncepta, se prepričamo tako, da v izrazu (6.4) povprečimo prispevke 10 bližnjih sosedov, ki se od primera X_i razlikujejo pri j-tem atributu. Pri takšni definiciji, označeni z "V, 10 sosedov", na sliki 6.1 postane obnašanje podobno kot pri definiciji iz (Vilalta, 1999).

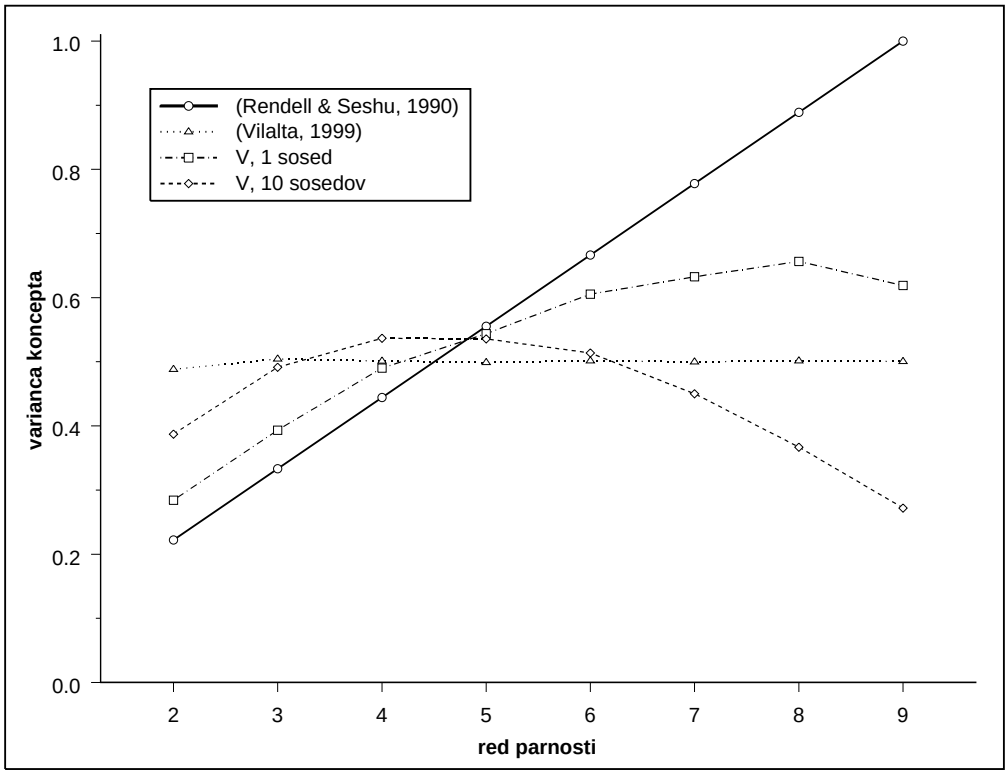
Opozarjamo, da smo v izrazu (6.4) uporabili funkcijo diff, zato lahko merimo spremenljivost koncepta tudi na regresijskih problemih.

6.1.2 Mere uspešnosti

V eksperimentalni analizi bomo opazovali delovanje algoritmov ReliefF in RReliefF na vrsti različnih problemov. Presojali bomo

- ali ocene kakovosti ločijo med pomembnimi atributi, ki vsebujejo informacijo o konceptu, in naključno generiranimi atributi ter
- ali razvrstitev atributov glede na oceno kakovosti ustreza dejanskemu vplivu atributov na koncept.

Za ocenjevanje uspešnosti algoritmov bomo uporabljali naslednji meri:



Slika 6.1: Obnašanje spremenljivosti koncepta pri različnih definicijah za koncepte parnosti reda 2 do 9 in 512 primeri.

Ločljivost s definiramo kot razliko v oceni kakovosti med najslabše ocenjenim pomembnim atributom in najboljše ocenjenim nepomembnim atributom:

$$s = W_{I_{worst}} - W_{R_{best}} \tag{6.5}$$

Pravimo, da je algoritem uspešen pri ločevanju med pomembnimi in nepomembnimi atributi, če je $s > 0$.

Uporabnost u je razlika v oceni kakovosti najbolje ocenjenega pomembnega atributa in najbolje ocenjenega nepomembnega atributa:

$$u = W_{I_{best}} - W_{R_{best}} \tag{6.6}$$

Pravimo, da so ocene uporabne, če velja $u > 0$, saj dobimo iz njih vsaj nekaj uporabne informacije (npr. najbolje ocenjeni pomembni atribut lahko uporabimo kot test v odločitvenem ali regresijskem drevesu). Velja neenakost $u \geq s$.

6.1.3 Primerjane cenilke kakovosti atributov

Pri nekaterih problemih želimo primerjati uspešnost algoritmov Relief z drugimi hevristikami za ocenjevanje kakovosti atributov. Za primerjavo smo izbrali najpogosteje uporabljane. Za klasifikacijske probleme je to razmerje informacijskega prispevka (Gain ratio), ki se uporablja npr. v sistemu C4.5 (Quinlan, 1993a), za regresijske probleme pa je to srednja kvadratna napaka povprečne napovedane vrednosti (MSE), ki jo uporablja npr. sistem CART (Breiman in sod., 1984).

Meri smo definirali v razdelku 2.3. Opozarjamo, da algoritmi Relief in Gain ratio pripišejo boljšim atributom večjo utež, MSE pa pripiše boljšim atributom manjšo utež. Zaradi primerljivosti krivulj za s in u podajamo zato za MSE te vrednosti z obrnjenim predznakom.

6.2 Razpoznavanje problemov in odvisnosti

Raziskali bomo zmožnosti algoritmov ReliefF in RReliefF, da prepoznata in po pomembnosti pravilno razvrstita atribute pri množici različnih problemov.

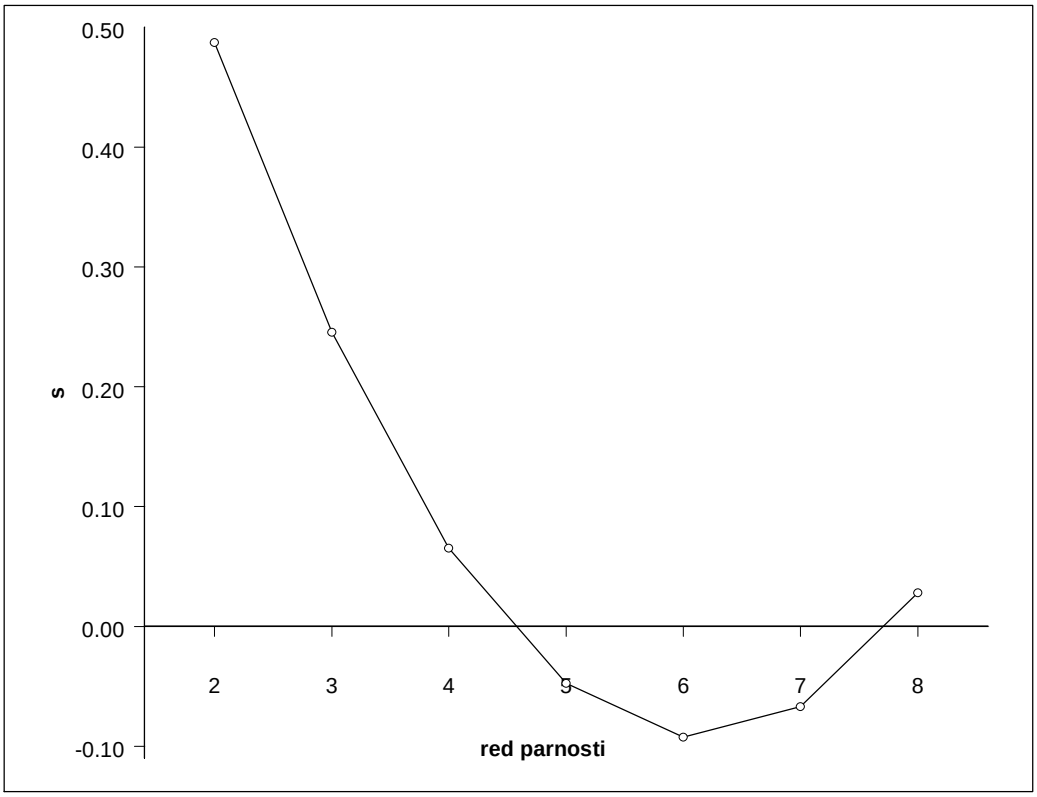
6.2.1 Vsota po modulu

Predstavitev zmožnosti algoritmov Relief in RReliefF bomo začeli s koncepti temelječimi na vsoti po modulu. Vsota po modulu p je celoštevilčna generalizacija konceptov parnosti, ki predstavljajo poseben primer, ko so atributi logične vrednosti (0 ali 1) in je koncept definiran kot vsota pomembnih atributov po modulu 2. Definirajmo družino konceptov Modulo-p-I, kjer je vsak problem opisan z množico atributov s celoštevilčnimi vrednostmi na intervalu $[0, p)$. Napovedana vrednost τ je vsota I pomembnih atributov po modulu p.

$$\mathrm{Modulo-p-I}: \qquad \tau = (\sum_{i=1}^{I} A_i) \bmod p \tag{6.7}$$

Začnimo z osnovnim problemom parnosti logičnih atributov ($p = 2$). Za primer si poglejmo probleme parnosti reda 2 do 8 ($I \in [2, 8]$) na 9 atributih (imamo 7 do 1 nepomembnih atributov) in s popolnim opisom problema (vseh 512 primerov). Slika 6.2 prikazuje krivuljo s za ta problem (krivulja u ji je identična, saj imamo popoln opis problema). Na tej in na vseh ostalih slikah v tem poglavju predstavlja vsaka točka povprečje 10 ocen.

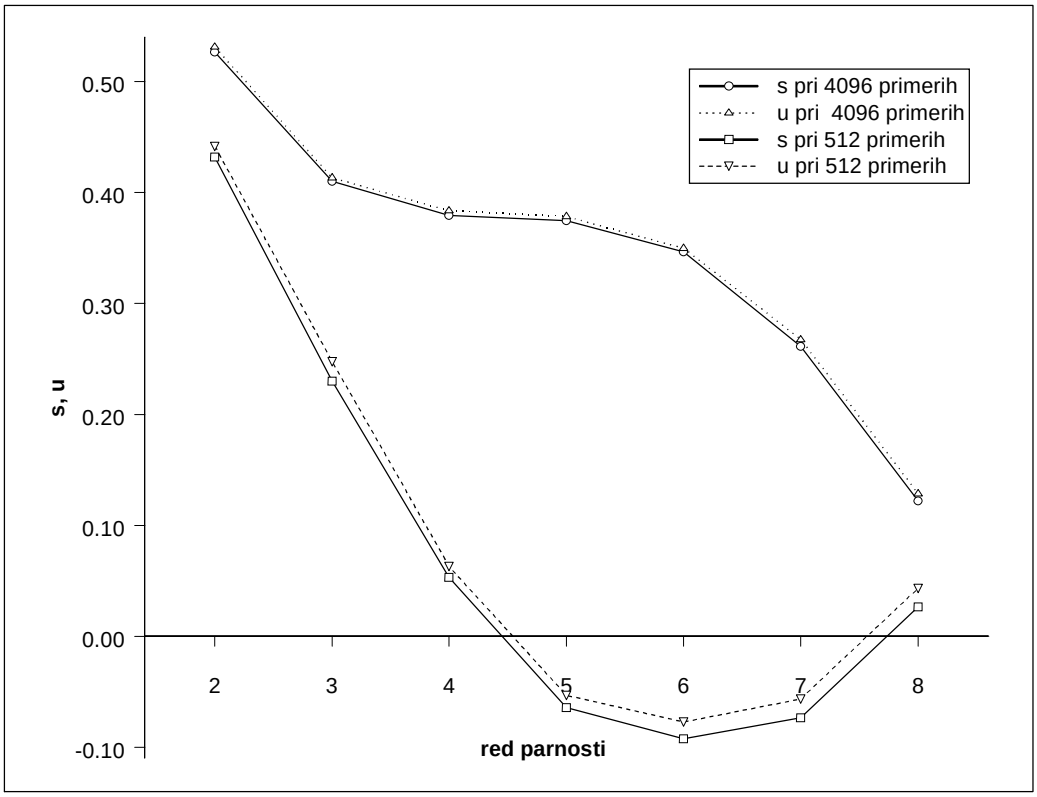
Vidimo, da ločljivost atributov pada z naraščanjem težavnosti problemov pri redih parnosti 2, 3 in 4. Pri parnosti reda 5, ko je pomembnih več kot polovica atributov postane ločljivost negativna (ocene kakovosti ne ločujejo več med pomemb-



Slika 6.2: Ločljivost na problemih parnosti reda 2 do 8 z 9 atributi (7 do 1 nepomembnimi) in z vsemi 512 primeri.

nimi in naključnimi atributi). Razlog za to je uporaba več kot enega bližnjega soseda (z uporabo 1 bližnjega soseda dobimo pozitiven s za vse brezšumne probleme parnosti). Ker število karakterističnih področij v problemu narašča z 2^I, število primerov pa ostaja konstantno (512), imamo vse manj primerov na karakteristično področje in večina bližnjih sosedov izbranega primera zato prihaja iz drugih karakterističnih področij. Pri $I = 5$, ko dobimo negativen s, število bližnjih sosedov iz karakterističnih področij na razdalji 1 (drugačna parnost) preseže število bližnjih primerov na karakterističnem področju, ki mu pripada izbrani primer. Zanimivo je, da pri $I = 8$ (ko je nepomemben le 1 atribut) postane s spet pozitiven. To razložimo tako, da število bližnjih sosedov iz izbranega karakterističnega področja in iz karakterističnega področja na razdalji 2 (parnost je spet enaka) preseže število bližnjih sosedov iz karakterističnega področja na razdalji 1.

Da je zadostno število primerov na karakteristično področje ključno za zanesljivost ocen algoritma ReliefF, se lahko prepričamo na sliki 6.3. Spodnji krivulji

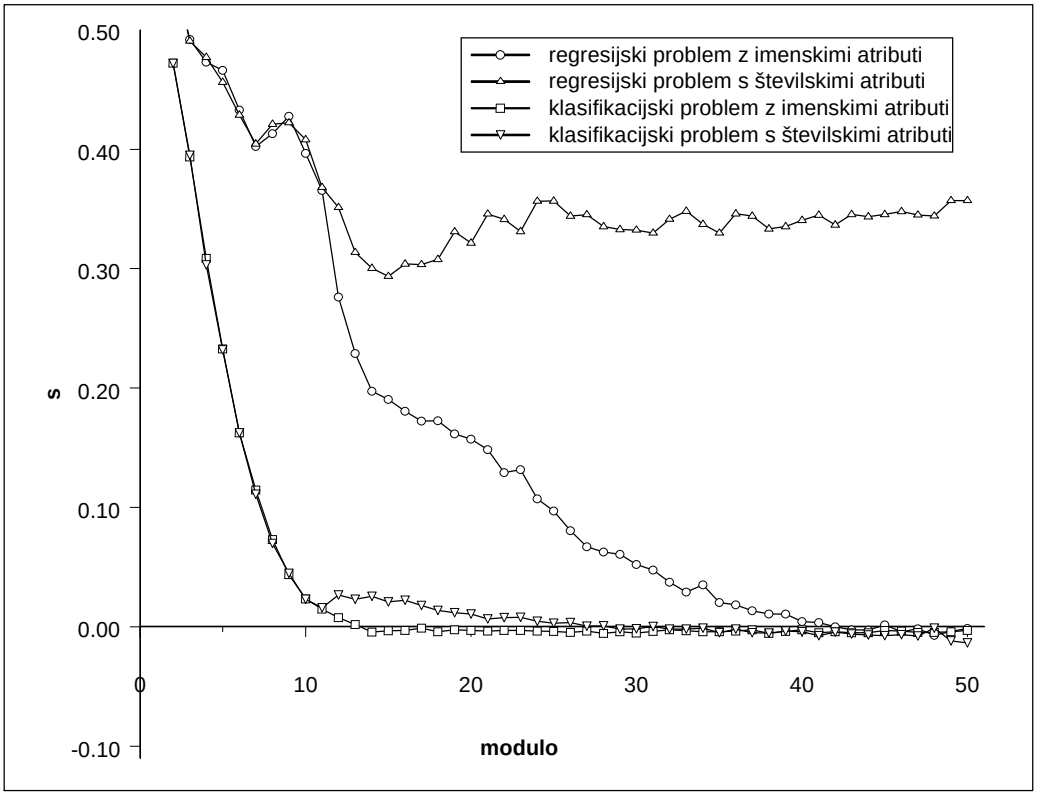


Slika 6.3: Ločljivost in uporabnost na problemih parnosti reda 2 do 8 z naključno izbranimi 512 oziroma 4096 primeri.

s in u prikazujeta isti problem kot na sliki 6.2, le da problem ni opisan z vsemi 512 primeri, ampak uporabljamo naključno generiranih 512 primerov. Vidimo, da je ločljivost s nekoliko nižja kot na sliki 6.2, saj smo dejansko zmanjšali število različnih primerov (v povprečju na 63.2% vseh). Zgornji krivulji s in u prikazujeta isti problem, le da imamo osemkrat več primerov (4096). Vidimo, da je pri tolikšnem številu primerov ločljivost za vse primere pozitivna.

Naslednji problem, ki si ga bomo ogledali, definiramo tako, da spreminjamo modulo (p), število pomembnih atributov postavimo na 2, število primerov pa na 512. Spodnji krivulji na sliki 6.4 prikazujeta ločljivost za klasifikacijski problem (imamo p razredov) in torej za ReliefF. Atribute lahko obravnavamo kot številske ali imenske, vidimo pa, da se obe krivulji obnašata podobno: ločljivost pada z naraščajočim modulom. Takšen potek je pričakovan, saj zahtevnost problema narašča s številom vrednosti razreda in atributov s p ter s številom karakterističnih

področij s p^I (polinomsko naraščanje). Spet bi se z uporabo več primerov pomaknile pozitivne vrednosti s bolj proti desni. Majhna, vendar opazna razlika med ločljivostjo imenskih in številskih atributov kaže, da dobimo pri tem problemu s številskimi atributi več informacije. Funkcija diff nam vrne 1 za katerikoli različni vrednosti imenskega atributa, pri številskih atributih pa diff vrne relativno razliko, kar pa prinaša več informacije (še posebej pri nekoliko večji vrednosti p). ReliefF torej pozitivno ovrednoti urejenost atributa, kadar ta vsebuje kaj informacije.



Slika 6.4: Ločljivost za ReliefF in RReliefF na Modulo klasifikacijskih in regresijskih problemih glede na modulo.

Iste probleme lahko obravnavamo tudi kot regresijske, atribute spet obravnavamo kot imenske ali številske. Zgornji krivulji na sliki 6.4 prikazujeta ločljivost za te regresijske probleme in torej za RReliefF. Opazimo, da dobimo pozitiven s pri večjih vrednostih modula v primerjavi s klasifikacijskimi problemi, če pa obravnavamo atribute kot številske, ločljivost sploh ne pada z naraščanjem modula. Razlog je v tem, da so klasifikacijski problemi dejansko težji, saj poskušamo napovedati p ločenih razredov (npr. vrednosti razreda 2 in 3 pri klasifikaciji sta

popolnoma različni), pri regresiji pa primerjamo številske vrednosti (napovedani vrednosti 2 in 3 sta različni glede na obseg vrednosti).

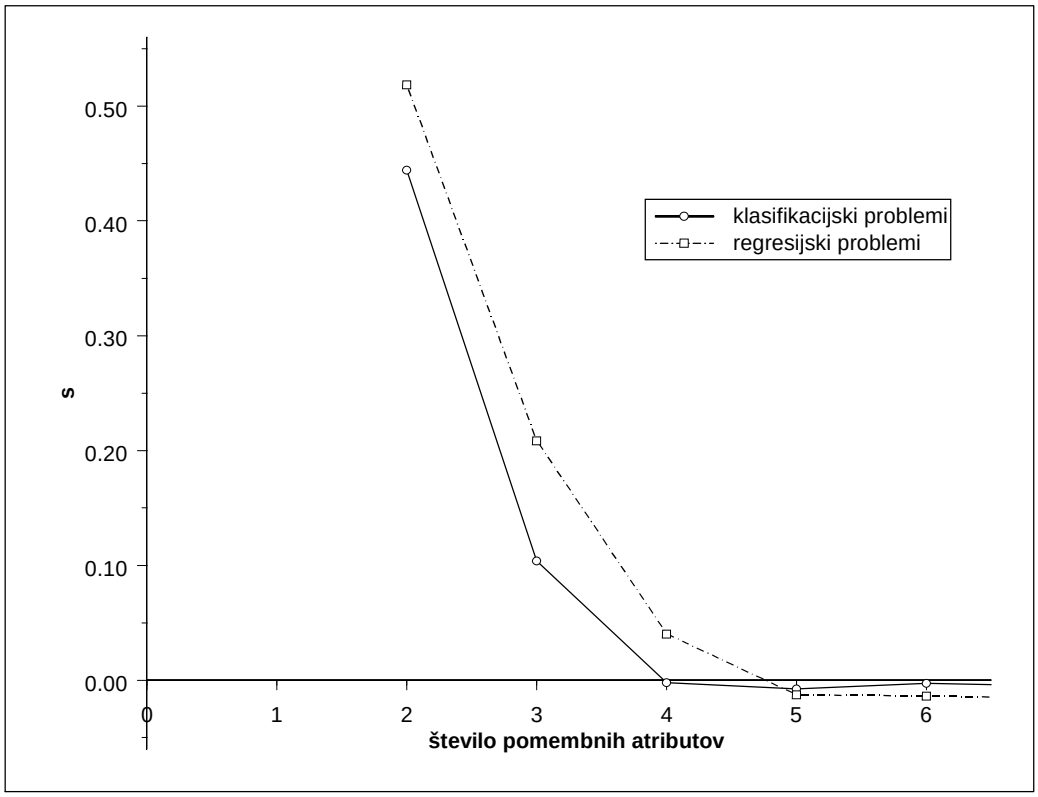
Na istem regresijskem problemu s številskimi atributi smo namesto manhattanske poskusili uporabiti krožno razdaljo. Uporaba te razdalje v modulo problemih je naravna, saj upošteva krožnost glede na modulo. Tako sta npr. pri modulu 8 vrednosti 1 in 7 na razdalji 2 in ne 6 kot pri manhattanski razdalji. V tem smislu je potrebno spremeniti funkcijo $\mathrm{diff}(A, I_1, I_2) = min(v(A, I_1) - v(A, I_2), v(A, I_2) - v(A, I_1))$, kjer $v(A, I)$ predstavlja vrednost atributa A pri primeru I in je potrebno negativne vrednosti jemati kot nasprotne vrednosti v grupi seštevanja po modulu ter jim prišteti vrednost modula. Rezultati, ki smo jih dobili, so skoraj povsem enaki tistim z manhattansko razdaljo (zgornja krivulja na sliki 6.4) in jih ne navajamo. Razlog za to podobnost je, da RReliefF pri popravkih uteži upošteva le bližnje primere, pri teh pa je vrednost razdalje pri manhattanski in krožni razdalji enaka, do razhajanj lahko pride le pri primerih z najmanjšimi in največjimi vrednostmi atributa, kar pa na rezultat skoraj ne vpliva.

Še en zanimiv problem dobimo, če postavimo modulo na neko majhno vrednost (npr. $p = 5$) in spreminjamo število pomembnih atributov. Slika 6.5 prikazuje krivulje s za 4096 primerov in 10 naključnih atributov. Pri modulu 5 ni opaznih razlik pri ocenah za imenske in številske atribute in zato podajamo krivulje le za imenske atribute. Vidimo, da s hitro upada z naraščanjem števila pomembnih atributov I. Spomnimo se, da zahtevnost problema (število karakterističnih področij) narašča s p^I (eksponentno).

Problemi Modulo so primer težkih problemov v smislu velike spremenljivosti koncepta (glej tabelo A.1). Opozorimo naj, da mere, temelječe na funkcijah nečistoče, kot sta na primer Gain ratio ali MSE, niso sposobne razločevati med pomembnimi in nepomembnimi atributi v nobenem problemu opisanem v tem razdelku.

6.2.2 Problemi MONK

Poglejmo si ocenjevanje atributov na znanih in priljubljenih problemih MONK, ki so bili originalno uporabljeni za primerjavo med različnimi sistemi strojnega učenja (Thrun in sod., 1991). MONK sestavljajo trije dvorazredni klasifikacijski problemi, temelječi na skupnem opisu iz šestih atributov. A_1, A_2 in A_4 lahko zavzamejo vrednosti 1, 2 ali 3, A_3 in A_6 imata lahko vrednost 1 ali 2, A_5 pa lahko dobi vrednost 1, 2, 3 ali 4. Možnih je 432 različnih primerov, mi pa smo uporabili naključno generirane učne množice s toliko primeri, kot so jih vsebovale naloge v originalni primerjavi med učnimi sistemi. Problemi so naslednji:



Slika 6.5: Ločljivost za ReliefF in RReliefF na problemih Modulo-5 glede na število pomembnih atributov.

- **problem M_1:** 124 učnih primerov za problem: $(A_1 = A_2) \vee (A_5 = 1)$
- **problem M_2:** 169 učnih primerov za problem: natančno dva atributa imata vrednost 1
- **problem M_3:** 122 učnih primerov (s 5 % napačnih klasifikacij) za problem: $(A_5 = 3 \wedge A_4 = 1) \vee (A_5 \neq 4 \wedge A_2 \neq 3)$.

Generirali smo 10 naključnih vzorcev dane velikosti za vsak problem in primerjali ocene algoritma ReliefF in hevristike Gain ratio. Rezultate, ki so povprečje desetih ocen, podaja tabela 6.1.

Pri prvem problemu vidimo, da ReliefF loči pomembne atribute (A_1, A_2 in A_5) od nepomembnih, medtem ko Gain ratio ne razpozna pomembnosti atributov A_1 in A_2.

Tabela 6.1: Ocene atributov za algoritem ReliefF in Gain ratio pri treh problemih MONK.

	M_1		M_2		M_3	
Atribut	ReliefF	Gain r.	ReliefF	Gain r.	ReliefF	Gain r.
A_1	0.054	0.003	0.042	0.006	-0.013	0.004
A_2	0.056	0.004	0.034	0.006	0.324	0.201
A_3	-0.023	0.003	0.053	0.001	-0.016	0.003
A_4	-0.016	0.007	0.039	0.004	-0.005	0.008
A_5	0.208	0.160	0.029	0.007	0.266	0.183
A_6	-0.020	0.002	0.043	0.001	-0.016	0.003

Pri drugem problemu so pomembni vsi atributi. ReliefF atributom, ki imajo manj vrednosti, pravilno priredi višje ocene, saj vsebujejo več informacije, medtem ko Gain ratio, nasprotno, bolje oceni atribute z več vrednostmi.

Pri tretjem problemu se ReliefF in Gain ratio obnašata podobno: razpoznata pomembne atribute in jih tudi rangirata enako.

6.2.3 Linearni in nelinearni problemi

V tipičnem regresijskem problemu so linearne odvisnosti pomešane z nelinearnimi. Pogledali si bomo nekaj takšnih tipičnih problemov. Opisani so s številskimi atributi z vrednostmi na intervalu $[0, 1]$ in 1000 primeri. V vsakem od problemov je poleg I pomembnih atributov še 10 naključnih.

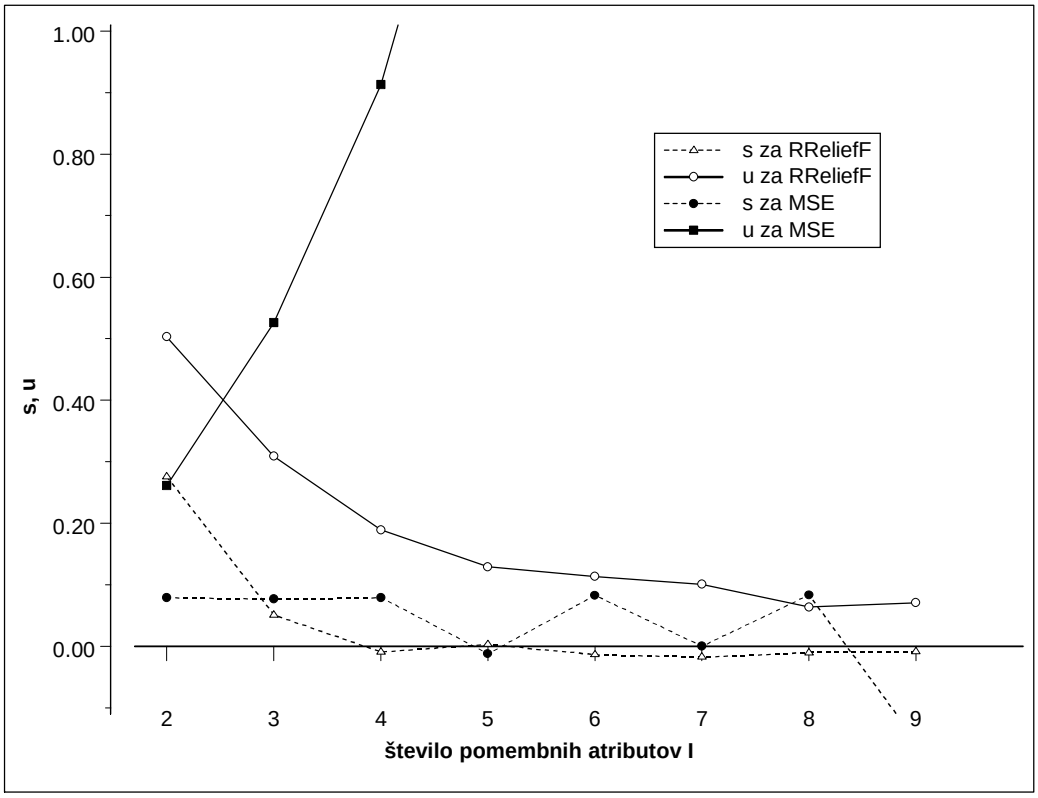
Začnimo s samimi linearnimi odvisnostmi in definirajmo problem:

$$\mathrm{LinInc-I}: \qquad \tau = \sum_{j=1}^{I} j \cdot A_j \tag{6.8}$$

Atributi z večjimi koeficienti imajo večji vpliv na napovedano vrednost in pričakujemo, da bodo bolje ocenjeni. Tabela 6.2 prinaša ocene kakovosti atributov za RReliefF (RRF) in MSE. RReliefF prireja boljšim atributom višje ocene, MSE pa obratno. Pri majhnih razlikah v pomembnosti tako RReliefF kot MSE ločita pomembne od nepomembnih atributov in jih pravilno razvrstita glede na njihov vpliv na napovedano vrednost. Ko postane razlika pri vplivu na napovedano vrednost večja (pri problemih LinInc-4 in LinInc-5), se lahko zgodi, da je eden od naključnih atributov ocenjen bolje kot najmanj vpliven pomemben atribut (A_1). To se zgodi pri problemu LinInc-4 algoritmu RReliefF pri problemu LinInc-5 pa MSE. Obnašanje s in u ponazarja slika 6.6.

Tabela 6.2: Ocene algoritma RReliefF in hevristike MSE za najboljši naključni atribut (R_{best}) in vse pomembne atribute v problemih LinInc-I.

	LinInc-2		LinInc-3		LinInc-4		LinInc-5	
Atribut	RRF	MSE	RRF	MSE	RRF	MSE	RRF	MSE
R_{best}	-0.040	0.424	-0.023	1.098	-0.009	2.421	-0.010	4.361
A_1	0.230	0.345	0.028	1.021	-0.018	2.342	-0.007	4.373
A_2	0.461	0.163	0.154	0.894	0.029	2.093	0.014	4.181
A_3			0.286	0.572	0.110	1.833	0.039	3.777
A_4					0.180	1.508	0.054	3.380
A_5							0.139	2.837

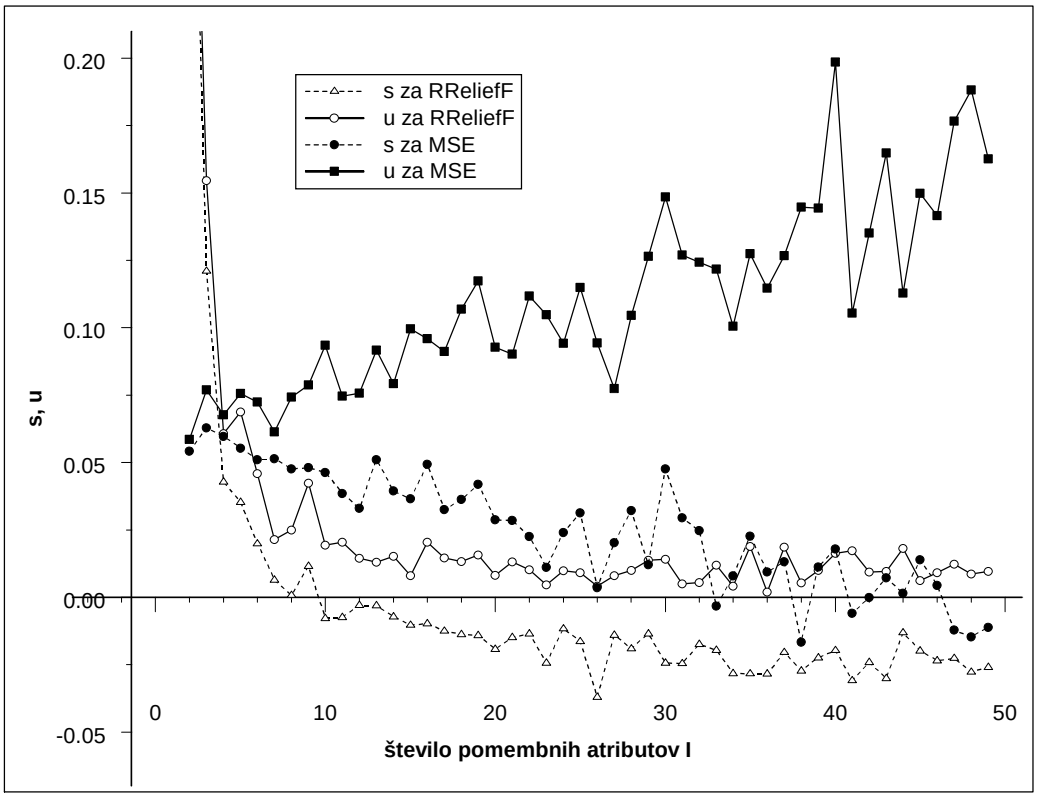


Slika 6.6: Ločljivost in uporabnost na problemih LinInc s 1000 primeri.

Raziskali bomo še eno obliko linearnih odvisnosti:

$$\mathrm{LinEq-I}: \qquad \tau = \sum_{j=1}^{I} A_j \tag{6.9}$$

Vsem atributom v tem problemu damo enak vpliv na napovedano vrednost, preveriti želimo koliko pomembnih atributov lahko imamo. Slika 6.7 prikazuje krivulji ločljivosti in uporabnosti za RReliefF in MSE.



Slika 6.7: Ločljivost in uporabnost na problemih LinEq s 1000 primeri.

Vidimo, da pri algoritmu RReliefF ločljivost pada in postane pri 10 atributih negativna, kar nas ne sme presenetiti, če upoštevamo, da Relief priredi vsakemu atributu utež, ki ustreza deležu razloženega koncepta (glej enačbo (4.22) v razdelku 4.4). S povečevanjem števila pomembnih atributov se manjša ocena kakovosti, ki pripada posameznemu atributu in veča verjetnost, da bo eden od naključnih atributov ocenjen bolje kot nek pomemben atribut. Enako velja za uporabnost pri algoritmu RReliefF, ki pa postane negativna pri dosti večjem številu atributov.

MSE ocenjuje atribute neodvisno drug od drugega, zato pri njem ne opazimo te težave, vendar pa tudi tu z naraščanjem števila pomembnih atributov narašča verjetnost za slabšo oceno enega od pomembnih atributov in vrednost s postane negativna. Nasprotno velja za uporabnost, kajti z večanjem števila pomembnih atributov narašča tudi verjetnost za boljšo oceno enega od pomembnih atributov in vrednost u narašča.

S povečanjem števila primerov na 4000, postane krivulja s pri algoritmu RReliefF negativna pri 16 pomembnih atributih, obnašanje s in u pri MSE pa se ne spremeni.

Pregled problemov končajmo z nelinearno odvisnostjo. Definirajmo funkcijo

$$\mathrm{Cosinus-Hills}: \qquad \tau = \cos 2\pi(A_1^2 + 3A_2^2). \tag{6.10}$$

Zaradi lepše grafične predstavitve smo 1000 primerov pri tem problemu opisali z atributi na intervalu$[-1, 1]$. Poleg dveh pomembnih smo generirali še 10 naključnih atributov. Vizualizacija tega problema se nahaja na sliki 6.8. Vidimo, da so napovedane vrednosti simetrične vzdolž obeh pomembnih atributov in da imamo trikrat več valov vzdolž A_2.

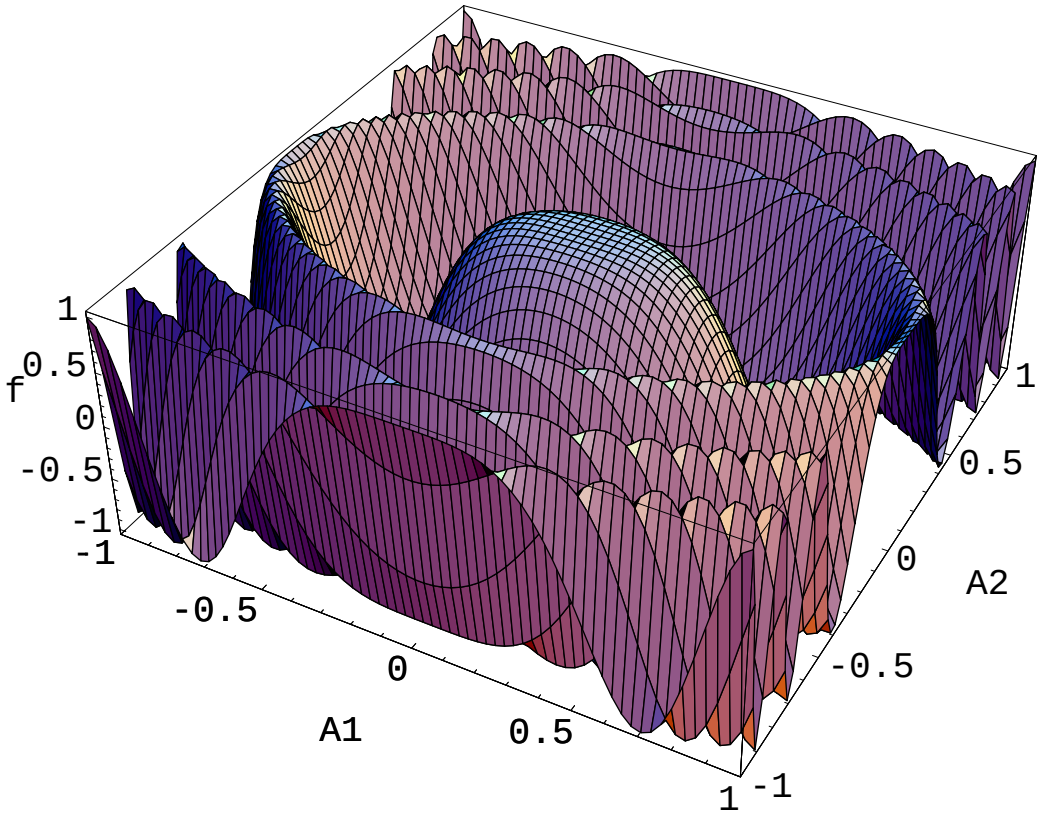
Ocene kakovosti za RReliefF in MSE se nahajajo v tabeli 6.3. RReliefF prireja boljšim atributom višje ocene, MSE pa obratno. Vidimo, da RReliefF loči A_1 in A_2 od naključnih atributov, hevristika MSE pa v tem problemu ni uspešna.

Tabela 6.3: Ocene kakovosti obeh pomembnih in najboljšega naključnega atributa (R_{best}) pri problemu Cosinus-Hills za RReliefF in MSE.

Atribut	RReliefF	MSE
A_1	0.0578	0.5001
A_2	0.0398	0.5003
R_{best}	0.0194	0.4992

6.3 Število primerov

Ugotoviti želimo, kako število učnih primerov vpliva na ocene, zato smo generirali nekaj umetnih problemov z različnim številom primerov in na njih opazovali ločljivost in uporabnost. Podatke o problemih in njihove karakteristike najdemo v tabeli A.1, ki se nahaja v dodatku. Za vsako število primerov smo desetkrat ponovili ocenjevanje in s Studentovim parnim t-testom testirali hipotezo, da sta s



Slika 6.8: Vizualizacija problema Cosinus-Hills.

in u večja od 0. Prag značilnosti odstopanja smo postavili na 0.99 in tako dobili mejno število primerov, ki je potrebno, da lahko značilno ločimo vse pomembne atribute od naključnih ali vsaj enega pomembnega od naključnih atributov. To mejno število za ReliefF ali RReliefF in za Gain ratio ali MSE prikazujemo na levi strani tabele 6.4. Število je indikator robustnosti cenilk atributov.

Opazimo, da število potrebnih primerov narašča z zahtevnostjo problemov znotraj posameznih skupin problemov (npr., Modulo-5, Modulo-10, LinInc, ...). Opozorimo naj, da med skupinami problemov to naraščanje ni povezano s spremenljivostjo koncepta V (izraz (6.4)), ki jo podajamo v tabeli A.1. To priča o določeni neusklajenosti med algoritmi ReliefF in mero V.

Poglejmo si primerjavo med algoritmoma ReliefF in RReliefF ter merama Gain ratio in MSE. Vidimo, da ti meri (zaradi kratkovidnosti) nekaterih problemov ne zmorejo pravilno oceniti (znak '-' v tabeli 6.4), v drugih (lažjih) problemih pa potrebujejo približno enako primerov za pozitivno uporabnost in nekoliko

Tabela 6.4: Prikaz vpliva števila učnih primerov in šuma v napovedanih vrednostih.

	število primerov				šum			
	ReliefF		Gain r.		ReliefF		Gain r.	
problem	s	u	s	u	s	u	s	u
Bool-Simple	100	50	-	40	10	45	-	20
Modulo-2-2-c	70	60	-	-	30	50	-	-
Modulo-2-3-c	170	110	-	-	5	50	-	-
Modulo-2-4-c	550	400	-	-	20	50	-	-
Modulo-5-2-c	150	130	-	-	5	5	-	-
Modulo-5-3-c	950	800	-	-	15	35	-	-
Modulo-5-4-c	5000	4000	-	-	10	50	-	-
Modulo-10-2-c	400	400	-	-	20	20	-	-
Modulo-10-3-c	4000	3000	-	-	25	55	-	-
Modulo-10-4-c	25000	22000	-	-	10	40	-	-
MONK-1	100	30	-	20	30	70	-	55
MONK-3	250	10	250	20	25	75	5	75
	RReliefF		MSE		RReliefF		MSE	
Modulo-5-2-r	60	50	-	-	25	50	-	-
Modulo-5-3-r	400	350	-	-	20	45	-	-
Modulo-5-4-r	2000	1600	-	-	15	40	-	-
Modulo-10-2-r	90	80	-	-	25	40	-	-
Modulo-10-3-r	800	600	-	-	10	40	-	-
Modulo-10-4-r	7000	4000	-	-	5	40	-	-
Fraction-2	80	70	-	-	20	40	-	-
Fraction-3	650	400	-	-	15	35	-	-
Fraction-4	4000	3000	-	-	10	35	-	-
LinInc-2	60	10	70	20	15	50	10	50
LinInc-3	400	20	150	10	10	65	20	70
LinInc-4	2000	40	450	10	5	80	35	85
LinEq-2	50	40	20	20	30	60	25	45
LinEq-3	180	50	50	20	25	40	30	50
LinEq-4	350	70	70	30	10	40	30	45
Cosinus-Lin	300	40	-	200	15	50	-	40
Cosinus-Hills	550	300	4000	2000	5	20	-	-

manj primerov za pozitivno ločljivost. Tudi to lahko pripišemo načinu ocenjevanja atributov s posameznimi hevristikami. Medtem, ko kratkovidne mere ocenjujejo vsak atribut neodvisno od drugih, jih algoritmi Relief ocenjujejo v kontekstu drugih atributov. Pomembni atributi si delijo oceno kakovosti, bolj pomembni dobijo večji delež, zato se lahko zabriše meja med manj pomembnimi in naključnimi atributi. Zaključimo lahko, da so, zaradi načina delovanja, algoritmi Relief rahlo pristranski v korist bolj pomembnih in v škodo manj pomembnih atributov.

6.4 Šum pri napovedani vrednosti

Preverili smo tudi robustnost algoritmov ReliefF in RReliefF glede šuma. Uporabili smo načrt poskusa kot v prejšnjem razdelku, le da smo število primerov postavili na vrednosti iz prvega s stolpca na levi strani tabele 6.4 ter v opis problemov dodali šum tako, da smo spremenili določen odstotek napovedanih vrednosti v naključno vrednost (iz zaloge vrednosti, ki jo lahko zavzame napovedana vrednost). Ta način dodajanja šuma pomeni, da lahko dobi napovedana vrednost tudi svojo originalno vrednost in je smiselno dodajanje tudi več kot 50% šuma. Odstotek spremenjenih vrednosti smo spreminjali v korakih po 5%. Desna stran tabele 6.4 podaja največji odstotek šuma, ki smo ga lahko dodali, da sta bila s in u še vedno pozitivna z veliko verjetnostjo (> 0.99).

Vidimo, da je sta algoritma ReliefF in RReliefF robustna glede na šum v vseh problemih. S povečanjem števila učnih primerov za desetkrat smo lahko v vseh problemih naključno določili več kot 80% vrednosti. Vključili smo tudi primerjavo z merama Gain ratio in MSE. V problemih, kjer sta ti dve meri uspešni, je primerljiva tudi njuna robustnost glede šuma.

6.5 Število iteracij

Potrebno število iteracij, ki jih potrebujeta ReliefF in RReliefF, smo preverjali na podoben način kot v prejšnjem razdelku, le da smo zdaj spreminjali število iteracij. Leva stran tabele 6.5 podaja minimalno število iteracij, ki smo jih potrebovali v vsakem od problemov, da sta bila s in u pozitivna z veliko verjetnostjo (> 0.99). Stolpca označena z '#Ex' prikazujeta število potrebnih iteracij, če imamo na voljo število primerov iz prvega s stolpca na levi strani tabele 6.4, stolpca označena z '$10\times$ #Ex' pa potrebno število iteracij pri desetkrat več primerih.

Zanimivo je, da pri najmanjšem potrebnem številu primerov za značilno razlikovanje med pomembnimi in naključnimi atributi za to niso potrebne vse iteracije, ampak je dejansko potrebno število iteracij včasih zelo majhno, kar kaže na možne prihranke pri računanju. Če pa imamo na razpolago več kot zadosti primerov, se

Tabela 6.5: Rezultati merjenja potrebnega števila iteracij in dodajanja nepomembnih atributov.

	število iteracij				nepomembni atr.	
	#Ex		$10\times$ #Ex		#Ex	
problem	s	u	s	u	s	u
Bool-Simple	83	1	21	1	12	150
Modulo-2-2-c	11	9	1	1	20	45
Modulo-2-3-c	144	15	2	1	14	25
Modulo-2-4-c	151	18	3	1	12	20
Modulo-5-2-c	51	35	1	1	17	30
Modulo-5-3-c	390	37	3	2	20	45
Modulo-5-4-c	3674	579	10	2	13	25
Modulo-10-2-c	31	26	1	1	110	160
Modulo-10-3-c	1210	263	3	1	50	85
Modulo-10-4-c	23760	3227	25	4	11	30
MONK-1	36	1	2	1	NA	NA
MONK-3	163	1	56	1	NA	NA
Modulo-5-2-r	15	12	2	1	19	20
Modulo-5-3-r	57	23	4	1	16	20
Modulo-5-4-r	975	182	17	3	11	19
Modulo-10-2-r	55	37	3	3	14	14
Modulo-10-3-r	316	164	16	9	13	19
Modulo-10-4-r	5513	929	126	17	13	25
Fraction-2	47	32	3	3	18	25
Fraction-3	379	51	29	11	14	20
Fraction-4	1198	78	138	13	13	18
LinInc-2	48	4	6	3	20	> 10000
LinInc-3	109	4	52	3	16	> 10000
LinInc-4	940	3	456	2	14	> 10000
LinEq-2	17	9	3	2	45	140
LinEq-3	96	10	16	7	25	170
LinEq-4	215	13	45	9	16	300
Cosinus-Lin	188	10	16	7	20	3200
Cosinus-Hills	262	51	110	30	12	20

lahko zadovoljimo z resnično malo iteracijami, kot vidimo iz stolpcev označenih z '$10\times$ #Ex' v tabeli 6.5.

6.6 Dodajanje nepomembnih atributov

ReliefF in RReliefF sta kontekstno občutljiva, zato pri ocenjevanju kakovosti nanju vplivajo vsi atributi v problemskem prostoru. Veliko število nepomembnih atributov ima torej na njune ocene večji vpliv kot na ocene kratkovidnih hevristik, ki ocenjujejo atribute neodvisno drug od drugega. Vpliv naključnih atributov smo preverili s podobno eksperimentalno zasnovo kot v prejšnjih razdelkih, problemom pa smo dodajali različno število naključnih atributov. Pri problemih MONK ni jasno, kakšni naj bi bili dodani naključni atributi, zato ju nismo vključili v poskus (označba NA).

Desna stran tabele 6.5 povzema rezultate. Predstavljeno je število nepomembnih atributov, ki smo jih lahko dodali, da ostaneta s in u z veliko verjetnostjo pozitivna (> 0.99). Vidimo, da lahko ReliefF in RReliefF za pozitivno ločljivost tolerirata veliko manj naključnih primerov kot za pozitivno uporabnost, kar spet pripisujemo rahli pristranosti v korist pomembnejših atributov. To je še posebej jasno vidno pri problemih Lin-Inc in Cosinus-Lin, kjer vsi atributi niso enako pomembni.

MSE in Gain ratio sta občutljiva na dodajanje naključnih atributov le v toliko, da z večanjem števila naključnih atributov narašča verjetnost pojavitve navidezno ,,dobrega" atributa, zato ju nismo vključili v ta poskus.

6.7 Število pomembnih atributov

Čeprav smo se z vprašanjem koliko pomembnih atributov lahko imamo pri nekem problemu, da bodo ocene algoritmov Relief še zanesljive, srečali že v prejšnjih razdelkih, bomo na tem mestu zbrali rezultate in jih pojasnili, kajti gre za pomembno praktično vprašanje.

Če se pomembni atributi razlikujejo v pomembnosti, jih bodo algoritmi Relief v idealnem primeru ocenili glede na delež koncepta, ki ga razložijo (glej izraza (4.22) in (4.28)). V praksi (pri omejenem številu primerov) pa dobijo manj pomembni primeri še manjši delež, kajti spremembe napovedanih vrednosti, ki jih povzročijo manj pomembni atributi (glej izraz 4.1) lahko prekrijejo večje spremembe, ki jih povzročijo pomembnejši atributi. Primer tovrstnega obnašanja vidimo na sliki 6.6 in v tabeli 6.2, ki vsebujeta rezultate za probleme, kjer atributi niso enako pomembni. S štirimi atributi je v teh pogojih ločljivost algoritma RReliefF padla pod 0, medtem ko je uporabnost v takih razmerah dosti bolj robustna

in ostane pozitivna tudi pri več sto pomembnih atributih.

Če so vsi atributi enako pomembni, je ocenjevanje zanesljivo tudi pri več atributih, kar prikazuje slika 6.7, vendar pa je natančno število odvisno od zadostnega števila primerov oziroma primernega pokritja problemskega prostora. Uporabnost tudi v tem primeru ostaja pozitivna z več sto pomembnimi atributi.

Na kratko lahko povzamemo, da je razpoznavanje večjega števila manj pomembnih atributov vprašljivo, medtem ko bolj pomembne atribute ReliefF in RReliefF razpoznata tudi v težjih pogojih.

6.8 Odvečni atributi

V tem razdelku bomo prikazali obnašanje algoritmov Relief, ko so v problemu prisotni odvečni atributi. Začeli bomo s preprostim problemom parnosti z dvema pomembnima in desetimi naključnimi atributi, ki jih ReliefF takole oceni:

I_1	I_2	R_{best}
0.445	0.453	-0.032

Zdaj podvojimo prvi pomembni atribut I_1 in dobimo:

I_{1,C_1}	I_{1,C_2}	I_2	R_{best}
0.221	0.221	0.768	-0.015

Obe izvoda atributa I_1 si zdaj delita oceno kakovosti pa tudi I_2 dobi nekaj višjo oceno. Preden razložimo te ocene, poskusimo še s tremi izvodi prvega pomembnega atributa:

I_{1,C_1}	I_{1,C_2}	I_{1,C_3}	I_2	R_{best}
0.055	0.055	0.055	0.944	-0.006

in nato še z desetimi izvodi:

I_{1,C_1}	I_{1,C_2}	...	$I_{1,C_{10}}$	I_2	R_{best}
0.000	0.000	...	0.000	1.000	0.006

Razlog za takšno obnašanje je v spremembi problemskega prostora, v katerem iščemo bližnje sosede, ki jo povzročijo dodatni izvodi atributa. Spomnimo se, da za vsak izbrani primer iščemo bližnje sosede iz istega in iz drugega razreda, potem pa popravimo oceno kakovosti glede na vrednosti funkcije diff (glej izraz (3.1)). Ocena se spremeni le, ko je vrednost diff funkcije neničelna, torej ko so vrednosti atributov različne. Če naredimo c izvodov nekega atributa, to pomeni, da so primeri z različno vrednostjo tega atributa zdaj na razdalji vsaj c, kar povzroči izpad takih primerov iz bližnje soseščine izbranega primera. Posledično se z

naraščanjem števila izvodov manjša tudi verjetnost popravljanja uteži podvojenih atributov in njihova ocena kakovosti konvergira k 0. Obraten učinek ima podvajanje atributov na preostale nepodvojene atribute: primeri z razlikami pri vrednostih nepodvojenih atributov se večkrat uvrstijo v bližnjo soseščino, zato so ocene teh atributov večkrat obnovljene in atributi dobijo višjo oceno, kot bi jo zaslužili.

Če razmnožimo vse pomembne atribute (kar v praksi sicer ni verjetno), se ta pojav izniči, kajti razdalje so simetrično povečane. Podajamo primer z desetimi izvodi obeh pomembnih atributov:

I_{1,C_1}	...	$I_{1,C_{10}}$	I_{2,C_1}	...	$I_{2,C_{10}}$	R_{best}
0.501	...	0.501	0.499	...	0.4999	-0.037

Sklenemo lahko, da so algoritmi Relief občutljivi na odvečne atribute, saj ti spremenijo problemski prostor. To občutljivost lahko interpretiramo tako, da so ocene algoritmov Relief sorazmerne deležu razloženega koncepta in, če si mora ta delež deliti več atributov, je dobitek vsakega ustrezno manjši. Kratkovidne hevristike, kot sta Gain ratio in MSE, niso občutljive na podvojene atribute.

Pri problemih opisanimi le s številskimi atributi lahko težavo omilimo z uporabo druge mere razdalje. Mahalanobisova razdalja upošteva odvisnost med atributi (ne pogojne odvisnosti glede na razred). Definirana je kot

$$D_M(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T \mathbf{K}^{-1} (\mathbf{x} - \mathbf{y}), \tag{6.11}$$

kjer $\mathbf{x}$ in $\mathbf{y}$ predstavljata vektorje vrednosti atributov primerov X in Y, K^-1 pa je inverz kovariančne matrike vrednosti atributov. Slabost tega pristopa je poleg tega, da ga lahko uporabimo le na problemih, ki so opisani samo s številskimi atributi, tudi nezmožnost obravnave manjkajočih vrednosti atributov in dodatna računska zahtevnost pri izračunu kovariančne matrike in njenega inverza.

V splošnem se moramo zavedati občutljivosti algoritmov Relief na odvečne atribute. Ker to povzroča težave tudi v nekaterih drugih postopkih strojnega učenja (npr. gradnji odločitvenih dreves), bi to slabost lahko izkoristili za odkrivanje odvečnih atributov. Ideja je v tem, da bi iz problemskega prostora po vrsti izločali po en atribut in opazovali spremembe ocen ostalih atributov. Izločitev odvečnega atributa bi povzročila povečanje vrednosti ocen ostalih atributov z isto informacijo in zmanjšanje ocen ostalim atributom. To idejo bo potrebno preveriti v nadaljnjem delu.

6.9 Povzetek empirične analize

Povzemimo ugotovitve tega poglavja in poskušajmo osvetliti dobre in šibke točke algoritmov Relief.

Algoritma ReliefF in RReliefF smo testirali na vrsti različnih problemov. Na problemih tipa Modulo smo opazili, da je zadostno število primerov na karakteristično področje problema ključno za uspešno razpoznavanje odvisnosti in ocenjevanje atributov. Če vsebujejo številski in imenski atributi isto informacijo, znajo algoritmi Relief ovrednotiti informacijo o urejenosti vrednosti in bolje ocenijo številske atribute. Sposobnost razločevanja pada z naraščanjem števila razredov in števila pomembnih atributov, kar je povezano z zadostnim pokritjem problemskega prostora. S privzetimi parametri je ReliefF uspešno razpoznal probleme nižje zahtevnosti, medtem ko smo za bolj zahtevne probleme potrebovali več primerov ali drugačno izbiro bližnje okolice.

Na problemih MONK smo primerjali ReliefF in Gain ratio, ki ne razpozna močnih pogojnih odvisnosti med atributi. Ugotovili smo, da ReliefF pri atributih, ki vsebujejo isto informacijo, favorizira tiste z manj vrednosti, kar je skladno z intuitivnimi pričakovanji, Gain ratio pa deluje obratno.

Pri številskih problemih z linearnimi in nelinearnimi odvisnostmi smo primerjali RReliefF z MSE. Pri linearnih problemih se je RReliefF odrezal slabše kot MSE, saj ocenjuje vse atribute hkrati in dobijo atributi oceno odvisno od deleža sprememb koncepta, ki jih zaznajo. Pri večjem številu pomembnih atributov tako manj pomembne atribute težko loči od nepomembnih. Nelinearne odvisnosti zmedejo MSE, medtem ko jih RReliefF pravilno zazna.

Za ločevanje bolj pomembnih od nepomembnih atributov potrebujeta ReliefF in RReliefF približno enako število primerov kot Gain ratio in MSE, za ločevanje manj pomembnih od nepomembnih atributov pa nekoliko več. Precejšnja in primerljiva je tudi robustnost glede šuma.

Računsko zahtevnost algoritmov Relief lahko precej zmanjšamo z nastavitvijo števila iteracij, saj že z nekaj izračunanimi iteracijami dobimo zadovoljive rezultate, še posebej če je na voljo zadosti primerov.

Z dodajanjem nepomembnih atributov spremenimo problemski prostor (ga povečamo) in algoritmi Relief so občutljivi na te spremembe. Njihova uspešnost je odvisna od zadostne pokritosti problemskega prostora. Na testnih problemih se je pokazalo, da lahko za uspešno ločevanje manj pomembnih atributov od nepomembnih tipično dodamo le nekaj atributov, za ločevanje bolj pomembnih atributov pa tipično nekaj deset atributov. Podobno velja za število pomembnih atributov v problemu: razpoznavanje večjega števila manj pomembnih atributov je vprašljivo, bolj pomembni atributi pa bodo razpoznani tudi v težjih pogojih.

Algoritmi Relief so občutljivi tudi na odvečne atribute. Ocene algoritmov Relief so sorazmerne deležu razlage koncepta in kadar si mora zasluge za razlago deliti več atributov, je dobitek vsakega ustrezno manjši.

Ugotovili smo naslednje pristranosti algoritmov Relief:

- proti večvrednostnim atributom,
- proti večvrednostnim razredom,
- za urejenost vrednosti atributov,
- za bolj pomembne atribute in proti manj pomembnim atributom ter
- proti odvečnim atributom.

7

Uporaba

Resnično reven je ta, ki napačno uporablja, kar ima na voljo. Resnično bogat je ta, ki pametno uporablja, kar ima na voljo.

Mikhaïl Naimy

V prejšnjih poglavjih smo si ogledali teoretično in empirično analizo algoritmov Relief, v tem poglavju pa bomo podali kratek pregled njihove uporabe. Začeli bomo na področjih, ki jim je bil Relief prvotno namenjen, in sicer z izbiro podmnožice pomembnih atributov in z uteževanjem atributov, nadaljevali pa bomo z uporabo v drevesnih modelih, pri diskretizaciji številskih atributov, v induktivnem logičnem programiranju ter povezovalnih pravilih.

7.1 Izbira podmnožice pomembnih atributov in uteževanje atributov

Pri problemu izbire podmnožice pomembnih atributov želimo med vsemi atributi izbrati majhno podmnožico, ki bo v idealnem primeru potrebna in zadostna za opis danega koncepta. Relief je bil prvotno razvit prav v ta namen (Kira in Rendell, 1992a;b) in ocenjujejo ga za enega najboljših algoritmov pri tej nalogi (Dietterich, 1997).

Da bi izbrali podmnožico najbolj pomembnih atributov, je bila uvedena hevristika s pragom pomembnosti θ (Kira in Rendell, 1992a). Če je ocena nekega atributa manjša od θ, smatramo, da je ta atribut nepomemben in ga ne vključimo v izbrano množico. Predlagane so tudi meje za θ in sicer $0 < \theta \leq \frac{1}{\sqrt{\alpha m}}$, kjer je α verjetnost, da je dani atribut nepomemben, čeprav ima oceno kakovosti večjo od praga θ, in m število uporabljenih iteracij (glej sliko 3.1). Zgornja meja za θ je zelo ohlapna in v praksi lahko uporabljamo precej manjše vrednosti.

Na uteževanje atributov lahko gledamo kot na posplošitev izbire podmnožice pomembnih atributov, saj vsakemu atributu namesto dvojiške uteži (0-1, vključi-izključi) priredimo poljubno realno utež. Atribute nato pri uporabi upoštevamo glede na velikost uteži (npr. pri predikciji po metodi bližnjih sosedov). Če uporabljamo algoritme Relief na ta način, ne potrebujemo praga pomembnosti, pač pa lahko za uteževanje atributov uporabimo neposredno njihove ocene kakovosti. Algoritem ReliefF je bil testiran kot metoda uteževanja atributov pri zakasnjenem učenju (Wettschereck in sod., 1997) in ugotovljeno je bilo, da je zelo koristen.

7.2 Izgradnja drevesnih modelov

Pri učenju drevesnih modelov (odločitvenih ali regresijskih dreves) potrebujemo v vsakem vozlišču drevesa hevristiko za ocenjevanje atributov, ki nam bo izbrala ustrezno razbitje. Običajno se v ta namen uporabljajo hevristike, temelječe na funkcijah nečistoče, npr. indeks Gini (Breiman in sod., 1984) ali Gain ratio (Quinlan, 1993a) pri odločitvenih drevesih ter MSE (Breiman in sod., 1984) pri regresijskih drevesih. Te ocene so kratkovidne in ne zmorejo razpoznati močnih pogojnih odvisnosti, so pa tudi neprimerno pristranske pri večvrednostnih atributih (Kononenko, 1995). ReliefF je bil uspešno uporabljen pri gradnji klasifikacijskih (Kononenko in Šimec, 1995; Kononenko in sod., 1997) in regresijskih dreves (Robnik Šikonja in Kononenko, 1996; 1997). Če v podatkih ni močnejših pogojnih odvisnosti med atributi, dobimo drevesa s podobno napako kot s kratkovidno hevristiko. V primeru, da močne pogojne odvisnosti obstajajo, pa lahko dobimo z njuno uporabo mnogo boljše rezultate. Pri analizi novega, neznanega problema je nespametno zavračati možnost, da ta vsebuje močne pogojne odvisnosti in se omejiti le na uporabo kratkovidnih hevristik.

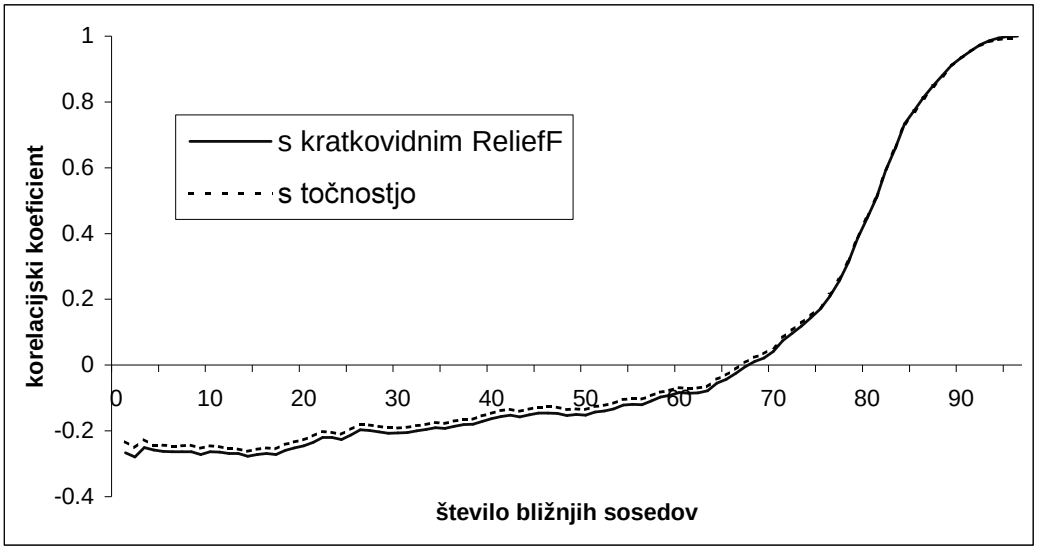
Uporaba funkcij nečistoče za ocenjevanje atributov blizu listov odločitvenega drevesa vodi do neoptimalnih razbitij, upoštevajoč predikcijsko točnost. Kot rešitev tega problema so predlagali (Brodley, 1995; Lubinsky, 1995) preklop na točnost na učni množici kot cenilko atributov. Razvili smo metodo za ta preklop pri algoritmih ReliefF in RReliefF (Robnik Šikonja in Kononenko, 1999), ki jo opisujemo v naslednjem razdelku.

7.2.1 Preklop na drugo hevristiko

Med gradnjo odločitvenega ali regresijskega drevesa z uporabo algoritmov Relief razpoznamo in uporabimo vse pomembne pogojno odvisne atribute že v delu drevesa bliže korenu in pridemo do točke, ko med atributi ni več pomembnih odvisnosti. To je točka, ko postane globalni pogled enak lokalnemu, in lahko začnemo optimizirati drevo glede točnosti napovedanih vrednosti. To točko preklopa lahko

zaznamo tako, da opazujemo korelacijo ocen atributov med kratkovidnim globalnim pogledom, ki smo ga izpeljali z izrazom (4.15), in nekratkovidnim pogledom, ki predstavlja povprečje lokalnih ocen, to pa so kar ocene kakovosti algoritmov Relief (izraz (3.6)).

Izkaže se, da so ocene algoritmov Relief pri spreminjanju lokalnosti zelo podobno korelirane tako s hevristiko kratkovidni ReliefF (izraz (4.15)) kot s točnostjo na učni množici (oziroma MSE) kot cenilko atributov, kar si lahko ogledamo na sliki 7.1 za problem parnosti z dvema pomembnima, desetimi naključnimi atributi ter 200 primeri. Spreminjanje lokalnosti simuliramo s spremembo števila bližnjih sosedov.



Slika 7.1: Korelacija med ocenami algoritma ReliefF ter hevristikama kratkovidni ReliefF in točnost na problemu parnosti.

Točnost na učni množici kot cenilka kakovosti atributov ima precej slabosti v smislu strukture drevesa (Brodley, 1995), vendar je lahko koristna potem, ko smo že izrazili vse odvisnosti in želimo optimizirati drevo za čim manjšo napako napovedane vrednosti. Pri klasifikaciji z večinskim razredom je točnost definirana kot:

$$Acc = \sum_{V} P(V) \max_{C} P(C|V) \tag{7.1}$$

kjer je V vrednost atributa in C vrednost razreda. Če želimo izračunati točnost za dvojiško razbitje večvrednostnega atributa, združimo vrednosti atributa v dve skupini in upoštevamo verjetnosti za skupini v izrazu (7.1). Pri regresijskih problemih

uporabimo za ocenjevanje atributov namesto točnosti MSE (izraz (2.10)).

Ker je podobna korelacija značilna tudi v drugih problemih, nas to vodi k nadaljnji poenostavitvi izbire točke preklopa med hevristikami za ocenjevanje atributov: z opazovanjem korelacije med ocenami algoritmov Relief in točnostjo (MSE) se odločimo, kdaj ni več pomembnih odvisnosti v lokalnem podprostoru (vozlišču drevesa). Ko je korelacijski koeficient zadosti velik, npr. ≥ 0.8, lahko začnemo optimizirati drevo za čim manjšo napako napovedane vrednosti in za cenilko atributov v poddrevesih vozlišča uporabimo izraz (7.1) ali (2.10).

Metodo s preklopom smo vgradili v naš učni sistem in jo empirično ovrednotili na vrsti klasifikacijskih in regresijskih problemov. Primerjali smo jo z gradnjo dreves brez preklopa pri uporabi algoritma ReliefF oz. RReliefF ali hevristike (7.1) oz. (2.10). Statistično pomembnost zmanjšanja napake predikcije smo merili z enostranskim parnim Studentovim t-testom z Bonferonijevim popravkom. Prag zaupanja smo postavili na 0.95. Pokazalo se je, da sta oba pristopa, ki uporabljata Relief, večkrat dosegla značilno manjšo napako kot pristop, ki je za ocenjevanje uporabljal natančnost oz. MSE. Razlike v napaki predikcije med uporabo zgolj algoritmov Relief ter algoritmov Relief s preklopom niso bile nikoli značilne, pač pa so bila značilno manjša drevesa, zgrajena le z algoritmom ReliefF. Podrobnosti so opisane v (Robnik Šikonja in Kononenko, 1999).

Sklenemo lahko, da ReliefF na odločitvenih ter RReliefF na regresijskih drevesih ne potrebujeta tega preklopa, ker ga njune ocene kakovosti implicitno že upoštevajo skozi upoštevanje lokalnosti in normalizacijske faktorje (glej komentarje k izrazu (4.15)).

7.2.2 Konstruktivna indukcija

Konstruktivna indukcija razširja opisni jezik modelov in z njeno pomočjo lahko včasih dobimo nov vpogled v zakonitosti problema. Največji težavi pri uporabi konstruktivne indukcije sta velika računska zahtevnost in nevarnost pretirane prilagoditve podatkom oziroma slaba posplošljivost modelov.

Algoritme Relief smo uporabljali za usmerjanje procesa konstruktivne indukcije med gradnjo odločitvenih in regresijskih dreves. Kot kandidate za konstrukcijo smo izbrali le najbolje ocenjene atribute in jih poskušali združiti z različnimi operatorji (konjunkcija, disjunkcija, vsota, produkt). Ta metoda je zmanjšala računsko zahtevnost v primerjavi z metodo konstruktivne indukcije, ki je uporabljala funkcije nečistoče ob primerljivi napaki končnih modelov.

7.2.3 Razumljivost

Pri realnih problemih iz medicine in ekologije, na katerih smo se učili drevesnih modelov, smo opazili, da so bila drevesa zgrajena z algoritmoma ReliefF ali RRe-

liefF bolj razumljiva strokovnjakom (Dalaka in sod., 2000; Cukjati in sod., 2001).

Na ekološkem problemu modeliranja fotosinteze trave Stipa bromoides (Dalaka in sod., 2000) so drevesni modeli, zgrajeni s funkcijo RReliefF kot cenilko kakovosti atributov, odkrili za strokovnjake potencialno zanimivo novo znanje in sicer, da obstaja prag relativne vlažnosti okoli 35%, kjer se stopnja fotosinteze bistveno razlikuje glede na temperaturo, vlažnost in iradiacijo.

Na medicinskem problemu napovedovanja hitrosti celjenja ran (Cukjati in sod., 2001) smo pri gradnji drevesnih modelov v klasifikaciji in regresiji preskusili več mer za ocenjevanje atributov. Že v začetni fazi analize podatkov so strokovnjaki ocenili, da so najbolj smiselna drevesa, ki smo jih zgradili z uporabo algoritmov ReliefF in RReliefF. Struktura drevesa in izbrane delitve so bile bližje njihovemu načinu razmišljanja, kot pri drevesih, ki smo jih zgradili s pomočjo cenilk Gain ratio in MSE. Čeprav razumljivosti ne moremo jasno in nedvoumno definirati, si večjo razumljivost modelov dobljenih z algoritmi Relief razlagamo z lastnostmi algoritmov (kontekstna občutljivost, upoštevanje lokalnosti, interpretacijo ocen kakovosti kot deleža razloženih sprememb v konceptu), ki jih druge cenilke nimajo.

7.3 Diskretizacija atributov

Diskretizacija razdeli vrednosti številskega atributa v več intervalov, vsak interval pa lahko obravnavamo kot eno vrednost novega, imenskega atributa.

Diskretizacija je v strojnem učenju pomemben korak pri predprocesiranju primerov. Obstaja več razlogov zakaj je potrebna: nekateri algoritmi strojnega učenja ne znajo ali ne morejo obravnavati številskih atributov in potrebujejo diskretizacijo, zmanjšuje računsko zahtevnost, mejne vrednosti intervalov pa morda vsebujejo koristno informacijo (Richeldi in Rossotto, 1995). Ločimo metode diskretizacije, ki ne upoštevajo napovedane vrednosti, in temeljijo na statistikah vrednosti številskih atributov (npr. enaka frekvenca in enaka širina intervala), ter tiste, ki upoštevajo porazdelitev napovedanih vrednosti, ter poskušajo maksimizirati neko hevristiko za ocenjevanje atributov (Cestnik, 1989; Kerber, 1992; Fayad in Irani, 1993; Robnik Šikonja in Kononenko, 1995). Algoritma ReliefF in RReliefF uporabljamo za diskretizacijo številskih atributov v našem učnem sistemu, ki se uči drevesnih modelov.

V (Robnik Šikonja in Kononenko, 1995) se je na umetnih problemih pri uporabi algoritma ReliefF pokazalo, da lahko vsebujejo močno pogojno odvisni številski atributi pomembne meje, ki jih ne moremo odkriti s kratkovidnimi merami. Pri učenju drevesnih modelov dobimo na realnih problemih pri diskretizaciji z algoritmi Relief primerljive rezultate kot z računsko manj zahtevno diskretizacijo, ki uporablja princip minimalne dolžine opisa (Fayad in Irani, 1993).

7.4 ILP in povezovalna pravila

V induktivnem logičnem programiranju (ILP) je bil uporabljen algoritem ReliefF za ocenjevanje kakovosti literalov, ki so kandidati za razširitev trenutnega stavka med indukcijo teorij v logiki prvega reda (Pompe in Kononenko, 1995). Za prilagoditev posebnostim pri ocenjevanju literalov je bilo potrebno spremeniti funkcijo diff, ki je postala asimetrična. Učni sistem, ki je uporabljal to verzijo hevristike ReliefF, je dosegel dobre rezultate na vrsti učnih problemov.

Uporaba algoritma ReliefF skupaj s klasifikatorjem, temelječem na povezovalnih pravilih (Jovanoski in Lavrač, 1999), je zahtevala podobno asimetrično spremembo funkcije diff kot pri uporabi v ILP. ReliefF je bil na tem področju uporabljen za preprocesiranje, v katerem je izbiral podmnožico relevantnih atributov.

8

Sklep

Rad bi te zaprosil, kolikor najbolj morem - bodi potrpežljiv do vsega, kar ostaja nerazvozlano v tvojem srcu, in poskusi vzljubiti vprašanja sama, kot zaprte sobe in kot knjige, ki so napisane v nekem zelo tujem jeziku. Ne išči odgovorov, ki ti ne morejo biti dani, ker jih ne bi bil sposoben živeti. Smisel je v tem, da živiš vse. Živi vprašanja v tem trenutku. Morda ti bo potem nekoč, ne da bi to opazil, kakšen oddaljen dan prinesel odgovor.

Rainer Maria Rilke

8.1 Zaključki

Raziskali smo lastnosti, parametre in uporabo algoritmov iz družine Relief. Ti algoritmi so splošni in uspešni pri ocenjevanju atributov in še posebej dobro zaznavajo pogojne odvisnosti. Omogočajo enoten pogled na ocenjevanje atributov v regresijskih in klasifikacijskih problemih. Ocene kakovosti, ki jih izračunavajo, imajo intuitivno razumljivo interpretacijo. Zaradi teh lepih lastnosti in uspešne uporabe na številnih področjih smo jih analizirali in sistematično raziskali različne lastnosti in parametre.

Predstavili smo verjetnostno interpretacijo ocen kakovosti in pojasnili, da ocene atributov predstavljajo razliko verjetnosti, da atribut loči bližnje primere z različno predikcijo in verjetnosti, da atribut loči bližnje primere s podobno predikcijo. Ta interpretacija je temelj za enoten pogled na ocenjevanje atributov v klasifikaciji in regresiji.

Analiza računske zahtevnosti algoritmov Relief pokaže, da imajo vse inačice algoritma asimptotično enako časovno zahtevnost $O(m \cdot n \cdot a)$. Z uporabo k-d dreves za iskanje bližnjih primerov jo lahko zmanjšamo na $O(a \cdot n \cdot \log n)$, kar je enako zahtevnosti algoritmov za urejanje z več ključi. Relacija med ocenami algoritmov Relief in funkcijami nečistoče pojasni vlogo lokalnosti in konteksta pri

ocenjevanju atributov ter pokaže, da Relief implicitno vsebuje normalizacijo za večvrednostne atribute in razrede. Splošen okvir algoritmov za ocenjevanje kakovosti atributov, ki temeljijo na kontekstni podobnosti med vrednostmi atributa in rešitvijo učnega primera, je primeren za razlago lastnosti ter medsebojno primerjavo teh algoritmov. Preko različnih mer podobnosti lahko pridemo do razširitev in prilagoditev algoritma Relief za različne namene, npr. za induktivno logično programiranje ali cenovno občutljivo učenje. Oceno algoritma Relief v limiti, ko gre število primerov v neskončnost, lahko interpretiramo kot delež sprememb koncepta, ki jih ta atribut razloži. Ta interpretacija daje ocenam kakovosti lahko predstavljivo intuitivno razlago in omogoča razložiti obnašanje ocen kakovosti pri naraščanju števila pomembnih atributov in pri odvečnih atributih.

Teoretična in eksperimentalna analiza različnih parametrov algoritmov Relief pokaže neobčutljivost algoritmov glede na uporabo evklidske ali manhattanske razdalje ter večjo občutljivost pri različnih obravnavah razdalje v problemskem prostoru. Določanje števila in uteženosti bližnjih sosedov je povezano z globalnostjo ocen in kontekstno občutljivostjo, vendar so algoritmi Relief robustni glede tega vprašanja. Pri problemih, ki so opisani z mešanico imenskih in številskih atributov, je koristna uporaba praga različnosti pri številskih atributih. Velikost vzorca in število iteracij določimo glede na pokritosti problemskega prostora s primeri. S tema parametroma lahko kontroliramo tudi računsko zahtevnost.

Spremenili smo definicijo spremenljivosti koncepta tako, da deluje na realnih primerih in upošteva lokalnost prostora. Na podlagi te mere težavnosti koncepta smo skušali razložiti razlike v ocenah kakovosti atributov v različnih problemih. Vpeljali smo meri ločljivosti in uporabnosti ocen atributov. Empirično smo ovrednotili delovanje algoritmov ReliefF in RReliefF na nekaj tipičnih problemih različnih težavnosti. Problemi Modulo-p-I, MONK ter nekaj linearnih in nelinearnih odvisnosti so pokazali, da je uspešnost algoritmov Relief odvisna od zadostnega pokritja karakterističnih območij s primeri. Pokazala se je rahla pristranost ocen v korist pomembnejših atributov. Glede na veliko verjetnost ločljivosti in uporabnosti smo opazovali število potrebnih primerov, število potrebnih iteracij, število nepomembnih atributov v problemu in stopnjo šumnosti napovedane vrednosti. Pokazala se je precejšnja robustnost algoritmov glede na te dejavnike. Rezultati na problemih brez močnih pogojnih odvisnosti kažejo na primerljivost s hevristikama Gain ratio in MSE. Pri večjem številu pomembnih atributov se moramo zavedati, da bolj pomembni atributi lahko prikrijejo učinke manj pomembnih. Odvečni atributi deformirajo problemski prostor in dobijo slabše ocene.

Algoritmi Relief so bili v praksi uporabljeni za izbiro podmnožice pomembnih atributov, uteževanje atributov, izgradnjo drevesnih modelov, usmerjanje konstruktivne indukcije, diskretizacijo številskih atributov, v induktivnem logičnem programiranju in pri klasifikaciji s pomočjo povezovalnih pravil. Pri gradnji drevesnega modela se izkaže, da ne potrebujemo prehoda na minimizacijo napake,

saj ocene kakovosti implicitno že vsebujejo lokalnost.

Pri analizi novega, neznanega problema je nespametno zavračati možnost, da ta vsebuje močne pogojne odvisnosti in se omejiti le na uporabo kratkovidnih hevristik. V zelo velikih problemih, kjer uporabo algoritmov Relief postavi pod vprašaj večja računska zahtevnost (v primerjavi s funkcijami nečistoče), sta rešitev vzorčenje in paralelizacija.

8.2 Nadaljnje delo

Čeprav je bilo na algoritmih Relief opravljenih že dosti raziskav, ostaja precej vprašanj še nerešenih. Predlagali bomo nekaj idej v zvezi s paralelizacijo, inkrementalno uporabo, v časovnih vrstah, v cenovno občutljivih problemih in v meta učenju.

8.2.1 Paralelizacija

Algoritmi Relief so računsko bolj zahtevni kot kratkovidne hevristike, imajo pa možnost, da jih lahko zelo naravno razbijemo v več med seboj neodvisnih nalog, kar je predpogoj za uspešno paralelizacijo algoritma. Vsaka iteracija algoritma je kandidat za poseben proces oziroma nalogo, kar bi spremenilo algoritem v drobnozrnati paralelni algoritem.

8.2.2 Inkrementalno učenje in časovne vrste

Pri inkrementalnem učenju vsi podatki niso na voljo že na začetku učenja ampak prihajajo postopno. Algoritme Relief bi lahko prilagodili za tovrstno učenje, saj algoritmi že sami po sebi delujejo podobno, ko postopno, iz iteracije v iteracijo izboljšujejo zanesljivost ocene. Naši poskusi kažejo (npr. leva stran tabele 6.5), da postanejo rezultati uporabni že po nekaj iteracijah. Rešiti je potrebno vprašanje normalizacije obnovljenih uteži, ki se pojavi s spreminjanjem števila učnih primerov.

Preučiti bi bilo potrebno tudi idejo, da bi opazovali spreminjanje ocene kakovosti pri dodajanju novih podatkov. Na ta način bi morda lahko detektirali pomembne prehode pri učenju časovnih vrst, recimo spremembo konteksta.

8.2.3 Cenovna občutljivost

Algoritmi strojnega učenja ponavadi ne upoštevajo cene napačne klasifikacije, ampak merijo le točnost. Obstaja pa cela vrsta različnih tehnik, ki rešujejo ta problem. Za ReliefF je bila uspešno preskušena sprememba uteži atributov, ki

odraža njihovo cenovno občutljivo apriorno verjetnost (Kukar in sod., 1999). Cenovno občutljivost bi lahko dosegli tudi s spremembo funkcije diff, v katero bi pretvorili ustrezno normalizirano cenovno matriko. Odprto vprašanje ostaja spet normalizacija.

8.2.4 Meta učenje

Eden od pristopov k meta učenju je, da zberemo bazo podatkov različnih značilnosti podatkov (meta podatke) in razpoložljivih učnih algoritmov in se nato poskušamo naučiti izbire najprimernejšega algoritma za dani problem. Nekatere značilnosti, ki jih bomo uporabili so očitne, npr. število in tip atributov, število razredov, število primerov, uporabljajo pa se tudi bolj zapletene mere npr. (Vilalta, 1999) uporablja spremenljivost koncepta. Ocene atributov, ki jih izračunajo algoritmi Relief, so lahko koristen vir meta podatkov. Uporabili bi oceno najboljšega atributa, razliko med najboljšim in najslabšim atributom, korelacijo z ocenami kratkovidne hevristike, itd.

A

Dodatek: opis problemov

Predstavljamo nekatere značilnosti problemov, ki smo jih uporabljali pri analizi in za demonstracijo lastnosti algoritmov Relief. V tabeli A.1 po vrsti podajamo za vsak problem ime, sklicevanje na opis ali definicijo, število razredov ali oznako r, če gre za regresijski problem, število pomembnih atributov (I), število naključnih atributov (R), tip atributov in spremenljivost koncepta, izračunano iz 1000 primerov po enačbi (6.4) (razen za problema MONK-1 in MONK-3, kjer smo V izračunali iz popolnega opisa problema, iz 432 primerov).

Vseh problemov predhodno še nismo izrecno definirali, zato podajamo definiciji tu:

Cosinus-Lin je nelinearen regresijski problem, ki ga sestavljata dve periodi funkcije kosinus, pomnoženi z linearno kombinacijo dveh atributov:

$$\mathrm{Cosinus-Lin}: \qquad \tau = \cos(4\pi A_1) \cdot (-2A_2 + 3A_3) \tag{A.1}$$

Atributi so številski z vrednostmi med 0 in 1.

Fraction-I je normalizirana inačica problema Modulo-∞-I (glej definicijo (6.7). Napovedana vrednost in atributi so normalizirani na interval $[0, 1]$. Problem lahko opišemo kot posplošitev problema parnosti reda I na realna števila: napovedana vrednost je namreč neceli del vsote I pomembnih atributov:

$$\mathrm{Fraction-I}: \qquad \tau = \sum_{j=1}^{I} A_j - \left\lfloor \sum_{j=1}^{I} A_j \right\rfloor \tag{A.2}$$

Tabela A.1: Kratek opis uporabljenih problemov.

ime	definicija	#razr.	I	R	tip atr.	V
Bool-Simple	izraz (5.7)	2	4	10	imenski	0.283
Modulo-2-2-c	izraz (6.7)	2	2	10	imenski	0.278
Modulo-2-3-c	izraz (6.7)	2	3	10	imenski	0.379
Modulo-2-4-c	izraz (6.7)	2	4	10	imenski	0.444
Modulo-5-2-c	izraz (6.7)	5	2	10	imenski	0.651
Modulo-5-3-c	izraz (6.7)	5	3	10	imenski	0.748
Modulo-5-4-c	izraz (6.7)	5	4	10	imenski	0.790
Modulo-10-2-c	izraz (6.7)	10	2	10	številski	0.808
Modulo-10-3-c	izraz (6.7)	10	3	10	številski	0.891
Modulo-10-4-c	izraz (6.7)	10	4	10	številski	0.902
MONK-1	razdelek 6.2.2	2	3	3	imenski	0.213
MONK-3	razdelek 6.2.2	2	3	3	imenski	0.145
Modulo-5-2-r	izraz (6.7)	r	2	10	imenski	0.658
Modulo-5-3-r	izraz (6.7)	r	3	10	imenski	0.755
Modulo-5-4-r	izraz (6.7)	r	4	10	imenski	0.803
Modulo-10-2-r	izraz (6.7)	r	2	10	številski	0.801
Modulo-10-3-r	izraz (6.7)	r	3	10	številski	0.885
Modulo-10-4-r	izraz (6.7)	r	4	10	številski	0.892
Fraction-2	izraz (A.2)	r	2	10	številski	0.777
Fraction-3	izraz (A.2)	r	3	10	številski	0.845
Fraction-4	izraz (A.2)	r	4	10	številski	0.869
LinInc-2	izraz (6.8)	r	2	10	številski	0.642
LinInc-3	izraz (6.8)	r	3	10	številski	0.707
LinInc-4	izraz (6.8)	r	4	10	številski	0.709
LinEq-2	izraz (6.9)	r	2	10	številski	0.660
LinEq-3	izraz (6.9)	r	3	10	številski	0.693
LinEq-4	izraz (6.9)	r	4	10	številski	0.726
Cosinus-Lin	izraz (A.1)	r	3	10	številski	0.588
Cosinus-Hills	izraz (6.10)	r	2	10	številski	0.848

B

Dodatek: učni sistem CORE

Vsi algoritmi opisani v tem delu so implementirani v učnem sistemu CORE. Ta izvira iz regresijskega sistema (Robnik Šikonja, 1997), ki je bil razširjen tudi na klasifikacijo. Programa za klasifikacijo in regresijo imata v izvorni kod precej skupnih modulov in vsebujeta nekaj nad 30.000 vrstic v ANSI C++ programskem jeziku. CORE lahko poganjamo na Windows ali Unix platformi. Je prosto razširljiv pod GNU licenco.

Opisali bomo različne algoritme in zmožnosti, ki jih bomo razdelili na več sklopov: vhodno - izhodne operacije, ocenjevanje atributov, modele za predikcijo, konstruktivno indukcijo in rezanje dreves.

Vhodno-izhodne operacije so skupne za regresijski in klasifikacijski del sistema in imajo naslednje značilnosti:

- podatke preberemo v oblike tabele, ločene z vejico ali tabulatorjem,
- podprta so vnaprej definirana razbitja na učno in testno množico ter razbitja generirana po načelu večkratnega prečnega preverjanja,
- rezultate izpisujemo na zaslon in v datoteke,
- drevesne modele izpisujemo s tekstovno grafiko ter v formatu *dot* za opis grafov (Koutsofios in North, 1991), ki nam omogoča predstavitev drevesnih modelov v formatih postscript (EPS), HPGL, PCL-5 in FrameMaker (MIF).

Ocenjevanje atributov; številske ter večvrednostne imenske atribute lahko ocenjujemo kot dvojiške, kar je smiselno, če bodo nastopali v dvojiškem drevesnem modelu. V tem primeru je ocena atributa maksimum po vseh možnih dvojiških razbitjih (mogoče je vključiti tudi hevristično iskanje najboljših dvojiških razbitij).

Pri učenju drevesnih modelov je mogoče vključiti prehod na drugo cenilko blizu listov dreves po metodi opisani v (Robnik Šikonja in Kononenko, 1999).

Vključenih je več mer za ocenjevanje kakovosti atributov.

- Za klasifikacijske probleme:
  - Relief (slika 3.1),
  - ReliefF (slika 3.2) z inačicami:
    - z enakim vplivom k najbližjih sosedov (kot na sliki 3.2),
    - z eksponentno padajočim vplivom k najbližjih primerov na podlagi vrstnega reda po padajoče urejeni razdalji (enačba (5.4)),
    - z najboljše ocenjenim številom bližnjih sosedov (stran 41),
    - z upoštevanjem dejanske razdalje (enačba (5.5)),
    - z upoštevanjem kvadrata dejanske razdalje (enačba (5.6)),
  - informacijski prispevek, (gain v enačbi (2.6)),
  - razmerje informacijskega prispevka (Gain ratio, enačba (2.6)),
  - indeks Gini (Breiman in sod., 1984),
  - MDL (Kononenko, 1995) in
  - točnost (izraz (7.1)).
- Za regresijske probleme:
  - RReliefF (slika 3.3) z inačicami:
    - z enakim vplivom k najbližjih sosedov (enačba (3.13)),
    - z eksponentno padajočim vplivom k najbližjih primerov na podlagi vrstnega reda po padajoče urejeni razdalji (enačba (3.12)),
    - z najboljše ocenjenim številom bližnjih sosedov (stran 41),
    - z upoštevanjem dejanske razdalje (enačba (5.5)),
    - z upoštevanjem kvadrata dejanske razdalje (enačba (5.6)),
  - srednja kvadratna napaka (MSE) povprečja napovedanih vrednosti (enačba (2.10), kjer kot model uporabimo povprečje napovedanih vrednosti primerov),
  - srednja kvadratna napaka izbranega modela (enačba (2.10)),
  - srednja absolutna napaka izbranega modela (namesto kvadrata, uporabimo absolutno vrednost v enačbi (2.11)),
  - RReliefF in MSE kombinirana s principom MDL (Robnik Šikonja, 1997).

Modeli za predikcijo; večino modelov lahko uporabljamo samostojno ali v listih drevesnega modela. Za klasifikacijo in regresijo je implementiranih več različnih modelov.

- Za klasifikacijske probleme:
  - večinski razred,
  - k najbližjih sosedov in
  - naivni Bayesov klasifikator.
- Za regresijske probleme:
  - povprečje napovedanih vrednosti na učni množici,
  - mediana napovedanih vrednosti na učni množici,
  - linearna regresija po metodi dekompozicije singularnih vrednosti (Press in sod., 1988),
  - linearna regresija z odstranjevanjem nepomembnih atributov kot v sistemu M5 (Quinlan, 1993b),
  - linearna regresija z odstranjevanjem nepomembnih atributov po principu MDL (Robnik Šikonja in Kononenko, 1998),
  - k najbližjih sosedov in
  - regresija z jedrnimi funkcijami.

Konstruktivna indukcija dovoljuje uporabo naslednjih operatorjev:

- konjunkcija členov, kjer so členi lahko ena ali več vrednosti imenskega atributa ter intervali vrednosti številskih atributov,
- vsota številskih atributov,
- produkt številskih atributov.

Rezanje dreves; tako pri odločitvenih kot pri regresijskih drevesih uporabljamo iste principe rezanja:

- z m-oceno verjetnosti po principu (Niblett in Bratko, 1986),
- po principu cene napake in velikosti drevesa (Breiman in sod., 1984) in
- po principu MDL (Robnik Šikonja in Kononenko, 1998).

CORE lahko uporabljamo v paketnem načinu z različnimi opcijami ali iz tekstovnih menujev. Poleg učenja različnih modelov in ocenjevanja atributov z različnimi metodami lahko zahtevamo tudi izpis spremenljivosti koncepta, uporabljene diskretizacije številskih atributov in ocen verjetnosti predikcije za posamezen primer.

Literatura

Jon Luis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 15(9):509–517, 1975.

Avrim L. Blum, Pat Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, Charles J. Stone. *Classification and regression trees*. Wadsworth Inc., Belmont, California, 1984.

Alan J. Broder. Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23(1/2):171–178, 1990.

Carla E. Brodley. Automatic selection of split criterion during tree growing based on node location. V *Machine Learning: Proceedings of the Twelfth International Conference (ICML'95)*, str. 73–80. Morgan Kaufmann, 1995.

Carla E. Brodley, Paul E. Utgoff. Multivariate decision trees. *Machine Learning*, 19(1):45–77, 1995.

Bojan Cestnik. Informativity-based splitting of numerical attributes into intervals. V *Expert Systems Theory & Applications, Proceedings of the IASTED International Symposium*. Acta Press, 1989.

Bojan Cestnik, Igor Kononenko, Ivan Bratko. Assistant 86: A knowledge-elicitation tool for sophisticated users. V Ivan Bratko, Nada Lavrač, uredniki, *Progress in Machine Learning, Proceedings of European Working Session on Learning EWSL'87*, str. 31–36, Wilmslow, 1987. Sigma Press.

David Cukjati, Marko Robnik-Šikonja, Stanislav Reberšek, Igor Kononenko, Damijan Miklavčič. Prognostic factors, prediction of chronic wound healing and electrical stimulation. *Medical & Biological Engineering & Computing*, 2001. (submitted).

Anastasia Dalaka, Boris Kompare, Marko Robnik-Šikonja, Stefanos Sgardelis. Modeling the effects of environmental conditions on apparent photosynthesis of Stipa bromoides by machine learning tools. *Ecological Modelling*, 129:245–257, 2000.

Janez Demšar, Blaž Zupan. Orange, machine learning library in Python, 2001. http://magix.fri.uni-lj.si/orange/.

Kan Deng, Andrew W. Moore. Multiresolution instance-based learning. V *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)*, str. 1233–1239. Morgan Kaufmann, 1995.

Thomas G. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.

Thomas G. Dietterich, Jude W. Shavlik, uredniki. *Readings in Machine Learning*. Morgan Kaufman, 1990.

Pedro Domingos. Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review*, 11:227–253, 1997.

Usama M. Fayad, Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. V *Proceedings of the 13th International Joint Conference on Aartificial Intelligence (IJCAI'93)*, str. 1022–1027. Morgan Kaufmann, 1993.

Jerome H. Friedman, Jon Luis Bentley, Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. Technical Report STAN-CS-75-482, Stanford University, 1975.

Se June Hong. Use of contextual information for feature ranking and discretization. Technical Report RC19664, IBM, July 1994.

Se June Hong. Use of contextual information for feature ranking and discretization. *IEEE transactions on knowledge and data engineering*, 9(5):718–730, 1997.

Earl B. Hunt, Janet Martin, Philip J. Stone. *Experiments in Induction*. Academic Press, New York, 1966.

Viktor Jovanoski, Nada Lavrač. Feature subset selection in association rules learning systems. V Marko Grobelnik, Dunja Mladenič, uredniki, *Prooceedings of the Conference Analysis, Warehousing and Mining the Data (AWAMIDA'99)*, str. 74–77, 1999.

Randy Kerber. ChiMerge: Discretization od numeric attributes. V *Proceedings of AAAI'92*, 1992.

Kenji Kira, Larry A. Rendell. The feature selection problem: traditional methods and new algorithm. V *Proceedings of AAAI'92*, 1992a.

Kenji Kira, Larry A. Rendell. A practical approach to feature selection. V D.Sleeman, P.Edwards, uredniki, *Machine Learning: Proceedings of International Conference (ICML'92)*, str. 249–256. Morgan Kaufmann, 1992b.

Igor Kononenko. Estimating attributes: analysis and extensions of Relief. V Luc De Raedt, Francesco Bergadano, uredniki, *Machine Learning: ECML-94*, str. 171–182. Springer Verlag, 1994.

Igor Kononenko. On biases in estimating multi-valued attributes. V *Proceedings of the International Joint Conference on Aartificial Intelligence (IJCAI'95)*, str. 1034–1040. Morgan Kaufmann, 1995.

Igor Kononenko. *Strojno učenje*. Založba FE in FRI, 1997.

Igor Kononenko, Edi Šimec. Induction of decision trees using ReliefF. V G. Della Riccia, R. Kruse, R. Viertl, uredniki, *Mathematical and Statistical Methods in Artificial Intelligence, CISM Courses and Lectures No. 363*. Springer Verlag, 1995.

Igor Kononenko, Edvard Šimec, Marko Robnik-Šikonja. Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence*, 7:39–55, 1997.

Eleftherios Koutsofios, Stephen North. Drawing graphs with *dot*. Technical Report 910904-59113-08TM, AT&T Bell Laboratories, Murray Hill, NJ, September 1991.

Matjaž Kukar, Igor Kononenko, Ciril Grošelj, Katarina Kralj, Jure Fettich. Analysing and improving the diagnosis of ischaemic heart disease with machine learning. *Artificial Intelligence in Medicine*, 16:25–50, 1999.

Huan Liu, Hiroshi Motoda, uredniki. *Feature extraction, construction and selection: a data mining perspective*. Kluwer Academic Publishers, 1998.

David J. Lubinsky. Increasing the performance and consistency of classification trees by using the accuracy criterion at the leaves. V *Machine Learning: Proceedings of the Twelfth International Conference (ICML'95)*, str. 371–377. Morgan Kaufmann, 1995.

Ramon Lopez Mantaras. ID3 revisited: A distance based criterion for attribute selection. V *Proceedings of Int. Symp. Methodologies for Intelligent Systems*, Charlotte, North Carolina, USA, October 1989.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

Andrew W. Moore, Jeff Schneider, Kan Deng. Efficient locally weighted polynomial regression predictions. V Douglas H. Fisher, urednik, *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*, str. 236–244. Morgan Kaufmann, 1997.

Patrick M. Murphy, David W. Aha. UCI repository of machine learning databases, 1995. http://www.ics.uci.edu/ mlearn/MLRepository.html.

Tim Niblett, Ivan Bratko. Learning decision rules in noisy domains. V M. Bramer, urednik, *Developement in Expert Systems*. Cambridge University Press, 1986.

Eduardo Perèz, Larry A. Rendell. Learning despite concept variation by finding structure in attribute-based data. V *Machine Learning: Proceedings of the Thirteenth International Conference (ICML'96)*, str. 391–399, 1996.

Uroš Pompe, Igor Kononenko. Linear space induction in first order logic with ReliefF. V G. Della Riccia, R. Kruse, R. Viertl, uredniki, *Mathematical and Statistical Methods in Artificial Intelligence, CISM Courses and Lectures No. 363*. Springer Verlag, 1995.

William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. *Numerical recipes in C*. Cambridge University Press, 1988.

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

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993a.

J. Ross Quinlan. Combining instance-based and model-based learning. V *Machine Learning: Proceedings of the X. International Conference (ICML'93)*, str. 236–243. Morgan Kaufmann, 1993b.

Richard R. Redner, Homer F. Walker. Mixture densities, maximum likehood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

Larry A. Rendell, Raj Seshu. Learning hard concepts through constructive induction: Framework and rationale. *Computational Intelligence*, 6:247–270, 1990.

Francesco Ricci, Paolo Avesani. Learning a local similarity metric for case-based reasoning. V *Proceedings of the international conference on case-based reasoning (ICCBR-95)*, Sesimbra, Portugal, 1995.

Marco Richeldi, Mauro Rossotto. Class-driven statistical discretization of continuous attributes. V *Machine Learning: ECML-95*, 1995.

Marko Robnik Šikonja. Speeding up Relief algorithm with k-d trees. V *Proceedings of Electrotehnical and Computer Science Conference (ERK'98)*, str. B:137–140, Portorož, Slovenia, 1998.

Marko Robnik Šikonja. CORE - a system that predicts continuous variables. V *Proceedings of Electrotehnical and Computer Science Conference (ERK'97)*, str. B145–148, Portorož, Slovenia, 1997.

Marko Robnik Šikonja, Igor Kononenko. Discretization of continuous attributes using ReliefF. V *Proceedings of Electrotehnical and Computer Science Conference (ERK'95)*, str. B149–152, Portorož, Slovenia, 1995.

Marko Robnik Šikonja, Igor Kononenko. Context sensitive attribute estimation in regression. V Miroslav Kubat, Gerhard Widmer, uredniki, *Proceedings of ICML'96 workshop on Learning in context sensitive domains*, str. 43–52. Morgan Kaufmann, 1996.

Marko Robnik Šikonja, Igor Kononenko. An adaptation of Relief for attribute estimation in regression. V Douglas H. Fisher, urednik, *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*, str. 296–304. Morgan Kaufmann, 1997.

Marko Robnik Šikonja, Igor Kononenko. Attribute dependencies, understandability and split selection in tree based models. V Ivan Bratko, Sašo Džeroski, uredniki, *Machine Learning: Proceedings of the Sixteenth International Conference (ICML'99)*, str. 344–353. Morgan Kaufmann, 1999.

Marko Robnik Šikonja. Razvoj hevristik za usmerjanje učenja regresijskih dreves. Univerza v Ljubljani, Fakulteta za računalništvo in informatiko, 1997. (magistrsko delo).

Marko Robnik Šikonja, Igor Kononenko. Pruning regression trees with MDL. V Henri Prade, urednik, *Proceedings of 13th European Conference on Artificial Intelligence, ECAI'98*, str. 455–459, Brighton, UK, 1998. John Willey & Sons.

Padhraic Smyth, Rodney M. Goodman. Rule induction using information theory. V Gregory Piatetsky-Shapiro, William J. Frawley, uredniki, *Knowledge Discovery in Databases*. MIT Press, 1990.

Padhraic Smyth, Jeff Mellstrom. Detecting novel classes with applications to fault diagnosis. V Derek Sleeman, Peter Edwards, uredniki, *Machine Learning: Proceedings of the Ninth International Workshop*, str. 416–425, 1992.

Sebastian B. Thrun, Jerzy W. Bala, Eric Bloedorn, Ivan Bratko, Bojan Cestnik, John Cheng, Kenneth De Jong, Sašo Džeroski, Scott E. Fahlman, Douglas H. Fisher, R. Hamann, Kenneth A. Kaufman, Stefan F. Keller, Igor Kononenko, Jürgen Kreuziger, Ryszard S. Michalski, Tom Mitchell, Peter W. Pachowicz, Yoram Reich, H. Vafaie, Walter Van de Welde, Walter Wenzel, Janusz Wnek, Jianping Zhang. The MONK's problems - a performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, December 1991.

Ricardo Vilalta. Understanding accuracy performance through concept characterization and algorithm analysis. V *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work*, str. 3–9, 1999.

Mark Allan Weiss. *Data Structures and Algorithm Analysis*. The Benjamin/Cummings, 1995.

Dietrich Wettschereck, David W. Aha, Takao Mohri. A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, 11:273–314, 1997.

Ian H. Witten, Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

IZJAVA

Izjavljam, da sem doktorsko disertacijo z naslovom ,,Lastnosti in uporaba hevristične funkcije Relief v strojnem učenju" samostojno izdelal pod vodstvom mentorja prof. dr. Igorja Kononenka. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.

Marko Robnik Šikonja