UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Mitja Pugelj

Napovedovanje strukturiranih vrednosti z metodo najbližjih sosedov

DIPLOMSKO DELO NA INTERDISCIPLINARNEM UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Janez Demšar
Somentor: prof. dr. Sašo Džeroski

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00015/2010
Datum: 05.04.2010

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko ter Fakulteta za matematiko in fiziko izdaja naslednjo nalogo:

Kandidat: **MITJA PUGELJ**

Naslov: **NAPOVEDOVANJE STRUKTURIRANIH VREDNOSTI Z METODO NAJBLIŽJIH SOSEDOV**

PREDICTION OF STRUCTURED VALUES USING K NEAREST NEIGHBOURS

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Metoda k najbližjih sosedov je preprost primer lenega učenja, ki ga pogosto uporabljamo v klasifikaciji in regresiji. V okviru diplomskega dela sestavite in implementirajte različico algoritma za napovedovanje strukturiranih vrednosti in raziščite njegovo uporabnost. Primerjajte različne nastavitve algoritma, kot so izbor algoritma za iskanje najbližjih sosedov, število uporabljenih sosedov in njihovo uteževanje. Algoritem primerjajte tudi z drugimi uveljavljenimi algoritmi za tovrstne učne probleme, pri čemer uporabite čim širši nabor primernih problemskih domen.

Mentor:

doc. dr. Janez Demšar

Dekan Fakultete za računalništvo in informatiko:

prof. dr. Franc Solina

Somentor:

prof. dr. Sašo Džeroski

Dekan Fakultete za matematiko in fiziko:

prof. dr. Andrej Likar

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Mitja Pugelj, z vpisno številko 63040293, sem avtor diplomskega dela z naslovom: **Napovedovanje strukturiranih vrednosti z metodo najbližjih sosedov**.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Janeza Demšarja in somentorstvom prof. dr. Sašota Džeroskega,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ilirski Bistrici, dne 16.9.2010 Podpis:

Zahvala

Iskrena hvala predvsem mentorju na IJS Sašu Džeroskemu in mentorju na FRI Janezu Demšarju za predlog teme, mentorstvo, potrpežljivost in preglede diplomskega dela. Hvala tudi Dragiju Kocevu in ostalim na IJS, ki so pomagali z nasveti okoli sistema Clus, za uporabo podatkovnih baz, rezultatov ter strežnikov.

Zahvala iz srca gre tudi staršem za podporo in vsem prijateljem za družbo tekom študijskih let.

Simboli in kratice

n Število primerkov v podatkovni bazi.

d Število opisnih atributov v podatkovni bazi, število dimenzij.

k Število najbližjih sosedov, ki jih uporabimo za napovedovanje.

r Število ciljnih spremenljivk.

$\mathbf{d^{-1}}$ Inverzno obteževanje glasovanja, glej stran 11.

$\mathbf{d_{-1}}$ Obratno obteževanje glasovanja, glej stran 11.

PCT Drevesa za napovedno razvrščanje (*Predictive clustering trees*), glej stran 14.

PCR Pravila za napovedno razvrščanje (*Predictive clustering rules*), glej stran 16.

CA Klasifikacijska točnost (*Classification accuracy*), glej razdelek 2.1.

RMSE Koren pričakovane vsote kvadratov napak (*Root Mean Square Error*), glej razdelek 2.1.

QDM Kvalitativna mera razdalje (*Qualitative distance measure*), glej razdelek 2.2.3.

`RF` Obteževanje atributov z metodo `Random Forest`, glej stran 32.

`VP` Metoda iskanja najbližjih sosedov, ki implementira vp iskalno drevo.

`KD` Metoda iskanja najbližjih sosedov, ki implementira kd iskalno drevo.

`BF` Metoda iskanja najbližjih sosedov, ki temelji na navadnem iskanju.

Kazalo

Povzetek

V diplomskem delu preverimo uspešnost metode najbližjega soseda za napovedovanje strukturiranih vrednosti, kot so večciljna klasifikacija in regresija, hierarhična večznačna klasifikacija in kratke časovne vrste. Predstavimo probleme in tehnike napovedovanja omenjenih podatkov.

Za testiranje uspešnosti uporabimo različne podatkovne baze iz različnih (večinoma okoljskih) domen. Preizkusimo tudi vpliv obteževanja glasovanja in obteževanja atributov z metodo Random Forest. Uspešnost metode primerjamo z drevesi in pravili za napovedno razvrščanje. Izkaže se, da se lahko metoda najboljšega soseda za omenjene naloge pogosto kosa z drevesi in pravili, od slednjih je v določenih primerih tudi značilno boljša. V delu glasujemo z do največ petnajstimi najbližjimi sosedi (pogosto precej manj) in glede na rezultate lahko pričakujemo še boljšo uspešnost metode pri glasovanju več sosedov.

Za namene iskanja najbližjega soseda smo implementirali tri metode: navadno iskanje, kd drevo in vp drevo. Metode med seboj primerjamo glede na čas, ki ga porabijo za iskanje v prostoru, v katerega so primeri umeščeni enakomerno naključno. Ugotovimo, da vp drevo sicer hitreje išče v prostorih visokih dimenzij, a tudi njega hitro zadane problem predimenzioniranosti in najučinkovitejša metoda postane navadno iskanje. Za naše implementacije eksperimentalno poiščemo pravilo, ki glede na število primerov, dimenzij in iskanih sosedov izbere predvidoma najhitrejšo metodo.

Implementacijo napišemo v programskem jeziku Java kot del sistema za napovedno razvrščanje Clus.

Ključne besede: podatkovno rudarjenje, napovedovanje strukturiranih podatkov, metoda najbližjega soseda, vp drevo, Clus.

Abstract

In this work we are interested in prediction accuracy of nearest neighbour method for predicting structured values; multiclass classification and regression, hierarchical multilabel classification and short time series. Problems and techniques for dealing with this kind of data are presented.

Prediction accuracy is tested on various datasets from various (but mostly enviromental) domains. For some cases we also check influence of different vote (distance) weighting schemes and feature weighting using Random Forest method. Method's accuracy is compared to predictive clustering rules and trees. We show that nearest neighbour method is capable of predicting structured data with accuracy comparable to trees and rules. Furtherwore, method is in some cases significantly better than rules. In our work we have only tested prediction with at most fifteen voting neighbours. We expect that method could perform even better when more neighbours are used.

We implement three nearest neighbour search methods: simple search, kd tree and vp tree. All methods are compared regarding to time spent for searching in space (where instances are distributed according to random uniform distribution). We conclude that vp tree is faster than kd for high dimension spaces, but has also limitations (dimensionality cource) and simple search outperforms it for high-dimension spaces. For our implementations, we derive a simple (experiment-based) rule that decides which method to use according to number of instances, dimensions and voting neighbours.

Implementation is done in Java as part of Clus - system for predictive clustering.

Keywords: data mining, structured data prediction, nearest neighbour method, vp tree, Clus.

Poglavje 1

Uvod

Podatkovno rudarjenje se ukvarja s pridobivanjem znanja in vzorcev iz podatkovnih baz. Uporaba metod podatkovnega rudarjenja sega na mnoga področja: od marketinga, sprejemanja poslovnih odločitev, strojništva pa vse do bioinformatike, medicine in odkrivanja enačb. Pogosta naloga podatkovnega rudarjenja je tudi napovedovanje neznanih vrednosti novim primerom na podlagi podatkovne baze, ki vsebuje primere istega tipa in katerih vrednosti so znane.

Napovedovanje spremenljivk se torej ubada z gradnjo modelov, ki zmorejo napovedati ciljno spremenljivko glede na vrednosti opisnih spremenljivk nekega primera. Postopku gradnje modela pravimo učenje. Vhod v učenje je navadno množica primerov X, sestavljena iz parov $x = (D_x, T(x))$, kjer je $D_x \in D$ seznam opisnih atributov in $T(x)$ ciljna spremenljivka, ki je v učnih primerih znana. Rezultat učenja je model, funkcija

$$m : D \mapsto T.$$

Od modela pričakujemo, da bo sposoben napovedovati ciljne spremenljivke z določeno natančnostjo, točnostjo oz. karseda majhno napako.

Navadno se podatkovno rudarjenje ubada s primeri, kjer je $T(x)$ enostavna spremenljivka, ki zavzema realno ali nominalno, npr. [“sončno”, “oblačno”].

Včasih pa si želimo napovedovati strukturirane podatke kot so seznam spremenljivk (večciljno napovedovanje), drevesa ali aciklični usmerjeni grafi (hierarhično večznačno napovedovanje), časovne vrste, ipd. V tem delu se ubadamo z napovedno

uspešnostjo metode najbližjega soseda za napovedovanje omenjenih strukturiranih podatkov. Metoda najbližjega soseda je ena najenostavnejših metod podatkovnega rudarjenja in je kot taka uporabna za uvodno rudarjenje po podatkih ali kot metoda za primerjavo uspešnosti ostalih klasifikatorjev. Pogosto pa se izkaže povsem zadovoljivo ali celo odlično tudi kot dejanski klasifikator. Metoda se za napovedovanje zanaša na glasovanje najbližnjih nekaj primerov v prostoru opisnih atributov.

Implementirali smo tri različne pristope k iskanju najbližjega soseda: **navadno iskanje**, ki pregleda celotno učno množico, in metodi na osnovi **kd** ter **vp** dreves. Slednji metodi zgradita drevesi z rekurzivno delitvijo prostora na dva podprostora. Bolj znana kd drevesa to počnejo na podlagi vrednosti posameznih vrednosti atributov, vp drevesa pa izkoriščajo posebno projekcijo razdalje in omogočajo iskanje tudi v neevklidskih prostorih.

Vse tri metode primerjamo na umetnih in realnih podatkih ter tako dobimo vpogled v to, kdaj katera od metod poišče najbližje elemente hitreje. Rezultate uporabimo za avtomatično preklapljanje med metodami, kar doprinese k hitrejšemu rudarjenju za uporabnika sistema.

Poglavje 2

Ozadje

V tem razdelku se seznanimo s problemom napovedovanja spremenljivk v splošnem in z napovedovanjem z uporabo metode najbližjega soseda. Predstavimo problem napovedovanja strukturiranih podatkov, s katerim se ubadamo v poznejših poglavjih. Seznanimo se tudi z drevesi in pravili za napovedno razvrščanje, s katerimi primerjamo metodo najbližjega soseda.

2.1 Napovedovanje vrednosti spremenljivk

Napovedovanje spremenljivk je naloga strojnega učenja, katere namen je določitev ciljnih vrednosti objektom oz. primerom, ki imajo te vrednosti neznane. Postopek poskuša posnemati učenje ljudi, ki na podlagi opazovanj, izkušenj zgradijo modele, s katerimi nato vrednotijo nove situacije in reči.

Vhod v učenje predstavlja nabor učnih primerov $X = x_1, x_2, \ldots, x_n$. Posamezen primerek x_i je opisan s seznamom opisnih atributov $(x_{i,1}, x_{i,2}, \ldots, x_{i,d})$, kjer je $x_{i,j} \in D_j$, seznam $D = (D_1, D_2, \ldots, D_d)$ pa je seznam domen posameznih atributov. Ko bo nedvoumno, bomo x_i pisali kot x, $x_{i,j}$ pa kot x_j. Za učne primere poznamo tudi ciljne vrednosti, ki jih označimo s $T(x_i) \in T$. Neznane ciljne vrednosti $T(y)$ novih primerov y dobimo z modelom $m(y)$. V najbolj enostavnem primeru je vrednost $T(y)$ ena sama spremenljivka, in sicer ena od treh možnosti:

realno število Vrednost je poljubno realno število, predstavljeno poljubno natančno

(seveda znotraj omejitev sistema). *Npr.: 3, 3.14, -0.5, 6.1e-4,..* Napovedovanju realnih vrednosti pravimo **regresija**.

nominalna vrednost Ciljna spremenljivka nosi eno od vrednosti iz seznama možnih vrednosti. Možnim nominalnim vrednostim pravimo tudi razredi. *Npr.: "sončno, oblačno", "moški, ženska", "zelena, rumena, modra, oranžna",...* Za nominalne vrednosti je značilno, da nad njimi ni definirana urejenost. Napovedovanju nominalnih vrednosti pravimo **klasifikacija**. V primeru, ko imamo na voljo le dve vrednosti (npr.: "da, ne"), govorimo o **binarnem razredu** in **binarni klasifikaciji**. Tipično se klasifikacija binarnih razredov ukvarja z vprašanjem ali nek primerek pripada določenem razredu ali ne, npr.: "Je pacient bolan?" ali "Bo stranka vrnila kredit?", ipd.

ordinalna vrednost Gre za podobno kategorijo, kot so nominalne vrednosti, le da je nad zalogo vrednosti definirana urejenost, npr.: "slab, povprečen, dober", "1, 2, 3, 4, 5". Urejenost nam daje dodatno znanje, s katerim lahko izboljšamo kvaliteto napovedovanja. Pogosto pa ordinalne vrednosti obravnavamo enako kot nominalne.

V zadnjih letih postaja aktualnejše tudi napovedovanje **strukturiranih podatkov**, kot so vektorji spremenljivk, drevesa, časovne vrste,... Z napovedovanjem slednjih se ukvarjamo v tem delu.

Čeprav so opisni atributi v večini podatkovnih baz bodisi numerični bodisi nominalni (to velja tudi za vse atribute v našem delu), si lahko predstavljamo tudi postavitve, kjer bi vrednost napovedovali iz strukturiranih podatkov, ordinalne vrednosti, ipd. Vse naštete vrste ciljnih spremenljivk lahko torej uporabimo tudi kot opisne atribute.

Da lahko govorimo o uspešnosti metode in primerjamo metode med seboj, potrebujemo merilo uspešnosti metode. Tu predstavimo dva pogosta načina ocenjevanja metod, ki ju (s prilagoditvami) uporabljamo preko celotnega dela.

V kolikor imamo opravka s klasifikacijo, navadno govorimo o klasifikacijski točnosti metode. *Klasifikacijska točnost* (angleško *Classification Accuracy - CA*) je odstotek pravilno napovedanih primerov. Bližje kot je *CA* vrednosti 1, boljši je

klasifikator. Navadno pričakujemo, da se metoda obnese vsaj bolje od privzetega klasifikatorja, tj. klasifikatorja, ki napoveduje večinski razred (*CA* le-tega je enaka apriorni verjetnosti, da primerek pripada večinskemu razredu).

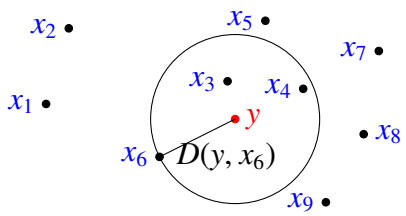
V delu pri regresiji (in tudi pri ocenjevanju uspešnosti napovedovanja kratkih časovnih vrst) uporabljamo vrednost *koren pričakovane vsote kvadratov napak* (angleško *Root Mean Square Error - RMSE*). *RMSE* je merilo napake in ne uspešnosti, zato so zaželene nižje vrednosti *RMSE*. Metoda, ki ima *RMSE* enak 0, torej pri napovedovanju ne stori napake.

Da zagotovimo veljavno oceno uspešnosti oz. napake, vse metode v tem delu ocenjujemo na enega od dveh ustaljenih načinov. V prvem primeru ločimo množico primerov na množico za učenje in množico za testiranje, pri čemer testna množica vsebuje približno $1/3$ vseh primerov (učna torej $2/3$). Metodo "učimo" na učni množici, uspešnost oz. napako metode pa ocenimo na primerih iz testne množice. V drugem primeru množico primerkov X razdelimo na 10 množic $X_1, \ldots, X_{10}$ (približno) enake velikosti. Test poženemo desetkrat, kjer je vsakič ena od množic X_i testna, ostale pa učne. Tako je vsak primerek enkrat v testni množici. Postopek delitve učne množice na m delov in testiranje na omenjen način se imenuje *m-kratna navzkrižna validacija*.

Modele, ki jih zgradijo metode, lahko poleg uporabe za napovedovanje vrednosti, izrabimo tudi za *pridobivanje znanja in vzorcev* iz podatkov. Slednje je pomembno, saj nam model tako ponuja novo, še nepoznano znanje in povezave, omogoča pa tudi boljšo interpretacijo napovedanih vrednosti (še posebej pomembno za strokovnjake z domene v interesu). Posebno dobro lahko znanje izluščimo npr. v primeru dreves in pravil.

2.1.1 Napovedovanje z metodo najbližjega soseda

Metoda najbližjega soseda (`kNN`) sodi med najenostavnejše metode podatkovnega rudarjenja. Metoda ne zgradi modela, torej ne poskuša izluščiti znanja iz podatkov, temveč se zanaša le na vhodne podatke v osnovni, načeloma nespremenjeni obliki. Takšnim metodam pravimo tudi *lene metode*. Postopek učenja izhaja iz ideje, da imajo primeri, ki ležijo blizu v prostoru opisnih atributov, enako/podobno vrednost



Slika 2.1: Napovedovanje z metodo najbližjega soseda. Ciljna vrednost primera y je odvisna od glasovanja najbližjih treh učnih primerov v (2-dimenzionalnem) prostoru - x_3, x_4 in x_6.

ciljne spremenljivke. Podobno sklepanje lahko opazimo tudi na primerih v naravi (tudi pri človeku). Predpostavka o razporeditvi primerov v prostoru opisnih atributov naredi metodo zelo občutljivo na domeno in še bolj na same vhodne podatke (dejansko so vse metode bolj ali manj občutljive na problem in podatke, ki jih priskrbimo). Še ena večja pomanjkljivost je odsotnost modela iz katerega bi lahko izluščili naučeno znanje, zato metode `kNN` ne moremo uporabiti za iskanje vzorcev in pridobivanje znanja iz podatkov.

Po drugi strani je enostavnost metode tudi njena prednost. Metoda je namreč zelo enostavna za implementacijo, še posebej v primeru, ko iskanje implementiramo z navadnim iskanjem. Uporabnost se kaže tudi v primerih, ko nove primere pridobivamo med samim učenjem, saj nam ni potrebno spreminjati modela, kot bi to morali početi pri npr. drevesih, temveč le dodamo nove primere v učno množico.

2.1.1.1 Glasovanje najbližjih sosedov

Metoda `kNN` torej napoveduje vrednosti glede na k najbližjih primerov v prostoru opisnih atributov. Predstavljajmo si, da najdeni primeri oddajo k glasov za napoved ciljne spremenljivke. Način, na katerega bomo upoštevali glasove, lahko vpliva na uspešnost napovedovanja. Naj bo $d_i = D(y, x_i)$ razdalja od iskanega primera y do primera x_i. Primeri $x_1, \ldots, x_k$ predstavljajo glasove za napoved vrednosti ciljne spremenljivke $m(y)$. Spomnimo se, da je $T(x_i)$ ciljna vrednost primera x_i. Poglejmo si tri različne načine glasovanja, ki jih implementiramo v naši metodi.

navadno glasovanje, pri katerem upoštevamo vseh k glasov z enako težo.

$$T(y) = \frac{1}{k} \cdot \sum_{i=1}^{k} T(x_i)$$

1/d oz. inverzno glasovanje, ki ga označimo z d^{-1} obteži vsak glas z obratno vrednostjo razdalje, pri čemer poskrbimo, da so vse uteži tudi normalizirane (izpostavljena prva vsota).

$$T(y) = \left(\sum_{i=1}^{k} d_i \right) \cdot \sum_{i=1}^{k} \frac{T(x_i)}{d_i}$$

1-d oz. obratno glasovanje označimo z d_{-1} in za razliko od d^{-1} ohrani linearno razmerje med utežmi ter jih zgolj preslika, da bližje razdalje odražajo glas z večjo težo. Tudi tu poskrbimo za ustrezno normalizacijo uteži.

$$T(y) = \left(\sum_{i=1}^{k} \frac{1}{1 - d_i} \right) \cdot \sum_{i=1}^{k} (1 - d_i) T(x_i)$$

V primeru regresije so vrednosti $T(x_i)$ realna števila in je posledično tudi $T(y)$ obteženo povprečje glasov. S tem dobimo kar samo napoved.

$$m(y) = T(y) \tag{2.1}$$

V primeru klasifikacije so stvari malce drugačne. Poglejmo si primer binarne klasifikacije. Tu so vrednosti $T(x_i)$ iz zaloge vrednosti $\{0, 1\}$. Po glasovanju dobimo vrednost $T(y) \in (0, 1)$, na katero lahko gledamo kot na verjetnost, da y pripada razredu. V kolikor je $T(y)$ večja od nekega praga p, y klasificiramo v razred. Prag p je navadno enak 0.5 (tako je tudi v sistemu Clus). Tako klasificiramo v razred z največjo verjetnostjo, kar je enostavno razširljivo tudi na primere večrazredne (enociljne) klasifikacije.

$$m(y) = \begin{cases} 1 & T(y) > p \\ 0 & \text{sicer} \end{cases} \tag{2.2}$$

Izračun prototipov v primeru strukturiranih podatkov predstavimo v razdelku 3.4, kjer je govora o sami implementaciji.

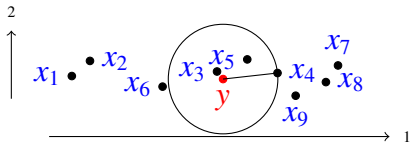
2.1.1.2 Napovedovanje v domenah visokih dimenzij

Eno od področij, kjer se metoda `kNN` v svojih različicah pogosto uporablja je tudi računalniški vid, kjer z metodo iščemo podobne izreze značilnih regij, ipd. Tu lahko dimenzionalnost prostora opisnih atributov hitro doseže rang 1000 in več dimenzij. Visok rang najdemo tudi v drugih domenah (genetika,..); z njim se soočamo tudi pri nekaj uporabljenih podatkovnih bazah v nadaljevanju tega dela. V [1] predstavijo problem iskanja najbližjega soseda v prostorih visokih dimenzij. Na podlagi teoretične izpeljave trdijo, da v splošnem primeru v prostorih visokih dimenzij, kontrast med najbližjim in najbolj oddaljenim primerom zbledi. Z drugimi besedami: pomenskost najbližjega primera postane vprašljiva, ker je mnogo primerov, ki so le malo dlje od njega. Rezultati se prenesejo tudi v primeru iskanja večih sosedov. Empirični rezultati, opravljeni v omenjenem članku kažejo, da problem nastane že v prostorih s 15 dimenzijami. Kljub temu dopuščajo možnost, da se metoda `kNN` obnese dobro na posameznih podatkovnih bazah.

2.1.1.3 Obteževanje atributov

Ideja obteževanja atributov sloni na opazki, da nekateri atributi nosijo več informacij od ostalih in bi jih zato morali upoštevati v večji meri. Medtem ko nekatere metode, kot npr. drevesa, takšne atribute deloma prepoznajo in upoštevajo že med učenjem, to ne drži za lene metode (med katere spada tudi `kNN`). Zato nabor normaliziranih uteži $w_1, w_2, \ldots, w_d$ za metodo `kNN` pridobimo ločeno in jih nato posredujemo metodi.

V primeru metode `kNN` obteževanje atributov vpliva le na izračun razdalje. For-



Slika 2.2: Sprememba iskalnega prostora (iz slike 3.1 na strani 25) pri uporabi obteževanja atributov (uteži $w_1 = 0.8$ in $w_2 = 0.2$.) Zaradi transformacije prostora je nabor najbližjih elementov drugačen (x_4,x_5 in x_4).

mula za izračun evklidske razdalje

$$D_2(x, y) = \sum_{i \in [1..r]} (x_i - y_i)^2$$

se v tem primeru spremeni v

$$D_2(x, y) = \sum_{i \in [1..r]} w_i(x_i - y_i)^2,$$

kjer i teče po vseh atributih. Podobno bi storili tudi v primeru manhattanske, chebysheve ali kakšne druge razdalje. Poglejmo si, kako sprememba razdalje transformira prostor iskanja k najbližjih sosedov. Večja utež w_i (pomembnejši atribut) bo raztegnila dimenzijo i. Manjše vrednosti uteži w_j (manj pomembni atributi) pa bodo dimenzijo j skrčile. Takšna transformacija prostora približa primere glede na pomembnejše atribute in v kolikor smo si izbrali primerne uteži, bo iskanje k najbližjih sosedov v novem prostoru, vrnilo "boljše" sosede.

Slika 2.2 prikazuje spremembo dvo-dimenzionalnega iskalnega prostora po uporabi obtežitve atributov.

V grobem lahko tehnike obteževanja atributov razdelimo v tri kategorije (označimo jih kar z njihovimi angleškimi termini):

filter Z določeno metodo ocenimo kvaliteto posameznih atributov, kvaliteto pretvorimo v uteži, te nato podamo metodi. Primeri so izračun informacijskega prispevka atributa, korelacije med atributom in ciljno spremenljivko,...

wrapper V tem primeru kvaliteto celotnega nabora uteži ocenjujemo kar z uspešnostjo

same metode. Iskanje nabora uteži navadno usmerjamo z eno od optimizacijskih metod (simulirano ohlajanje, genetski algoritmi, iskanje z omejeno širino, ...). Slabost tega načina je, da zahteva mnogo poganjanj metode in je zato zelo počasen. Prednost je, da ocenjujemo celoten nabor uteži hkrati in optimizacija dejanske končne uspešnosti metode.

embedded Obteževanje atributov je vključeno v samo učenje in je del metode. Predstavnik slednjega so npr. odločitvena drevesa (dejansko drevesa ne obtežujejo atributov, temveč jih samo izbirajo - to pa si lahko predstavljamo kot podproblem obteževanja).

2.1.2 Drevesa za napovedno razvrščanje

Odločitvena drevesa [2] so eden najbolj enostavnih in učinkovitih načinov gradnje modela za napovedovanje spremenljivk. Vozlišča v odločitvenem drevesu predstavljajo teste, na podlagi katerih delimo primere v enega od poddreves. Listi predstavljajo napovedi za določeno pot (zaporedje testov).

Odločitvena drevesa so dobra tudi s stališča predstavitve znanja, saj lahko iz testov v notranjih vozliščih razberemo nekatere odvisnosti med atributi. V splošnem je gradnja optimalnega odločitvenega drevesa NP-poln problem, zato si pomagamo s hevristiko. Hevristika določa izbiro odločitvenih pogojev, ki pripadajo notranjim vozliščem. Znani primeri hevristik so: informacijski prispevek, gini indeks, gini razmerje, reliefF, itd., ki najdejo "najboljši" atribut na podlagi katerega razdelimo primere med dve poddrevesi.

Poglejmo si še eno od nalog podatkovnega rudarjenja, in sicer **razvrščanje v skupine**. Naloga razvrščanja v skupine je združevanje primerov v skupine, tako da so skupine karseda homogene (navadno definiramo homogenost samo z opisnimi atributi, v tem primeru je govora o nenadzorovanem učenju). Najbolj znani metodi razvrščanja v skupine sta k-means in hierarhično razvrščanje. Vsaki od skupin navadno pripišemo prototip, primerek, ki je najbolj reprezentativen za skupino. Pri tem ni nujno, da je prototip eden od primerov, ki smo jih uvrstili v skupino - lahko ustvarimo novega in ga uporabimo le za namene predstavitve skupine.

V tem razdelku predstavimo drevesa za napovedno razvrščanje, angleško *Predictive Clustering Trees* (PCT) [3], ki omogočajo tako napovedovanje kot tudi razvrščanje primerov v skupine. Drevesa PCT so del sistema *Clus* [4], ki implementira ogrodje in metode za nenadzorovano razvrščanje v skupine in napovedno modeliranje ter omogoča enostavno razširitev na napovedovanje in razvrščanje strukturiranih podatkov, kot so večciljno napovedovanje, hierarhična večznačna klasifikacija in časovne vrste. Clus skupaj razvijata *Declarative Languages and Artificial Intelligence group of the Katholieke Universiteit Leuven, Belgium* in *Oddelek za tehnologije znanja, Inštitut Jožef Stefan, Ljubljana*. Sistem je izdan pod GPL licenco in je prosto dosegljiv na domači spletni strani [5].

Poglejmo si, kako drevesa zgradimo ter kako jih uporabimo za napovedovanje.

Algoritem 1 PCT

Require: I
1: $(t^*, \mathcal{P}^*)$ = NajboljšaDelitev(I)
2: **if** $t^* \neq none$ **then**
3: **for all** $I_k \in \mathcal{P}^*$ **do**
4: $tree_k$ = PCT(I_k)
5: **vrni** vozlišče $(t^*, \bigcup_k tree_k)$
6: **else**
7: **vrni** list(Prototip(I))

PCT induciramo podobno kot običajna odločitvena drevesa z indukcijo od zgoraj navzdol. Algoritem gradnje je opisan v metodi PCT (Algoritma 1 in 2). Gradnja poteka rekurzivno. Izhod algoritma je drevo, katerega notranja vozlišča vsebujejo odločitvene teste, listi pa učne primere. Listi predstavljajo skupine v primeru razvrščanja v skupine in dajejo napoved v primeru uporabe modela za napovedovanje spremenljivk.

Test v vsakem od notranjih vozlišč razdeli množico primerov na podmnožice. To počnemo, dokler je delitev smiselna (npr. množice primerov, ki si delijo isto vrednost ciljne spremenljivke, navadno ni smiselno deliti). Smiselnost delitve kot tudi izbira najboljšega testa delitve se določa z definicijo *variance* skupine/lista.

Algoritem 2 NajboljšaDelitev

Require: I
1: $(t^*, h^*, \mathcal{P}^*) = (none, 0, \emptyset)$
2: **for all** možne teste t **do**
3: $\mathcal{P}$ = delitev množice I glede na t
4: $h = Var(I) - \sum_{I_k \in \mathcal{P}} \frac{I_k}{I} Var(I_k)$
5: **if** $(h > h^*) \wedge$ Dopusten$(t, \mathcal{P})$ **then**
6: $(t^*, h^*, \mathcal{P}^*) = (t, h, \mathcal{P})$
7: **vrni** $(t^*, \mathcal{P}^*)$

Termin *varianca* izvira s področja razvrščanja v skupine in označuje kvaliteto[1] skupine.

Za vsakega od listov/skupin izračunamo prototip. Prototip je tipičen predstavnik skupine, ki jo predstavlja. V sistemu *Clus* je koncept prototipa razširjen tudi na glasovanje. Neznani primerek, ki ga bo drevo uvrstilo v list l_i, bo namreč dobil isto ciljno vrednost, kot jo nosi prototip lista l_i.

Drevesa za napovedno razvrščanje se od običajnih odločitvenih dreves ločijo po odprti definiciji prototipa in variance. S tem lahko z enim samim modelom opravimo ne samo nalogi razvrščanja v skupine in napovedovanja, temveč tudi napovedovanje strukturiranih podatkov, kjer je definicija prototipa in variance težavnejša.

Opazimo, da algoritma 1 in 2 kličeta tri še nedefinirane metode: *Prototip*, *Var* in *Dopusten*. Prvi dve smo že predstavili in jih definiramo pozneje pri posamezni uporabi dreves PCT. Metoda *Dopusten* preveri dopustnost testa in je odvisna od problema. Definira množico dopustnih testov in delitev. Nedopustna delitev je recimo delitev, kjer je ena od podmnožic prazna.

2.1.3 Pravila za napovedno razvščanje

Ena od implementiranih metod v Clus ogrodju so tudi pravila za napovedno razvrščanje (angleško *Predictive Clustering Rules* - PCR). Tako kot PCT, tudi PCR zmorejo razvrščanje v skupine in napovedovanje spremenljivk različnega tipa. Na tem mestu

[1] Kvaliteto skupine tipično definiramo na podlagi vsote razdalj med primeri v skupini. V primeru dreves PCT pa lahko v izračun kvalitete vključimo tudi ali zgolj ciljno spremenljivko in tako zagotovimo tudi ali zgolj napovedno uspešnost.

se seznanimo le z osnovno idejo učenja pravil. Podrobneje jih predstavi Ženko v [6] in [7].

Algoritem PCR je predelava algoritma CN2 [8]. Slednji uporablja verzijo algoritma za pokrivanje, ki se lahko nauči tako urejena kot neurejena pravila. Uporabimo pa ga lahko tudi za napovedovanje večrazrednih spremenljivk.

Na vsakem koraku glavne zanke iščemo najboljše pravilo in ga dodamo v seznam pravil. Pokrite primere odstranimo iz X in zanko ponovimo dokler je X neprazna ali nismo izpolnili kakšnega drugega zaustavitvenega pogoja. Iskanje najboljšega pravila usmerja hevristika iskanja, ki zajema število pokritih primerov, natančnost pokrivanja in druge parametre v primeru napovedovanja kompleksnejših spremenljivk.

2.2 Napovedovanje strukturiranih podatkov

Čeprav lahko za napovedovanje strukturiranih podatkov uporabimo konvencionalne tehnike napovedovanja (običajno klasifikacijo in regresijo), takšen pristop običajno ne izkorišča vsega razpoložljivega znanja, opravlja nepotrebno dodatno delo ali pa ne zadovolji povsem natančno vsem zahtevam. Od tehnik, ki izkoriščajo strukturo podatkov zato pričakujemo, da bodo prilagojene ustreznim strukturam in jo (v kolikor je možno) tudi izkoriščale. V nadaljevanju predstavimo 3 probleme in nekatere tehnike napovedovanja strukturiranih podatkov s katerim se srečamo v tem delu - večciljno klasifikacijo in regresijo, hierarhično večznačno klasifikacijo in napovedovanje kratkih časovnih vrst.

2.2.1 Večciljno napovedovanje

Večciljno napovedovanje se ukvarja s hkratnim napovedovanjem večih ciljnih spremenljivk. Metoda, ki implementira večciljno napovedovanje, vrne model m

$$m : D \mapsto (T_1, T_2, \ldots, T_r),$$

torej en sam model za napovedovanje vseh r ciljnih spremenljivk. Pri uporabi klasičnih metod podatkovnega rudarjenja bi za vsako od r ciljnih spremenljivk

zgradili svoj model. Imenujmo slednji pristop *posamično napovedovanje*. Večciljno napovedovanje ima torej vsaj eno prednost, in sicer za faktor r manjši čas gradnje modela in same napovedi. Če ne bo navedeno drugače, povsod v delu z r označujemo število ciljnih spremenljivk oz. razredov.

Navedimo glavne razlike pri gradnji in napovedovanju spremenljivk pri uporabi večciljnega in posamičnega napovedovanja:

večciljno napovedovanje Za napovedovanje zgradimo samo en model (drevo, nabor pravil). Varianco skupine ocenjujemo na podlagi vseh r razredov, prav tako izračunamo en sam prototip za napovedovanje vseh razredov.

posamično napovedovanje Za vsakega od razredov zgradimo svoj model. Rezultat je r modelov, kjer je vsak specializiran za napovedovanje enega od razredov.

Ker so zgrajeni modeli pri posamičnem napovedovanju specializirani za vsakega od ciljnih razredov, bi pričakovali, da se bo posamično napovedovanje obneslo bolje, a rezultati primerjav za PCT in PCR kažejo, da so uspešnosti pogosto primerljive [7, 9]. Poleg tega pa nam večciljno napovedovanje ponuja enoten model pridobljenega znanja za vse ciljne spremenljivke in ima zato večjo izrazno moč.

Omenili smo že, da o modelu pri metodi `kNN` pravzaprav ne moremo govoriti, saj se metoda ne uči, temveč je lena. Posledica je, da se večciljno napovedovanje ne razlikuje od posamičnega, saj se vedno odločamo na podlagi (istih) najbližjih primerov. Pristopa se ločita le na konceptualni ravni - pri posamičnem napovedovanju metodo poženemo r krat in združimo vseh r napovedi skupaj, pri večciljnem napovedovanju pa enkrat samkrat in napovedi združimo že pri pripravi prototipa.

Enaka uspešnost `kNN` metode ne glede na pristop k večciljnem napovedovanju je seveda vse prej kot slaba, v primeru večciljnega napovedovanja lahko tako prihranimo na času za faktor r z zagotovilom, da bo uspešnost natanko enaka, kot če se napovedovanja lotimo na konvencionalen način.

Kot pri enociljnem napovedovanju, tudi tu ločimo med napovedovanje nominalnih in numeričnih atributov. V prvem je govora o večciljni klasifikaciji, v drugem o večciljni regresiji.

2.2.2 Napovedovanje hierarhičnih večznačnih spremenljivk

Hierarhična večznačna klasifikacija se ubada z napovedovanjem skupine (vektorja) binarnih razredov $C = (c_1, c_2, ..., c_r)$ nad katerimi je definirana stroga delna urejenost $<_h$. Urejenost $<_h$ predstavlja hierarhično urejenost razredov. Velja $c_1 <_h c_2$ natanko tedaj, ko je c_2 podrejen razredu c_1.

Hierarhična večznačna predikcija je uporabna na večih področjih, med drugim pri prepoznavanju in kategorizaciji objektov, klasifikaciji tekstov in funkcionalni genomiki. V diplomskem delu se osredotočamo na podatke iz slednjega področja. Tu poskušamo napovedati funkcije genov določenega genoma. Vsak gen lahko opravlja eno ali več funkcij. Te funkcije so urejene v hierarhično shemo (specifične funkcije spadajo v okvir splošnih funkcij). V tem delu uporabljamo dve različni klasifikacijski shemi za funkcije genov:

- Prva je **FunCat**, ki je podrobneje opisana v [10] in prosto dostopna na spletni strani [11]. V shemi FunCat so funkcije genov urejene v drevesno strukturo. Drevesna struktura nam določa tudi urejenost $<_h$. Za vozlišče c_o in njegovega sina c_s velja $c_o <_h c_s$ in obratno. Na sliki 2.3 lahko vidimo izsek iz sheme FunCat.

- Druga uporabljena klasifikacijska shema je **GO** (Gene Ontology), podrobneje predstavljena v [12] in prosto dostopna na spletni strani [13]. V nasprotju s shemo FunCat so tu funkcije predstavljene kot usmerjen acikličen graf. Stroga delna urejenost je tu izomorfna usmerjenosti grafa: če za vozlišči c_x, c_y velja $c_x \rightarrow c_y$, potem velja tudi $c_y <_h c_x$ in obratno. Tako definirana urejenost pomeni, da ima lahko vozlišče več "nadrejenih" vozlišč, kar je v domeni enako temu, da je neka specifična funkcija gena del večih (bolj splošnih) funkcij.

Tabela 2.1 povzema glavne lastnosti obeh uporabljenih klasifikacijskih shem.

Na prvi pogled je naloga hierarhične večznačne klasifikacije enaka večciljni klasifikaciji - napovedovanju binarnega vektorja - ki smo jo spoznali v prejšnjem razdelku. Vendar slednja ni direktno uporabna v tem primeru. Hierarhično urejeni

```
+ TRANSCRIPTION
- PROTEIN SYNTHESIS
    - ribosome biogenesis
        - ribosomal proteins
    - translation
        - translation initiation
        - translation elongation
        - translation termination
    - translational control
    - aminoacyl-tRNA-synthetases
    - selenocystein biosynthesis and incorporation
+ PROTEIN FATE (folding, modification, destination)
```

Slika 2.3: Izsek iz klasifikacijske sheme **FunCat**. Hierarhija funkcij genov je urejena v drevo.

Tabela 2.1: Lastnosti uporabljenih klasifikacijskih shem. Zaradi primerjav opravljenih v poglavju 6, uporabimo iste verzije shem kot v [3].

	FunCat	GO
Verzija sheme	2.1 (9.1.2007)	1.2 (11.4.2007)
Opis kvasovke	16.3.2007	7.4.2007
Vseh razredov v shemi	1362	22960

razredi postavljajo napovedi dodaten pogoj dopustnosti: da bi lahko nek primerek klasificirali v razred c_2, ga moramo tudi v razred c_1, ki je razredu nadrejen ($c_1 <_h c_2$). To omejitev imenujmo **hierarhična omejitev** in vpliva na postopek učenja in/ali priprave prototipa.

V poglavju 6 primerjamo metodo `kNN` z dvema metodama, pridobljenima s pomočjo dreves PCT - `SC` in `HMC`. Tu povzamemo delovanje teh dveh metod, ki sta natančno opisani v [3].

Metoda `SC` za vsakega od ciljnih razredov C_i zgradi svoje drevo (model) m_i, ki napoveduje vrednosti za tisti razred. Model m_i za primerek y poda verjetnost $c_i = m_i(y)$, da primerek y pripada razredu C_i. Za vsakega od razredov določimo prag p_i, primerek y pa klasificiramo v C_i, če je $c_i \geq p_i$. Metoda `SC` ima štiri pomanjkljivosti: **1)** ne zadosti hierarhični omejitvi, saj ni zagotovljeno, da bo primerek, ki je klasificiran v nek razred, klasificiran tudi v nadrejenega. To lahko (do neke mere)

rešimo v fazi napovedovanja ali z naknadnim popravljanjem verjetnosti pri pripravi prototipa **2)** je neučinkovita, ker moramo zgraditi r dreves **3)** frekvence razredov (tj. odstotek učnih primerov, ki pripadajo razredu) so precej neenakomerne - višje v hierarhiji so frekvence visoke, nižje pa lahko precej nizke. Mnogi klasifikatorji imajo probleme z napovedovanjem močno neenakomerno razporejenih razredov [14]. **4)** iz modela ni moč dobro izluščiti znanja, saj so višje v modelih atributi, ki so pomembni za posamezen razred in ne za celoten nabor funkcij .

V nasprotju s `SC` poskuša `HMC` napovedati vse razrede hkrati in pri tem zadovoljiti hierarhični omejitvi. Uporabimo PCT drevo in pametneje definirajmo funkciji variance in prototipa. Razrede predstavimo kot binarne vektorje dolžine r. Varianca je definirana kot povprečje kvadratov razdalj vseh primerov od srednje vrednosti skupine $\bar{x}$. Vrednost $\bar{x}$ je povprečje vseh vektorjev v skupini.

$$Varianca(X') = \frac{\sum_{x_k \in X'} D(x_k, \bar{x})^2}{|X'|} \tag{2.3}$$

Razdalja $D(\cdot\,, \cdot)$ (2.4) upošteva vse elemente vektorja, s tem da element i obteži glede na globino razreda c_i v hierarhiji. Opomnimo, da v tem primeru izračun prototipa uporablja obteženo evklidsko razdaljo, kjer uteži padajo z globino v hierarhiji. Po želji bi lahko uporabili tudi drugo funkcijo razdalje (D_1, D_∞).

$$D(x_1, x_2) = \sqrt{\sum_{i \in [1..r]} w(c_i)(x_{1,i} - x_{2,i})^2} \tag{2.4}$$

$$w(c) = w_0^{depth(c)} \quad w_0 \in (0, 1) \tag{2.5}$$

2.2.3 Napovedovanje kratkih časovnih vrst

Časovna vrsta je urejeno zaporedje meritev neke zvezne vrednosti, ki se spreminja s časom. Intervali med meritvami so navadno uniformni, čeprav to ni nujno. Tipična časovna vrsta je npr. gibanje finančnih indeksov; v tem delu pa se ukvarjamo s časovnimi vrstami, ki zajemajo spremembe odziva genov v genomu kvasovke glede

na različne spremembe v okolju. Različnih nalog, ki vključujejo časovne vrste, je več, npr.: **1)** razvrščanje časovnih vrst v skupine **2)** napovedovanje prihodnjih vrednosti v časovni vrsti na podlagi preteklih **3)** iskanje odvisnosti med dvema časovnima vrstama.

Naša naloga je napovedovati kratke časovne vrste iz opisnih binarnih atributov. V [15] se avtorji ukvarjajo z razvrščanje profilov izražanja gena v skupine s pomočjo dreves PCT. Izražanje gena je predstavljeno kot kratka časovna vrsta.

Ker drevesa PCT združujejo nalogi razvrščanja v skupine in napovedovanja, članek [15] predstavi tudi uspešnost dreves PCT za napovedovanje kratkih časovnih vrst na treh podatkovih bazah. S tem dobimo primerjalni model za našo metodo.

Da lahko napovedujemo časovne vrste z drevesi PCT, moramo ustrezno definirati varianco in prototip skupine. Uporabimo lahko standardno definicijo variance (enako kot pri običajni klasifikaciji), če prej definiramo ustrezno razdaljo med dvema elementoma prostora - časovnima vrstama. Poglejmo si nekatere primerne razdalje. Časovno vrsto lahko predstavimo kot vektor r realnih vrednosti, kjer vrednosti v vektorju predstavljajo vrednosti spremenljivke v eni od r časovnih točk. V takšni predstavitvi lahko uporabimo že omenjene standardne razdalje D_2, D_1 in D_∞. Uporabimo lahko tudi razdaljo D_{cr}, ki uporablja korelacijo $cr(a, b)$ med dvema časovnima vrstama a in b:

$$cr(a,b) = \frac{E[(a - E[a])(b - E[b])]}{E[(a - E[a])^2]E[(b - E[b])^2]} \quad , \quad D_{cr}(a,b) = \sqrt{2(1 - cr(a,b))}.$$

Pogosto se uporablja tudi Dynamic Time Warping (DTW) razdalja, ki lahko zajame tudi nelinearne raztege vzdolž časovne osi in je še posebej primerna, ko časovni vrsti nista popolnoma sinhronizirani (časovni vrsti si delita obliko, a z zamikom). V slednjem primeru bi ostale razdalje izmerile relativno visoko razdaljo med sicer podobnima časovnima vrstama. Omenjene razdalje imajo nekaj slabosti: korelacijo težko ocenimo na krajših časovnih vrstah, zato razdalja D_{cr} ni primerna za uporabo v našem primeru. Ostale razdalje zajamejo gibanje časovne vrste in njeno višino. Dve časovni vrsti, kjer je ena le y-premik ali y-razteg druge, bosta zato daleč narazen. Ker je naš cilj obravnava odzivnosti gena, nas same natančne vrednosti ne

Tabela 2.2: Definicija funkcije Diff(q_1, q_2) je odvisna od kvalitativnih sprememb časovnih vrst q_1, q_2 med dvema točkama na časovni osi. Primer: če med dvema točkama časovna vrsta q_1 raste, q_2 pa pada, je vrednost Diff(q_1, q_2) enaka 1.

Diff($\mathbf{q_1}, \mathbf{q_2}$)	rast	brez sprememb	upad
rast	0	0.5	1
brez sprememb	0.5	0	0.5
upad	1	0.5	0

zanimajo, zanima nas le groba oblika časovne vrste. Zato raje uporabimo razdaljo *Qualitative Difference Measure* (QDM), ki jo predlaga Todorovski [16].

Isto razdaljo uporabijo tudi v [15] za gradnjo dreves PCT, kar nam omogoča primerjavo obeh metod. QDM ocenjuje časovno vrsto s kvalitativnim pristopom in je predvsem primerna za krajše časovne vrste, s katerimi imamo opravka tudi v tem delu. Razdalja zazna grobo obliko časovne vrste in se ne ubada z natančnimi vrednostmi. Je edina od omenjenih razdalj, kjer je razdalja med dvema vrstama z isto obliko a nelinearnim raztegom y vrednosti, ničelna. V nadaljevanju opišemo izračun QDM razdalje.

Vzemimo dve časovni vrsti a in b ter si izberimo par časovnih točk i in j. Opazujemo kvalitativno spremembo q v vrednosti časovne vrste med točkama i in j ($i < j$). Za časovno vrsto a (enako za b) ločimo tri možnosti: rast ($a_i < a_j$), brez spremembe ($a_i \sim a_j$) in upad ($a_i > a_j$). Razdaljo D_q izračunamo tako, da povprečimo kvalitativne spremembe glede na vse pare časovnih točk:

$$D_q(a, b) = \frac{4}{r \cdot (r-1)} \cdot \sum_{i<j} \text{Diff}(q(a_i, a_j), q(b_i, b_j)), \tag{2.6}$$

kjer je r število vseh meritev v časovni vrsti, Diff() pa funkcija, ki ovrednoti kvalitativni spremembi $q(\cdot, \cdot)$ dveh vrst. Vrednosti funkcije Diff() so podane v tabeli 2.2.

Poglavje 3

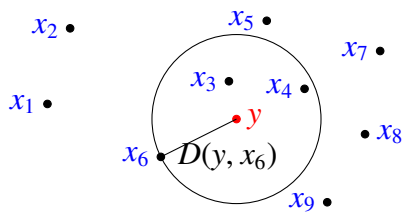
Implementacija

V tem poglavju se posvetimo implementaciji metode najbližjega soseda za napovedovanje strukturiranih podatkov. Za iskanje najbližjih sosedov smo implementirali 3 iskalne metode: navadno iskanje (`BF`) in kd ter vp drevo, ki jih združimo v našo metodo `kNN`. Podrobnosti metod predstavimo v nadaljevanju poglavja, v poglavju 4 pa ugotavljamo, katera od metod se obnese hitreje v določenih okoliščinah. Kot je bilo uvodoma omenjeno, smo metodo implementirali znotraj sistema Clus [4]. V Clusu je sicer izračun prototipov za uporabljene strukturirane podatke že implementiran, a ga zavoljo natančnosti in popolnosti vseeno predstavimo tudi na tem mestu. Na koncu poglavja predstavimo tudi tehnične podrobnosti implementacije, možne nastavitve in uporabo metode.

3.1 Iskanje najbližjega soseda

Poglejmo si problem iskanja najbližjega soseda na enostavnem primeru prikazanem na sliki 3.1. V dvodimenzionalnem prostoru imamo 6 točk $x_1, x_2, \ldots x_6$ in izpostavljeno točko y, katere najbližje sosede iščemo. Odločimo se za iskanje $k = 3$ najbližjih točk. Te točke so x_3, x_4 in x_6.

Najprej si oglejmo navadno iskanje (algoritem 3). Metoda je preprosta in mora zato preiskati vse točke v X (teh je n). Njena časovna zahtevnost je $O(n)$.



Slika 3.1: Enostaven primer iskanja najbližjega soseda v dvo-dimenzionalnem prostoru. V ilustriranem primeru od iskalne metode zahtevamo, da najde 3 y-onu najbližje elemente (to so x_3, x_4 in x_6).

Algoritem 3 bfSearch(y, k, X) Navadno iskanje najbližjih k točk.

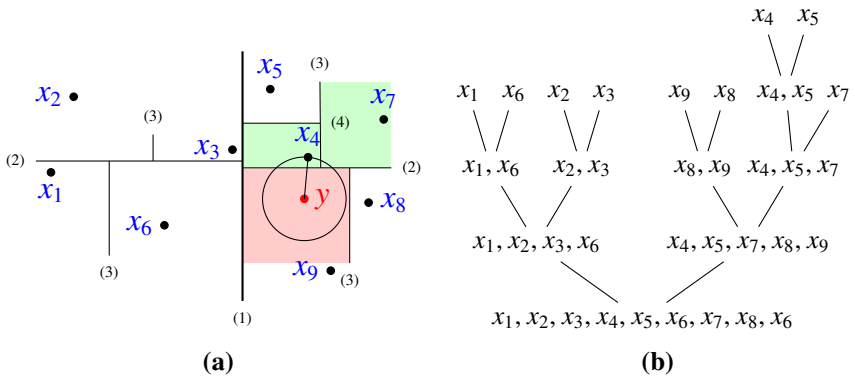
Require: $y, X, k, D(\cdot, \cdot)$
Ensure: Y množica najbližjih k primerov

```
1: d_limit = ∞
2: for x ∈ X do
3:     d = D(x, y)
4:     if d < d_limit then
5:         Y = Y ⋃ x
6:         if |Y| > k then
7:             Odstrani najslabšo iz Y
8:             d_limit = max_{y'∈Y} D(y', y)
```

Seveda lahko z uporabo boljših podatkovnih struktur dosežemo asimptotično hitrejše iskanje kot $O(n)$. Eno od možnosti nam predstavljajo binarna iskalna drevesa, ki razdelijo iskalni prostor in ga predstavijo kot binarno drevo. Iskanje v takšnem drevesu je časovne zahtevnosti $O(\log n)$. Vsako vozlišče v_i v drevesu T razdeli prostor na dva dela. Različni načini delitve prostora nam določajo različne tehnike gradnje in iskanja. V nadaljevanju na kratko predstavimo dve tehniki, ki ju tudi implementiramo.

3.1.1 Kd drevesa

Kd drevo, krajše za k-dimenzionalno drevo, je binarno iskalno drevo, ki za delitev prostora na dva podprostora uporabi na koraku i dimenzijo ($i \pmod d$). V primeru dvodimenzionalnega prostora torej delimo glede na dimenzijo x, y izmenoma. Takšno



Slika 3.2: Zgrajeno kd drevo (b) in primer delitve iskalnega prostora (a). Pobarvani del prostora je preiskani del v primeru iskanja y-onu najbližjih sosedov.

situacijo prikazuje slika 3.2. V vsakem vozlišču izračunamo srednjo vrednost atributa, ki pripada dimenziji, za vse primere v podprostoru, ki ga vozlišče zastopa. Na podlagi srednje vrednosti razdelimo primere med dve novi podrejeni vozlišči.

V primeru nominalnih atributov, ko delitev v določeni dimenziji ni možna, storimo dvoje:

1. Pogledamo, če je delitev možna v kakšni od drugih dimenzij, in uporabimo prvo, ki jo najdemo.

2. Če delitev ni možna v nobeni dimenziji, vozlišča ne delimo naprej. Listu pripadajo vsi primeri.

Omenjena situacija je možna tudi v izrojenem primeru pri uporabi realnih atributov.

Iskanje najbližjega soseda s pomočjo kd drevesa je sestavljeno iz dveh korakov:

1. Poišči list v drevesu, ki ustreza podprostoru, katerega del je y. Določi prototip lista za najbližji primerek (v primeru na sliki 3.2 je to x_9).

2. Preišči preostalo drevo, če ni kateri od preostalih primerov bližje y-onu kot trenutno najbližji. Prispevek k zmanjšanju števila iskanj nam prinese struktura podatkov - v obzir vzamemo dejstvo, da podprostor, ki je od y oddaljen dlje kot trenutno najbližji najden učni primer, ne more vsebovati bližjega primera.

S tem se velikost preiskanega prostora drastično zmanjša. Preiskani prostor v našem primeru je na sliki obarvan.

3.1.1.1 Ravnanje z nominalnimi atributi

Nominalnim vrednostim pripišemo zaporedna naravna števila. Deljenje elementov v dimenziji, ki pripada nominalnemu atributu poteka, kot da bi bil atribut numeričen, vrednosti pa ustrezna naravna števila.

Ker so kd drevesa precej znana in dobro opisana v literaturi, ostale podrobnosti implementacije izpustimo.

3.1.2 Vp drevesa

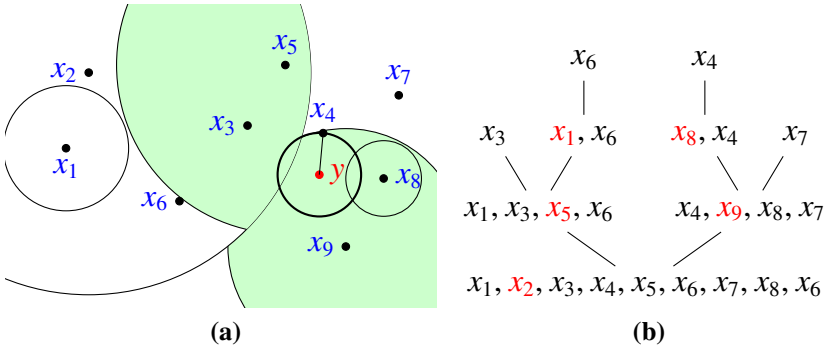
Tretja implementirana metoda je prav tako binarno iskalno drevo, ki pa podprostor deli precej drugače od omenjenega kd drevesa. Izhajamo iz opazke, da ima vsaka točka v prostoru svoj pogled na celoten prostor. To informacijo poskušamo uporabiti za delitev prostora na način, ki bo omogočal hitro iskanje tudi v primeru, ko prostor ni evklidski in v primeru prostorov z višjimi dimenzijami.

Na vsakem koraku gradnje vp drevesa si izberemo točko pogleda[1] v iz množice vseh učnih primerov X. Nato poiščemo vrednost $d_v = \sum_i D_{(x^i,v)}/(|X'| - 1)$, kjer X' predstavlja množico primerov, ki jih še nismo uvrstili v drevo. Vrednost d_v je srednja vrednost oddaljenosti preostalih primerov od točke v in nam služi za delitev primerov v dve množici oz. podprostora. Točke, ki so od v oddaljene več kot d_v, uvrstimo v desno podrevo, ostale v levo. Slika 3.3 nam prikazuje zgrajeno vp drevo in delitvene sfere na dvodimenzionalnem primeru.

3.1.2.1 Izbira točke pogleda

Na vsakem koraku želimo izbrati takšno točko, ki nam bo zagotovila karseda hitro iskanje. Lastnost takšne točke je, da je varianca razdalj od točke pogleda do vseh ostalih točk velika. Možni kandidati za točko pogleda so vse točke v prostoru,

[1] VP je kratica za *vantage point*, točko pogleda, razgleda.



Slika 3.3: Zgrajeno vp drevo (b) in primer delitve iskalnega prostora (a). Pobarvani del prostora je preiskani del v primeru iskanja y-onu najbližjih sosedov. Pobarvani rdeči primeri predstavljajo izbrano točko pogleda.

vendar je iskanje optimalne točke v celotnem prostoru prezahtevno. Tudi zmanjšanje možnih kandidatov le na točke, ki jih predstavljajo učni primeri, je še vedno prevelik zalogaj, ko je primerov veliko. Odločimo se za kompromis in za najboljšo točko pogleda preiščemo le naključno podmnožico S vseh učnih primerov X. Za dodatno pohitritev vsak primer $x_{m\in[1..s]}$ testiramo le na naključni podmnožici vseh preostalih učnih primerov S_m (ki je drugačna za vsak primer). Če je X množica vseh točk, ki še niso bile preiskane, so velikosti množic S in S_m v naši implementaciji naslednje:

$$\begin{aligned} |S| &= \max(\min(|X|, 3), \sigma_1 \cdot |X|) \\ |S_m| &= \max(\min(|X|, 2), \sigma_2 \cdot |X|) \\ \sigma_1 &= 0.1 \quad \sigma_2 = 0.08 \end{aligned} \tag{3.1}$$

Vrednosti σ_1 in σ_2 sta bili izbrani ekperimentalno in sta se izkazali za dober kompromis med hitrostjo gradnje in hitrostjo iskanja (kvaliteto točke pogleda). V kolikor bi želeli hitrejše iskanje, bi σ_1 in σ_2 ustrezno povečali in seveda plačali s počasnejšo gradnjo drevesa. Z uvedbo naključnega izbora vzorca za iskanje optimalne točke pogleda postane algoritem naključen in ne več deterministične kot kd drevo. Dobimo tudi dva nova parametra, s katerimi lahko vplivamo na čas gradnje in iskanja.

Algoritem 4 vpSearch(v,y) Iskanje najbližjega soseda v vp drevesu.

Require: novi primer y, razdalja $D(\cdot,\cdot)$, koren drevesa n
Ensure: Y najbližji učni primeri

```
 1: d = D(y, v.p)
 2: best.push(d)
 3: m = (v.bounds_left,high + v.bounds_right,low)/2
 4: global τ = best.worst()
 5: if d < m then
 6:     if x <= v.bounds_left,high + τ then
 7:         vpSearch(v.left, y)
 8:     if x => v.bounds_right,low + τ then
 9:         vpSearch(v.right, y)
10: else
11:     if x => v.bounds_right,low + τ then
12:         vpSearch(v.right, y)
13:     if x <= v.bounds_left,high + τ then
14:         vpSearch(v.left, y)
```

3.1.2.2 Iskanje v vp drevesu

Oglejmo si delovanje algoritma 4. V pomoč si pripravimo podatkovno strukturo `best`, ki hrani seznam najbližjih k najdenih učnih primerov. Struktura temelji na enostavnem vstavljanju. Z metodo `push()` dodamo nov primer v seznam, če je le-ta dovolj blizu. Metoda `worst` vrne najslabšo razdaljo med k najboljšimi. Isto strukturo uporabljamo tudi pri ostalih dveh metodah (kd drevesu in navadnem iskanju).

Algoritem pokličemo nad korenom vp drevesa. Izračunamo razdaljo $d = D(y, v.x)$, razdaljo med neznanim primerom y in točko pogleda $v.p$ v vozlišču v. Na podlagi vrednosti m se odločimo, katero od poddreves preiščemo najprej. Vrednost m predstavlja srednjo vrednost razdalj od točke pogleda do najbolj desnega primera v levem poddrevesu in najbolj levega primera v desnem poddrevesu.

Vsako poddrevo obiščemo le, če je možno, da bomo v njem našli dovolj dober primer (vrstice 6,8,11,13). To informacijo hranimo v vrednosti τ, ki je globalna in vedno vsebuje razdaljo do najbolj oddaljenega primera med k najboljšimi med preiskanimi. Globalno vzdrževanje vrednosti τ in pravilni vrstni red preiskovanja

poddreves nam zagotavljata, da preiščemo karseda majhen del prostora.

V algoritmu na večih mestih uporabimo podatkovno strukturo `bounds`, ki jo shranimo že ob gradnji drevesa. Vsebuje največjo in najmanjšo razdaljo do točke pregleda $v.p$ za obe polovici poddrevesa. Dodatna informacija nam pove, koliko boljše učne primere lahko najdemo s preiskovanjem določenega dela prostora in pomembno vpliva na velikost preiskanega prostora.

Naj bo $D(a, b)$ dejanska (običajno evklidska) razdalja med dvema primeroma v prostoru. Definirajmo razdaljo med primeroma a in b kot jih vidi primer p: $D_p(a, b) = |D(a, p) - D(p, b)|$. Zaradi trikotniške neenakosti velja:

$$D(a, b) \geq |D(a, p) - D(b, p)| = D_p(a, b). \tag{3.2}$$

Med gradnjo drevesa operiramo samo z razdaljo D_p. Zato so tudi razdalje mejnih primerov do točke pogleda razdalje D_p. Te vrednosti nato podeduje tudi τ. Iz enačbe (3.2) neposredno sledi

$$D_p(a, b) \geq \tau \Rightarrow D(a, b) \geq \tau. \tag{3.3}$$

Če torej poznamo razdaljo D_p do nekega primera in je ta večja od τ, potem bo večja od τ tudi razdalja v dejanski metriki, torej primer ni kandidat za najbližjega soseda. Ta lastnost nam prihrani na času iskanja, saj omogoča, da učinkovito zmanjšamo del preiskanega prostora.

3.1.3 Možne izboljšave iskalnih metod

Ena od možnih enostavnejših izboljšav bi bila porezati iskalni drevesi (to lahko storimo tako v primeru kd kot vp dreves) in v listih hraniti več kot en primer, npr. k primerov. Rezanje bi bilo toliko bolj uporabno v primeru višjih vrednosti k, saj bi že takoj imeli več kandidatov za vseh k sosedov. Rezanje načeloma prinaša manj sestopanja v drevesu in manj obiskanih listov, vprašanje pa je, do katerega nivoja je rezanje smiselno.

Kd drevesa lahko izboljšamo s tem, da vpeljemo podobno vrednost τ, kot to storimo pri vp drevesih. S tem bi se hitrost kd dreves dvignila, a bi asimptotično

ostala enaka.

Yianilos v članku [17] omenja še dve različici vp dreves: vps in vpsb drevesa. Za vps navaja asimptotično boljši čas iskanja napram kd in vp metodi v primeru konstantne dimenzije in naraščajočega števila primerov, a je v tem primeru navedena precej nizka dimenzija - 8.

V [18] podajajo algoritem na podlagi Voronoievih diagramov, za katerega navajajo, da ima optimalen pričakovani čas iskanja najbližjih sosedov (in drugih sorodnih problemov). Algoritem je zanimiv dosežek računske geometrije a ga Yianilos v [17] zavrača kot neprimernega za visoke dimenzije. Tudi avtorji pričakujejo neučinkovitost metode že za prostore dimenzij 4 z manj kot 10.000 primeri. Kot razlog navajajo *problem predimenzioniranosti*, ki prej ali slej zadane vsako metodo za iskanje najbližjih primerov v prostorih visokih dimenzij. Zaradi majhnih razlik v razdalji med primeri jih ne moremo učinkovito izločiti iz iskanja in moramo zato preiskati veliko število možnih kandidatov za najbližjega soseda, kar pomeni iskanje, ki je počasnejše od navadnega. Problem predimenzioniranosti je druga posledica istega problema, ki povzroči tudi izgubo pomenskosti najbližjih sosedov (omenjeno v 2.1.1.2).

3.2 Obravnava manjkajočih vrednosti

V določenih podatkovnih bazah nekateri primeri nimajo vseh vrednosti atributov. Razlogi za to so odvisni od domene in načina, s katerimi smo pridobili podatke (podatkov ni mogoče izmeriti, osebe niso želele izdati osebnih podatkov, ipd.). Robustna metoda za napovedovanje spremenljivk mora rešiti tudi vprašanje, kako izračunati razdaljo, ko koordinate primerov v iskalnem prostoru niso enolično določene (primeru manjkajo vrednosti določenih atributov).

V naši implementaciji se odločimo za pristop, podoben tistemu v sistemu Weka [19], kjer je v primeru manjkajočega atributa razdalja enaka največji možni. Za primere razdalj D_2, D_∞ in D_1 lahko spremenimo le način računanja razlike med dvema primeroma v določenem atributu (omenjene razdalje so agregatorska funkcija razlik v atributu). Predpostavimo normalizirane vrednosti atributov (jih zahtevamo,

ko imamo opravka z manjkajočimi vrednostmi). Naj bosta x_m in x_n primera, za katera računamo razdaljo. Namesto razlike $\left|x_{m,j} - x_{n,j}\right|$ za atribut j, bomo upoštevali vrednost glede na naslednje 3 primere.

1. Če sta prisotni obe vrednosti, potem vrnemo razliko $\left|x_{m,j} - x_{n,j}\right|$.

2. Če ena od vrednosti manjka, naj bo to $x_{m,i}$, vrnemo večjo od vrednosti $1 - x_{n,j}$ in $x_{n,j}$.

3. Če manjkata obe vrednosti, vrnemo največjo možno razdaljo, tj. 1.

Zgornji primeri zahtevajo normalizirane koordinate, zato tudi naša `kNN` implementacija zahteva normalizirane podatke, ko podatki niso popolni. Normalizacijo podatkov je moč enostavno vključiti skozi nastavitve poskusa v *Clus* sistemu. Obravnava je še enostavnejša, ko imamo opravka z nenumeričnimi tipi. V kolikor sta oba atributa prisotna in imata enako vrednost, vrnemo 0, sicer vrnemo 1 (kot največjo možno razdaljo).

3.3 Obtežitev atributov

Za pridobitev uteži atributov uporabimo metodo `Random forest` [20], ki je v Clusu že implementirana [21] in zna delati s strukturiranimi podatki. `Random forest` sodi v razred filter metod, saj pridobi uteži atributov ločeno od metode, ki jo kasneje uporabimo za napoved. V privzetem načinu uporabimo 100 naključnih dreves, možno pa je navesti poljubno število. `Random forest` nam vrne range atributov glede na njihove pojavitve v drevesih. Višji rang pomeni večjo uspešnost. Range normaliziramo in preslikamo v uteži. Naj bo rang atributa j enak R_j, potem je utež w_j atributa enaka

$$w_j = 1 - \frac{R_j}{\sum_i R_i}$$

Metodi je v nastavitveni datoteki možno navesti tudi poljubne uteži (ki jih recimo pridobimo iz drugih metod).

3.4 Napovedovanje strukturiranih podatkov

Zaradi enostavnosti metode potrebujemo za delovanje napovedovanja strukturiranih podatkov zgolj spremembe v načinu izračuna prototipa.

3.4.1 Večciljna klasifikacija in regresija

Ko imamo enkrat na voljo izračun prototipa za navadno (enociljno) regresijo in klasifikacijo, je razširitev na večciljno regresijo ali klasifikacijo preprosta. Za vsako od r ciljnih spremenljivk izračunamo prototip, kot bi to storili v primeru enociljnega napovedovanja. Vrnemo seznam r prototipov.

3.4.2 Hierarhična večznačna klasifikacija

Prototip izračunamo na enak način, kot to napravijo v [3]. Oznake razredov najprej pretvorimo v binarni vektor, ki označuje pripadnost primera posameznemu razredu. Vrednost 1 na i-tem mestu pomeni, da primerek pripada razredu c_i. Tako dobljene vektorje povprečimo za vse primere, ki sestavljajo prototip. Povprečje vektorjev označimo z $\bar{v}$. V primeru obteževanja glasov posamezne vektorje seveda obtežimo z normaliziranimi utežmi. V $\bar{v}_i$ je shranjen odstotek primerov klasificiranih v ta razred. Določimo še parameter p_i, ki določa minimalno vrednosti v_i, da primerek klasificiramo v razred i. Parameter p_i določimo glede na cilje našega učenja. Spomnimo se na hierarhično omejitev, ki zahteva, da primerek klasificiramo v nadrejeni razred, kadarkoli klasificiramo v podrejenega. Da zadostimo hierarhični omejitvi, je dovolj, da izberemo takšne parametre p_i, da paroma velja $p_k \leq p_m$ n.t.k. $c_k <_h c_m$. Slednje velja, ker hierarhična omejitev velja za ciljne spremenljivke na učnih primerih in operacije nad $\bar{v}$ njeno veljavo "ohranjajo".

3.4.3 Časovne vrste

V primeru časovnih vrst se odločimo, da bo prototip takšna časovna vrsta, ki ima najmanjšo vsoto kvadratov razdalj do vseh časovnih vrst v napovedi (teh je k).

Izberemo torej vrsto

$$argmin_q \sum_{x \in x_1, x_2, \ldots, x_k} D^2(x, q).$$

V primeru obteževanja glasovanja ustrezno obtežimo posamezne časovne vrste. V tem primeru se enačba preoblikuje v

$$argmin_q \sum_{x \in x_1, x_2, \ldots, x_k} D^2(x, q)/w_q.$$

3.5 O programski kodi

Sistem Clus je implementiran v programskem jeziku Java, zato je v Javi napisan tudi naš prispevek k sistemu. V tabeli 3.5 so navedeni vsi napisani razredi. Razreda `KnnClassifier` in `KnnModel` implementirata klasifikator in model, s katerima Clus pokliče učenje in klasifikacijo.

Paket `distance` vsebuje različne implementacije razdalj in operacije, ki se nanašajo na razdalje. Razred `SearchDistance`, ki spada v ta paket, je abstrakten razred za razdaljo, ki poleg tega nudi tudi skupne metode za izračun razdalje in pa obravnavo manjkajočih vrednosti. Del paketa `distance` sta tudi paketa `distanceWeighting` in `attributeWeighting`. Prvi vsebuje abstrakten razred za obteževanje glasovanja `DistanceWeighting` in vse tri implementirane načine obteževanja. Paket `attributeWeighting` sestavljajo abstrakten razred za obteževanje atributov `AttributeWeighting` in različni načini obteževanja atributov.

Ostali razredi so del metod, ki implementirajo iskanje najbližjega soseda. Razred `SearchAlgorithm` nudi kot abstraktni razred osnovno strukturo za implementirane metode `BF`, `KD` in `VP`. Razred `NNStack` vsebuje podatkovno strukturo (in metode za delo z njo), omenjeno na strani 29. Podatkovna struktura učinkovito hrani in osvežuje seznam najbližjih primerov ter vrača najslabšega med njimi.

Poleg omenjenih razredov je bilo potrebnih še nekaj sprememb v osnovnem ogrodju za vključitev metode `kNN` med seznam možnih metod. Prav tako je bilo potrebnih še nekaj dodatnih razredov in prilagoditev za namene primerjanja metod v

Tabela 3.1: Opisi razredov, ki sestavljajo Clus kNN implementacijo in število vrstic kode, ki sestavlja razred. Vodoravne črte ločujejo razrede glede na njihov namen.

vrstic	razred
129	./**KnnClassifier**.java
288	./**KnnModel**.java
43	./distance/distanceWeighting/**DistanceWeighting**.java
37	./distance/distanceWeighting/**WeightConstant**.java
36	./distance/distanceWeighting/**WeightMinus**.java
37	./distance/distanceWeighting/**WeightOver**.java
76	./distance/attributeWeighting/**AttributeWeighting**.java
21	./distance/attributeWeighting/**NoWeighting**.java
129	./distance/attributeWeighting/**RandomForestWeighting**.java
27	./distance/attributeWeighting/**UserDefinedWeighting**.java
128	./distance/**SearchDistance**.java
48	./distance/**EuclideanDistance**.java
37	./distance/**ManhattanDistance**.java
35	./distance/**ChebyshevDistance**.java
74	./methods/**SearchAlgorithm**.java
89	./methods/**NNStack**.java
48	./methods/bfMethod/**BrutForce**.java
111	./methods/kdTree/**KDTree**.java
247	./methods/kdTree/**KDNode**.java
255	./methods/vpTree/**VPTree**.java
52	./methods/vpTree/**VPNode**.java
31	./methods/vpTree/**VPItem**.java
	skupaj 1978 vrstic

Tabela 3.2: Izpis parametrov metode in možnih vrednostih le-teh.

parameter	možne vrednosti	privzeto
k	1,2,3...	3
method	vp-tree, kd-tree, bf-method, auto	auto
distance	euclidean, manhattan, chebyshev	euclidean
distanceWeighting	none, 1/d, 1-d	none
attributeWeighting	none, RF, 'RF,100', [2,1.2,...], f.weight	none

poglavju 4.

3.6 Nastavitve metode

Nastavitve poskusa v sistemu Clus nastavljamo v ".s" datoteki, ki jo podamo programu. Datoteka vsebuje informacije o vhodnih podatkih, izhodnih modelih, predprocesiranju podatkov, ...

Za uporabo metode kNN v nastavitveno datoteko dodamo odsek [kNN], kjer navedemo vse nastavitve, ki vplivajo neposredno na našo metoda. Nastavitve, možne in privzete vrednosti so povzete v tabeli 3.2. Podajmo še opis posameznih parametrov:

method Izbira metode iskanja najbližjih sosedov. Z vrednostjo "auto"omogočimo samodejno izbiro metode glede na parametra n, k in d ter formulo 4.4.

distance Izbira uporabljene razdalje za iskanje najbližjih sosedov.

distanceWeighting Parameter določa način obteževanja glasov, ko je parameter k večji od 1.

attributeWeighting Določimo način obteževanja atributov. Pri RF (Random Forest) je privzeto število naključnih dreves 100, lahko pa izberemo tudi drugačno število, npr.: RF,50; RF,120;... Možno je tudi navesti .weight datoteko (ta se shrani ob uporabi RF obteževanja atributov) in tako določiti, katere uteži naj algoritem privzame. To omogoča večkratno poganjanje eksperimenta z istim

```
[Data]
File = ./learn_set.arff
TestSet = ./test_set.arff

[kNN]
k = 5
distance = euclidean
distanceWeighting = 1/d
attributeWeighting = none
```

Slika 3.4: Izsek iz Clus nastavitvene datoteke.

naborom uteži. Tretja možnost je uporaba poljubnih uteži, ki jih navedemo kar v nastavitveno datoteko.

Primer Clus nastavitvene datoteke lahko vidimo na sliki 3.4. Seveda je možnih nastavitev v sistemu Clus mnogo več, a njihova predstavitev presega obseg tega dela.

Poglavje 4

Primerjava algoritmov iskanja sosedov

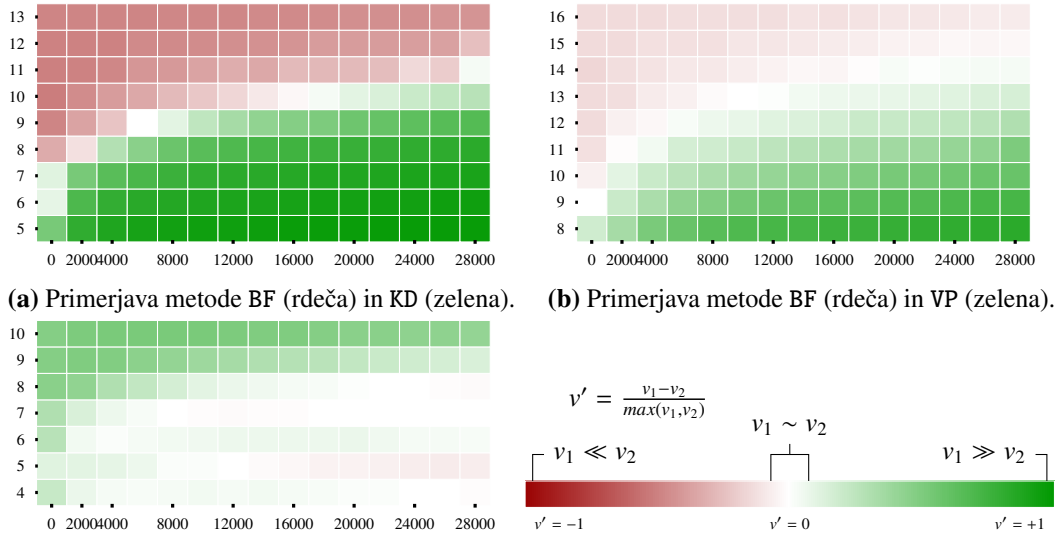
V tem razdelku primerjamo vse tri iskalne metode z namenom pridobitve znanja, kdaj je katera od metod hitrejša od ostalih. Naša motivacija je omogočiti samodejen preklop med implementiranimi metodami iskanja glede na število dimenzij in primerov v podatkovni bazi.

Za namen testiranja generiramo naključne podatke z $d \in [4..20]$ dimenzijami in $n \in [100..30000]$ primeri. Vsaka od d vrednosti v primeru je izbrana enakomerno naključno iz $(0, 1)$. Za vsak test nad posamezno množico smo opravili 10 korakov navzkrižne validacije in merili čas, ki ga metoda porabi za iskanje sosedov - meri se le čas porabljen v iskalni metodi. Navzkrižne validacije ne opravimo zaradi verodostojnosti napovedovanja, temveč zaradi večjega števila iskanj in posledično jasnejše razlike med metodami.

V razdelku 3.1.3 smo že omenili problem iskanja v prostorih visokih dimenzij. Problem predimenzioniranosti se pojavi nekje okrog krivulje (odvisno od podatkov):

$$n = O\left(2^d\right). \tag{4.1}$$

Število primerov mora torej eksponentno naraščati s številom dimenzij. Ker je v podatkovnem rudarjenju temu redko tako, pričakujemo, da se bosta metodi `KD` in `VP` v prostorih z višjimi dimenzijami odrezali slabše. Za vp drevo sicer Yianilos v [17]



(a) Primerjava metode `BF` (rdeča) in `KD` (zelena).

(b) Primerjava metode `BF` (rdeča) in `VP` (zelena).

(c) Primerjava metode `KD` (rdeča) in `VP` (zelena).

(d) Skala izboljšave.

Slika 4.1: Razlike med pari metod v času porabljenim za iskanje **enega** najbližjega soseda. Faktor izboljšave lahko ocenimo s skalo na sliki 4.1d. Barve v oklepaju poleg metod označujejo barvo kvadrata, v katerem se metoda izkaže bolje od nasprotnice.

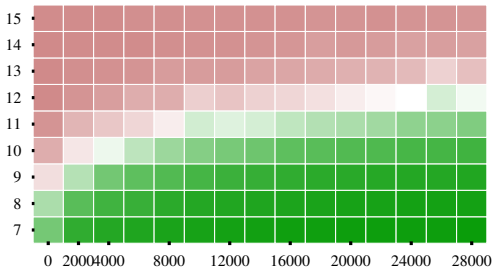
navaja zmožnost hitrejšega iskanja, a žal primerja le `KD` in `VP` metodi.

4.1 Vizualizacija primerjave

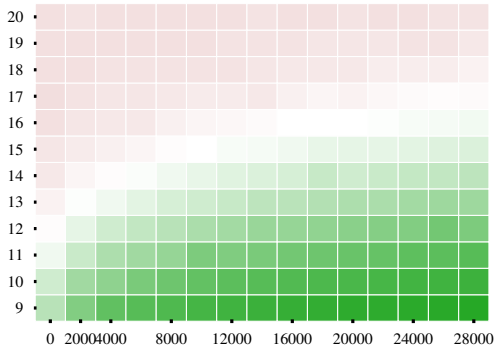
Želimo enostavno prikazati primerjavo med posameznimi pari algoritmov za "vse" pare vrednosti (n, d). Primerjave med posameznimi pari metod za različne vrednosti k so predstavljene na slikah 4.1a do 4.3c.

Na slikah abcisna os predstavlja število primerov v podatkovni bazi, ordinatna pa število atributov. Prostor (n, d) razdelimo v mrežo kvadratov. Vsak kvadrat predstavlja povprečje vrednosti testov, katerih n in d parametra padeta znotraj kvadrata. Za kvadrate, kjer ni bilo testov, smo vrednost interpolirali iz 4-soseščine. Takih kvadratov je največ okoli 5% na sliko in na grafično predstavitev nimajo pomembnejšega vpliva.

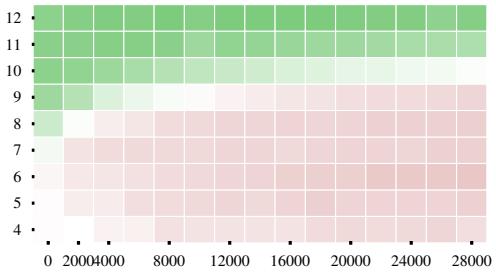
Barvno vrednost izračunamo na naslednji način:



(a) Primerjava metode `BF` (rdeča) in `KD` (zelena).



(b) Primerjava metode `BF` (rdeča) in `VP` (zelena).



(c) Primerjava metode `KD` (rdeča) in `VP` (zelena).

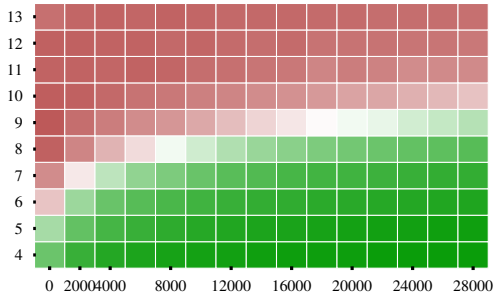
Slika 4.2: Razlike med pari metod v času porabljenim za iskanje **petih** najbližjih sosedov. Faktor izboljšave lahko ocenimo s skalo na sliki 4.1d. Barve v oklepaju poleg metod označujejo barvo kvadrata, v katerem se metoda izkaže bolje od nasprotnice.

1. Če je povprečna vrednost (čas) prve metode manjša od druge, bo kvadrat rdeče barve. Sicer bo zelene.

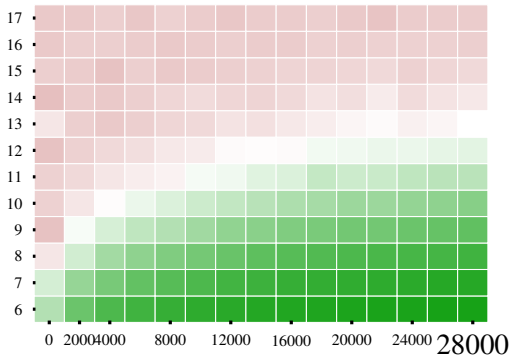
2. Inteziteto barve določimo po formuli $v = \left|\frac{v_1 - v_2}{\max(v_1, v_2)}\right|$. Večja kot je razlika med metodama, intezivnejša bo barva.

Prikazana razlika med metodama torej ni absolutna, temveč relativna glede na posamezen kvadratek. Pove nam odstotek izboljšave pri uporabi ene izmed metod. Pri kvadratih, ki so obarvani belo oz. z bledo barvo, ni večje razlike med metodama.

Podobne slike dobimo, če si namesto časa iskanja, pogledamo število obiskanih vozlišč.

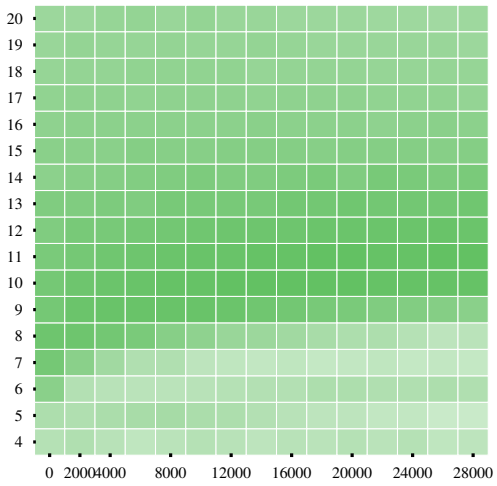


(a) Primerjava metode `BF` (rdeča) in `KD` (zelena).



(b) Primerjava metode `BF` (rdeča) in `VP` (zelena).

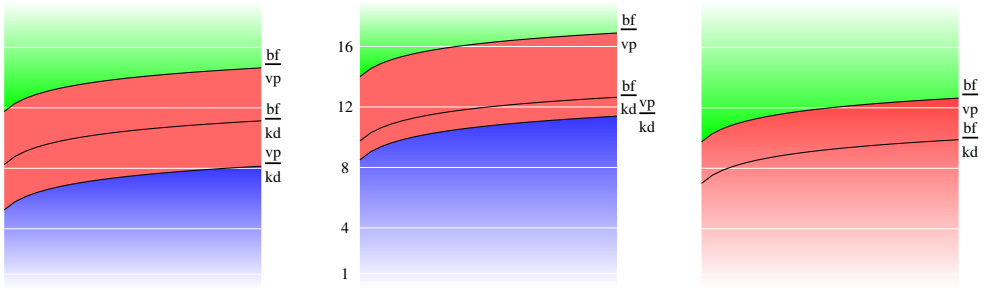
Slika 4.3: Razlike med pari metod v času porabljenim za iskanje **enajstih** najbližjih sosedov. Faktor izboljšave lahko ocenimo s skalo na sliki 4.1d. Barve v oklepaju poleg metod označujejo barvo kvadrata, v katerem se metoda izkaže bolje od nasprotnice.



(c) Primerjava metode `KD` (rdeča) in `VP` (zelena). Jasno je, da se metoda `VP` povsod obnese bolje kot `KD`. Ker na podlagi drugih grafov `KD` pričakujemo pri nižjih vrednostih d, lahko ugotovitev posplošimo na vse parametre n, d.

4.2 Povzetki primerjav

Iz primerjav metod vidimo, da se v prostorih visokih dimenzij najbolje obnese kar navadno iskanje. Slika 4.3 prikazuje približne dele prostora (n, d), kjer je posamezna metoda uspešnejša. Meja, ko sta tako metoda VP kot KD slabši od BF se pojavi najpozneje v primeru 16-ih dimenzij, kar je v skladu z ugotovitvami v članku [1]. Ker metode primerjamo na prostorih enakih lastnosti, kot to naredijo v [18], lahko sklepamo, da se VP obnese bolje kot metoda izpeljana iz Voronoiovih diagramov. Za slednje navajajo nepraktičnost za dimenzije med 4 in 6 in število primerov manj od 10.000. Sklepamo, da nepraktičnost pomeni slabšo uspešnost v primerjavi z navadnim iskanjem. V našem primeru je ta meja med 10 in 16 dimenzijami.

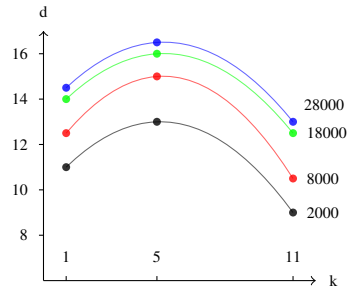


(a) $k = 1$ **(b)** $k = 5$ **(c)** $k = 11$

Slika 4.3: Diagram hitrosti metod KD, VP in BF za različne vrednosti k. Abcisa v diagramu predstavlja število primerov, ordinata pa število dimenzij. Črne krivulje predstavljajo meje, ko je ena od dveh primerjanih metod boljša oz. slabša od druge.

Na sliki 4.3 lahko opazimo tudi spreminjanje višine mej s številom iskanih sosedov. Graf 4.4 prikazuje odvisnost števila dimenzij od števila iskanih sosedov k za mejo med metodo VP in KD. Iz grafa sklepamo na kvadratično odvisnost med spremenljivkama.

Naslednji funkciji nam za dani k in n ocenita mejno vrednost d, na podlagi katere se odločimo, katero metodo je smotrneje uporabiti. Naj bo f funkcija, ki ocenjuje mejo med BF in VP ter g funkcija, ki ocenjuje mejo med VP in KD.



Slika 4.4: Spreminjanje meje, ko se metoda VP obnese bolje od KD, glede na število dimenzij d in število iskanih sosedov k za 4 različne vrednosti števila primerov n.

$$f(k,n) = 2.5\log(n+5015) - 0.1(k-5.2)^2 - 9.1 \tag{4.2}$$

$$g(k,n) = 1.2\log(n+2594) - (k-2.8)^2 - 7 \tag{4.3}$$

Z njima lahko za trojico (n,k,d) določimo uporabljeno metodo:

$$\texttt{metoda}(n,k,d) = \begin{cases} \texttt{BF} & ; \quad d > f(k,n) \\ \texttt{KD} & ; \quad d < g(k,n) \\ \texttt{VP} & \text{sicer} \end{cases} . \tag{4.4}$$

4.3 Čas za pripravo iskanja

Metoda BF ne potrebuje priprave iskanja, saj povsem enostavno preišče celoten prostor, zato je smiselno pogledati le razliko v pripravi iskanja med metodama VP in KD. V naši postavitvi je VP počasnejša tudi za faktor 40 (tabela 4.1). Spomnimo se, da je čas gradnje vp drevesa v veliki meri odvisen od vrednosti σ_1 in σ_2 v enačbi (3.1) in bi ga zato lahko zmanjšali/povečali. Omeniti je potrebno še, da za naloge, ko imamo dovolj veliko število neznanih primerov in le relativno majhno število znanih oz. ko opravimo dovolj iskanj, postane čas gradnje hitro zanemarljiv napram času namenjenem iskanju.

Tabela 4.1: Kvocient časov za pripravo iskanja metod KD in VP: $build_{VP}/build_{KD}$. Za uporabljene vrednosti σ_1 in σ_2 iz enačbe (3.1) je jasna prednost metode KD.

k	min	max	avg
1	2.08	41.31	21.29
5	2.11	41.99	21.89
11	1.92	29.20	15.69

Tabela 4.2: Primerjava hitrosti metod na različnih podatkovnih bazah iz realne domene. Zadnji stolpec vsebuje število numeričnih (N) in diskretnih (D) atributov. Iščemo **enega** soseda.

	iskanje			gradnja		lastnosti baz			
	bf	kd	vp	kd	vp	d	n	r	N+D
wine	434	492	**291**	6	23	13	2310	1	13+0
cpu	123	47	**23**	6	12	6	209	1	6+0
iris	297	149	**100**	9	21	4	150	1	4+0
edm	386	323	**177**	7	23	16	154	2	16+0
sigmeasim-dis	1275930	11166	**9816**	682	9092	11	10368	2	9+2
sigmeareal	3256	223	**154**	28	105	4	817	2	4+0

4.4 Metode na realnih podatkih

Hitrosti iskanja metod smo primerjali tudi na različnih bazah. Nekatere so dobro poznane iz UIC skladišča (*wine, cpu, iris*), druge pa uporabljamo v tem delu in so opisane pozneje (*edm, sigmeasim, sigmeareal*). Opazimo, da se metoda VP obnese bolje povsod, kar je v večini baz neskladno (v prid VP) z našo enačbo (4.4). Neskladnost je verjetno posledica neenakomerne razporeditve primerov v prostoru. Kljub temu pustimo parametre v enačbah (4.2, 4.3) nespremenjene, tj. bolj konzervativne.

4.5 Povzetek

Primerjave dajejo vsaj deloma pričakovane rezultate. Jasno je, da hitro naletimo na problem predimenzioniranosti, kjer odpovedo "modernejše" metode in je najbolje, da se zanesemo kar na navadno iskanje najbližjih sosedov. Za naše implementacije metod smo dobili mejo, ki jo lahko izkoristimo vsaj za približno merilo, kdaj

uporabiti BF in kdaj katero od drugih metod. Zaradi nekoliko drugačnih zasnov implementacije KD in VP metode je možno, da razlika med njima izhaja iz implementacije in ne načina delitve prostora. Iskalno drevo pri metodi VP je namreč manjše zaradi shranjevanja točk pogleda v notranja vozlišča, medtem ko so pri KD metodi vse točke v listih. Posledično (v vp drevesu) hitreje najdemo ustrezen list. Omenimo še pomembno prednost metode VP pred KD in sicer, da prva deluje v splošnih metričnih prostorih, medtem ko druga le v evklidskih. Zato lahko metodo uporabimo tudi, ko imamo med opisnimi atributi objekte, nad katerimi ni definirana evklidska razdalja. Dovolj je, da za razdaljo velja trikotniška neenakost, saj s tem drži tudi enačba (3.2) in poznejše izpeljave glede omejevanja preiskanega prostora.

Poglavje 5

Večciljno napovedovanje

V tem razdelku preizkusimo implementirano `kNN` metodo na problemu večciljne klasifikacije in regresije. Kot smo že omenili v poglavju 2, razlik v uspešnosti med večciljnim in posamičnim napovedovanje z uporabo metodo `kNN` ni. Zato nas v tem poglavju predvsem zanima, kako se metoda obnese napram drugim metodam, ki podpirajo večciljno klasifikacijo. V ta namen jo primerjamo s pravili in drevesi za napovedno razvrščanje, ki smo jih spoznali v poglavju 2. Ženko za pravila v [6, 7] pokaže, da je njihova uspešnost pri večciljnem napovedovanju primerjiva s posamično. Pokaže tudi primerljivo uspešnost z nekaterimi drugimi metodami, ki implementirajo učenje pravil (cn2, jrip). Podobno Struyf v [9] pokaže primerljivo uspešnost med večciljnim in posamičnim hierarhičnim večznačnim napovedovanjem za PCT.

5.1 Uporabljene podatkovne baze

V tem poglavju uporabimo 20 podatkovnih baz, od tega je 11 klasifikacijskih in 9 regresijskih. Tabeli 5.2b in 5.2a povzemata osnovne lastnosti uporabljenih baz.

Podatkovne baze izhajajo iz različnih področij, večina pa iz okoljskih problemov. V nadaljevanju sledijo opisi uporabljenih podatkovnih baz, ki so v veliki meri povzeti po [6].

Tabela 5.1: Lastnosti podatkovnih baz uporabljenih v poglavju. V stolpcu d je navedeno število diskretnih + število numeričnih atributov. Število ciljnih spremenljivk je označeno z r.

	n	d	r
EDM	154	0+16	2
Sigmea Real	817	0+4	2
Sigmea Simulated	10368	2+9	2
Water Quality	1060	0+16	14
Solar-flare 1	323	10+0	3
Collembola	393	8+39	3
Soil Quality	1944	0+142	3
Soil Resilience	27	1+7	8

(a) Podatkovne baze večciljne regresije.

	n	d	r
EDM	154	0+16	2
Sigmea Real	817	0+4	2
Sigmea Simulated	10368	2+9	2
Water Quality	1060	0+16	14
Solar-flare 1	323	10+0	3
Thyroid	9172	22+7	7
Emotions	593	0+72	6
Monks	432	6+0	3
Scene	2407	0+294	6
Yeast	2417	0+103	14
Bridges	102	4+3	5

(b) Podatkovne baze večciljne klasifikacije.

5.1.1 Problemi večciljne klasifikacije

Bridges Podatkovna baza je iz *UCI* skladišča in vsebuje podatke o 102 mostovih opisanih s 7 atributi (namen, leto izdelave,...). Za vsakega od mostov imamo 5 ciljnih spremenljivk - lastnosti, ki jih napovedujemo (uporabljeni material, razpon,...).

EDM (*Electrical Discharge Machining*) Problem je originalno regresijski in je podrobneje opisan v 5.1.2. Za namene klasifikacije sta bili ciljni spremenljivki diskretizirani v 3 nominalne vrednosti: "-1" za negativne vrednosti, "1" za pozitivne in "0" za ostale.

Monks Podatkovna baza je namenjena preizkušanju algoritmov strojnega učenja [22]. Vsak robot je opisan s šestimi nominalnimi atributi $(a_1, \ldots, a_6)$, kjer vsak od atributov zavzame od 2 do 4 možne vrednosti. Vseh primerov (robotov) je 432. Imamo 3 ciljne spremenljivke - binarne razrede:

monk-1 $\equiv a_1 = a_2 \vee a_5 = 1$
monk-2 $\equiv$ natančno dva atributa med vsemi imata vrednost 1
monk-3 $\equiv (a_5 = 3 \wedge a_4 = 1) \vee (a_4 \neq 4 \wedge a_2 \neq 3)$.

V zadnji razred je bilo dodano 3% šuma.

Sigmea Real je regresijska podatkovna baza in je podrobneje opisana v 5.1.2. Obe ciljni spremenljivki smo diskretizirali v "1" za pozitivne vrednosti spremenljivke in "0" za ničelne.

Sigmea Simulated je prav tako regresijska podatkovna baza in je podrobneje opisana v 5.1.2. Tudi tu smo obe ciljni spremenljivki diskretizirali v "1" za pozitivne vrednosti spremenljivke in "0" za ničelne.

Solar Flare Podatkovno bazo smo dobili iz *UCI* skladišča in predstavlja informacijo o sončevih izbruhih. Vsak izbruh zaznamo kot močno spremembo v svetlosti na površini sonca. V bazi je 323 izbruhov zabeleženih med 13. februarjem in 27. marcem 1969. Dejavnost vsakega izbruha v prejšnjih 24 urah je opisana z 10 atributi. Naloga je napovedati število in razred izbruhov v naslednjih 24 urah. Razred označuje inteziteto izbruha (C, M, X). Ker so izbruhi z manjšo inteziteto (C) pogostejši od intezivnejših (M) in najbolj intezivnih (X), je porazdelitev primerov po razredih močno neenakomerna.

Water Quality Podatkovna baza je bila prvotno regresijski problem, zato je podrobno predstavljena v 5.1.2. Vse ciljne spremenljivke smo diskretizirali v "1" za pozitivne vrednosti spremenljivke in "0" za ničelne.

Thyroid Podatkovna baza je iz *UCI* skladišča in nosi oznako *0387* (na voljo je več podatkovnih baz, ki se nanašajo na ščitnico). Baza vsebuje podatke o 9172 pacientih med leti 1984 in 1987. Za vsakega pacienta imamo zabeleženih 29 atributov, ki opisujejo njegove stanje in zdravljenje v preteklosti. Naloga je napovedati 7 različnih bolezni. Teh 7 bolezni predstavlja 7 binarnih razredov.

Emotions Tu se ukvarjamo s problemom napovedovanja čustev, ki jih povzroči poslušanje določenega zvočnega zapisa (skladbe) [23]. Podatkovna baza ima 593

zvočnih zapisov, vsak zapis je opisan z 72 atributi. Napovedujemo 6 razredov - čustev.

Scene Podatkovna baza [24] vsebuje 2407 primerov - slik scen, ki jih želimo klasificirat. Ker lahko scena vsebuje več različnih vsebin, ciljni razredi niso disjunktni. Zato problem vsebuje večciljno napovedovanje. V tem primeru iz 294 opisnih atributov napovedujemo 6 ciljnih razredov.

Yeast Podatkovna baza je s področja funkcionalne genomike kvasovk. Vsebuje 2417 primerov genov opisanih z 103 zveznimi atributi. Napovedujemo 14 razredov - funkcij genov (metabolizem, energija, razkroj proteinov, sinteza proteinov,...). Več v [25].

5.1.2 Problemi večciljne regresije

EDM (*Electrical Discharge Machining*) EDM ali žična erozija je postopek obdelave trdih elementov, ki so električno prevodni. Obdelovanec premikamo med dvema elektrodama skozi katere teče tok. Tok postopoma erodira v material. Točka obdelave je konstantno izpirana s prevodno tekočino. Stabilnost in kvaliteta postopka je odvisna od mnogih parametrov (obdelovanca, tipa prevodne tekočine, širine vrzeli, toka med elektrodama,...). Čeprav so parametri vnaprej nastavljeni, je moč skrajšati čas obdelav z dinamičnim prilagajanjem parametrov, za kar je navadno zadolžen človeški operater. Naloga je posnemati odzive operaterja. Podatkovna baza vsebuje 154 različnih odzivov operaterja, kjer spreminja dva parametra - ciljni spremenljivki (tok in vrzel). Obe spremenljivki imata tri možne vrednosti: -1 v primerih, ko je operater zmanjšal vrednost parametra, 1, ko jo je povečal in 0, ko ni spremenil ničesar. Opisne atribute sestavlja 16 numeričnih vrednosti, ki opisujejo električne impulze v zadnjih petih sekundah. Podrobnejši opis podatkov in njihova analiza se nahaja v [26].

Sigmea Real Z vedno večjo uporabo gensko spremenjenih organizmov se pojavljajo tudi mnogi agronomski in ekološki pomisleki, med katerimi je največji, kako

omejiti širjenje novih genov z gensko spremenjenih organizmov na ostale vrste. V okviru projekta *Sustainable Introduction of GMO's into European Agriculture (SIGMEA)* je bil izveden naslednji poskus: na gredico velikosti 10m x 10m je bila posejana gensko spremenjena oljčna ogrščica z odpornostjo na herbicide. Gredica je bila obdana s poljem velikost 90m x 90m, kjer sta bili posejani dve vrsti (MS in MF) oljčne ogrščice brez odpornosti na herbicide. Kot merilo širjenja cvetnega prahu je vzeta odpornost posamezne oljčne ogrščice na herbicide. Naloga je modelirati odpornost na herbicide obeh vrst oljčne ogrščice glede na lokacijo v polju, oddaljenost od centra, oddaljenostjo od najbližje stranice gredice, ipd. Ogrščice so bile na polju posejane v mreži 29 x 29 repic. Posamezne ogrščice (tiste izven gredice) predstavljo primere v bazi. Analiza in podroben opis podatkov je v [27].

Sigmea Simulated Tudi *Sigmea Simulated* se podobno kot *Sigmea Real* ukvarja s prehajanjem genov iz gensko spremenjene na običajne vrste oljčne ogrščice. Tudi ta podatkovna baza izhaja iz *SIGMEA* projekta, le da podatki ne predstavljajo praktičnega eksperimenta, temveč simulacijo prehajanja genov v času in prostoru. Simulacija je bila opravljena s pomočjo *GeneSys* modela [28]. Ciljni spremenljivki označujeta širjenje cvetnega prahu in semen. Podatki so bili že analizirani v [29], kjer je na voljo tudi podroben opis.

Solar Flare Gre za iste podatke kot v klasifikacijskem primeru, le da ciljne vrednosti (število sončnih izbruhov) obravnavamo kot numerične podatke. Več informacij v 5.1.1.

Water quality Podatkovna baza vsebuje podatke o kvaliteti vode v Slovenskih rekah v obdobju 1990-1995. Bazo je pripravila *Agencija za okolje Republike Slovenije*. Biološki vzorci so zajeti dvakrat letno (poleti in pozimi), fizikalna in kemična analiza pa je opravljena večkrat letno za vsakega od območij vzorčenja. Fizikalni in kemični parametri zajemajo 16 opisnih atributov, biološke atribute (teh je 14) pa obravnavamo kot cilje spremenljivke. Baza vsebuje vsega 1060 vzorčenj.

Collembola Podatkovna baza zajema podatke o številčnosti skakačev (Collembola) v poljedeljski pokrajini na jugu Nemčije. Vsak primerek v bazi (teh je 396) predstavlja vzorec prsti, ki je opisan z 47 atributi. Ti zajemajo kemične, biološke in fizične lastnosti prsti. Napovedujemo številčnost oz. odsotnost skakačev.

Soil Quality Podatkovna baza iz ECOGEN projekta poskuša napovedati kvaliteto prsti glede na agronomske procese na njej v preteklosti. Več v [30].

Soil Resilience Podatkovna baza zajema problem prožnosti prsti na škotskem. Ciljne spremenljivke predstavljajo prožnost prsti glede na različne vplive. Več v [31].

5.2 Ocenjevanje uspešnosti metode

V primeru klasifikacije vzamemo kot mero uspešnosti na celotni bazi $\overline{CA} = \sum_i \frac{CA_i}{|C|}$, povprečje klasifikacijske točnosti CA_i po vseh ciljnih spremenljivkah i.

Za ocenjevanje regresije pa uporabimo $\overline{RRMSE}$, povprečje $RRMSE$ ocen. $RRMSE$ ali relativni $RMSE$ ocenjuje relativno napako regresije glede na napako privzetega modela. Privzeti model v primeru regresije vrača povprečno vrednost ciljne spremenljivke vseh učnih primerov. S podanim $RMSE^{\text{def}}$ privzetega modela, lahko $RRMSE_i$ za ciljno spremenljivko i izračunamo kot ulomek:

$$RRMSE_i = \frac{RMSE_i}{RMSE_i^{\text{def}}}. \tag{5.1}$$

Mero na celotni bazi izrazimo kot povprečje $RRMSE$ napak po vseh ciljnih spremenljivkah:

$$\overline{RRMSE} = \frac{1}{n} \sum_{i=1}^{n} \frac{RMSE_i}{RMSE_i^{\text{def}}}. \tag{5.2}$$

5.3 Primerjava

Vse teste opravimo z 10-kratno navzkrižno validacijo. Uspešnost metode `kNN` merimo za $k \in \{1, 3, 5, 15\}$ in dve različici glasovanja (brez, d^{-1}).

V primeru večciljne regresije primerjavo opravimo na petih podatkovnih bazah (edm, sigmea-real, sigmea-sim, solar-flare, water-quality), ki imajo skupaj 24 ciljnih spremenljivk. Pri večciljni klasifikaciji vzamemo 7 podatkovnih baz (edm, sigmea-real, sigmea-sim, solar-flare, water-quality) s skupno 33 ciljnimi spremenljivkami.

Podamo vse *CA* in *RMSE* ocene za posamezno ciljno spremenljivko in pa tudi ustrezna povprečja nad celotno bazo ($\overline{CA}$, $\overline{RMSE}$).

Uspešnost `kNN` metode primerjamo z metodama `PCR` in `PCT`. V ta namen uporabimo podatke o uspešnostih omenjenih metod iz [6]. V tem delu se Ženko med drugim ubada s primerjavo 4 različic `PCR` metode napram metodi `PCT`. Uporabimo podatke za pravila `PCR UN MULTIPLE`, torej neurejena pravila za večciljno napovedno razvrščanje.

5.4 Rezultati primerjav

Tabeli 5.2 in 5.3 (strani 53 in 56) podajata uspešnosti za metodi `PCR` in `PCT` ter za metodo `kNN` za vrednosti $k \in \{1, 3, 5, 15\}$ in z ter brez uporabe obteževanja glasovanja d^{-1}. Spomnimo se, da obteževanje glasovanja nima vpliva v primeru enega samega glasu. Rezultate Nemenyijevih testov smo v tem in naslednjih poglavjih grafično predstavili z diagrami, kot Demšar predlaga v [32].

5.4.1 Vpliv obtežitve glasov

Najprej preverimo vpliv obteževanja glasovanja na uspešnost napovedovanja. Z Wilcoxonovim testom preverimo vse pare metod z in brez obteževanja glasovanja: za $k \in \{3, 5, 15\}$, za klasifikacijo in regresijo. Rezultate prikazuje tabela 5.4, kjer vidimo, da se v vseh primerih, razen v primeru klasifikacije za $k = 15$, metoda `kNN`

Tabela 5.2: Klasifikacijske točnosti *CA* za metode PCR, PCT in kNN ter privzeti klasifikator def. Na levi navajamo podatkovne baze in posamezne cilje spremenljivke. Vrednosti za PCR in PCT so povzete iz [6].

	def	PCR	PCT	1-NN	3-NN		5-NN		15-NN	
						d^{-1}		d^{-1}		d^{-1}
monks	55.5	76.5	82.8	64.5	68.1	68.1	75.5	75.5	81.2	84.9
monk-1	46.5	82.4	90.7	53.7	74.5	74.5	90.5	90.5	79.4	90.0
monk-2	67.1	64.1	58.1	48.4	47.0	47.0	47.7	47.7	65.7	65.0
monk-3	52.8	82.9	99.5	91.4	82.9	82.9	88.4	88.4	98.6	99.5
sigmea-real	63.7	75.1	70.6	69.5	73.7	73.7	76.2	75.1	77.5	77.1
mfo	68.7	74.9	70.1	71.0	74.1	74.3	76.3	75.5	77.6	77.4
mso	58.8	75.4	71.1	67.9	73.3	73.1	76.1	74.7	77.5	76.9
sigmea-sim	62.6	97.9	99.9	98.8	98.9	98.7	97.8	98.8	97.8	98.8
disp-rate	69.6	95.7	99.7	97.6	98.2	97.4	97.7	97.7	97.6	97.9
disp-seeds	55.6	100	100.	100.	99.6	100.	98.0	99.9	97.9	99.8
thyroid	96.1	97.5	99.0	93.3	95.4	95.4	95.7	95.7	96.1	96.1
hyper-thyroid	97.4	97.5	98.9	93.7	96.9	96.9	97.4	97.4	97.4	97.4
hypo-thyroid	92.7	96.3	99.2	89.1	91.0	91.0	91.4	91.4	92.7	92.7
bind-prot	95.5	96.6	98.4	91.5	95.3	95.3	95.4	95.4	95.5	95.5
gen-health	93.8	97.3	99.0	90.3	93.0	93.0	93.1	93.1	93.7	93.7
repl-theory	96.1	97.0	99.0	93.7	94.3	94.4	95.1	95.2	96.1	96.1
antithyro-tr	99.6	99.6	99.7	99.4	99.6	99.6	99.6	99.6	99.6	99.6
disc-results	97.9	98.0	99.0	95.3	97.8	97.8	97.8	97.8	97.9	97.9
water-quality	65.8	68.2	65.7	63.5	66.0	65.9	67.2	66.8	69.1	69.1
clad-sp	57.0	49.6	59.2	61.4	63.9	63.6	64.1	63.4	65.3	64.7
gong-inc	73.3	71.6	66.4	61.2	66.1	65.8	68.4	68.2	71.6	71.4
oedo-sp	71.2	70.1	69.8	65.8	69.6	69.1	70.3	69.9	71.0	71.8
tige-ten	79.4	79.2	76.1	73.6	74.2	74.2	77.3	76.4	79.7	79.8
melo-var	52.8	58.6	56.8	56.0	58.1	57.5	59.2	58.5	61.2	60.4
nitz-pal	55.8	69.4	64.6	61.5	63.0	63.7	64.1	63.6	65.1	65.5
audo-cha	75.3	75.8	69.8	66.9	70.7	71.4	72.0	71.8	75.2	75.7
erpo-oct	71.8	73.5	68.9	66.4	70.3	69.7	72.5	71.7	72.7	72.3
gamm-foss	58.5	62.2	62.2	59.3	61.3	61.0	61.4	60.2	63.8	63.5
baet-rhod	67.6	67.5	67.6	64.8	68.0	68.0	70.9	70.6	72.0	71.8
hydro-sp	56.1	61.8	60.5	59.8	61.3	60.7	62.3	62.2	66.0	66.0
rhya-sp	68.5	69.2	68.4	65.1	66.6	66.7	67.5	67.2	68.6	68.3
simu-sp	63.3	62.7	59.4	61.3	60.4	60.1	60.7	60.6	62.7	63.0
tubi-sp	70.7	72.9	70.2	66.2	70.3	70.8	70.2	71.3	72.3	73.3

Nadaljevanje tabele na naslednji strani.

Tabela 5.2: Nadaljevanje tabele s prejšnje strani - uspešnosti metod za problem večciljne klasifikacije.

	def	PCR	PCT	1-NN	3-NN		5-NN		15-NN	
						d^{-1}		d^{-1}		d^{-1}
solar-flare	92.3	89.6	90.4	89.4	91.6	90.6	92.0	91.0	92.3	91.0
c-class	88.9	86.4	86.7	85.8	88.2	87.3	88.5	87.6	88.9	87.6
m-class	90.1	85.1	86.7	87.0	89.5	87.9	89.5	88.2	90.1	88.2
x-class	97.8	97.2	97.8	95.4	97.2	96.6	97.8	97.2	97.8	97.2
edm	69.5	67.8	77.2	75.3	74.0	73.7	69.8	70.8	69.8	74.7
d-flow	83.1	87.7	88.3	90.9	89.6	89.6	87.0	87.7	84.4	90.3
d-gap	55.8	53.9	66.2	59.7	58.4	57.8	52.6	53.9	55.2	59.1
povprečje	73.4	79.6	80.3	76.1	78.3	78.1	79.3	79.2	80.8	81.5

obnese bolje, če glasov ne obtežujemo. V primeru klasifikacije in glasovanja s petimi sosedi se metoda brez obteževanja obnese celo značilno boljše.

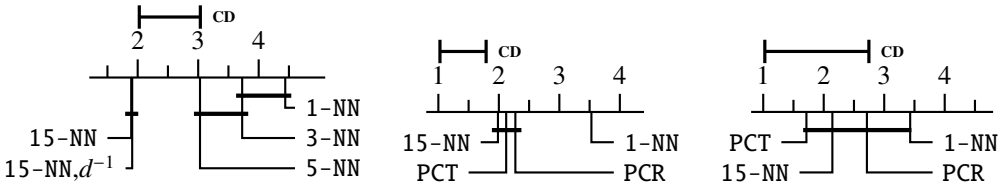
Sklepamo, da se bo v splošnem metoda brez obteževanja glasovanja odrezala bolje od obteževanja d^{-1}, zato primerjave v nadaljevanju tega poglavja uporabljajo metodo kNN brez obteževanja glasovanja.

5.4.2 Primerjava metod na problemih večciljne klasifikacije

Sprva (diagram 5.1a) primerjamo metodo kNN pri različnih vrednosti k. Tudi v primeru večciljne klasifikacije se za boljše izkaže napovedovanje, kjer v obzir vzamemo večje število sosedov. Za razliko od regresije pa je od 15-NN značilno slabša tudi 5-NN.

V drugem diagramu (5.1b) primerjamo kNN napram PCT in PCR. Tu se kot najboljša presenetljivo obnese metoda 15-NN, sledita ji PCR in PCT; vse tri niso značilno različne. Značilno najslabša pa je pričakovano 1-NN.

Podobno opravimo primerjavo tudi na diagramu (5.1c), kjer si ogledamo povprečja uspešnosti za posamezno podatkovno bazo - $\overline{CA}$. Čeprav je zaradi majhnega števila podatkovnih baz (7) ta test precej manjše moči, nas zanima, kakšna je razvrstitev metod v tem primeru. Motivacija je v tem, da so lahko ciljne spremenljivke



(a) Primerjava uspešnosti več različic metode kNN.

(b) Primerjava uspešnosti metod PCR, PCT in kNN.

(c) Podobno kot 5.1b, a tu metode primerjamo glede na $\overline{CA}$, povprečno uspešnost na bazi.

Slika 5.1: Diagrami povprečnih rankov za problem večciljne klasifikacije; klasifikacijske točnosti vseh metod po posamezni ciljni spremenljivki so primerjane med seboj z Nemenyijevem testom. Metode, ki niso značilno različne ($\alpha = 0.05$) po uspešnosti, so povezane.

znotraj ene podatkovne baze korelirane in bi metoda, ki se na takšni podatkovni bazi obnese dobro, v primeru večjega števila ciljnih spremenljivk pridobila na povprečnem rangu. Kljub temu se razvrstitev metod po uspešnosti tudi v tem primeru ne spremeni, a zaradi majhne množice podatkovnih baz test ne pokaže značilnih razlik med metodami.

5.4.3 Primerjava metod na problemih večciljne regresije

Slika (5.4.3) prikazuje diagrame primerjav različnih metod z Nemenyijevem testom. Na diagramu (5.2a) primerjamo napako metode glede na različne vrednosti k. Izkaže se prednost metode pri uporabi višjih vrednosti k. Naslednji diagram (5.2b) primerja metodo kNN z metodama PCR in PCT. Kot je razvidno iz diagrama se najboljše izkaže PCR, ki pa ni značilno boljša od 15-NN. Od obeh je značilno slabša metoda PCT, od te pa (pričakovano) tudi 1-NN. S podobno motivacijo kot v primeru klasifikacije, opravimo še test podoben (5.2c), le da metode primerjamo glede na povprečne napake na bazi - $\overline{RRMSE}$. Vrstni red uvrstitve metode se ohrani; zaradi majhne množice podatkovnih baz (le 5) pa test ne pokaže značilnih razlik med metodami.

Tabela 5.3: Uspešnosti metod PCR, PCT in kNN za probleme večciljne regresije. Podane vrednosti so *RRMSE*. Na levi navajamo podatkovne baze in posamezne cilje spremenljivke. Vrednosti za PCR in PCT so povzete iz [6].

	PCR	PCT	1-NN	3-NN		5-NN		15-NN	
					d^{-1}		d^{-1}		d^{-1}
edm	0.92	0.72	0.85	0.75	0.74	0.77	0.73	0.77	0.70
d-flow	1.04	0.69	0.75	0.69	0.66	0.74	0.68	0.79	0.66
d-gap	0.80	0.76	0.94	0.81	0.80	0.80	0.78	0.75	0.73
sigmea-real	0.60	0.61	0.81	0.66	0.65	0.64	0.64	0.66	0.63
mfo	0.70	0.62	0.88	0.73	0.72	0.67	0.68	0.66	0.65
mso	0.50	0.61	0.73	0.58	0.58	0.62	0.60	0.66	0.62
sigmea-sim	0.40	0.03	0.54	0.38	0.43	0.36	0.41	0.34	0.40
disp-rate	0.44	0.03	0.61	0.44	0.49	0.41	0.47	0.39	0.45
disp-seeds	0.35	0.03	0.45	0.32	0.35	0.30	0.34	0.29	0.33
solar-flare	1.10	1.00	1.25	1.12	1.18	1.06	1.16	1.01	1.16
c-class	1.02	0.99	1.11	1.07	1.10	1.02	1.06	1.00	1.06
m-class	1.05	0.98	1.12	1.08	1.15	1.05	1.15	1.00	1.14
x-class	1.24	1.02	1.47	1.21	1.28	1.10	1.26	1.03	1.26
water-quality	0.99	0.96	1.28	1.06	1.06	1.01	1.01	0.96	0.96
clad-sp	0.99	0.99	1.27	1.05	1.05	0.99	0.99	0.95	0.95
gong-inc	1.01	0.99	1.40	1.12	1.13	1.08	1.08	1.02	1.02
oedo-sp	0.99	0.99	1.31	1.06	1.06	1.02	1.02	0.97	0.97
tige-ten	0.97	0.93	1.21	1.04	1.03	0.98	0.98	0.95	0.94
melo-var	1.00	0.98	1.37	1.07	1.08	1.02	1.03	0.96	0.96
nitz-pal	0.96	0.90	1.28	1.03	1.04	0.98	0.99	0.93	0.92
audo-cha	1.00	0.98	1.37	1.09	1.10	1.05	1.05	0.98	0.98
erpo-oct	0.98	0.95	1.24	1.03	1.04	0.96	0.97	0.95	0.95
gamm-foss	0.96	0.93	1.19	0.99	0.99	0.93	0.93	0.90	0.90
baet-rhod	0.99	0.98	1.25	1.05	1.05	0.99	0.99	0.95	0.94
hydro-sp	1.00	0.97	1.29	1.09	1.09	1.04	1.04	0.96	0.97
rhya-sp	0.99	0.95	1.29	1.08	1.08	1.03	1.03	0.97	0.97
simu-sp	1.03	1.00	1.26	1.09	1.09	1.06	1.06	1.00	1.00
tubi-sp	0.94	0.89	1.18	1.04	1.03	0.97	0.97	0.92	0.92
povprečje	0.91	0.83	1.10	0.92	0.93	0.88	0.90	0.85	0.86

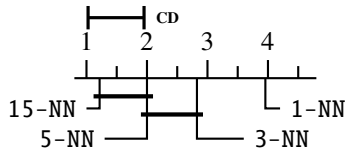
Tabela 5.4: Wilcoxonov test ($\alpha = 0.05$) uspešnosti metode brez in z obteževanja glasovanja (d^{-1}) za različne vrednosti k. Da je ena od metod značilno boljša mora biti manjša vsota rangov (v stolpcu) manjša ali enaka 170 v primeru klasifikacije in 73 v primeru regresije. Značilno boljše metode označimo z $\oplus$.

	3	5	15
	377	$\oplus$ **406.5**	265
d^{-1}	184	154.5	**296**

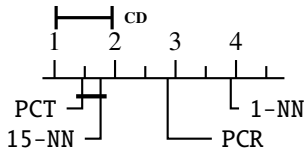
(a) Primerjava na klasifikacijskih podatkovnih bazah.

	3	5	15
	90	**86**	**134**
d^{-1}	186	190	142

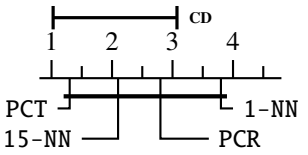
(b) Primerjava na regresijskih podatkovnih bazah.



(a) Primerjava uspešnosti metode kNN za različne vrednosti k.



(b) Primerjava metode kNN napram PCR in PCT.



(c) Podobno kot 5.2b, a tu uporabimo $\overline{RRMSE}$, povprečje napake na bazi.

Slika 5.2: Diagrami povprečnih rankov za problem večciljne regresije; vrednosti napak *RMSE* vseh metod po posamezni ciljni spremenljivki so primerjane med seboj z Nemenyijevem testom. Metode, ki niso značilno različne ($\alpha = 0.05$) po uspešnosti, so povezane. Pri vseh navedenih metodah kNN je uporabljeno obteževanje glasovanja d^{-1}.

5.5 Vpliv obteževanja atributov na uspešnost napovedovanja

Kot smo omenili v 2.1.1, obtežitev atributov spremeni prostor iskanja najbližjih sosedov tako, da razširi dimenzije, ki nosijo več informacij in skrči tiste, ki jih nosijo manj. V teoriji naj bi metoda tako našla učne primere, ki so neznanemu primeru bližje po lastnostih. Zanima nas kako obteževanje atributov vpliva na večciljno klasifikacijo in regresijo na podatkih iz realne domene. Postavimo hipotezo, da se bo metoda pri uporabi obteževanja atributov obnesla značilno bolje.

5.5.1 Primerjava in rezultati primerjav

Vse teste opravimo z 10-kratno navzkrižno validacijo. Za vsako podatkovno bazo preizkusimo metodo z in brez obtežitve glasovanja ter z in brez uporabe obtežitve atributov. Za obtežitev uporabimo `RF` metodo s 100 naključnimi drevesi. Iz rezultatov sestavimo $\overline{CA}$ oz. $\overline{RMSE}$ ocene, katere primerjamo v nadaljevanju. V poskusu uporabimo metodo `3-NN`.

Rezultati primerjav so prikazani v tabeli 5.5 in 5.6, od koder je že s hitrim pregledom razvidno, da večje razlike pri uporabi obteževanja atributov ni zaznati.

Tabela 5.5 prikazuje rezultate večciljne klasifikacije za različne nastavitve metode `kNN`. Prva dva stolpca se nanašata na rezulate z obteženim glasovanjem d^{-1}, pri drugih dveh pa imajo vsi glasovi enako težo. Pri vseh iščemo 3 sosede glede na evklidsko razdaljo. Stolpca (1, 3) označena z w = `RF` prikazujeta rezultate, kjer smo uporabili obtežitev atributov. Primerjamo razliko v delovanju metode z in brez uporabe obteževanja atributov. Z Wilcoxonovim testom ovržemo našo hipotezo, da obtežitev atributov izboljša večciljno klasifikacijo. Hipotezo testiramo posebej za teste z in brez uporabe obteženega glasovanja. Podrobnejši rezultati so prikazani v tabeli 5.8a.

Podobno testiramo hipotezo še na podatkih regresije. Tudi tu uporabimo Wilcoxonov test, ki ovrže tudi to hipotezo. Podrobnosti testa prikažemo v tabeli 5.8b.

Tabela 5.5: Primerjava uspešnosti metode kNN za večciljno klasifikacijo z in brez obtežitve atributov z metodo RF. Primerjamo uspešnost brez in z d^{-1} obteževanjem glasovanja.

	$d = d^{-1}$			
		$w = \mathrm{RF}$		$w = \mathrm{RF}$
monks	**0.753086**	0.732253	**0.746914**	0.729938
sigmeareal-dis	**0.729498**	**0.729498**	**0.731334**	0.730722
sigmeasim-dis	**0.921200**	0.921152	0.867622	**0.867766**
scene	0.903684	**0.903751**	0.902991	**0.903060**
solar-flare1-dis	**0.896801**	0.893705	**0.912281**	0.909185
thyroid	0.964364	**0.964379**	0.964597	**0.964613**
water-quality-nom	0.687129	**0.687332**	0.687197	**0.687399**
yeast	**0.786778**	0.786483	**0.786690**	0.786394
edm1-dis	0.785714	0.785714	0.785714	0.785714
emotions	**0.723440**	0.723159	**0.723159**	0.722877

Tabela 5.6: Primerjava uspešnosti metode kNN za večciljno regresijo z in brez obtežitve atributov z metodo RF. Primerjamo uspešnost brez in z d^{-1} obteževanjem glasovanja.

	d^{-1}			
		$w = \mathrm{RF}$		$w = \mathrm{RF}$
solar-flare1-num	0.435367	**0.430767**	0.392567	**0.391700**
soilresilience	**8.626762**	8.682638	**8.243038**	8.327800
edm1	0.372650	**0.367200**	0.386700	**0.380600**
sigmeareal	**3.755250**	3.756500	3.859500	**3.858950**
sigmeasim	**0.171550**	**0.171550**	**0.201900**	**0.201900**
solar-flare2-num	0.464967	**0.460167**	**0.439033**	0.450033
water-quality	**1.262179**	1.262507	**1.263986**	1.264629
collembolav2	**14.33596**	14.33816	14.31966	14.31966
soil_quality	**26946.52**	28037.51	**27197.96**	27738.95

5.6 Povzetek

Zaključimo lahko, da se metoda najbližjega soseda na problemih večciljne klasifikacije in regresije ne obnese značilno slabše od kompleksnejših metod, kot sta PCR in PCT. Celo več, napram metodi PCR, se 15-NN v primeru regresije izkaže kot značilno boljša. Iz rezultatov je tudi videti, da bi se lahko po uspešnosti s pravili in drevesi kosala že metoda 5-NN (kar je pomembno, ko želimo varčevati s procesorskim časom).

Pokažemo tudi, da uporaba obteževanja atributov, vsaj z metodo RF in pri iskanju 3 sosedov, ni smiselna. Večciljna regresija in klasifikacija se odrežeta celo bolje brez obteževanja (a ne značilno) tako v primeru sočasnega obteževanja glasov kot brez njega.

Tabela 5.7: Wilcoxonov test ($\alpha = 0.05$) obeh klasifikatorjev 3-NN z in brez obteževanja atributov RF za primer brez obteževanja glasovanja in primer z obteževanje glasovanja d^{-1}. Da bi bila razlika značilna, bi se morala manjša vsota v vrstici spustiti pod 5 v primeru klasifikacije (9 podatkovnih baz) oz. pod 8 v primeru regresije (10 podatkovnih baz).

	$w = 1$	$w =$ RF
	39.5	15.5
d^{-1}	40.5	14.5

(a) *Večciljna klasifikacija:* brez obteževanja atributov je klasifikator rahlo boljši, a ne značilno.

	$w = 1$	$w =$ RF
	18.5	26.5
d^{-1}	15.5	29.5

(b) *Večciljna regresija:* brez obteževanja atributov je napovedovanje rahlo boljše, a ne značilno.

Tabela 5.8: Wilcoxonov test ($\alpha = 0.05$) napovedne uspešnosti metode kNN napram PCT in PCR za klasifikacijo in regresijo. Da je ena od metod značilno boljša mora biti manjša vsota rangov (v vrstici) manjša ali enaka 170 v primeru klasifikacije in 73 v primeru regresije. Značilno boljše metode označimo z $\oplus$.

kNN : PCT	**350**	211
kNN : PCR	**334.5**	226.5

(a) Primerjava na klasifikacijskih podatkovnih bazah.

kNN : PCT	$\oplus$ **22**	254
kNN : PCR	192.5	**83.5**

(b) Primerjava na regresijskih podatkovnih bazah.

Poglavje 6

Napovedovanje hierarhičnih večznačnih spremenljivk

V tem razdelku nas zanima, kakšna je uspešnost metode kNN pri hierarhični večznačni klasifikaciji. Našo metodo preizkusimo na podatkovnih bazah iz funkcionalne genomike z uporabo klasifikacijskih shem FunCat in GO. Kot primerjalni klasifikator si izberemo metodi HMC in SC, ki sta v Clus sistemu implementirana s pomočjo dreves PCT. Obe metodi smo podrobneje predstavili že v razdelku 2.2.2. V [3] so opravili primerjavo metod HMC, HSC in SC in pokazali, da je metoda HMC značilno boljša od obeh ostalih. Značilna razlika je bila pokazana z Wilcoxonovim testom med metodo HMC in SC ter HMC in HSC.

Iz omenjenega članka povzamemo rezultate metode HMC kot najuspešnejše in SC kot najpreprostejše. Z enakimi parametri poskusa in nad istimi podatki preizkusimo tudi metodo kNN, za različne vrednosti k in z ter brez uporabe obtežitve glasovanja.

V koncu razdelka rezultate prikažemo ter primejamo med seboj. Pokažemo, kakšna je uspešnost metode kNN v primerjavi z metodama iz članka [3] ter katere nastavitve uporabiti za dosego boljše uspešnosti metode kNN.

6.1 Uporabljene podatkovne baze

Uporabimo 12 podatkovnih baz, ki jih uporabi Clare v [33]. Podatkovne baze opisujejo različne gene genoma kvasovke in njihove funkcije. Grobe lastnosti baz prikazuje tabela 6.1. V nadaljevanju povzamemo predstavitve podatkov po [3], kjer so navedene tudi reference na podrobnejše opise in uporabe posameznih podatkovnih baz.

Tabela 6.1: Lastnosti podatkovnih baz uporabljenih v hierarhični večznačni klasifikaciji

	Podatki	n	d
D_1	Sequence	3932	478
D_2	Phenotype	1592	69
D_3	Secondary struture	3851	19.628
D_4	Homology search	3867	47.034
D_5	Spellman et al.	3766	77
D_6	Roth et al.	3764	27
D_7	DeRisi et al.	3733	63
D_8	Eisen et al.	2425	79
D_9	Gasch et al. 1	3773	173
D_{10}	Gasch et al. 2	3788	52
D_{11}	Chu et. al.	3711	80
D_{12}	All microarray	3788	551

D_1 zajema statistiko zaporedja, ki je odvisna od zaporedja amino kisline proteina, ki ga gen kodira. Statistika zajema razmerje amino kislin, dolžino zaporedja, molekulsko težo in sposobnost topnosti v vodi.

D_2 vsebuje podatke fenotipa, ki opisujejo rast ali upad rasti v mutantih z odstranjenim genom, ki nas zanima. Organizem, ki je dobljen z odstranitvijo ali onesposobljenjem gena je gojen v različnih okoljih in tako testiran, na katere substance je odporen ali občutljiv. Atributi so diskretni, podatki pa nepopolni (vsi mutanti niso bili testirani v vseh pogojih).

D_3 hrani lastnosti, izračunane iz sekundarne strukture proteinov kvasovke. Struktura ni znana za vse gene kvasovke, vendar je lahko določena iz zaporedja proteina z dovolj dobro natančnostjo (za kar se uporabi program Prof). Clare [33] je izvedla predpripravo podatkov in poiskala najpogostejše vzorce v strukturi - ti so potem vključeni v D_3 kot binarni atributi.

D_4 hrani za vsak gen informacije o drugih homolognih genih. Homolognost je definirana glede na podobnost zaporedja. Zaporedja genov so bila primerjana z uporabo orodja PSI-BLAST skupaj z ostalimi zaporedji in z geni navedenimi v SwissProt 39. Za vsakega od homolognih genov so bile nato izločene nekatere značilnosti (ključne besede, dolžina zaporedja, ...). Podobno kot v D_3 je Clare tudi tu predprocesirala podatke in tako pridobila binarne atribute.

D_{5-12} predstavljajo podatkovne baze, pridobljene pri različnih merjenjih vpliva genov z uporabo mikromreže.

V vsaki podatkovni bazi predstavimo funkcije genov z uporabo klasifikacijskih shem FunCat in GO. Tako pridobimo 24 podatkovnih baz.

6.2 Ocenjevanje uspešnosti metode

Ker je naš cilj primerjava metode z drevesi za napovedno razvrščanje, kot so ocenjena v [3], privzamemo po članku tudi način ocenjevanja uspešnosti klasifikatorja.

Želimo si mere, ki bi bila neodvisna od izbire parametra p - parametra, ki določa, kdaj primerek klasificiramo v izbrani razred in kdaj ne.

Mere uspešnosti bomo sestavili iz krivulje natančnosti-odzivnosti (PRC), definirane za problem binarne klasifikacije. Definirajmo natančnost kot $Prec = \frac{TP}{TP+FP}$, odzivnost pa kot $Rec = \frac{TP}{TP+FN}$, kjer je

TP število primerov, ki so bili pravilno klasificirani kot pozitivni

FP število primerov, ki so bili nepravilno klasificirani kot pozitivni

TN število primerov, ki so bili nepravilno klasificirani kot negativni

Opazimo, da tako *Prec* kot *Rec* nista odvisni od števila primerov, ki so bili pravilno klasificirani kot negativni. Za našo postavitev poskusa ima to smisel, saj večina primerov pripada le majhni množici razredov (posamezni geni imajo malo funkcij glede na nabor vseh funkcij v klasifikacijski shemi). Nagrajevanje klasifikatorja za pravilno negativno klasifikacijo bi se zato izkazalo kot preveč optimistično. PR krivulja riše graf, kjer je *Prec* odvisen od *Rec*. S spreminjanjem parametra p lahko dosežemo vse vrednosti *Rec* (te so na intervalu [0..1]). Vsaka točka na krivulji torej ustreza določeni vrednosti parametra p. Mero uspešnosti posameznega razreda definiramo kot površino pod PR krivuljo. S tem dobimo eno samo mero, ki opisuje uspešnost klasifikatorja za določen razred, dobimo pa tudi neparametrično mero, saj upošteva vse možne vrednosti parametra p.

Še vedno pa ostane vprašanje, kako združiti ocene posameznih razredov v eno skupno oceno. Glede na to ločimo tri mere:

$\overline{\text{AUPRC}}$ $= (\sum_i AUPRC_i)/|C|$, kjer je $AUPRC_i$ površina pod krivuljo za razred i.

$\overline{\text{AUPRC}}_w$ $= \sum_i w_i \cdot AUPRC_i$, kjer je w_i normalizirana pogostost razreda c_i v podatkih. Mera je podobna $\overline{\text{AUPRC}}$, le da so razredi drugače obteženi. Ideja za uporabo tovrstnih uteži je v tem, da so pogosteje uporabljeni razredi, pomembnejši.

AU($\overline{\text{PRC}}$) Prejšnji dve meri vzameta povprečje površin pod krivuljama. AU($\overline{\text{PRC}}$) pa vzame površino pod povprečno krivuljo. V ta namen definiramo

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i}, \quad \text{in} \quad \overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i},$$

kjer i teče po vseh razredih. S takšno definicijo natančnosti in odzivnosti dobimo za vse parametre p povprečno krivuljo. Površina, ki jo obdaja z abciso, je iskana mera.

6.3 Primerjava

Za obe klasifikacijski shemi poženemo metodo `kNN` na vseh 12 podatkovnih bazah za $k \in \{1, 3, 5\}$ z in brez obtežitve glasov (skupaj 120 testov, ker obtežitev ne vpliva

na 1-NN). Točke na krivulji ocenimo le za $p \in [0, 0.02, 0.04, \ldots, 0.96, 0.98, 1]$ in ostale vrednosti interpoliramo. Za vsakega od testov izračunamo ocene $\overline{\text{AUPRC}}$, $\overline{\text{AUPRC}}_w$ in AU($\overline{\text{PRC}}$).

Podatkovna baza je bila razdeljena na učno množico, ki jo je sestavljalo 2/3 primerov in testno (1/3). Takšno razdelitev uporabimo tudi v naših testih. V [15] pa iz učne množice odstranijo 1/3 primerov in jih uporabijo v validacijski množici za optimizacijo uspešnosti metode. Opomnimo, da lahko slednja razlika v testih vpliva na razliko v uspešnosti posameznih metod, čeprav verjamemo, da nima večjega vpliva.

6.4 Rezultati primerjav

Tabela 6.2: Vpliv obtežitve glasovanja na natančnost hierarhične večznačne klasifikacije za $k \in \{3, 5\}$ pri uporabi klasifikacijske sheme FunCat. Opazno je izboljšane uspešnosti pri uporabi obtežitev glede na vse tri uporabljene mere.

baza	$k = 3$		$k = 5$		$k = 3$		$k = 5$		$k = 3$		$k = 5$	
		d^{-1}		d^{-1}		d^{-1}		d^{-1}		d^{-1}		d^{-1}
cellcycle	0.045	0.048	0.052	0.055	0.158	0.164	0.173	0.179	0.107	0.112	0.142	0.147
church	0.024	0.026	0.026	0.027	0.115	0.118	0.122	0.124	0.063	0.062	0.086	0.083
derisi	0.029	0.031	0.030	0.034	0.122	0.124	0.128	0.130	0.070	0.072	0.094	0.094
eisen	0.079	0.093	0.078	0.094	0.200	0.208	0.200	0.214	0.151	0.158	0.175	0.187
expr	0.037	0.043	0.039	0.045	0.132	0.136	0.138	0.143	0.082	0.084	0.104	0.108
gasch1	0.070	0.079	0.074	0.086	0.183	0.189	0.193	0.202	0.135	0.140	0.167	0.173
gasch2	0.049	0.060	0.060	0.068	0.157	0.164	0.174	0.180	0.108	0.114	0.144	0.149
hom	0.112	0.150	0.109	0.155	0.252	0.278	0.263	0.289	0.211	0.234	0.239	0.264
pheno	0.032	0.034	0.034	0.034	0.123	0.124	0.127	0.126	0.070	0.069	0.093	0.086
seq	0.030	0.042	0.030	0.043	0.124	0.137	0.124	0.142	0.072	0.082	0.088	0.103
spo	0.036	0.042	0.035	0.041	0.127	0.131	0.131	0.135	0.075	0.077	0.095	0.099
struc	0.047	0.063	0.048	0.068	0.151	0.163	0.156	0.172	0.102	0.112	0.123	0.139
μ	0.049	0.059	0.051	0.063	0.154	0.161	0.160	0.170	0.104	0.110	0.129	0.136
σ	0.026	0.035	0.025	0.036	0.040	0.046	0.042	0.048	0.044	0.049	0.047	0.053

(a) $\overline{\text{AUPRC}}$ **(b)** $\overline{\text{AUPRC}}_w$ **(c)** AU($\overline{\text{PRC}}$)

Tabela 6.2 vsebuje rezultate uspešnosti metode kNN na podatkih z uporabljeno klasifikacijsko shemo FunCat za $k \in \{3, 5\}$ z in brez uporabe obteževanja glasovanja d^{-1}. Podobno tabela 6.3 vsebuje rezultate na podatkih z uporabljeno GO klasifikacijsko shemo. Spomnimo se, da obtežitev glasovanja vpliva le za $k > 1$. Opazimo, da se metoda v večini primerov obnese boljše, če uporabimo obteževanje glasov.

Tabela 6.3: Vpliv obtežitve glasovanja na natančnost hierarhične večznačne klasifikacije za $k \in \{3, 5\}$ pri uporabi klasifikacijske sheme GO. Opazno je izboljšanje uspešnosti pri uporabi obtežitev glede na vse tri uporabljene mere.

baza	$k=3$		$k=5$		$k=3$		$k=5$		$k=3$		$k=5$	
		d^{-1}		d^{-1}		d^{-1}		d^{-1}		d^{-1}		d^{-1}
cellcycle	0.035	0.041	0.037	0.045	0.338	0.344	0.351	0.356	0.319	0.324	0.378	0.382
church	0.018	0.020	0.017	0.020	0.298	0.301	0.308	0.307	0.277	0.276	0.329	0.333
derisi	0.019	0.023	0.019	0.024	0.305	0.306	0.307	0.309	0.285	0.286	0.335	0.336
eisen	0.053	0.065	0.050	0.064	0.377	0.384	0.382	0.393	0.369	0.374	0.414	0.421
expr	0.025	0.030	0.025	0.031	0.315	0.319	0.318	0.324	0.297	0.300	0.348	0.352
gasch1	0.051	0.060	0.053	0.067	0.363	0.369	0.373	0.381	0.346	0.351	0.400	0.405
gasch2	0.032	0.043	0.044	0.049	0.314	0.343	0.350	0.355	0.138	0.326	0.379	0.382
hom	0.111	0.149	0.103	0.150	0.409	0.432	0.419	0.440	0.396	0.415	0.442	0.459
pheno	0.022	0.023	0.022	0.025	0.293	0.298	0.298	0.297	0.270	0.274	0.311	0.302
seq	0.025	0.039	0.024	0.040	0.316	0.329	0.317	0.333	0.294	0.302	0.344	0.349
spo	0.022	0.026	0.023	0.028	0.307	0.310	0.308	0.313	0.288	0.290	0.338	0.341
struc	0.037	0.055	0.036	0.059	0.331	0.343	0.334	0.350	0.313	0.322	0.361	0.372
μ	0.038	0.048	0.038	0.050	0.331	0.340	0.339	0.347	0.299	0.320	0.365	0.370
σ	0.026	0.035	0.024	0.035	0.035	0.040	0.037	0.042	0.064	0.042	0.039	0.043

(a) $\overline{\text{AUPRC}}$ **(b)** $\overline{\text{AUPRC}}_w$ **(c)** AU($\overline{\text{PRC}}$)

Tabela 6.4: Odstotek izboljšanja v uspešnosti hierarhične večznačne klasifikacije pri uporabi obteževanja glasov za $k \in \{3, 5\}$ glede na vse tri uporabljene mere. Opazimo izboljšanje v vseh primerih.

	FunCat		GO	
	$k=3$	$k=5$	$k=3$	$k=5$
$\overline{\text{AUPRC}}$	20.4	23.5	26.3	32.6
$\overline{\text{AUPRC}}_w$	4.5	6.5	2.7	2.4
AU($\overline{\text{PRC}}$)	5.8	5.4	7	1.4

Tabela 6.4 prikazuje procentualno izboljšanje metode v primeru obteževanja. Pri $\overline{\text{AUPRC}}$ meri so odstotki nekoliko višji predvsem zaradi manjših vrednosti. Ker obteževanje glasov porabi zanemarljivo malo dodatnega procesorskega časa in ker je izboljšanje uspešnosti metode očitno (četudi majhno), je obteževanje smiselno uporabiti. Zato v poznejših primerjavah povsod opazujemo le rezultate, pridobljene z obteževanjem glasovanja.

Tabela 6.5 vsebuje rezultate metode `kNN` za obe klasifikacijski shemi in vse tri ocene uspešnosti. Opazimo, da se metoda skoraj vedno najbolje obnese pri $k = 5$, čeprav tudi pri $k = 3$ rezultati niso dosti slabši. Zaradi relativno majhnih razlik med $k = 3$ in $k = 5$ lahko predvidevamo, da bi naraščanje uspešnosti metode za večje k

hitro prenehalo oz. bi uspešnost celo začela upadati.

Poglejmo si primerjavo metode kNN napram HMC in SC. Tabeli 6.6 in 6.7 vsebujeta tako rezultate iz članka [3] (HMC in SC) kot iz naših testov (1-NN in 5-NN). Vse 4 različne metode (HMC, SC, 1-NN 5-NN) so primerjane glede na $\overline{\text{AUPRC}}$, $\overline{\text{AUPRC}}_w$, AU($\overline{\text{PRC}}$) ocene uspešnosti napovedovanja.

Klasifikatorje primerjamo z Nemenyijevem testom. Primerjamo 4 klasifikatorje na 12 podatkovnih bazah za 2 klasifikacijski shemi in 3 različne načine ocenjevanja uspešnosti napovedovanja. Poglejmo si sliko 6.4, ki prikazuje diagrame za klasifikacijsko shemo FunCat. Iz nje razberemo

5-NN $\sim$ HMC Ker 5-NN in HMC nista značilno različna, lahko v splošnem uporabimo metodo 5-NN namesto HMC. Upoštevajoč kompleksnost klasifikatorja HMC in enostavnost 5-NN, gre za precej nepričakovan rezultat

kNN $\sim$ SC Tudi tu gre za podobno nepričakovan rezultat. Kljub enostavnosti klasifikatorja SC napram HMC, je ta še vedno precej kompleksnejši od 1-NN in celo 5-NN. Od prvega je značilno boljši le pri $\overline{\text{AUPRC}}_w$ oceni. 5-NN pa je glede na $\overline{\text{AUPRC}}$ celo značilno boljši; glede na $\overline{\text{AUPRC}}_w$ in AU($\overline{\text{PRC}}$) pa sicer boljši, a ne značilno.

1-NN $\nsim$ 5-NN Za $\overline{\text{AUPRC}}$ in $\overline{\text{AUPRC}}_w$ sta metodi različni, s čimer potrdimo smotrnost uporabe 5-ih sosedov pri glasovanju.

Podobne zaključke dobimo tudi iz diagrama 6.4, ki prikazuje klasifikacijo pri uporabi klasifikacijske sheme GO:

5-NN $\sim$ HMC Tudi pri uporabi klasifikacijske sheme GO, se klasifikatorja ne razlikujeta značilno.

1-NN $\sim$ SC Klasifikatorja nista značilno različna za nobeno od ocen uspešnosti napovedovanja (pri FunCat klasifikaciji sta značilno različni za $\overline{\text{AUPRC}}_w$).

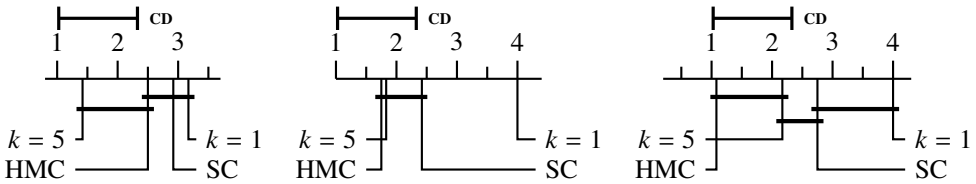
Tabela 6.5: Uspešnost metode na posameznih podatkovnih bazah za $k \in \{1, 3, 5\}$ glede na vse tri uporabljene mere. V vseh primerih uporabimo tudi obteževanje glasovanja d^{-1}. Opazno je izboljšanje pri glasovanju večih sosedov.

baza	$\overline{\text{AUPRC}}$			$\overline{\text{AUPRC}}_w$			AU($\overline{\text{PRC}}$)		
	1	3	5	1	3	5	1	3	5
cellcycle	0.036	0.048	0.055	0.133	0.164	0.179	0.056	0.112	0.147
church	0.022	0.026	0.027	0.107	0.118	0.124	0.034	0.062	0.083
derisi	0.027	0.031	0.034	0.113	0.124	0.130	0.040	0.072	0.094
eisen	0.063	0.093	0.094	0.156	0.208	0.214	0.074	0.158	0.187
expr	0.030	0.043	0.045	0.120	0.136	0.143	0.045	0.084	0.108
gasch1	0.052	0.079	0.086	0.145	0.189	0.202	0.065	0.140	0.173
gasch2	0.039	0.060	0.068	0.133	0.164	0.180	0.057	0.114	0.149
hom	0.089	0.150	0.155	0.193	0.278	0.289	0.110	0.234	0.264
pheno	0.029	0.034	0.034	0.116	0.124	0.126	0.038	0.069	0.086
seq	0.031	0.042	0.043	0.120	0.137	0.142	0.045	0.082	0.103
spo	0.032	0.042	0.041	0.115	0.131	0.135	0.041	0.077	0.099
struc	0.046	0.063	0.068	0.129	0.163	0.172	0.054	0.112	0.139
μ	0.041	0.059	0.063	0.132	0.161	0.170	0.055	0.110	0.136
σ	0.019	0.035	0.036	0.024	0.046	0.048	0.021	0.049	0.053

(a) Napovedne uspešnosti pri uporabi klasifikacijske sheme FunCat.

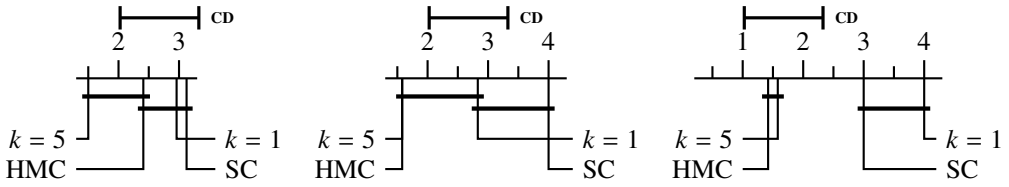
baza	$\overline{\text{AUPRC}}$			$\overline{\text{AUPRC}}_w$			AU($\overline{\text{PRC}}$)		
	1	3	5	1	3	5	1	3	5
cellcycle	0.029	0.041	0.045	0.316	0.344	0.356	0.139	0.324	0.382
church	0.015	0.020	0.020	0.291	0.301	0.307	0.106	0.276	0.333
derisi	0.019	0.023	0.024	0.296	0.306	0.309	0.115	0.286	0.336
eisen	0.043	0.065	0.064	0.339	0.384	0.393	0.172	0.374	0.421
expr	0.021	0.030	0.031	0.305	0.319	0.324	0.125	0.300	0.352
gasch1	0.040	0.060	0.067	0.331	0.369	0.381	0.157	0.351	0.405
gasch2	0.032	0.043	0.049	0.314	0.343	0.355	0.138	0.326	0.382
hom	0.092	0.149	0.150	0.371	0.432	0.440	0.208	0.415	0.459
pheno	0.020	0.023	0.025	0.290	0.298	0.297	0.125	0.274	0.302
seq	0.025	0.039	0.040	0.309	0.329	0.333	0.128	0.302	0.349
spo	0.021	0.026	0.028	0.299	0.310	0.313	0.118	0.290	0.341
struc	0.038	0.055	0.059	0.315	0.343	0.350	0.139	0.322	0.372
μ	0.033	0.048	0.050	0.315	0.340	0.347	0.139	0.320	0.370
σ	0.021	0.035	0.035	0.023	0.040	0.042	0.028	0.042	0.043

(b) Napovedne uspešnosti pri uporabi klasifikacijske sheme GO.



(a) $\overline{\text{AUPRC}}$ **(b)** $\overline{\text{AUPRC}}_w$ **(c)** AU($\overline{\text{PRC}}$)

Slika 6.1: Diagram povprečnih rangov v Nemenyijevem testu in značilne različnosti. Klasifikatorji, ki niso povezani z odebeljeno črto (so narazen vsaj za kritično razdaljo **CD**), so značilno različni. Manjši rang pomeni uspešnejši klasifikator. Rezultati veljajo pri uporabi klasifikacijske sheme **FunCat**.



(a) $\overline{\text{AUPRC}}$ **(b)** $\overline{\text{AUPRC}}_w$ **(c)** AU($\overline{\text{PRC}}$)

Slika 6.2: Diagram povprečnih rangov v Nemenyijevem testu in značilne različnosti. Klasifikatorji, ki niso povezani z odebeljeno črto (so narazen vsaj za kritično razdaljo **CD**), so značilno različni. Manjši rang pomeni uspešnejši klasifikator. Rezultati veljajo pri uporabi klasifikacijske sheme **GO**.

Tabela 6.6: Napovedne uspešnosti metod kNN ($k \in \{1, 5\}$) HCM in SC glede na različne mere uspešnosti pri uporabi klasifikacijske sheme FunCat. Podatki za metodi HCM in SC so povzeti iz [3].

baza	$\overline{\text{AUPRC}}$				$\overline{\text{AUPRC}}_w$				AU($\overline{\text{PRC}}$)			
	HMC	SC	$k = 5$	$k = 1$	HMC	SC	$k = 5$	$k = 1$	HMC	SC	$k = 5$	$k = 1$
cellcycle	0.034	0.038	0.055	0.036	0.142	0.146	0.179	0.133	0.172	0.106	0.147	0.056
church	0.029	0.031	0.027	0.022	0.129	0.128	0.124	0.107	0.170	0.128	0.083	0.034
derisi	0.033	0.028	0.034	0.027	0.137	0.122	0.130	0.113	0.175	0.089	0.094	0.040
eisen	0.052	0.055	0.094	0.063	0.183	0.173	0.214	0.156	0.204	0.132	0.187	0.074
expr	0.052	0.050	0.045	0.030	0.179	0.167	0.143	0.120	0.210	0.123	0.108	0.045
gasch1	0.049	0.047	0.086	0.052	0.176	0.153	0.202	0.145	0.205	0.104	0.173	0.065
gasch2	0.039	0.037	0.068	0.039	0.156	0.147	0.180	0.133	0.195	0.119	0.149	0.057
hom	0.089	0.076	0.155	0.089	0.240	0.205	0.289	0.193	0.254	0.153	0.264	0.110
pheno	0.030	0.031	0.034	0.029	0.124	0.127	0.126	0.116	0.160	0.149	0.086	0.038
seq	0.053	0.042	0.043	0.031	0.183	0.154	0.142	0.120	0.211	0.095	0.103	0.045
spo	0.035	0.034	0.041	0.032	0.153	0.139	0.135	0.115	0.186	0.098	0.099	0.041
struc	0.041	0.040	0.068	0.046	0.161	0.152	0.172	0.129	0.181	0.114	0.139	0.054
μ	0.045	0.042	0.063	0.041	0.164	0.151	0.170	0.132	0.194	0.118	0.136	0.055
σ	0.017	0.013	0.036	0.019	0.032	0.023	0.048	0.024	0.026	0.021	0.053	0.021

1-NN $\nsim$ 5-NN S tem potrdimo domnevo iz tabele 6.5.

Za klasifikatorja HMC in 5-NN opravimo še dodaten statistični test. Klasifikatorja nas najbolj zanimata, ker je klasifikator HMC z najboljšo uspešnostjo v omenjenem članku, 5-NN pa najboljši klasifikator pridobljen iz naše metode kNN. Med klasifikatorjema opravimo Wilcoxonov test za vse tri ocene uspešnosti napovedovanja in obe klasifikacijski shemi. Rezultate povzamemo v tabeli 6.8. Številka v stolpcu klasifikatorja predstavlja vsoto rangov v Wilcoxonovem testu. Glede na število podatkovnih baz mora biti, da bi bila klasifikatorja značilno različna, vsota rangov manjša od 13. Opazimo, da sta klasifikatorja enakovredna, razen v primeru ($\overline{\text{AUPRC}}$, GO), kjer doseže 5-NN občutno boljše rezultate in v primeru (AU($\overline{\text{PRC}}$), FunCat), kjer je občutno boljši HMC. Pri ($\overline{\text{AUPRC}}$, FunCat) je 5-NN skoraj značilno boljši (rangi so ravno na meji).

6.5 Povzetek

Opravljeni testi nam dajejo dober vpogled na uspešnost metode kNN za hierarhično večznačno klasifikacijo primerov s področja funkcionalne genomike. Ker je obseg

Tabela 6.7: Napovedne uspešnosti metod kNN ($k \in \{1, 5\}$) HCM in SC glede na različne mere uspešnosti pri uporabi klasifikacijske sheme GO. Podatki za metodi HCM in SC so povzeti iz [3].

baza	$\overline{\text{AUPRC}}$				$\overline{\text{AUPRC}}_w$				AU($\overline{\text{PRC}}$)			
	HMC	SC	$k = 5$	$k = 1$	HMC	SC	$k = 5$	$k = 1$	HMC	SC	$k = 5$	$k = 1$
cellcycle	0.036	0.035	0.045	0.029	0.373	0.279	0.356	0.316	0.386	0.197	0.382	0.139
church	0.021	0.021	0.020	0.015	0.299	0.238	0.307	0.291	0.337	0.316	0.333	0.106
derisi	0.025	0.026	0.024	0.019	0.328	0.262	0.309	0.296	0.358	0.228	0.336	0.115
eisen	0.051	0.052	0.064	0.043	0.389	0.313	0.393	0.339	0.401	0.252	0.421	0.172
expr	0.021	0.020	0.031	0.021	0.335	0.267	0.324	0.305	0.357	0.252	0.352	0.125
gasch1	0.018	0.017	0.067	0.040	0.316	0.247	0.381	0.331	0.348	0.289	0.405	0.157
gasch2	0.019	0.017	0.049	0.032	0.321	0.246	0.355	0.314	0.355	0.218	0.382	0.138
hom	0.036	0.031	0.150	0.092	0.362	0.294	0.440	0.371	0.380	0.270	0.459	0.208
pheno	0.030	0.026	0.025	0.020	0.353	0.282	0.297	0.290	0.371	0.239	0.302	0.125
seq	0.024	0.023	0.040	0.025	0.347	0.278	0.333	0.309	0.365	0.267	0.349	0.128
spo	0.026	0.020	0.028	0.021	0.324	0.254	0.313	0.299	0.352	0.213	0.341	0.118
struc	0.029	0.028	0.059	0.038	0.353	0.284	0.350	0.315	0.368	0.249	0.372	0.139
μ	0.028	0.026	0.050	0.033	0.342	0.270	0.347	0.315	0.365	0.249	0.370	0.139
σ	0.009	0.010	0.035	0.021	0.026	0.022	0.042	0.023	0.018	0.034	0.043	0.028

Tabela 6.8: Vsota rangov klasifikatorjev HMC in kNN v Wilcoxonovem testu za $\alpha = 0.05$. Z $\oplus$ označimo klasifikator v primeru, ko je značilno boljši od drugega. Meja za značilno različnost je 13 (manjša od vsot).

	Fun		GO	
	HMC	$k = 5$	HMC	$k = 5$
$\overline{\text{AUPRC}}$	13	***65***	7	$\oplus$**71**
$\overline{\text{AUPRC}}_w$	30	48	41	37
AU($\overline{\text{PRC}}$)	$\oplus$**77**	1	38	40

preizkušenih podatkovnih baz dokaj velik, lahko rezultate projeciramo tudi na druga področja.

Rezultati testov so presenetljivi. HMC metoda zgradi model - drevo, kjer uporablja nekaj modernih pristopov strojnega učenja, na drugi strani pa je metoda kNN preprosta. Rezultati kažejo, da je z uporabo obtežitve glasov in glasovanjem 5-ih sosedov dobimo metodo, ki statistično ni značilno slabša od HMC. Metoda 5-NN je za mero AU($\overline{\text{PRC}}$) celo značilno boljša od SC. Metoda SC ni značilno boljša niti od še enostavnejše 1-NN različice naše metode.

Ugotovili smo, da je smiselno uporabiti glasovanje 5-ih sosedov. V prihodnje bi bilo zanimivo preizkusiti, če je moč metodo še izboljšati z več sosedi (in kje je meja izboljšanja). Prav tako je iz rezultatov očitno, da d^{-1} obtežitev vedno izboljša uspešnost metode v hierarhični večznačni klasifikaciji.

Metoda kNN je torej lahko primeren kandidat tudi za klasifikacijo hierarhičnih večznačnih ciljnih spremenljivk.

Poglavje 7

Napovedovanje kratkih časovnih vrst

V tem razdelku preizkusimo `kNN` metodo za napovedovanje kratkih časovnih vrst. Primerjamo uspešnost za različne vrednosti parametrov metode in ugotovimo najprimernejše vrednosti le-teh. Dalje rezultate primerjamo z uspešnostjo `PCT` dreves na istih podatkih.

7.1 Uporabljene podatkovne baze

Vsaka podatkovna baza zajema spremembe v izražanju genov kvasovke za različne spremembe v okolju; podatkovna baza predstavlja eno od sprememb v okolju, vsak primerek v bazi pa nek gen. Opisni atributi so funkcije gena, ciljna spremenljivka pa časovna vrsta, ki predstavlja spremembe v izražanju gena.

Tako kot v [15] tudi mi uporabimo podatkovne baze iz študije, izvedene s strani

Tabela 7.1: Število primerov v podatkovnih bazah uporabljenih v tem razdelku. Vse baze imajo 933 opisnih atributov.

	n		n
AAstarvGO	5393	HeatShock25to37GO	3391
DiamideGO	5208	HyperOsmoticGO	3391
DTT1GO	2489	MenadioneGO	4925
H2O2GO	4570	NDepletionGO	5315

Gasch in sodelavcev [34]. Uporabimo 8 podatkovnih baz, ki zajemajo različne spremembe v okolju (temperaturne, kemične, pomanjkanje snovi). Podatkovne baze razširimo z opisi genov iz sheme GO (izberemo termine iz klasifikacijske sheme glede na `part_of` in `is_a` relacijo). Uporabimo le termine, ki se pojavijo v vsaj 50 od približno 5000 genov.

7.2 Ocenjevanje uspešnosti metod

Uspešnost napovedovanja časovnih vrst merimo z mero *RMSE*. Upoštevajoč uporabljeno razdaljo D_q, se formula za *RMSE* preoblikuje v

$$RMSE(m) = \sqrt{\frac{1}{|X|}\sum_{x\in X} D_q(C_x, m(x))^2}, \tag{7.1}$$

kjer je $m(x)$ časovna vrsta, ki jo napove m, C_x pa dejanska časovna vrsta za učni primerek x. V primerjavah klasifikatorjev uporabimo mero $\Delta RMSE$ (7.2), ki prikazuje % izboljšanja (ali poslabšanja) glede na privzeti klasifikator.

$$\Delta RMSE = \frac{RMSE - RMSE_{def}}{RMSE_{def}} \tag{7.2}$$

7.3 Primerjava

Na vsaki od podatkovnih baz opravimo 10 kratno navzkrižno validacijo. S tem dobimo verodostojno $\Delta RMSE$ oceno. Uspešnost različnih variant `kNN` metode primerjamo s privzetim modelom in z uspešnostjo metode `PCT`. Te podatke dobimo iz raziskave, ki so jo opravili Slavkov in sodelavci v [15]. Namen raziskave je sicer najti opisne skupine kratkih časovnih vrst s `PCT`, a kot stranski produkt zabeležijo tudi napovedno uspešnost pridobljenega modela. V raziskavi uporabijo vseh 934 atributov, med tem ko v našem testiranju zadnji atribut zaradi napake izpustimo. Posledično rezultati niso popolnoma primerljivi, a lahko s primerjavo $\Delta RMSE$ vrednosti, kjer vzamemo uspešnost privzetega klasifikatorja iz posameznega testa, dobimo vsaj grobo primerjavo obeh modelov.

Tabela 7.2: Odstotek spremembe napovedna napake `kNN` metode za različne vrednosti $k \in \{1, 3, 5, 11\}$ in tri načine obteževanja atributov (brez, d_{-1}, d^{-1}). Negativne vrednosti pomenijo boljšo napovedno uspešnost glede na privzeti klasifikator (manjšo napako).

k	1	3			5			11		
			d^{-1}	d_{-1}		d^{-1}	d_{-1}		d^{-1}	d_{-1}
AAstarvGO	-7.17	-8.88	-8.90	-2.32	-10.32	-10.10	3.94	-12.38	-12.42	12.53
DTT1GO	-6.83	-8.89	-8.95	-2.08	-9.85	-10.27	3.42	-13.67	-13.78	14.14
DiamideGO	-14.33	-16.22	-16.35	-11.54	-17.95	-18.17	-3.65	-20.09	-20.63	8.63
H2O2GO	-4.45	-6.04	-6.63	-2.99	-6.45	-6.96	-0.93	-7.56	-7.38	4.37
HeatShock25to37GO	-6.64	-9.78	-10.02	-2.80	-12.11	-12.16	3.96	-15.93	-15.72	17.47
HyperOsmoticGO	-8.43	-10.36	-10.31	-5.19	-11.38	-11.82	-0.40	-14.06	-14.23	7.43
MenadioneGO	-4.54	-5.23	-5.23	-3.96	-6.58	-6.61	-1.52	-8.40	-8.36	2.81
NDepletionGO	-16.92	-18.33	-18.68	-12.11	-18.64	-18.72	-4.71	-20.71	-21.26	5.48

`kNN` model zgradimo in preizkusimo za $k \in \{1, 3, 5, 11\}$ in 3 različna obteževanja glasovanja. Spomnimo se, da so za $k = 1$ ostala obteževanja enaka privzetemu (enakomernemu).

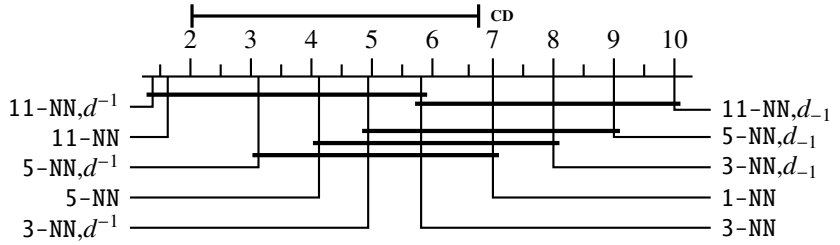
7.4 Rezultati primerjav

Tabela 7.2 prikazuje uspešnost metode glede na različne vhodne parametre. Negativne vrednosti pomenijo izboljšavo v odstotkih glede na privzeti klasifikator. Pozitivne vrednosti ravno obratno - odstotek poslabšanja.

Na sliki 7.4 prikažemo rezultate Nemenyijevega testa za vrednosti iz tabele 7.2. Iz slike lahko potegnemo nekaj zaključkov:

- Obteževanje glasov d_{-1} se obnese slabše od drugih obteževanj. Pri primerjavi navadnega in inverznega obteževanja za posamezne vrednosti k opazimo, da je inverzno vedno boljše od navadnega.

- Razen v primeru d_{-1} glasovanja, lahko trdimo, da je boljše, če uporabimo višje vrednosti k. Za $k = 11$ namreč dobimo precej boljše rezultate napram nižjim vrednostim k.



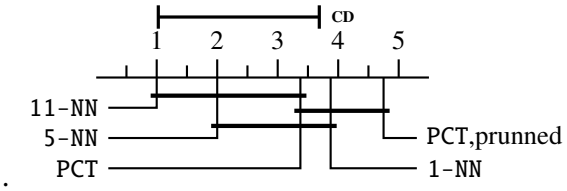
Slika 7.1: Diagram povprečnih rangov različnih `kNN` klasifikatorjev v Nemenyijevem testu in značilne različnosti. Klasifikatorji, ki niso povezani z odebeljeno črto (so narazen vsaj za kritično razdaljo **CD**), so značilno različni. Manjši rang pomeni uspešnejši klasifikator.

Tabela 7.3: Primerjava *RMSE* ocen klasifikatorjev. Prvi 4 stolpci so rezultati našega poskusa, zadnji tri pa poskusa iz [15]. Stolpca `def` označujeta uspešnost privzetega klasifikatorja v posameznem poskusu.

	`def`	`1-NN`	`5-NN`	`11-NN`	`def`	`PCT`	`PCT`, pruned
AAstarvGO	.457	.424	.411	.400	.498	.463	.451
DiamideGO	.464	.397	.379	.368	.483	.430	.419
DTT1GO	.448	.417	.402	.386	.471	.442	.426
H2O2GO	.485	.464	.452	.450	.498	.485	.477
HeatShock25to37GO	.429	.401	.377	.362	.453	.414	.400
HyperOsmoticGO	.478	.438	.422	.410	.497	.458	.450
MenadioneGO	.480	.458	.448	.440	.499	.474	.467
NDepletionGO	.478	.397	.389	.376	.493	.431	.425

Tabela 7.4: Primerjava Δ*RMSE* ocen glede na privzeti klasifikator v posameznem poskusu. Negativne vrednosti predstavljajo odstotek izboljšanja glede na privzeti klasifikator.

	`1-NN`	`5-NN`	`11-NN`	`PCT`	`PCT`, pruned
AAstarvGO	-7.2	-10.1	-12.4	-7.1	-9.4
DiamideGO	-14.3	-18.2	-20.6	-11.0	-13.3
DTT1GO	-6.8	-10.3	-13.8	-6.3	-9.5
H2O2GO	-4.5	-7.0	-7.4	-2.6	-4.3
HeatShock25to37GO	-6.6	-12.2	-15.7	-8.6	-11.7
HyperOsmoticGO	-8.4	-11.8	-14.2	-7.8	-9.5
MenadioneGO	-4.6	-6.6	-8.4	-5.1	-6.4
NDepletionGO	-16.9	-18.7	-21.3	-12.6	-13.9



Slika 7.2: Diagram povprečnih rangov v Nemenyijevem testu in značilne različnosti. Klasifikatorji, ki niso povezani z odebeljeno črto (so narazen vsaj za kritično razdaljo **CD**), so značilno različni. Manjši rang pomeni uspešnejši klasifikator.

Tabela 7.3 zajema uspešnost nekaj klasifikatorjev `kNN` in dveh klasifikatorjev iz poskusa. *RMSE* ocene iz te tabele smo nato preračunali v $\Delta RMSE$ ocene in jih zapisali v tabeli 7.4. Nad slednjimi vrednostimi opravimo Nemenyijev test, katerega rezultati so vidni na sliki 7.4. Diagram na sliki nakazuje na dobro uspešnost `kNN` metod. Zaradi napake v poganjanju testov sicer ne moremo zaključiti, da se metoda `kNN` obnese bolje od `PCT`, a lahko pričakujemo, da bi bili ustrezni rezultati podobni našim.

Poglavje 8

Zaključek

V delu smo predstavili napovedovanje spremenljivk z metodo najbližjega soseda. Ubadali smo se predvsem z napovedovanjem nekaterih tipov strukturiranih podatkov in na slednjih preizkusili uspešnost metode. Uspešnost smo primerjali z drevesi in pravili za napovedno razvrščanje. Tako pravila kot drevesa so v literaturi [6, 3, 15, 9] že primerjana tudi z drugimi sorodnimi metodami, od koder je razvidno, da se obnesejo dobro. Izkaže se, da se naša metoda za napovedovanje strukturiranih podatkov odreže bolje od pričakovanega. V večini primerov je bila napovedna uspešnost v podobnem rangu kot uspešnost odločitvenih dreves in pogosto značilno boljša od pravil.

Slednje velja tudi za večciljno napovedovanje, kjer smo ugotovili še, da (vsaj glede na našo postavitev poskusa) obteževanje atributov ni izboljšalo uspešnosti metode. Pri hierarhičnem večznačnem napovedovanju se je metoda povsem dobro kosala z metodo `HMC`, opazno pa je bilo tudi izboljšanje v uspešnosti pri uporabi inverznega obteževanja glasovanja. Podobne rezultate smo dobili tudi pri napovedovanju kratkih časovnih vrst.

Omenimo še, da je bilo število iskanih najbližjih sosedov pri vseh poskusih relativno nizko. Da bi lahko dosegli še boljšo napovedno uspešnost nakazuje dejstvo, da so pri večini poskusov najvišje uspešnosti metod pri najvišjem številu iskanih sosedov. V prihodnje bi bilo zato smiselno preizkusiti tudi glasovanje z večjim številom sosedov.

Omenimo še, da lahko rezultate poglavja 5 projeciramo tudi na običajno klasifikacijo in regresijo. Glede na rezultate iz celotnega dela bi lahko uspešnost metode površno uvrstili nekam med drevesa in pravila.

Zanimivo bi bilo metodo primerjati še z drugimi pristopi k učenju in jo poskusiti izboljšati z drugimi načini obteževanja atributov ali izborom podmnožice atributov (recimo z genetskim algoritmom). Prav tako v delu nismo natančneje preizkusili še ostalih dveh implementiranih razdalj - chebysheve in manhattanske. Z njima smo opravili le uvodne poskuse na večciljnem napovedovanju, kjer se nista izkazali za najbolj uspešne (v primerjavi z evklidsko). To (in ogromno poskusov) je tudi razlog, da ju v poskusih, ki jih opisuje delo nismo uporabljali.

Vsebinsko ločen del diplomskega dela se ubada s hitrostjo iskanja najbližjih sosedov treh različnih metod. Čeprav [17] navaja dobro uspešnost metode `VP` v prostorih višjih dimenzij, ugotavljamo, da ima tudi slednja svojo omejitev in da je tudi ta omejitev relativno nizka. Za podatkovne baze z mnogo atributi (in relativno malo primerki) je torej še vedno najboljša izbira navadno iskanje. Zanimivo bi bilo preizkusiti še izboljšano metodo `VPS`, nekatere druge pristope k iskanju najbližjih sosedov (recimo [18]) in mogoče aproksimacijske metode.

Omenili (razdelek 2.1.1.2) smo tudi problem izgube pomenskosti najbližjih sosedov in upad uspešnosti zaradi visokih dimenzij. V podatkovnih bazah, ki smo jih preizkusili v tem delu in imajo pogosto mnogo atributov (od 4 pa do 47034) teh problemov nismo zaznali, je pa možno, da do njih pride.

Izkazala se je še ena prednost metode `kNN`, in sicer njena enostavnost. Celotna implementacija s tremi iskalnimi metodami, tremi implementacijami razdalj, obtežitvami glasovanja in atributov zavzema le okoli 2000 vrstic kode. Hkrati pa implementacijo puščamo odprto za enostavno razširitev.

Tabele

Slike

Literatura

[1] K. Beyer, J. Goldstein, R. Ramakrishnan in U. Shaft, "When is "nearest neighbor" meaningful?" *Lecture Notes in Computer Science*, zv. 1540, strani 217–235, 1999.

[2] R. J. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[3] C. Vens, J. Struyf, L. Schietgat, S. Džeroski in H. Blockeel, "Decision trees for hierarchical multi-label classification."

[4] H. Blockeel, L. De Raedt in J. Ramon, "Top-down induction of clustering trees," v *Proceedings of the 15th International Conference on Machine Learning*. Morgan Kaufmann, 1998, strani 55–63, uRL: http://www.cs.kuleuven.ac.be/cgi-bin-dtai/publ_info.pl?id=20419.

[5] "Clus: A predictive clustering system," 2010. Dostopno na: http://www.cs.kuleuven.be/~dtai/clus/index.html

[6] B. Ženko, "Učenje pravil za napovedno razvrščanje," doktorska dizertacija, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, 2007.

[7] B. Ženko in S. Džeroski, "Learning classification rules for multiple target attributes."

[8] P. Clark in T. Niblett, "The CN2 induction algorithm," v *Machine Learning*, 1989, strani 261–283.

[9] J. Strufy in S. Džeroski, "Constraint based induction of multiple-objective regression trees."

[10] A. Ruepp, A. Zollner, D. Maier, K. Albermann, J. Hani, M. Mokrejs, I. Tetko, U. Güldener, G. Mannhaupt, M. Münsterkötter in H. W. Mewes, "The funcat, a functional annotation scheme for systematic classification of proteins from whole genomes," *Nucl. Acids Res.*, zv. 32, št. 18, strani 5539–5545, 2004.

[11] "Mips functional catalogue," 2010. Dostopno na: http://mips.helmholtz-muenchen.de/proj/funcatDB/

[12] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin in G. Sherlock, "Gene ontology: tool for the unification of biology. the gene ontology consortium." *Nature genetics*, zv. 25, št. 1, strani 25–29, 2000.

[13] "Gene ontology," 2010. Dostopno na: http://www.geneontology.org/

[14] G. M. Weiss in F. Provost. Learning when training data are costly: The effect of class distribution on tree induction.

[15] I. Slavkov, V. Gjorgjioski, J. Strufy in S. Džeroski, "Finding explained groups of time-course gene expression profiles with predictive clustering trees."

[16] L. Todorovski, B. Cestnik, M. Kline, N. Lavrač in S. Džeroski, "Qualitative clustering of short time-series: A case study of firms reputation data."

[17] P. N. Yianilos. (1993) Data structures and algorithms for nearest neighbor search in general metric spaces.

[18] J. L. Bentley, B. W. Weide in A. C. Yao, "Optimal expected-time algorithms for closest point problems," *ACM Trans. Math. Softw.*, zv. 6, št. 4, strani 563–580, 1980.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann in I. H. Witten, "The weka data mining software: an update," *SIGKDD Explorations*, zv. 11, št. 1, strani 10–18, 2009.

[20] L. Breiman, "Random forests," *Machine Learning*, zv. 45, strani 5–32, 2001, 10.1023/A:1010933404324. Dostopno na: http://dx.doi.org/10.1023/A:1010933404324

[21] D. Kocev, I. Slavkov in D. Sašo, "More in better : ranking with multiple targets for biomarker discovery," *MLSB 08 / The Second International Workshop on Machine Learning in Systems Biology, 13-14*, stran 133, 2008.

[22] S. B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. D. Jong, S. Dzeroski, S. E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R. Michalski, T. Mitchell, P. Pachowicz, Y. Reich, H. Vafaie, W. V. D. Welde, W. Wenzel, J. Wnek in J. Zhang, "The monk's problems a performance comparison of different learning algorithms," Tech. Rep., 1991.

[23] K. Trohidis, G. Tsoumakas, G. Kalliris in I. Vlahavas, "Multilabel classification of music into emotions," v *Proc. 9th International Conference on Music Information Retrieval (ISMIR 2008), Philadelphia, PA, USA, 2008*, 2008.

[24] M. R. Boutell, J. Luo, X. Shen in C. M. Brown, "Learning multi-label scene classification," *Pattern Recognition*, zv. 37, št. 9, strani 1757–1771, 2004.

[25] A. Elisseeff in J. Weston, "A kernel method for multi-labelled classification," v *In Advances in Neural Information Processing Systems 14*, zv. 14. MIT Press, 2001, strani 681–687.

[26] A. Karalič in I. Bratko, "First order regression," *Machine Learning*, zv. 26, št. 2-3, strani 147–176, 1997.

[27] D. Demšar, M. Debeljak, C. Lavigne in S. Džeroski, "Modelling pollen dispersal of genetically modified oilseed rape within the field," v *The Annual Meeting of the Ecological Society of America*, 2005.

[28] N. Colbach, C. Clermont-Dauphin in J. Meynard, "Genesys: a model of the influence of cropping system on gene escape from herbicide tolerant rapeseed crops to rape volunteers - i. temporal evolution of a population of rapeseed volunteers in a field," *Agriculture, Ecosystems Environment*, zv. 83, št. 3, strani 235–253, 2001.

[29] S. Džeroski, N. Colbach in A. Messean, "Analysing the effect of field characteristics on gene flow between oilseed rape varieties and volunteers with regression trees," v *2nd Int'l Conference on Co-existence between GM and non-GM based agricultural supply chains*, 2005.

[30] D. Demšar, S. Džeroski, T. Larsen, J. Struyf, J. Axelsen, M. Pedersen in P. Krogh, "Using multi-objective classification to model communities of soil," *Ecological Modelling*, zv. 191, št. 1, strani 131–143, 2006.

[31] M. Debeljak, D. Kocev, W. Towers, M. Jones, B. S. Griffiths in P. D. Hallett, "Potential of multi-objective models for risk-based mapping of the resilience characteristics of soils: demonstration at a national level," *Soil Use and Management*, zv. 25, strani 66–77, 2009.

[32] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, zv. 7, strani 1–30, 2006.

[33] A. Clare, "Machine learning and data mining for yeast functional genomics," doktorska dizertacija, University of Wales, Aberystwyth, 2003.

[34] A. P. Gasch, P. T. Spellman, C. M. Kao, O. Carmel-Harel, M. B. Eisen, G. Storz, D. Botstein in P. O. Brown, "Genomic expression programs in the response of yeast cells to environmental changes." *Molecular biology of the cell*, zv. 11, št. 12, strani 4241–4257, 2000.

[35] I. Bratko, *Prolog Programming for Artificial Intelligence*, 3. izdaja. Addison Wesley, 2000.

[36] I. H. Witten in E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2. izdaja, ser. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 2005.