Analiza algoritma uniclass

Jure Ham - 63080514

June 9, 2011

1 Opis programa

Program uniclass je celovit paket namenjen testiranju in nadgradji algoritma uniclass. Grafični vmesnik omogoča nalaganje testnih primerov, spreminjanje osnovnih nastavitev algoritma ter sproten prikaz delovanja. Ker je algoritem računsko zahteven, je program implementiran v javi, ki omogoča dober kompromis med hitrostjo izvajanja in hitrostjo razvoja.

1.1 Branje podatkov

Program podpira nalaganje testnih primerov v formatu tab, ki je osnoven format programskega paketa orange. Implementacija je omejena na zvezne in neurejene diskretne vrednosti ter na eno samo meta vrednost, ki postane ime entitete.
Manjkajoči podatki so obravnavani kot posebne vrednosti, kar pomeni, da je razdalja med entiteto z vsemi podatki in entiteto brez podatkov največja, razdalja med dvema entitetama brez podatkov pa je najmanjša. Razlog za to odločitev je zelo enostavna implementacija.

1.2 Računanje matrike razdalj

Razdaljo med dvema primeroma izračunamo kot vsoto vseh razdalj med atributi. V primeru, da je aribut diskreten in neurejen, je razdalja diskretna in sicer 0, če sta vrednosti enaki in 1, če sta vrenosti različni. V primeru, da je atribut zvezen, pa se razdalja inračuna kot

$$dist = \left| \frac{value_1 - value_{min}}{value_{max} - value_1} - \frac{value_2 - value_{min}}{value_{max} - value_2} \right|$$

Kjer je $value_1$ vrednost atributa pri prvem primeru, $value_2$ vrednost atributa pri drugem primeru, $value_{max}$ največja vrednost atributa in $value_{min}$ najmanjša vrednost atributa.
Vse razdalje se pomnožijo še s pomembnostjo atributa, ki je izračunana z algoritmom information gain in se lahko spreminja v grafičnem vmesniku, ter s pomembnostjo vrednosti, ki je definirana kot unikatnost vrednosti. Unikatnost izračunamo na diskretiziranih vrednostih in sicer tako, da izračunamo povrečno število primerov z enako vrednostjo, nato pa to število delimo s številom primerov, ki imajo enako vrednost kot primer, ki nas zanima.

$$uni = \frac{\frac{\| E \|}{\| V_u \|}}{\| E_i = v \|}$$

Kjer je E množica vseh entitet, V_u množica vseh unikatnih vrednosti za atribut in v vrednost za katero računamo unikatnost.
Ko primerjamo dva primera, je unikatnost določena kot produkt unikatnosti prve in druge vrednosti.

1.3 Simuliranje sil

Algoritem uniclass deluje tako, da primere predstavi kot delce v dvodimenzionalnem prostoru. Vsem delcem dodeli enako maso, nato pa med njimi računa sile, ki delce premikajo.

1.3.1 Vztrajnost

Za razliko od algoritma MDS, uniclass upošteva tudi vztrajnost delcev. Vztrajnost računamo tako, da ima vsak delec poleg atributa pozicija tudi atribut hitrost, ki se spreminja glede na sile, ki nanj delujejo. S spreminjanjem mase delcev lahko uravnavamo kaotičnost sistema, saj je vpliv sil na hitrost delca obratno sorazmeren z njegovo maso. Začetna masa delcev je sorazmerna z vsoto vseh sil med delci, tako da kaotičnost sistema ni odvisna od testnih podatkov.

1.3.2 Izračun sile med delcema

Algoritem deli delce na podobne in na ne podobne. Podobni delci so tisti, med katerimi je moč povezave manjša od določene meje, ki je na začetku definirana kot povprečna moč povezave med vsemi delci, kasneje pa jo lahko preko uporabniškega vmesnika tudi spreminjamo. Moč povezave je definirana kot $1 - razdalja$.
Podobni delci se privlačijo s silo, ki je definirana kot $\frac{(f-k)*d^2}{C}$, kjer je f moč povezave, k je prej omenjena meja, d je razdalja med delcema v prostoru, C pa konstanta.
Delci, ki si niso podobni, se odbijajo po formuli $\frac{(f-k)*C}{d}$.
Tako izračunana sila se nato še deli z maso delca. Ker so razdalje lahko zelo majhne, je določena tudi največja možna sila, kar prepreči kaotičnost sistema.

1.3.3 Zaletavanje delcev

Ker algoritem predstavi primere kot delce z maso in radijem, ki je večji od 0, se delci med seboj tudi zaletavajo. S tem preprčimo to, da bi se več delcev zbralo na isti oziroma zelo podobni poziciji v prostoru, hkrati pa omogočimo združevanje podobnih delcev, kar bi bilo v nasprotnem primeru zaradi vztrajnosti skoraj nemogoče.
Da se delci, ki si niso podobni, nebi združevali, definiramo togost delcev pri odboju kot obratno verdnost njune podobnosti. Tako dva zelo podobna delca po trku potujeta v skoraj enako smer, dva zelo različna pa vsak v svojo.

1.3.4 Meje polja

Delci, ki z ostalimi primeri nimajo močnih povezav, nam bodo hitro pobegnili izven vidnega polja, saj jih bo večina delcev odbijala. Ta problem vsaj delno rešimo s tem, da okoli polja postavimo mejo, od katere se vsi delci odbijajo. Tako take delce ujamemo na robu polja, kjer se bodo poskušali čim bolj približati delcem, ki so jim podobni in čim bolj oddaljiti od delcev, ki jih odbijajo. Da delci nebi ostali popolnoma prilepljeni na mejo, v algoritem uvedemo krčenje prostora, ki v vsakem ciklu računanja delce malo pomakne proti sredini. Razdalja za katero se delci premaknejo, je odvisna od oddaljenosti od središča.

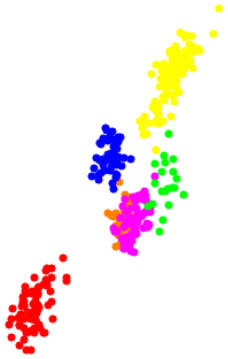
1.4 Barvanje delcev

Za boljšo preglednost delovanja delce pobarvamo glede na razred kateremu pripadajo. Razrede pobarvamo tako, da jih čim bolj enakomirno razporedimo po robu barvnega kroga.

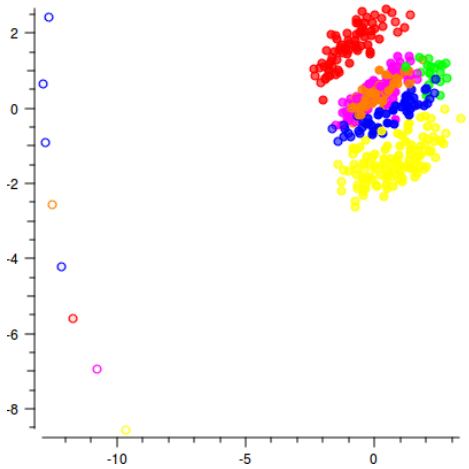
Poleg barvanja razredov delcem dodelimo še dve vizualni lastnosti. Prva lastnost je temnost, ki je odvisna od velikosti sile, ki deluje nanj, druga lastnost pa je rdeča obroba, ki se pokaže, kadar algoritem kNN iz pozicije delca v prostoru ni pravilno določil njegovega razreda.

2 Vizualizacija podatkov

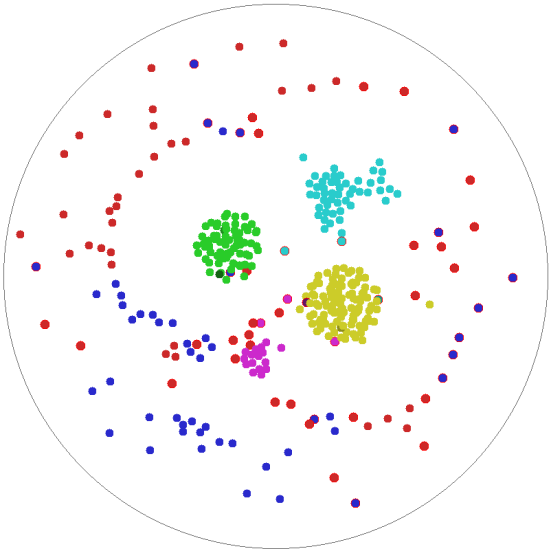
Ker algoritem uniclass obdeluje podatke v dvodimenzionalnem prostoru, je zelo primeren za vizualizacijo podatkov. S spreminjanjem nastavitev algoritma lahko v živo spremljamo kaj se s delci dogaja in pri tem izvemo zanimive lastnosti podatkov. Zaradi te lastnosti algoritma slika ne odseva vseh vizualnih lastnosti podatkov, je pa kljub temu informativna.
Pri sliki 1(c) lahko vidimo, da sta zelen in rumen razred dominantna, saj imata veliko močno povezanih primerov, moder razred ima ravno tako veliko primerov, vendar le ti niso zelo enotni. To lastnost izvemo iz oblike lika, ki ga delci oblikujejo. Vijoličen razred je majhen, vendar relativno dobro povezan. Iz pozicije lahko sklepamo, da je nekoliko podoben rumenemu razredu, hkrati pa ima skupne lastnosti z nekaterimi člani rdečega razreda. Moder in rdeč razdred sta očitno precej drugačna od ostalih, vendar sta znotraj razreda tako slabo povezana, da se pozamezni primeri me morejo združiti v skupine. Če bi opazovali animacijo in ne le slike, bi lahko opazili še, da se delci v zelenem razredu najhitreje giblejo, kar pomeni, da so sile med njimi največje.
Dejstvo, da se moder in rdeč razred ne združita v skupino je hkrat problem in prednost algoritma uniclass. Ker se nista združila lahko sklepamo, da so si primeri znotraj razreda precej različni, hkrati pa ne moremo jasno videti v kateri razred spada kateri primer zaradi česar bi bila klasifikacija nerazporejenih primerov izredno zahtevna.
Objektivno bi morda lahko ocenjevali kvaliteto vizualizacije s ponočjo clusteringa, ki bi ga izvedli s ponočjo pozicije delcev v prostoru. Ta metoda nam bi verjetno pokazala kako dobro določen algoritem razdeli podatke po dvodimenzionalnem prostoru, nebi nam pa povedala, kaj vse lahko o podatkih izvemo s pomočjo algoritma, kar je glavni cilj vizualizacije.
Ker ima vsaka metoda svoje prednosti in slabosti, ne bomo ugotavljali kater algoritem je najbolši, temveč bomo priporočili uniclass kot dodatno orodje za iskanje zakonitosti v podatkih.



(a) FreeViz



(b) MDS

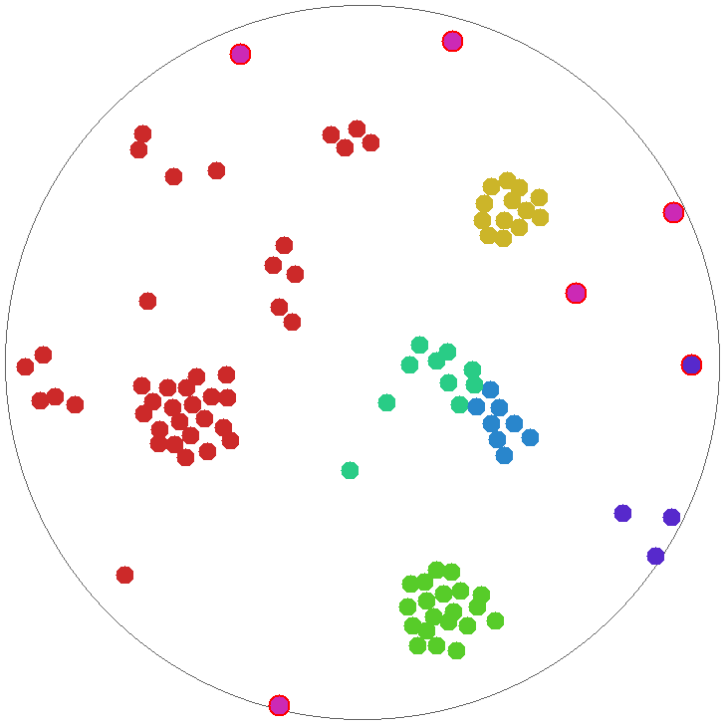


(c) uniclass

Slika 1: Primerjava algoritmov za vizualizacijo na podatkih dermatology.tab. (Barve pri različnih primerih ne predstavljajo nujno istega razreda.)

3 Klasifikacija

Poleg vizualizacije trenutna implementacija algoritma podpira tudi testiranje klasifikacijske točnosti algoritma. Razred primera se določi tako, da s pomočjo malo spremenjenega algoritma kNN iz najbližjih sosedov izberemo najbolj primernega kandidata. Za namene testiranja uporabimo princip izpusti enega(leave-one-out), kar pomeni, da poznamo razred vseh sosedov. Program pri tem nekoliko goljufa, saj računa pomembnost atributov tudi na tistem primeru, za katerega kasneje ugotavlja razred.
Glavna moč algoritma uniclass pri klasifikaciji je vizualna komponenta, saj lahko sami vidimo v kakšni okolici se nahaja določen primer. Če bi se na sliki 2 naznan primer nahajal v sredini rdeče skupine bi lahko z zagotovostjo trdili, da tja tudi spada. Če bi se nahajal nekje ob robu, pa bi lahko napovedali več različnih potencialnih razredov.
Razdalja med primeri je določena kot kvadrat razdlaje med primeroma v prostoru. Poleg razdalje agoritem upošteva tudi razpršenost primerov, tako da se iskanje sosedov ustavi takoj, ko je razdalja do naslednjega primera občutno večja od trenutne povprečne razdalje. S tem preprečimo napačno klasifikacijo primerov, ki se nahajajo v zelo majhni skupi v bližini velike skupine.
Na sliki 2 lahko vidimo katere primere je algoritem pravilno klasificiral in katere napačno(rdeča obroba). Klasifikacijska točnost za ta primer je 94%.



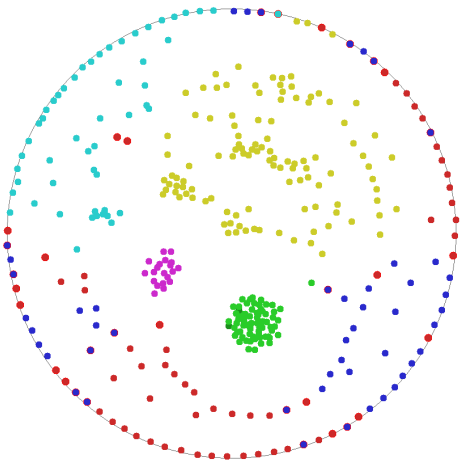
Slika 2: Predstavitev podatkov zoo.tab z algoritmom uniclass.

3.1 Primerjava klasifikacijske točnosti

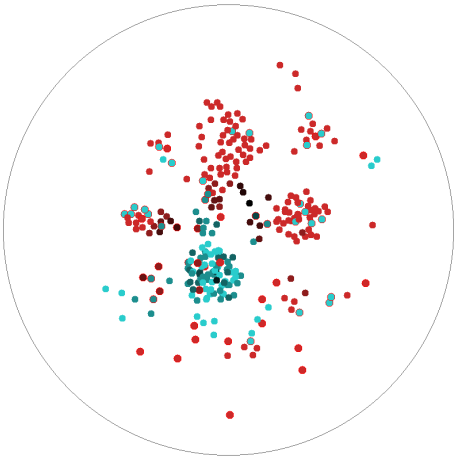
Za primerjavo klasifikatorjev sem si izbral štiri popularne klasifikatorje. Pri interpretiranju rezultatov je potrebno upoštevati tudi dejstvo, da je uniclass delač najpočasnejša in najmanj avtomatska metoda, saj je za zglede klasifikacjske rezultate potrebno ročno nastaviti mejo tako, da so skupine primerov čim bolj ločene. To se zdi kot goljufanje vendar ni, saj lahko vizualno ločimo skupine tudi, če poznamo razrede le majhnega števila primerov.

	dermatology.tab	zoo.tab	heart.tab
uniclass	0.91 slika 3(a)	0.94 slika 2	0.82 slika 3(b)
Večinski	0.31	0.41	0.54
SVM	0.97	0.95	0.83
Naivni Bayesov	0.97	0.92	0.83
kNN	0.96	0.97	0.77

Iz zbranih podatkov lahko ugotovimo, da algoritem uniclass ni najmočnejša metoda, če želimo golo klasifikacijo. Poleg počasnosti ima algoritem problem tudi s klasifikacijsko točnostjo. Zaradi teh ugotovitev priporočamo funkcijo klasificiranja podatkov le kot pomoč pri vizualizaciji podakov, kjer nam pri določenih primerih manjkajo podatki o razredu.



(a) dermatology.tab

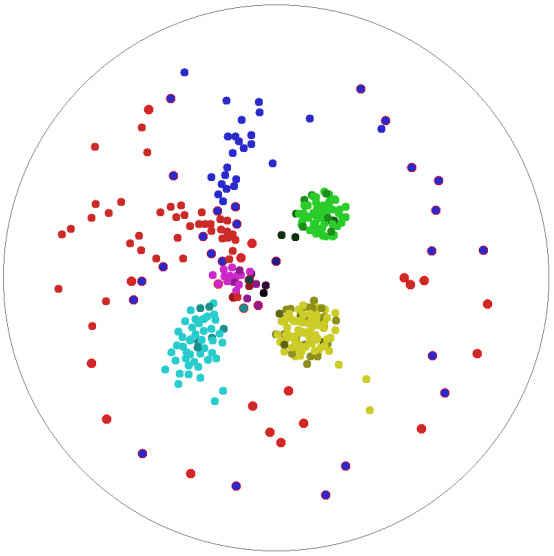


(b) heart.tab

Slika 3: Klasifikacija podatkov z algoritmom uniclass.

4 Clustering

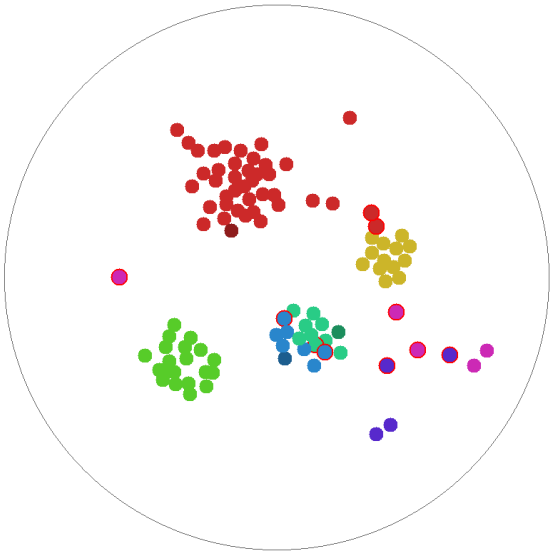
Ker algoritem posamezne primere združuje v skupine najbolj podobnih, se možnost clusteringa zdi očitna. Trenutna implementacija algoritma za izračun sil upošteva tudi pomembnost atributov, za kar potrebujemo razrede, vendar grafični umesnik podpira ponastavitev vseh atributov in s tem preprečitev goljufanja. Ko atribute ponastavimo, ugotovimo da se kljub temu primeri skoraj enako dobro razporedijo po skupinah(slika 4). Ker program nima implementirane funkcije clusteringa, si bomo pomagali s paktom orange in sicer z vtičnikoma k-Means Clustering ter Hierarchical clustering.



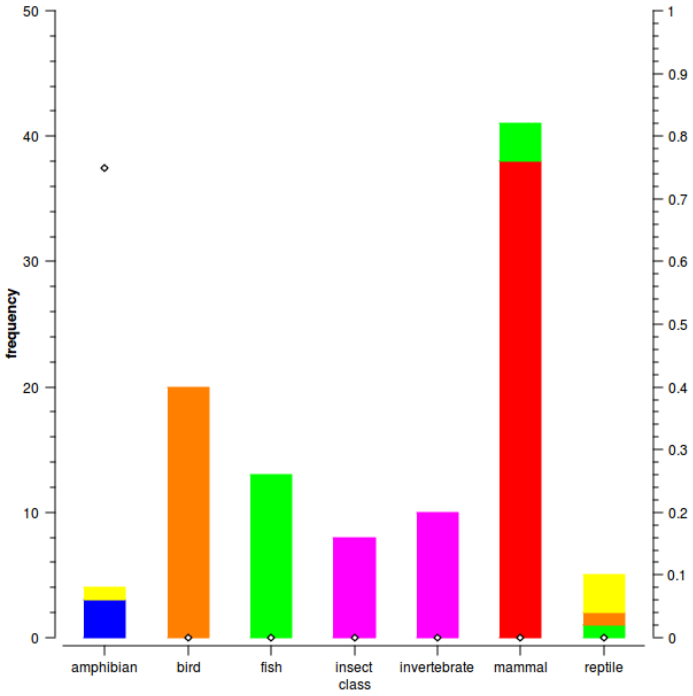
Slika 4: Predstavitev podatkov dermatology.tab brez upoštevanja pomembnost atributov.

4.1 Primerjava algoritmov

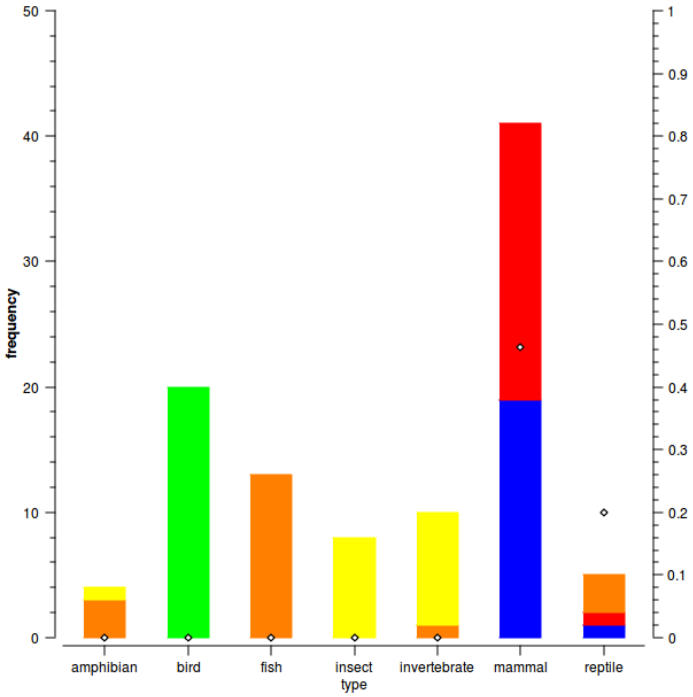
Za primerjavo algoritmov smo si izbrali podatke zoo.tab. V programi uniclass smo ponastavili pomomebnost atributov ter dobili sliko 5. Podatke smo izvozili tako, da smo imeli poleg podatka o razredu le podatek o poziciji, ter jih uvozili v paket orange. Nato smo uvozili še originalne podatke, ter jih za prvo primerjavo poslali v widget k-Means Clustering, ki smo ga nastavili tako, da si je sam izbral optimalno število clusterjev. Na sliki 6(a) si lahko ogledate rezultate clusteringa z uporabo algoritma uniclass, na sliki 6(b) pa rezultate pridobljene le z uporabo algoritma k-Means. Opazimo lahko, da je algoritem uniclass bolje razporedil sesalce, ostali razredi pa so razporejeni podobno dobro.
Za drugo primerjavo smo podatke obdelali z algoritmom hierarchical clustering, pri računanju povezav pa smo uporabili wardovo povezavo. Ker algoritem od nas zahteva, da sami nastavimo mejo za ločitev razredov, rezultati niso odvisni le od algoritmov, vendar tudi od raziskovalca. Na sliki 7(a) si lahko ogledate rezultate pridobljene s pomočjo algoritma uniclass, na sliki 7(b) pa rezultate brez uniclassa. Rezultati so si zopet zelo podobni. Pri uporabi uniclass smo napačno združili insekte in nevretenčarje, brez uniclassa pa dvoživke in nevretenčarje.



Slika 5: Podatki zoo.tab v programu uniclass brez ocene atributov.

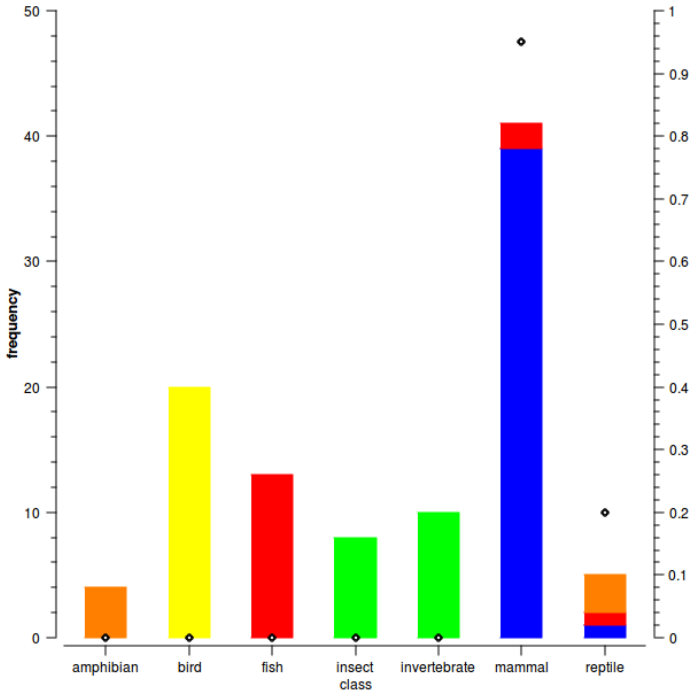


(a) Z uniclass

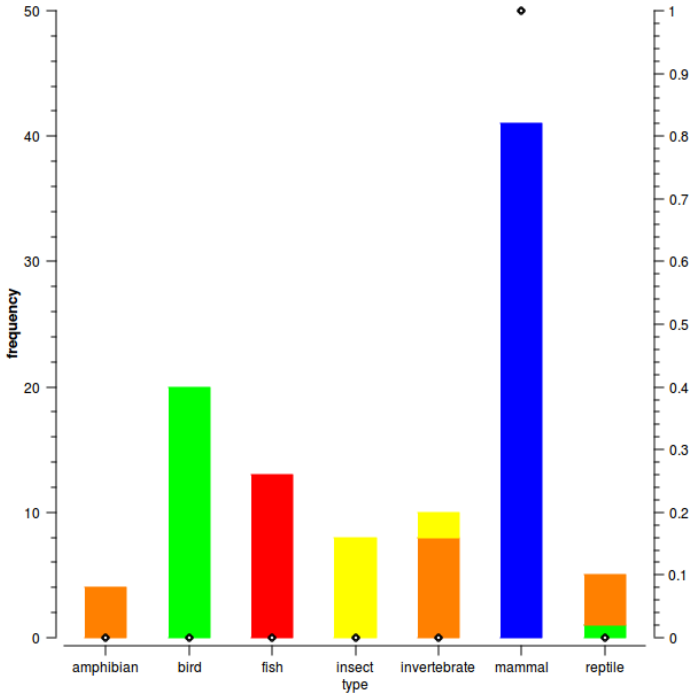


(b) Brez uniclass

Slika 6: Primerjava clusteringa podatkov z uporabo algoritma k-Means Clustering



(a) Z uniclass

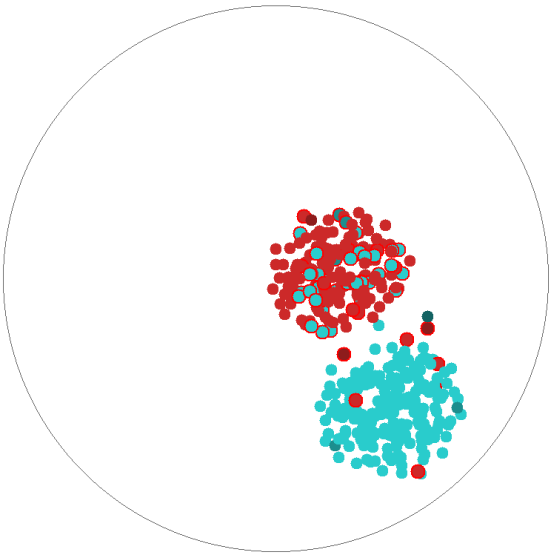


(b) Brez uniclass

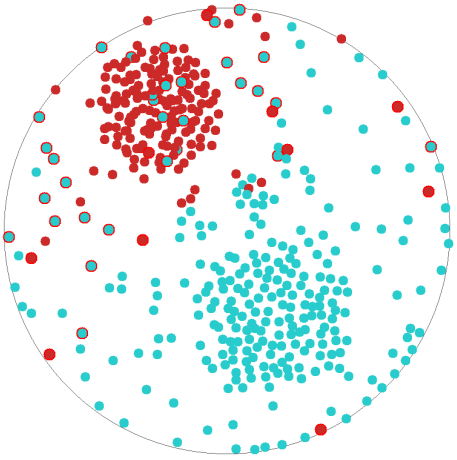
Slika 7: Primerjava clusteringa podatkov z in brez uporabe algoritma Hierarchical clustering.

5 Zaključek

Skozi seminarsko nalogo smo se dotaknili treh potencialnih aplikacij algoritma uniclass. Ugotovili smo, da je največja moč algoritma v vizualizaciji podatkov, saj omogoča dodatno perspiktivo pri opazovanju podatkov. Poleg opazovanja razpršenosti podatkov lahko s pomočjo opazovanja premikanja ločimo močno in šibko povezane primere. S spreminjanjem nastavitev v grafičnem umesniku pa lahko v živo opazujemo spremembe, kar nam omogoča odkrivanje bolj skritih zakonitosti in vzorcev(slika 8). Tudi pri clusteringu podatkov se algoritem dobro odreže, saj zelo učinkovito zmanjša število atributov protrebnih za clustering.



(a) Meja postavljena nizko



(b) Meja postavljena visoko

Slika 8: Opazovanje podatkov votes.tab, kjer lahko pri nizko postavljeni meji vidimo, da je med demokrati veliko takih, ki glasujejo podobno kot republikanci, pri visoko postavljeni meji, pa vidimo, da so si republikanci veliko bolj enotni kot demokrati.

6 Povezave

Izvorna koda projekta ter vsi testni primeri.