Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Peter Šaponja

Odkrivanje skupin s pomočjo argumentiranega strojnega učenja

magistrsko delo

študijski program druge stopnje
računalništvo in informatika

Mentor: doc. dr. Matej Guid

Ljubljana, 2015

Izjava o avtorstvu magistrskega dela

Spodaj podpisani Peter Šaponja sem avtor magistrskega dela z naslovom:

Odkrivanje skupin s pomočjo argumentiranega strojnega učenja

S svojim podpisom zagotavljam, da:

- sem magistrsko delo izdelal samostojno pod mentorstvom doc. dr. Matej Guida,
- so elektronska oblika magistrskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko magistrskega dela,
- soglašam z javno objavo elektronske oblike magistrskega dela v zbirki "Dela FRI".

V Ljubljani, 11. September 2015 Podpis avtorja:

Zahvaljujem se mentorju doc. dr. Mateju Guidu za pomoč, veliko zelo koristnih nasvetov in predlogov ter veliko mero potrpežljivosti. Zahvalil bi se tudi družini in vsem ostalim, ki so mi v tem času nudili podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ABML	argument-based machine learning	argumentirano strojno učenje
RFID	radio frequency identification	radiofrekvenčna identifikacija
ML	must-link constraint	pozitivna omejitev
CL	cannot-link constraint	negativna omejitev
CKM	constrained k-means	metoda voditeljev z omejitvami
PCKM	pairwise constrained k-means	metoda voditeljev s parnimi omejitvami
ABKM	argument-based k-means	metoda voditeljev z argumenti
PCA	principal component analysis	analiza glavnih komponent

Povzetek

Potrebe po izboljšanju odkrivanja skupin (angl. *clustering*) v podatkih dandanes vedno bolj zahtevajo možnost interakcije z domenskimi strokovnjaki, kar je vodilo do razvoja algoritmov odkrivanja skupin z omejitvami (angl. *constrained clustering*). Ti algoritmi uporabljajo domensko znanje v obliki pozitivnih (angl. *must-link*) in negativnih omejitev (angl. *cannot-link*) na pare učnih primerov, kar omogoča izboljšanje procesa odkrivanja skupin. Med slabo raziskanimi problemi na tem področju pa je sama učinkovitost procesa zajemanja omejitev. Postopek zajemanja omejitev lahko pomembno vpliva na kakovost odkrivanja skupin z omejitvami, vendar je tipično zelo zahteven celo za domenske strokovnjake.

V magistrskem delu smo zasnovali in razvili metodo voditeljev z argumenti (angl. *Argument-based k–means*, AB k-means), ki je namenjena učinkovitemu odkrivanju skupin in temelji na paradigmi argumentiranega strojnega učenja. Pri iterativni zanki za zajemanje znanja s pomočjo argumentiranega strojnega učenja domenski strokovnjak s pojasnjevanjem avtomatsko izbranih problematičnih primerov vnaša domensko znanje, metoda pa nato s pomočjo "protiprimerov" izpostavlja morebitne pomanjkljivosti strokovnjakovih razlag in mu omogoča izboljševanje podanih argumentov. Omenjeno iterativno zanko smo prilagodili potrebam odkrivanja skupin, tako da nova metoda z izpostavljanjem slabše in bolje umeščenih primerov v skupine tekom postopka zajemanja znanja iz domenskega strokovnjaka pridobi omejitve, ki so ključne za izboljšanje rezultatov odkrivanja skupin. Hkrati pa pridobljene omejitve vodijo do oblikovanja skupin, ki so skladne s strokov-

njakovim znanjem v izbrani domeni.

Razvili smo tudi aplikacijo za interaktivno odkrivanje skupin s pomočjo novo razvite metode. Učinkovitost našega pristopa smo empirično ovrednotili na treh eksperimentalnih domenah s pomočjo primerjave z običajnim algoritmom za odkrivanje skupin z omejitvami in pri tem dobili spodbudne rezultate.

Ključne besede

delno nadzorovano učenje, odkrivanje skupin, grupiranje, metoda voditeljev, odkrivanje skupin z omejitvami, argumentirano strojno učenje, iterativna zanka za zajemanje znanja, zajemanje omejitev, metoda voditeljev z argumenti

Abstract

The need for improvement of data clustering methods demanded more interactive options with domain experts, which led to the development of algorithms, coined as *constrained clustering*. These algorithms use domain knowledge in the form of positive *must-link* and negative *cannot-link* constraints to improve the quality of detected groups. One of the most overlooked issues in this filed is the effectiveness of constraint elicitation. While the process of constraint elicitation can be a tedious task it can have a significant impact on the quality of clustering.

In this thesis we designed and developed a method named *Argument-based k-means* (AB k-means), which is designed for a more efficient clustering and is based on the paradigm of *argument-based machine learning* (ABML). *The knowledge refinement loop* enables the domain expert to articulate his domain knowledge by argumenting automatically chosen problematic cases, while the method with the help of counter examples highlights any shortcomings in the expert's arguments. We adapted the knowledge refinement loop to the needs of clustering by exposing badly and well clustered cases when eliciting constraints, which are crucial for the improvement of clustering. At the same time the obtained constraints lead to clusters that are consistent with the knowledge of the expert in their chosen domain.

For an easier use of the new method we have also developed an interactive application. The effectiveness of our approach was empirically tested on three different experimental domains and compared favourably with an ordinary algorithm for constrained clustering.

Keywords

semi-supervised learning, clustering, k-means, constrained clustering, argument-based machine learning, knowledge refinement loop, constraint elicitation, argument-based k-means

Poglavje 1

Uvod

Že od nekdaj posamezniki, organizacije in vlade zbirajo podatke različnih oblik za doseganje želenih ciljev. Tehnologija zaznavanja in shranjevanja podatkov je v dobi računalnikov občutno napredovala. Naraščajoče število storitev, kot so internetni iskalniki, digitalne slike in video nadzor, pa je povzročilo izjemen preskok v zbiranju in obdelavi podatkov. Tako je danes v elektronskih medijih shranjen izjemno obsežen fond digitalnih podatkov, kar zagotavlja velik potencial za avtomatsko analizo teh podatkov.

A porasta nismo zaznali le v količini, temveč tudi v obliki podatkov (besedilo, slika, video, zvok,...). S cenovno dostopnimi digitalnimi kamerami smo dobili enormne arhive slik in videov. Z razširjenostjo radiofrekvenčne identifikacije (angl. *radio frequency identification*, RFID) tehnologije je nastalo več kot milijon senzorjev, ki oddajajo podatke. Svetovni splet pa z več bilijoni spletnimi stranmi vsak dan ustvari dodatne terabajte podatkov. S prihodom "WEB 2.0", so uporabniki vključeni v razvoj spleta in kot aktivni uporabniki lahko sodelujejo pri ustvarjanju vsebin na spletu[1]. S tem postanejo razlike med avtorji in potrošniki le še bolj zamegljene. Tako dobimo vtis, da je količina podatkov, beležk poizvedovanj, objav na blogih in socialnih omrežjih ter slikah in videih neskončna.

Nenehno naraščajoča količina podatkov nas tako vedno znova poziva k večjemu številu orodij za avtomatsko raziskovanje in procesiranje podatkov.

V ta namen so bile razvite statistične tehnike, kot so linearna regresija, diskriminantna analiza, analiza glavnih komponent (angl. *principal component analysis*, PCA) in analiza skupin.

Prepoznavanje vzorcev, kot je analiza podatkov, temelji na napovednih modelih. S pomočjo učnih podatkov poskušamo ustvariti napovedni model, ki bo napovedal obnašanje še ne videnih testnih podatkov. Ta postopek imenujemo strojno učenje. Ločimo med nadzorovanim (angl. *supervised learning*) in nenadzorovanim učenjem (angl. *unsupervised learning*). Pri nadzorovanem učenju imamo učne podatke z znanimi oznakami, pri nenadzorovanem pa podatke brez oznak. Zaradi odsotnosti teh oznak je s postopkom nenadzorovanega učenja gradnja napovednega modela težja.

Med najbolj razširjene metode avtomatskega nenadzorovanega učenja spada odkrivanje skupin (angl. *clustering*), nadzorovanega učenja pa klasifikacija (angl. *classification*) [2]. Pri odkrivanju skupin algoritmi poskušajo iz podanih informacij povečati podobnost podatkov v isti skupini in hkrati narediti čim večje razlike med točkami v različnih skupinah. Pri klasifikaciji pa imamo opravka z učnimi primeri z znanimi razredi, kjer naučimo algoritem karakteristik razredov, v obliki pravil, s katerimi uvrsti nadaljnje primere, pri katerih so razredi neznani.

Klasifikacija je zelo odvisna od učnih podatkov. Ko jih pridobimo, lahko dobimo kvalitetne rezultate pri najrazličnejših nalogah in domenah. Ob tem imamo veliko mero nadzora, saj z izborom učnih primerov lahko usmerjamo algoritem do rezultata, ki ga pričakujemo. Prav zato je tudi pridobivanje učnih primerov zelo pomembna naloga pri klasifikaciji. Odkrivanje skupin deluje po nasprotnem principu. Odkrite skupine niso vnaprej natančno definirane, zato težko razločimo, kaj točno povzroči razliko med podatki in do katere mere moramo to upoštevati. Imamo manj nadzora nad celotnim procesom in smo omejeni na oblikovanje rešitev s pomočjo intuicij ali korenite spremembe algoritma. Končni produkti so tipično bolj skromni, a koristni rezultati.

Zaradi tipično skromnejših rezultatov odkrivanja skupin in vse večjega

povpraševanja po boljših rezultatih, je interes vedno bolj usmerjen v novo vejo odkrivanja skupin z omejitvami, ki sodi v okvir delno nadzorovanega učenja (angl. *semi–supervised learning*). Ti algoritmi imajo znane oznake le za manjši del učnih podatkov. Medtem ko pri klasifikaciji neoznačene podatke navadno odstranimo, jih pri delno nadzorovanem učenju vključimo v učni proces. Podobno kot pri klasifikaciji, podatki z oznakami služijo algoritmu kot domensko znanje, s pomočjo katerega lahko uporabnik vodi proces odkrivanja skupin in izboljša kvaliteto rezultatov.

Pri delno nadzorovanem učenju je pridobivanje omenjenih oznak težak proces, zato algoritmi za odkrivanje skupin največkrat namesto oznak postavljajo omejitve nad dvema učnima primeroma. Čeprav nosijo oznake več domenskega znanja o skupinah, zaradi podanega podatka o ciljni skupini, je zaradi nepoznavanja skupin lažje podati domensko znanje v obliki parnih omejitev. Parne omejitve predstavljajo strokovnjakovo (ekspertovo) trditev, ali naj omenjeni par spada v isto ali različno skupino. Lahko se jih upošteva kot mehke prepovedi (angl. *soft constraint*), katere izražajo le naklonjenost k tej prepovedi ali trde prepovedi (angl. *hard constraint*).

1.1 Motivacija

Analiza skupin je uporabna v katerikoli disciplini za analiziranje podatkov. Odkrivanje skupin se med drugim uporablja za naravno klasifikacijo, kjer poiščemo podobnost objektov, za iskanje osnovne strukture podatkov, s katero dobimo vpogled v podatke, ustvarjamo hipoteze, dektiramo anomalije in iščimo pereče atribute, ter za kompresijo, kjer organiziramo in povzemamo podatke preko skupin [10].

Zaradi številčnosti znanstvenih domen in njihove uporabe analize skupin je težko izčrpno opisati vse. Za lažjo predstavo pomembnosti področja analize skupin sledi kratek seznam domen in aplikacij.

- Računalniški vid: segmentacija slik [4], odkrivanje skupin pri prepoznavi črk [5].

- Video: sledenje objektom [6].
- Marketing: odkrivanje skupin kupcev [7].
- Prostorske informacije: iskanje GPS poti [8].

Delno nadzorovano odkrivanje skupin je močno in fleksibilno orodje, s katerim drastično izboljšamo odkrivanje skupin ne glede na domeno in naravo podatkov. S tem dobimo priročen način uporabe domenskega znanja, katerega bi v nenadzorovanem učenju spustili. Tako lahko uporabniku omogočimo vodenje algoritma do hipoteze, ki bo skladna z njegovimi znanjem in željami. Kako hitro se hipoteza odkrivanja skupin spremeni, so prikazali Elias Pampalk in sodelavci na primeru podatkov šestnajstih živali s trinajstimi atributi [3]. Slika 1.1

(a) Z uteževanjem dobimo skupini predatorji in ne–predatorji.

(b) Z uteževanjem dobimo skupini sesalci in ptiči.

Slika 1.1: Primer podatkovne zbirke, pri kateri dobimo različne skupine, glede na interpretacijo.

Podatke so ob dodajanju višjih uteži atributom, ki predstavljajo aktivnosti, razdelili na dve skupini: predatorje in nepredatorje; medtem ko so ob dodajanju višjih uteži izgledu podatke razdelili na dve drugi skupini: sesalce in ptiče. Obe hipotezi sta pravilni, saj je strokovnjak tisti, ki se odloči, katera mu bolj ustreza.

V primeru uporabe odkrivanja skupin z omejitvami nam ni treba spreminjati delovanja primerjave primerov in imamo bolj direkten način vpeljave naše hipoteze. Z vpeljavo omejitev v odkrivanje skupin postavljamo omejitve čez celoten prostor hipotez in s tem zmanjšamo nevarnost pretiranega

prilagajanja podatkom (angl. *overfitting*) ter vodimo algoritem k boljšim hipotezam, hkrati pa je pridobljena hipoteza načeloma bolj skladna z znanjem strokovnjaka.

Kljub prednostim, ki jih prinaša odkrivanje skupin z omejitvami, je bilo raziskovanje doslej usmerjeno le v teoretične vidike omejitev, medtem ko je nekaj praktičnih problemov ostalo nerešenih. Predlagani algoritmi so bili osredotočeni na čim bolj učinkovito uporabo informacije, ki jo nosi omejitev. Praktični problemi, kot so dolgo in utrujajoče ustvarjanje omejitev ter zmotljivost strokovnjaka pa ostali nenaslovljeni.

V tem magistrskem delu predlagamo nov način uporabe odkrivanja skupin z omejitvami ter se posvetimo praktičnim vidikom in problemom, ki morajo biti upoštevani, ko se spopadamo z realnimi podatki.

1.2 Prispevki k znanosti

V magistrskem delu izpostavimo vprašanja, ki se pojavijo pri praktični izvedbi odkrivanja skupin z omejitvami. V nadaljevanju izpostavimo težave, ki se pojavijo pri uporabi teh metod v realnih primerih. Bolj podrobno si ogledamo problem pridobivanja omejitev in predlagamo možno rešitev.

Dosedanji razvoj je bil usmerjen v nove algoritme, ki na kreativne načine uporabijo dobljene omejitve. Ko se algoritme primerja, avtorji v veliki večini uporabijo omejitve, dobljene z resničnimi razredi učnih primerov, ki v praksi niso na voljo. Tako so omejitve postavljene pod idealnimi pogoji s predpostavko, da poznamo resnične skupine primerov.

V magistrski nalogi predlagamo metodo za odkrivanje skupin z aktivnim zajemanjem znanja iz strokovnjaka, ki se spopade s težavo zajemanja omejitev, ko ne poznamo zlate resnice (angl. *golden truth*). Strokovnjaku izpostavimo bolj problematične dele trenutne baze znanja in mu s tem omogočimo postavljanje omejitev na primere, ki vodijo do večjih izboljšav. Tako tudi drastično zmanjšamo število potrebnih omejitev, strokovnjak pa porabi manj časa za postavljanje omejitev.

Predlagano metodo vpeljemo s pomočjo argumentiranega strojnega učenja (angl. *argumented based machine learning*, ABML) [25], katere cilj je zajemanje strokovnjakovega znanja in postavljanje omejitev, s katerimi se zmanjša prostor hipotez ter izpostavi hipoteze, ki so bolj skladne s strokovnjakovimi argumenti. Z vpeljavo argumentov v odkrivanje skupin strokovnjak obrazloži svojo odločitev, zakaj se je odločil za takšno omejitev. S to novo informacijo iščemo nadaljnje možne omejitve, ki sovpadajo z njegovo obrazložitvijo.

Strokovnjak tako vodi odkrivanje skupin, s pomočjo omejitev, kot tudi selekcijo omejitev nad pari, s pomočjo argumentov. Strokovnjaka vodimo skozi podatke in mu jih s tem pomagamo odkrivati ter analizirati. S sprotno analizo strokovnjak odkriva skupine in lahko s tem znanjem postavi strožje omejitve in tako hitreje pride do želene hipoteze.

Z združitvijo ABML in odkrivanja skupin z omejitvami dobimo metodo, ki ima naslednje lastnosti:

1. Neodvisna od načina vpeljave omejitev v algoritem odkrivanja skupin.
2. Upošteva strokovnjakove argumente določenega primera pri oblikovanju skupin.
3. Išče primere, s katerimi postavimo bolj informativne omejitve.
4. Išče primerne pare za postavljanje omejitev na podlagi strokovnjakovih vnešenih argumentov.

Iz navedenih lastnosti lahko izpeljemo naslednje prednosti:

1. Strokovnjak ima več svobode in alternativ, kako podajati omejitve.
2. Izoblikovane skupine postajajo vse bolj konsistentne s strokovnjakovim znanjem.
3. Možnost učinkovitega dodajanja ekspertovega znanja v domeno.
4. Izboljša izbor omejitev in s tem zmanjša njihov nabor.

1.3 Pregled magistrskega dela

Glavni prispevki magistrskega dela so predstavljeni v poglavjih 4, 5 in 6. Poglavji 2 in 3 pa vsebujeta uvod v sorodna dela.

- Poglavje 2 predstavlja opis in kratek pregled odkrivanja skupin z omejitvami. Poglavje vsebuje opis nekaterih algoritmov in njihove prednosti. Ob koncu poglavja naštejemo možnosti za izboljšave teh algoritmov.
- Poglavje 3 opisuje metodologijo argumentiranega strojnega učenja. Izpostavimo, kako argumentirano strojno učenje rešuje problem zajemanja znanja iz strokovnjaka in navedemo korake, potrebne za implementacijo metode v odkrivanje skupin.
- Poglavje 4 opiše našo implementacijo argumentiranega strojnega učenja v odkrivanje skupin z omejitvami. Znotraj poglavja razložimo, kako smo reševali določene ovire in kako smo prilagajali algoritem, da smo prišli do boljših rezultatov.
- Poglavje 5 predstavi aplikacijo, s katero smo vizualizirali delovanje predlagane metode in hkrati olajšali strokovnjakovo interkacijo z algoritmom.
- Poglavje 6 opisuje metodologijo opravljenih eksperimentov z uporabljenimi množicami podatkov in mer za evalvacijo rezultatov.
- Poglavje 7 predstavi rezultate eksperimentov.
- Poglavje 8 zaključi s končnimi ugotovitvami in možnimi izboljšavami.

Poglavje 2

Odkrivanje skupin z omejitvami

Med klasifikacijo in odkrivanjem skupin je glavna razlika v količini kontrole, ki jo ima uporabnik nad celotnim procesom. Pri klasifikaciji je celoten proces odvisen od primerov, ki jih poda uporabnik (zato je tehnika, s katero zajamemo te primere, zelo pomembna). Pri klasičnem odkrivanju skupin temu ni tako: vnos strokovnjakovega znanja je bolj problematičen, saj je omejen na vnos domenskega znanja preko izbora mere razdalj med primeri ali pa na korenite spremembe delovanja algoritma.

Zaradi te problematičnosti se je začela razvijati veja strojnega učenja, imenovana delno nadzorovano strojno učenje, katere namen je vpeljava domenskega znanja [12]. To uporabniku omogoči, da vodi proces odkrivanja skupin do bolj smiselnih razlag. Domensko znanje se vstavlja v obliki omejitev nad primeri. Te omejitve se ponavadi izražajo kot razmerje med dvema učnima primeroma. Razmerje poda informacijo o tem, ali dovolimo, da sta izbrana primera v isti skupini ali ne.

Čeprav so takšne omejitve ponavadi manj informativne kot postavljene oznake na učnih primerih pri klasifikaciji, so ob dovolj veliki količini omejitev in primerno izbranimi učnimi primeri prav tako koristne [2]. Kljub postavljanju pravil primerom, še vedno govorimo o procesu odkrivanja skupin. Ker pa teh skupin ne poznamo, je zato strokovnjaku lažje postaviti omejitve nad parom učnih primerov, kot napovedati morda nepoznano skupino nekega

primera.

2.1 Algoritmi za odkrivanje skupin

Odkrivanje skupin je ena izmed temeljnih nalog strojnega učenja. Njeni začetki segajo v leto 1950[19]. Do danes je bilo to področje obsežno raziskano in aktivno ter je postalo eno od najbolj razširjenih oblik avtomatskega nenadzorovanega učenja. Naloga odkrivanja skupin je poiskati skupine, v katerih si bodo primeri v istih skupinah čim bolj podobni in primeri v različnih skupinah čim bolj različni. V tem procesu algoritem uporabi le podatke iz baze podatkov. Oziroma bolj natančno: algoritem uporabi podatke, kakršne mu jih sami predstavimo.

Kot je rekel Vladimir Estivill–Castro, "Clustering is in the eye of the beholder" [20], kar je še kako pomembna trditev, saj je podobnost in s tem določanje skupin predvsem subjektivno in zato velikokrat ne dobimo pričakovanih in želenih rezultatov. Klasifikacija, ki je med najbolj razširjenimi metodami nadzorovanega učenja, je v tem pogledu njeno nasprotje. Pri klasifikaciji uporabnik točno ve, katere skupine so prisotne in algoritem s pomočjo primerov nauči, kako postaviti skupine. Tako ponavadi dobimo skupine, kot si jih zamislimo.

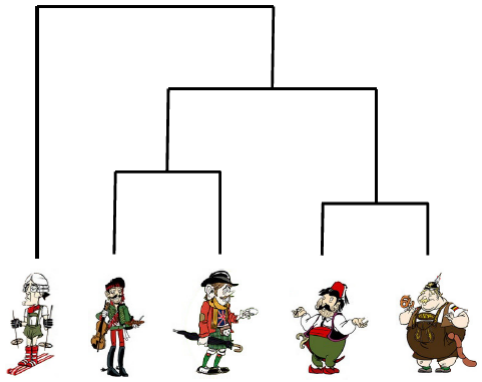
Kljub tolikšni raznolikosti domen in možnih končnih hipotez, imajo sodobni algoritmi za odkrivanje skupin nekaj skupnih zaželenih lastnosti:

- skalabilnost (tako glede časovne kot tudi prostorske zahtevnosti),
- zmožnost dela z različnimi tipi podatkov,
- le minimalno potrebno znanje o domeni,
- neobčutljivost na šum (angl. *noise*) in prisotnost osamelcev (angl. *outliers*),
- neodvisnost od podanega vrstnega reda učnih primerov,

- fleksibilnost (možnost vodenja algoritma k rešitvi),
- enostavnost interpretacije rezultatov, uporabnost.

2.1.1 Hierarhično odkrivanje skupin

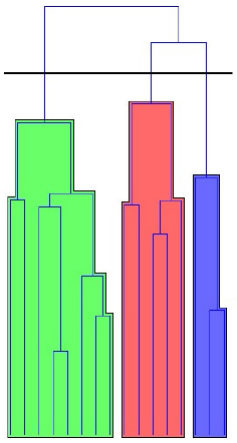
Hierarhično odkrivanje skupin primere razvrsti v skupine, ki se med seboj ne prekrivajo. Vsak primer pripada natanko eni skupini. Rezultat je organiziran v množico gnezdenih skupin, ki tvorijo hierarhično drevo.



Slika 2.1: Primer dendograma hierarhije skupin.

Pri hierarhičnem odkrivanju skupin lahko skupine ustvarimo od spodaj navzgor (angl. *bottom–up: agglomerative*), kjer na začetku vsak primer tvori svojo skupino, nato pa algoritem poišče najboljši par za združitev v novo skupino. Ta proces ponavlja, dokler niso vse skupine združene v eno. Drugi način delovanja je od zgoraj navzdol (angl. *top–down: divisive*), kjer na začetku vsi primeri tvorijo eno skupino, nato pa algoritem razdeli skupino po najboljši poti in rekurzivno nadaljuje postopek na obeh straneh, dokler ne pride do skupin, v katerih je le en sam primer, oziroma dokler ga ne ustavimo. Zaradi načina delovanja ta tip algoritmov ob vsaki iteraciji sam ustvari nove skupine in ni treba v naprej opredeliti števila skupin. Slika 2.2 prikazuje, kako preprosto je končni dendogram razrezati na želeno število skupin.

Pri večjem številu primerov se hitro prikaže problem skalabilnosti, saj so ti algoritmi ponavadi časovno zelo potratni. Prav tako ne vodijo nujno k



Slika 2.2: Razrez dendograma na tri skupine.

želeni rešitvi.

2.1.2 Metoda voditeljev: k-means

Metoda voditeljev (angl. *k-means*) [13] je najbolj razširjena metoda particijskega odkrivanja skupin. Zanje je prav tako, kot za hierarhično odkrivanje, značilno, da vsak primer spada v natanko eno skupino, ki se ne prekriva z ostalimi skupinami. Skupine se ne gradijo hierarhično, zato mora biti število skupin vnaprej znano. Voditelji (angl. *centroid*) določajo te skupine, oziroma so središča teh skupin, katerih lega se spreminja med izvajanjem algoritma. K-means je iterativni algoritem, ki išče distribucijo podatkov v skupine tako, da minimizira vrednost:

$$SSE(\Omega) = \sum_{i=1}^{K} \sum_{x \in C_i} d(m^{(i)} - x)^2$$

kjer je Ω končna rešitev algoritma (k skupin in pripadajoči primeri v vsaki skupini $\{\omega_1, \omega_2, ...\omega_k\}$ in $m^{(i)}$ je voditelj (centroid) skupine i. Algoritem 1 prikazuje psevdokodo k-means metode. Prvi korak algoritma je inicializacija skupin, kjer je vsaka od k skupin inicializirana s točko v prostoru skupin. Ta točka se imenuje *seme* (angl. *seed*) skupine. Popularna metoda inicializacije je izbor naključnih k točk iz podatkov, ki nato služijo kot seme.

Algoritem 1 k-means

Input: X = podatki, ki jih grupiramo; k = število skupin; maxI = maksimalno število iteracij

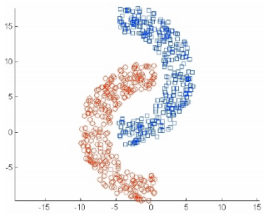
Output: $\Omega = \{\omega_1, \omega_2, .., \omega_k\}$ k skupin in pripadajoči primeri v vsaki skupini

1: **foreach** $\omega \in \Omega$ **do** Initializiraj(ω)
2: **while** konvergenca ni dosežena ali $i \geq maxI$ **do**
3: **foreach** $\omega \in \Omega$ **do** PonovnoIzračunajCentroid(ω)
4: **foreach** $x \in X$ **do**
5: $k \leftarrow argmin_{j \in 1..k}(\text{Distance}(x, \bar{\omega_j}))$
6: Assign(x, ω_k)

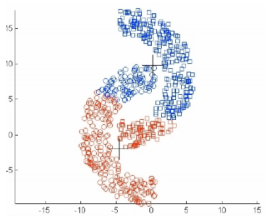
V vsaki iteraciji algoritem priredi vsak primer najbližjemu voditelju in nato izračuna nove voditelje. Ta postopek se ponavlja, dokler se voditelji ne ustalijo in se spreminjajo za dovolj majhno razliko. Ponavadi se iteracije prekinejo tudi, ko je doseženo maksimalno število iteracij brez konvergence, da ne pride do neskončne zanke.

Sama metoda je enostavna in zelo učinkovita, vendar ima poleg tega, da moramo prej nastaviti število skupin, tudi druge omejitve oziroma slabosti. Največja slabost tega pristopa je, da je občutljiva na začetna semena, ki jih postavimo. Dobro postavljena semena lahko pripeljejo algoritem do dobre in hitre rešitve, slabo postavljena semena pa bodo algoritem pripeljale do lokalnega minimuma. Žal je težko resnično vedeti *a priori*, kako postaviti dobra semena, zato je velikokrat odgovor na ta problem večkratna ponovitev delovanja algoritma.

K–means metoda ni odporna na šum ter na prisotnost osamelcev. Prav tako ni primerna za podatke, kjer imamo skupine, ki niso okrogle oblike (glej Sliko 2.3). Težave se tudi pojavijo, ko imamo skupine različnih velikosti in/ali gostot (glej Sliko2.4).

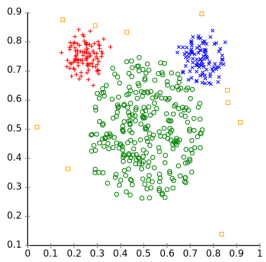


(a) Pravilna razporeditev.

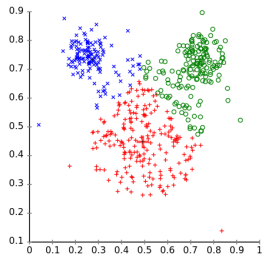


(b) Razporeditev z uporabo k-means.

Slika 2.3: Primer problema oblike skupin, ker skupine niso kroglaste oblike.



(a) Pravilna razporeditev.



(b) Razporeditev z uporabo k-means.

Slika 2.4: Primer problema nagnjenosti k-means metode k skupinam iste velikosti.

2.1.3 Mere razdalj

Tipično so učni primeri vektorji z n komponentami, ki jih imenujemo atributi ali značke. Preko teh atributov primerjamo učne primere. Če je rezultat meritve različnih učnih primerov večje število, potem se ta mera imenuje mera različnosti, če je število manjše, pa mera podobnosti. Mero podobnosti lahko pretvorimo v mero različnosti in obratno. Npr. z odštevanjem dejanske vrednosti z maksimalno vrednostjo mere podobnosti dobimo mero različnosti.

Izbor mere je pomembna odločitev, saj lahko izbor ustrezne mere znatno spremeni kvaliteto hipoteze. Kot smo že omenjali, je bil to eden redkih načinov, s katerim je uporabnik lahko vplival na rezultat odkrivanja skupin.

Naštejmo le nekaj popularnih metod za računanje razdalj:

- **Evklidska razdalja** nam prikaže razliko reprezentacije med dvema vektorjema. Ta mera se ponavadi uporablja za vektorje nizkih dimenzij (manj atributov) z zveznimi atributi. Recimo, da imamo podan primer

vektorjev d_i in d_j dolžine n. Potem je enačba evklidske razdalje:

$$D_{euc}(d_i, d_j) = \sqrt{\sum_{k=1}^{n} (\omega_{ik} - \omega_{jk})^2} \tag{2.1}$$

- **Kosinusna razdalja** opazuje kot med vektorjema. Manjši, kot je kot, bolj sta si vektorja podobna. Če je kot med vektorjema 0, potem sta identična, pri 1 pa sta si popolnoma različna. Posebnost kosinusne razdalje je, da dolžina vektorja ne vpliva na podobnost. Enačba kosinusne razdalje je:

$$D_{cos}(d_i, d_j) = 1 - \frac{\sum_{k=1}^{n}(\omega_{ik} \cdot \omega_{jk})}{\sqrt{\sum_{k=1}^{n} \omega_{ik}^2}\sqrt{\sum_{k=1}^{n} \omega_{jk}^2}} \tag{2.2}$$

- **Pearsonov korelacijski koeficient** se izračuna po enačbi:

$$sim_{pear}(d_i, d_j) = 1 - \frac{\sum_{k=0}^{k}(\omega_{ik} - \bar{\omega_{ik}}) \cdot \sum_{k=0}^{k}(\omega_{jk} - \bar{\omega_{jk}})}{\sqrt{\sum_{k=0}^{k}(\omega_{ik} - \bar{\omega_{ik}})^2} \cdot \sqrt{\sum_{k=0}^{k}(\omega_{jk} - \bar{\omega_{jk}})^2}} \tag{2.3}$$

  Ker je to dejansko mera podobnosti, lahko mero različnosti izračunamo po enačbi:

$$D_{pear} = \begin{cases} 1 - sim_{pear}(d_i, d_j), & \text{če } sim_{pear} \leq 0 \\ |sim_{pear}(d_i, d_j)|, & \text{če } sim_{pear} > 0 \end{cases}$$

2.2 Algoritmi za odkrivanje skupin z omejitvami

Z vpeljavo delno nadzorovanega učenja se je pojavilo veliko algoritmov za odkrivanje skupin s pomočjo omejitev. Pogosto ti algoritmi uporabljajo dve vrsti omejitev, ki sta jih uvedla Wagstaff in Cardle leta 2000 [11]:

- Pozitivne omejitve (angl. *must-link*, ML), ki pravijo, da morata biti dva primera v isti skupini.

- Negativne omejitve (angl. *cannot-link*, CL), ki pravijo, da ne smeta biti dva primera v isti skupini.

Strogost uveljavitve omejitev je odvisna od algoritma. Nekateri algoritmi postavljajo trde ali absolutne omejitve (angl. *hard constraints*) in ne bodo vrnili odkritih skupin, če npr. primera s pozitivno omejitvijo nista v isti skupini. Drugi pa postavljajo mehke omejitve (angl. *soft constraints*), s katerimi uravnavajo pravilnost omejitev s funkcijo kvalitete odkrivanja skupin. Tako so omejitve kot vodilo do pravilnega končnega odkrivanja skupin.

Kot je omenil Wagstaff [8], so pozitivne omejitve transitivne ($ML(a,b) \wedge ML(b,c) \rightarrow ML(a,c)$) in simetrične ($ML(a,b) = ML(b,a)$). Zaradi teh lastnosti lahko izpeljemo skupek pozitivnih omejitev, ki jih imenujemo soseščine (angl. *neighbourhoods*). Ob dovolj številnih in dobro postavljenih pozitivnih omejitvah bi na ta način dobili celote skupine.

Negativne omejitve so prav tako simetrične, toda niso transitivne. Wagstaff [8] omenja, da lahko z njimi kljub netransitivnosti izpeljemo druge negativne omejitve, ko jih upoštevamo skupaj s pozitivnimi omejitvami. Če se negativna omejitev navezuje na omejitev v soseščini, zgrajeni s pozitivnimi omejitvami, potem ta negativna omejitev velja za vse točke v tej soseščini pozitivnih omejitev (CL(a,b) $\wedge$ ML(b,c) $\rightarrow$ CL(a,c)).

2.2.1 k-means z omejitvami

Wagstaff je leta 2001 [8] predlagal razširitev popularne k-means metode z vpeljavo omejitev nad primeri. Z vpeljavo pozitivnih in negativnih omejitev nad primeri, k-means z omejitvami (angl. *Constrained k-means*, CKM) ne doda vedno vsakega primera najbližjemu centroidu, temveč pred tem preveri, ali bi s tem kršil kakšno omejitev. Če se s tem prekrši katera od omejitev, se tega primera ne doda k skupini. Če se katere omejitve prekrivajo tako, da kakšne omejitve ne more upoštevati, potem nimamo sprejemljive rešitve.

Algoritem 2 prikazuje psevdokodo CKM. V vrstici 6 vidimo, da pred dodelitvijo primera k najbližji skupini algoritem preveri, da učni primer x ne

Algoritem 2 Constrained k-means

Input: X = podatki, ki jih grupiramo; k = število skupin; maxI = maksimalno število iteracij; ML in CL omejitve, ki jih želimo upoštevati

Output: $\Omega = \{\omega_1, \omega_2, .., \omega_k\}$ k skupin in pripadajoči primeri v vsaki skupini

1: **foreach** $\omega \in \Omega$ **do** Initializiraj(ω)
2: **while** konvergenca ni dosežena ali $i \geq maxI$ **do**
3: **foreach** $\omega \in \Omega$ **do** PonovnoIzračunajCentroid(ω)
4: **foreach** $x \in X$ **do**
5: $k \leftarrow argmin_{j \in 1..k}(\text{Distance}(x, \bar{\omega_j}))$
6: $\neg$ krsiOmejitev(x, ω_j)
7: **if** $\nexists$ k **then** odkrivanje skupin je neuspešno
8: **else**Assign(x, ω_k)

9: **function** KRSIOMEJITEV(x, ω, ML, CL)

Input: x = učni primer; ω = skupina; ML = pozitivne omejitve; CL = negativne omejitve; $x^{'}$ = učni primer za katerega je podana omejitev

Output: ali vstavljanje x v skupino ω, krši kakšno omejitev

10: **foreach** $(x, x^{'}) \in ML$ **do**
11: **if** $x^{'} \notin \omega$ **then** return true
12: **foreach** $(x, x^{'}) \in CL$ **do**
13: **if** $x^{'} \in \omega$ **then** return true

krši nobene od pozitivnih ali negativnih omejitev. Če krši katero od omejitev, se točke ne postavi v to skupino in se preveri naslednjo skupino.

Kot je razvidno iz psevdokode Algoritma 2, algoritem postavlja trde omejitve. Zaradi načina vpeljave teh omejitev lahko hitro nastanejo problemi. Vsi učni primeri, ki so povezani s pozitivnimi omejitvami, bodo slepo postavljeni v skupino prvega učnega primera s pozitivno omejitvijo. Pri tem ne upošteva podobnosti s centroidi. Lahko se zgodi tudi, da dobimo učni primer, ki ima negativno omejitev na vse skupine. V tem primeru je nadaljnje odkrivanje skupin nemogoče in se algoritem ustavi. Temu bi se sicer lahko izognili, če bi učne primere pregledovali v drugačnem vrstnem redu.

2.2.2 K-means s parnimi omejitvami

K-means s parnimi omejitvami (angl. *Pairwise Constrained k-means*, PCKM) [14] prav tako temelji na izboljšavi osnovnega k-means algoritma s pomočjo omejitev. Za razliko od CKM ta metoda uporablja pozitivne in negativne omejitve tudi pri inicializaciji algoritma.

V začetnem koraku uporabnik algoritma PCKM določi omejitve, s pomočjo katerih se inicializira skupine po naslednjem postopku. Z uporabo transitivne lastnosti omejitev algoritem pridobi soseščine točk, ki smo jih omenjali v poglavju 2.2. Od teh soseščin algoritem vzame k največjih in jih uporabi za inicializacijo skupin. Če je teh soseščin manj kot k, potem se ustvari novo skupino tako, da izbere točko, če ta seveda obstaja, ki ima negativne omejitve z vsemi skupinami. Če ta učni primer ne obstaja ali je takih primerov premalo, se ostale skupine inicializira s pomočjo perturbacije.

V svojem delu avtorji predlagajo funkcijo $\mathcal{J}$, katera se med procesom odkrivanja skupin minimizira. Motivacijo za novo funkcijo so avtorji iskali s pomočjo postavitve naključnega polja Markova čez podatke. Iskanje maksimalne *a posteriori* verjetnosti so posplošili na minimiziranje funkcije $\mathcal{J}_{pckm}$:

$$\mathcal{J}_{pckm}(\Omega) = \frac{1}{2}\sum_{i=1}^{k}\sum_{x\in\omega_i}\|x\bar{\omega_i}\|^2 + \sum_{(x_i,x_j)\in ML}\omega_{ij}\mathbb{1}[l_i \neq lj] + \sum_{(x_i,x_j\in CL}\bar{\omega_{ij}}\mathbb{1}[l_i = l_j] \tag{2.4}$$

Prvi del funkcije 2.4 je SSE kriterij kompaktnosti skupin iz osnovnega k-means algoritma. Drugi del funkcije 2.4 predstavlja utež, ki kaznuje pozitivne omejitve, če ob osnovni izvedbi k-means omejitev ne drži. To pomeni, da če imata točki x_i in x_j, s pozitivno omejitvijo med sabo, različen razred, potem je treba dodati utež ω_{ij}. Tretji del funkcije 2.4, podobno kot prejšni, opazuje negativne omejitve med pari in postavi w_{ij} utež za vsako kršeno negativno omejitev med izvedbo osnovnega k-means.

Iz psevdokode 3 je razvidno, da so v vrstici 5 spremenili dodelitev točke najbližji skupini zamenjali dodelitev točke najbližji skupini z vključevanjem prej omenjenih uteži in tako vključili postavljene omejitve nad učnimi primeri.

PCKM algoritem je glavni predstavnik odkrivanja skupin z ročnim pridobivanjem omejitev. Avtorji PCKM algoritma so se posvetili optimiziranju sheme za pridobivanje omejitev in s tem optimizirali število potrebnih omejitev. V svojem delu predlagajo iskanje parov z implementiranjem metode "iskanje najbolj oddaljenega najprej" (angl. *farthest–first traversal search*), ki bi bili bolj informativni kot naključni pari. Osnovna ideja je, da algoritem najde k točk, ki so najbolj oddaljene druga od druge.

Algoritem, ki so ga poimenovali Explore, najprej izbere naključno točko, nato pa poišče najbolj oddaljeno točko od te točke. Obe točki doda k skupini potencialnih točk. Nato poišče novo točko, ki je najbolj oddaljena od te skupine in jo prav tako doda k potencialnim točkam. Tako dobimo približek k voditeljev. Na podoben način se je s to metodo reševalo tudi problem skalabilnosti hierarhičnega odkrivanja skupin. Ko metoda z *Explore* algoritmom dobi skeletno strukturo skupin, se metoda prestavi v fazo *Consolidate*.

Bistvo *Consolidate faze* je, da sedaj, ko imamo skeletno strukturo, lahko s $(k-1)$ omejitvami določimo soseščino katerekoli naključne točke. Algoritem

Algoritem 3 Pairwise Constrained k-means

Input: X = podatki, ki jih grupiramo; k = število skupin; ML in CL omejitve, ki jih želimo upoštevati; $\omega^{'}$ = utež vsake omejitve

Output: $\Omega = \{\omega_1, \omega_2, .., \omega_k\}$ k skupin in pripadajoči primeri v vsaki skupini

1: **foreach** $\omega \in \Omega$ **do** InicializirajPCKM(k,X, ML, CL)

2: **while** konvergenca ni dosežena ali $i \geq maxI$ **do**

3: **foreach** $\omega \in \Omega$ **do** PonovnoIzračunajCentroid(ω)

4: **foreach** $x \in X$ **do**

5: $k \leftarrow argmin_{j \in 1..k}(\frac{1}{2}\left\|x - \omega_k^{'}\right\|^2 + DodeliKazni(x, \omega, ML, CL, \omega^{'}))$
Assign(x, ω_k)

6: **function** InicializirajPCKM(k,X, ML, CL)

Input: X = podatki, ki jih grupiramo; k = število skupin; ML in CL omejitve, ki jih želimo upoštevati

$N_1, ...N_z \leftarrow$ PridobiSoseščineRazvrščenePoVelikosti(ML,CL)

7: **if** $k \nleq z$ **then**

8: **foreach** $i \leftarrow 1$ do k **do** Inicializiraj(ω_i,Centroid(N_i))

9: **else**

10: **foreach** $i \leftarrow 1$ do z **do** Inicializiraj(ω_i,Centroid(N_i))

11: **if** x omejen z CL s celo skupino N_i **then** Inicializiraj(ω_{z+1}, x)

12: **foreach** ω, ki še vedno ni inicializiran **do** Inicializiraj(ω,RandomPermuntation(Centroid(X)))

13: **foreach** $(x, x^{'}) \in ML$ **do**

14: **if** $x^{'} \notin \omega$ **then** $p \leftarrow p + \omega^{'}$

15: **foreach** $(x, x^{'}) \in CL$ **do**

16: **if** $x^{'} \in \omega$ **then** $p \leftarrow p + \omega^{'}$

return p

17: **function** DodeliKazni($x, \omega, ML, CL, \omega^{'}$)

Input: x = podatkovni primer; ω = skupina; ML in CL = omejitve, ki jih želimo upoštevati; $\omega^{'}$ = utež vsake omejitve;$x^{'}$ = učni primer za katerega je podana omejitev

Output: p = utež, ki služi kot kazen, če je x postavljen v skupino ω $p \leftarrow 0$

18: **foreach** $(x, x^{'}) \in ML$ **do**

19: **if** $x^{'} \notin \omega$ **then** $p \leftarrow p + \omega^{'}$

20: **foreach** $(x, x^{'}) \in CL$ **do**

21: **if** $x^{'} \in \omega$ **then** $p \leftarrow p + \omega^{'}$

return p

consolidate izbere naključno točko in vpraša uporabnika za vsako skupino, če naključna točka spada v to skupino ali ne. Ko uporabnik določi pozitivno omejitev, algoritem vstavi to točko v pravilno skupino. Če uporabnik nastavi $(k-1)$ negativnih omejitev, se avtomatsko predpostavi, da točka spada v zadnjo skupino.

2.2.3 Ostali algoritmi za odkrivanje skupin z omejitvami

Algoritmov za odkrivanje skupin z omejitvami je zares veliko (in tudi vedno več), zato je težko narediti njihov izčrpen seznam in jih opisati. V nadaljevanju magistrske naloge opišemo še nekaj algoritmov za odkrivanje skupin z omejitvami, za nazornejši prikaz raznolikosti področja.

Normaliziran rez z omejitvami

Normaliziran rez z omejitvami(angl. *Constrained Normalised Cut*, CNC) je algoritem, ki temelji na spektralnem odkrivanju skupin [16]. Govorimo o družini algoritmov, ki problem odkrivanja skupin preslikajo v problem rezanja grafov. Iz podatkov se ustvari utežen graf, kjer je vsako oglišče (angl. *vertex*) točka iz podatkov ($V = \{v_1, v_2, ..., v_n\}$) in vsaka stranica (angl. *edge*)

utež ($W = \{\omega_1, \omega_2, ..., \omega_n\}$), ki predstavlja podobnost med točkami. Podobnost se računa z uporabo preproste evklidske razdalje ali s čim bolj kompleksnim, kot je glajenje z Gaussom (angl. *Gaussian smoothing*). Tudi povezave so lahko z najbližjimi sosedi ali s točkami, katerih podobnost je nad neko vrednostjo. Ko je graf zgrajen, se naloga odkrivanja skupin spremeni v iskanje dobrega prereza grafa. To pomeni, da se minimizira uteži stranic, ki povezujejo oglišča v različnih skupinah in maksimizira uteži stranic oglišč, ki povezujejo oglišča znotraj skupine. Ena od bolj popularnih metod rezanja grafa je Normaliziran rez. Predstavila sta jo Shi in Malik leta 2000 [17].

Leta 2006 sta Ji in Xu [9] predlagala normaliziran rez z omejitvami, ki minimizacijski funkciji za rezanje grafa dodata nov cilj. Ne le, da poskuša funkcija minimizirati uteži podobnosti med oglišči, z namenom dobiti dobre skupine, temveč tudi upošteva omejitve, ki jih dodeli uporabnik. Algoritem zapiše vse omejitve v matriko U, katera ima n vrstic. Za vsak primer svojo vrstico in kolono za vsako omejitev, ki pove, ali želimo točki i in j v isti skupini ali ne, z vrednostima 1 in -1. To matriko se množi z matriko H, ki hrani informacijo elementov točk skupine, s pomočjo Forbeniusove norme. Forbeniusova norma je definirana kot kvadratni koren vsote absolutnih kvadratov elementov matrike:

$$\|H\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2} \tag{2.5}$$

Tako dobimo manjše uteži za elemente z več omejitvami in večje, ko je omejitev manj.

Constrained Complete Link

Constrained Complete Link je del hierarhične družine algoritmov, ki je bil predlagan leta 2002 [18]. Ta družina algoritmov temelji na od spodaj navzgor (angl. *bottom–up*) hierarhičnem odkrivanju skupin, kjer je na začetku vsak primer svoja skupina. Te skupine algoritem ob vsaki iteraciji združi z drugo najbližjo skupino, dokler ne ostane le še ena skupina. Po končanih iteracijah

se po potrebi določi število skupin.

Klein [18] je leta 2002 predlagal nadgradnjo, ki podpira pozitivne in negativne omejitve. Omejitve se uvede na naslednji predlagan način. Če imata dve točki x_i in x_j pozitivno omejitev, algoritem postavi razdaljo med njunima skupinama na 0. Podobno, če imata točki negativno omejitev, se razdalja med njunima skupinama postavi na najvišjo možno vrednost. Avtor opozori, da z vpeljavo takšnih omejitev lahko izgubimo metričnost prostora (npr. trikotniška neenakost ne velja več). V primeru le pozitivnih omejitev je mogoče metričnost popraviti z računanjem poti med vsakim parom v novem metričnem prostoru, ki ima najkrajše poti in tako ohranimo metričnost prostora. Predpostavlja se, da je pozitivnih omejitev veliko manj kot dejanskih primerov v podatkih, zato je računanje najkrajše poti sprejemljivo, medtem ko pri negativnih omejitvah avtorji trdijo, da ni treba popravljati njihovega učinka, ker je računska zahtevnost previsoka, nudi pa premajhne izboljšave.

2.3 Prednosti odkrivanja skupin z omejitvami

Omejitve so priročen način vpeljave ekspertovega znanja v odkrivanje skupin, ki bi pri klasičnem odkrivanju skupin ostalo neuporabljeno. Omogočajo lahek in enoten način vstavljanja namigov želene postavitve podatkov v skupine, ne glede na domeno.

Za primer vzemimo stranke neke trgovine. Navadno bi vsako stranko prikazali kot vektor z realnimi vrednostmi, pridobljenimi s pomočjo lastnosti kupljenih izdelkov. Podobnost med njimi bi merili z mero, kot je kosinusna mera podobnosti. Toda v algoritem bi lahko vstavili tudi dodatne zunanje informacije, npr. uro nakupa, saj bi lahko trdili, da ob določenih urah stranke kupujejo bolj podobe artikle.

Če bi želeli to novo informacijo vstaviti v klasičen algoritem za odkrivanje skupin, bi morali spremeniti reprezentacijo primerov ali računati podobnosti na *ad–hoc* način, kar velikokrat privede v nezaželene stranske učinke in je zelo slabo prilagodljivo z drugimi informacijami, ki bi jih morda želeli vpeljati. Z

omejitvami te probleme zaobidemo, saj lahko informacijo o uri zapišemo v obliki pozitivnih in negativnih omejitev, s čimer bi dobili želeni učinek, hkrati pa imamo dovolj prostora za dodajanje drugačnih informacij. Ker nam omejitve podajo relativno veliko informacij, lahko zato z dobro izbranimi primeri dokaj hitro pridemo do vidnih sprememb v rezultatu odkrivanja skupin.

2.4 Slabosti in priložnosti za izboljšave

Odkrivanje skupin z omejitvami je relativno nova tema, vendar prikazuje dobre rezultate [2]. Kljub veliki raziskanosti področja, pa obstajajo problemi, ki nudijo zanimive raziskovalne priložnosti. Glavni izivi, ki se pri odkrivanju skupin z omejitvami pojavljajo, so:

- pridobivanje omejitev,
- robustnost algoritmov,
- izvedljivost omejitev,
- koristnost omejitev.

Pridobivanje omejitev

Do danes je bilo raziskovanje usmerjeno zlasti v razvijanje novih algoritmov. Ker je bil cilj teh del ustvariti metode, ki bodo čim bolj izčrpno uporabile informacije pridobljene z omejitvami, so eksperimenti v člankih predstavljeni z že vnaprej nastavljenimi omejitvami. V skoraj vseh primerih so te omejitve nastavljene z uporabo zlate resnice (angl. *golden truth*). V problemih vsakdanjega življenja to predstavlja veliko težavo, saj nikoli nimamo podanih skupin in moramo omejitve šele pridobiti s pomočjo strokovnjaka.

Pri ročni metodi pridobivanja omejitev strokovnjak označi, ali dana para sodita v isto skupino ali ne. Ker gre za dolgotrajen in intelektualno intenziven proces, je velik izziv pridobiti koristne omejitve in posledično tudi informacije

iz uporabnika. Ta problem je poskusil rešiti Basu leta 2004 [14]. Z uporabo aktivnega učenja je razvil metodo PCKM, ki smo jo opisali v poglavju 2.2.2.

V primeru avtomatskega pridobivanja omejitev je naloga še težja, saj algoritem sam postavlja omejitve le s strokovnjakovim splošnim domenskim znanjem. Gre za proces, pri katerem strokovnjak išče povezanost primerov, kar je osnovna naloga odkrivanja skupin. V ta namen se lahko uporabi zunanje informacije ali informacije, ki so že podane v podatkih, pri katerih želimo odkriti skupine. Kot primer avtomatskega pridobivanja omejitev, je Wagstaff [8] pri reševanju problema iskanja cestnih pasov z globalnim sistemom pozicioniranja (angl. *Global Positioning System*, GPS) postavil omejitve glede na medsebojno oddaljenost točk (pravokotno na sredino ceste). Postavil je splošno pravilo, s katerim opredeli negativno omejitev med dvema točkama takrat, ko sta ti dve točki med seboj oddaljeni za vsaj štiri metre.

Robustnost algoritmov

Za vse omejitve pri uporabi že znanih razredov učnih podatkov, ki so postavljene pri testiranju odkrivanja skupin z omejitvami, je značilno, da so resnične in brez napak. Dve točki, ki imata pozitivno omejitev, resnično spadata v isto skupino, in točki, ki nosita negativno omejitev, resnično ne spadata v isto skupino. V vsakdanji uporabi, ko je treba te omejitve pridobiti, takšnih idealnih pogojev ni. Ko se omejitve pridobiva ročno, so napake lahko posledica napačne presoje. Odkrivanje skupin je večinoma uporabljeno za raziskovanje podatkov in uporabnik tipično ne pozna podatkov in njihovih struktur dovolj dobro. Avtomatsko generirane omejitve ponavadi delujejo tako, da posplošujejo eksplicitne pojme domene, npr. objava novic mora biti v isti skupini, če je avtor isti. Ko se posplošuje pojme, pa je jasno, da posplošitev ne bo pravilna za čisto vsak primer.

Izvedljivost omejitev

Problem predstavlja tudi izvedljivost omejitev. Ko postavimo omejitev, spremenimo prostor rešitev in usmerimo algoritem v iskanje rešitev, ki so skla-

dne z omejitvami. Davidson in Ravi [22] sta izpostavila dvomljivost obstoja takšne rešitve. Ta dvom je upravičen predvsem, kadar imamo nekonsistentne omejitve, npr. ko imamo omejitev $ML(x, y)$ in hkrati $CL(x, y)$. Toda ta nerešljivost se lahko pojavi tudi v primerih, ko imamo konsistentne omejitve. To se zgodi, ko algoritem ne more več zadovoljiti omejitvi, ne da bi kršil druge omejitve. V primeru, ko imamo $k = 2$ in poskušamo odkriti skupine podatkovne množice $X = \{x_1, x_2, x_3\}$, je nemogoče priti do rešitve z zadovoljitvijo omejitev, ko so te: $ML(x_1, x_2), ML(x_2, x_3), CL(x_3, x_1)$.

Koristnost omejitev

Ponavadi, ko preverjamo rezultate algoritmov odkrivanj skupin z omejitvami, algoritem večkrat preizkusimo in rezultate preizkusov povprečimo. Leta 2006 je Davidson [21] izpostavil, da čeprav v povprečju naključne omejitve izboljšajo odkrivanje skupin, v nekaterih primerih omejitve dejansko poslabšajo rezultat. Rezultat je lahko celo slabši kot bi bil brez omejitev. To se zgodi, ko pozitivne omejitve postavijo dva primera v isti razred, vendar je sam razred napačen, saj ne podamo oz. nimamo nobene informacije o pravilnosti razreda.

V odgovor je Wagstaff predlagal dve meritvi: informacija in koherenca [21]. Informacija meri količino informacije, ki jo nosi omejitev, katere algoritem sam ne zazna. Omejitve nad dvema točkama z malo informacije ne pripomorejo ničesar k odkrivanju skupin, če algoritem že zazna povezavo med njima. Koherenca pa meri, koliko se omejitve strinjajo z razdaljo. Ta mera nam sporoči, kolikšna je verjetnost, da točke v bližini pozitivne omejitve niso negativne in obratno.

Poglavje 3

Argumentirano strojno učenje

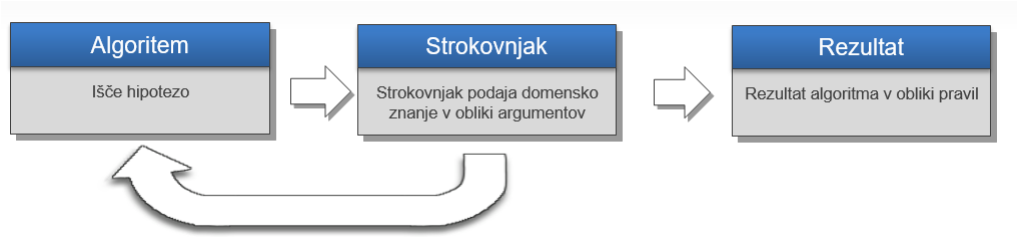
Pri zajemanju znanja iz strokovnjaka se ponavadi zahteva splošno domensko znanje od strokovnjaka, kar je večinoma težka naloga. To je problem, imenovan Feigenbaumovo ozko grlo [26]. Kljub mnogim raznovrstnim rešitvam [27] [28] [29] z opazovanji in analogijami, da bi zajeli čim več strokovnjakovega znanja, ostaja problem nerešen [30]. V iskanju rešitve ozkega grla pri pridobivanju podatkov (angl. *knowledge aquisition bottleneck*) je nastala nova metoda, za zajemanje znanja iz eksperta, imenovana argumentirano strojno učenje (angl. *argument-based machine learning*, ABML) [25]. S pomočjo argumentiranja, ABML omogoči strokovnjaku, da na naraven način vključi svoje znanje. Prav tako vodi do bolj razumljivih modelov, ki so konsistentni z znanjem eksperta [31].

ABML se je v preteklosti izkazal kot dober način izboljšanja natančnosti in razumljivosti obstoječih metod [25]. Pri črpanju znanja iz strokovnjaka je razumljivost še posebej pomembna, saj modelom, ki so nerazumljivi, ponavadi ne zaupamo in jih raje ne uporabimo. Velikokrat pridemo do hipotez, ki imajo znanje organizirano in izraženo v obliki, ki ni po želji strokovnjaka. Z vpeljavo argumentov v strojno učenje, omogočimo strokovnjaku, da artikulira svoje znanje tako, da bo čimbolj izčrpno za učno metodo, hkrati pa bo konsistentno z njegovim eksplicitnim znanjem.

ABML uporabi argumente za izboljšanje učenja iz podanih učnih prime-

rov [25]. Strokovnjak z argumenti razloži, zakaj se strinja oziroma zakaj se ne strinja z neko trditvijo. S trditvijo, ki jo strokovnjak argumentira, pomagamo vključiti njegovo splošno znanje. Ker ostaja nevarnost pretiranega prilagajanja (angl. *overfitting*), ponudimo strokovnjaku protiprimere, s katerimi utrdi svoj argument in zmanjša verjetnost pretiranega prilagajanja. S tem vodi algoritem do hipotez, ki so bolj jasne, in zavrže hipoteze, ki so neskladne z argumenti.

Diagram na sliki 3.1 prikazuje interakcijo strokovnjaka z algoritmom.



Slika 3.1: Potek ABML.

Argument predstavlja razlago za trditev le enega primera, ki je problematičen. S tem se osredotočimo le na en primer, zaradi česar strokovnjak lažje artikulira svoje znanje. Lažje je vnašati znanje za posamezen primer z argumenti, kot pa vnašati znanje za celotno domeno podatkovne baze. Ker strokovnjak argumentira le problematičen del podatkovne baze, zagotavlja bolj relevantno znanje. S pomočjo protiprimerov strokovnjak najde pomankljivosti svojega argumenta in ga nadgradi.

Primer težavnosti zajemanja znanja iz strokovnjaka je komentiranje šahovske igre. Strokovnjak s težavo posploši poteze šahovske igre in jih opredeli kot dobre ali slabe. Lažje se osredotoči na posamezno potezo in z argumenti izpostavi relevantne dejavnike ter njihov vpliv na kvaliteto poteze [31].

3.1 Paradigma argumentiranega strojnega učenja

V nadaljevanju prikažemo klasifikacijski primer, ki nazorno predstavi delovanje in potek ABML. Klasifikacija v klasični obliki kot opisana v poglavju

1, se loti problema na sledeč način:

- z učnimi primeri poišči pravila za umestitev primerov v razrede,
- vmesti nadaljnje primere v razrede z dobljenimi pravili.

V našem primeru se klasifikacijska metoda uči o odobritvi kredita. Vsak učni primer predstavlja atribute, pridobljene ob prijavi stranke za odobritev kredita. Podana je tudi odločitev uslužbenca o uspešnosti odobritve, kar predstavlja razred učnega primera. Vsaka stranka ima naslednje atribute: Ime, RedniPlačnik (vrednosti da ali ne), Premožen (vrednosti da ali ne), StatusRačuna (vrednosti pozitiven ali negativen), BarvaLas (vrednosti črna, blond, rjava ...) in KreditOdobren (vrednosti da, ne).

Tabela 3.1 prikazuje pet primerov naših učnih podatkov. Tipično bi se program za učenje pravil naučil naslednje pravilo:

ČE *BarvaLas = blond* POTEM *KreditOdobren = da* DRUGAČE *KreditOdobren = ne*

Pravilo je s stališča strojnega učenja dobro, ker je kratko in pravilno za vse učne primere. Vendar, strokovnjaku takšno pravilo ne bo smiselno. Z vpeljavo argumentiranja lahko dobimo bolj smiselna pravila. Za ponazoritev argumentiranja predpostavimo, da strokovnjak poda argument za Gdč. Bevk: "Gdč. Bevk ima odobren kredit, ker je redni plačnik mesečnih obro-

Tabela 3.1: Primeri podatkov odplačevanja kreditov.

Ime	RedniPlačnik	Premožen	StatusRačuna	BarvaLas	KreditOdobren
G. Bond	ne	da	negativen	blond	da
G. Medved	ne	ne	pozitiven	rjava	ne
Gdč. Bevk	da	ne	pozitiven	blond	da
Ga. Bogataj	da	da	pozitiven	blond	da
G. Novak	da	ne	negativen	črna	ne

kov." S takšnim argumentiranim primerom se učni problem spremeni v problem iskanja hipoteze konsistentne z učnimi primeri in njihovimi argumenti. Prejšnje pravilo sedaj ni več konsistentno z ekspertovim argumentom – "G. Bond je blond, vendar ni redni plačnik". Metoda strojnega učenja z argumentiranjem tako izpelje novo pravilo v skladu z argumentom:

Če *RedniPlačnik* = *da* IN *StatusRačuna* = *pozitiven*
POTEM *KreditOdobren* = *da* DRUGAČE *KreditOdobren* = *ne*

Algoritem ABML bo nato poskušal med učenjem posplošiti strokovnjakov argument tako, da ohrani konsistentnost z drugimi učnimi primeri. S tem dobimo nadgradnjo strokovnjakovega argumenta, ki vključuje trditev, da mora stranka imeti tudi pozitiven status računa.

S takšnim pravilom uvrstimo Gdč. Bevk v pravilen razred, hkrati pa je pravilo bolj skladno s strokovnjakovim znanjem. Prednosti takšnega učenja so:

- Argumentiranje posamičnega primera omogoči lažje izražanje domenskega znanja, s tem se tudi izognemo iskanju argumentov, ki bi veljali za celotno domeno.
- Z argumentiranjem zmanjšamo število možnih hipotez in s tem zmanjšamo kompleksnost iskanja boljše rešitve.
- Naučena hipoteza vsebuje strokovnjakovo razlago in je s tem bolj smiselna.

3.2 Interaktivna zanka za zajemanje ekspertnega znanja

ABML je sestavljen iz dveh glavnih delov:

- spremenjen algoritem strojnega učenja, ki uporablja argumente,

- iterativna ABML zanka, ki upravlja interakcijo med strokovnjakom in algoritmom.

Ponavadi je algoritem tisti, ki mu pripisujejo vse zasluge za uspešno delovanje metode, toda iterativna zanka za zajemanje strokovnjakovega znanja (angl. *knowledge refinement loop*) je vsaj tako pomembna kot algoritem [31]. Je metoda, ki jo strokovnjak uporablja med procesom zajemanja znanja, medtem pa je delovanje algoritma skrito v ozadju. V nadaljevanju prikažemo prednosti takšne interakcije med strokovnjakom in algoritmom strojnega učenja. Sledeči koraki povzamejo potek ABML zanke zajemanja znanja iz strokovnjaka:

1. Nauči se hipotezo z uporabo podanih podatkov.

2. Poišči najbolj kritičen primer in ga predstavi strokovnjaku. Če kritičnega primera ne najdemo, ustavi postopek.

3. Strokovnjak razloži kritični primer s podajanjem argumentov.

4. Ponovi prvi korak.

V nadaljevanju razložimo kritični primer, kako ga izbrati in kako podati argumente, da elicitiramo strokovnjakovo znanje za izbrani primer.

V iskanju kritičnih primerov

Glavna lastnost kritičnih primerov je, da jih dobljena hipoteza ne zna dobro razložiti. V dosedanjih implementacijah ABML se je iskanje kritičnih primerov izvedlo z računanjem statistične napake preko k-kratnega prečnega preverjanja (angl. *k–fold validation*) ponovljenega n krat (e.g. $n = 4, k = 5$) [25] [31] [33]. Primer, ki ima najvišjo statistično napako, je najslabše razložen v naši hipotezi in ga zato poimenujemo kritični primer.

Iskanje boljših argumentov

S podanimi argumenti vodimo učno metodo do boljše razlage kritičnih primerov, katere metoda ne zna dobro razložiti [32]. Strokovnjakove argumente je ponavadi treba dopolniti. Kadar metoda ne zmore sama dopolniti strokovnjakovega argumenta, ga mora strokovnjak dopolniti s pomočjo protiprimerov. Protiprimeri so primeri, ki so v različnih razredih od kritičnega primera in hkrati sovpadajo s strokovnjakovimi argumenti. Proceduro enega koraka argumentiranja lahko opišemo z naslednjimi koraki:

Korak 1: Razlaganje kritičnega primera. V tem koraku strokovnjaku predstavimo kritični primer in mu postavimo vprašanje, "Zakaj je ali zakaj ni ta primer v podanem razredu?". Odgovor je seznam argumentov $A_1, ..., A_k$, ki potrjujejo strokovnjakovo trditev. Ko je primer pretežak za argumentiranje, je odgovor strokovnjaka lahko tudi, "Ne vem". S tem zaključimo korak argumentiranja in poiščemo nov kritični primer.

Korak 2: Dodajanje argumentov primeru. Argumente, kot jih poda strokovnjak, je treba pretvoriti v domenski jezik (atribute). Ko so trditve argumentov podane z znanimi atributi, se te trditve preprosto dodajo primeru. Ko je trditev v obliki koncepta, kot je npr. razmerje dveh atributov in ta koncept še ni v domeni, je treba pred dodajanjem argumenta dodati še ustrezen atribut.

Korak 3: Odkrivanje protiprimerov. Protiprimere se uporabi za dopolnjevanje argumentov, ki so nepopolni in jih ABML ne zna dopolniti. Protiprimere iščemo v razredih, ki so različni od kritičnega primera in ga hkrati pokrivamo s pravilom, nastalim iz argumenta.

Korak 4: Izboljšanje argumentov. Strokovnjak mora sedaj pregledati svoj začetni argument upoštevajoč protiprimer. Ta korak je podoben prvemu in drugemu s korenito spremembo. Strokovnjaka tokrat vprašamo, "Zakaj je kritični primer v tem razredu in protiprimer

v drugem?" Njegov odgovor se doda začetnemu argumentu.

Korak 5: vrnitev na tretji korak, če je najden nov protiprimer.

V našem primeru odobritve kredita iz poglavja 3.1 bi z argumentom "Gdč. Bevk ima odobren kredit, ker je status računa pozitiven.", prišli do nepopolnega argumenta. ABML bi vrnil protiprimer: "G.Medved ima pozitiven račun, toda nima odobrenega kredita." Strokovnjak bi na poziv protiprimera izboljšal svoj prvotni argument z odgovorom: "Gdč. Bevk je redni plačnik računov." S takšnim postopkom ABML pride do pravila s 100-odstotno natančnostjo:

ČE *StatusRačuna = pozitiven* IN *RedniPlačnik = da* POTEM *OdobrenKredit = da*

3.3 Možnosti interakcije z argumentiranjem pri odkrivanju skupin

Iterativna ABML zanka 3 in odkrivanje skupin z omejitvami si delita skupen cilj: izboljšanje osnovne implementacije algoritma s podajanjem omejitev na konkretne primere. Kot omenjeno v poglavju 1.1, je odkrivanje skupin z omejitvami odlično orodje za izboljšanje obstoječih algoritmov, ki pa ima do sedaj slabo raziskane rešitve za elicitacijo znanja iz strokovnjaka. Zaradi narave interaktivne zanke za zajemanje znanja je ABML možna rešitev tega problema.

S kombinacijo teh dveh metod dobimo novo vejo ABML, imenovano odkrivanje skupin z argumentiranjem 4, kjer zajem podatkov z interaktivno zanko prilagodimo odkrivanju skupin. Dosedanje metode argumentiranega strojnega učenja so imele naloge klasifikacije, kjer so skupine učnih primerov znane. Pri kombiniranju metod ABML in odkrivanja skupin, moramo paziti na določene težave, ki nastanejo, ker ne poznamo skupin učnih primerov.

Čeprav morda poznamo skupine, ki jih želimo videti kot rezultat odkrivanja skupin, je razlikovanje med skupinami prav tako težka naloga. Zato argumenti argumentiranega strojnega učenja v obliki, kot so predstavljeni v poglavju 3.1, niso primerni.

Težave, ki nastanejo pri združitvi metod ABML in odkrivanja skupin, so:

- **Skupin ne poznamo**, kar oteži argumentranje primerov po načinu metode klasičnega ABML. Strokovnjak zato precej težje argumentira, zakaj primer spada v določeno skupino.

- **Razločevanje skupin** je za strokovnjaka težka naloga. Nastale skupine vsaj na začetku postopka elicitacije znanja nimajo pravega smisla, zaradi česar smo ponavadi manj prepričani, v katero skupino spada posamezen primer.

- **Kritičnost učnega primera** je težko določiti brez skupin. Medtem, ko dosedanje implementacije ABML izvajajo k–prečno validacijo za računanje kritičnosti, pri odkrivanju skupin to ni mogoče, saj ne poznamo dejanskih skupin učnih primerov.

Poglavje 4

Odkrivanje skupin z argumentiranjem

4.1 Algoritem AB k-means

Metoda voditeljev z argumenti (angl. *argument-based k-means*, AB k-means, ABKM) temelji na postopku argumentiranja, ki je opisan v poglavju 3.2. Ideja iterativne zanke za zajemanje znanja ostaja takšna, kot je opisana v prejšnih poglavjih. S pomočjo zanke želimo strokovnjaku ponuditi le tiste primere, ki so problematični. Problematični primeri pa so tisti, ki se ob odkrivanju skupin slabo umestijo v skupine. V nadaljevanju jih poimenujemo **kritični primeri**.

Postopek začnemo s primeri, ki so brez argumentov. Nato nadaljujemo po naslednjih korakih:

1. **korak: Iskanje hipoteze** – z metodo voditeljev z omejitvami poiščemo hipotezo.
2. **korak: Iskanje kritičnih primerov** – poiščemo kritične primere in dovolimo strokovnjaku, da sam izbere najbolj primeren kritični primer.
3. **korak: Podajanje argumentov** strokovnjak argumentira omejitev, ki vsebuje izbran kritični primer in nek drug učni primer.

4. **korak: Iskanje protiprimerov** – metoda lahko najde primer, ki ni skladen s strokovnjakovim argumentom. Strokovnjak nato argumentira omejitev, ki vsebuje izbran kritični primer in protiprimer.

5. **korak:** Vrni se na prvi korak z dobljenimi omejitvami.

V prvem koraku dobimo hipotezo z odkritimi skupinami, na podlagi katerih izračunamo, kateri so novi kritični primeri.

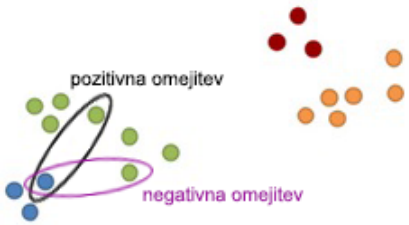
V drugem koraku izračunamo problematičnost uvrstitve posameznih primerov v določene in strokovnjaku prikažemo tiste, ki so najbolj kritični. Za računanje "kritičnosti" prečna validacija ni več primerna, zato smo jo izračunali s pomočjo silhuetne metode (angl. *silhouette*), ki oceni kakovost umeščenosti primerov v skupino [23]. Učni primeri z nižjo silhuetno vrednostjo se smatrajo kot bolj kritični primeri. Strokovnjak izbere enega izmed podanih kritičnih primerov.

V tretjem koraku poiščemo **reprezentativen primer** ene od skupin, v kateri ni kritičnega primera. Reprezentativen primer je tisti, ki ima visoko silhuetno vrednost in je dobro umeščen v skupino. Strokovnjaku prikažemo kritičen in reprezentativen primer v obliki poizvedbe o omejitvi. Strokovnjak nato postavi omejitev nad ta par in argumentira odločitev za takšno omejitev. Z izborom takšnega para dobimo bolj prepričljivo omejitev. Negotovost umestitve kritičnega primera v pravilno skupino zmanjšamo, ko dodamo informacijo v obliki omejitve z reprezentativnim primerom, ki se z veliko stopnjo zanesljivosti nahaja v drugi skupini kot kritični primer. Slika

Slika 4.1: Omejitev na kritični in reprezentativni primer.

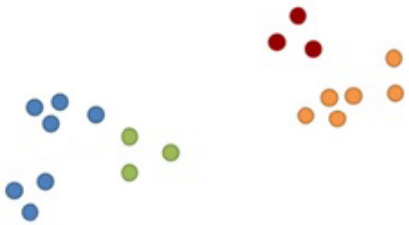
4.1 prikazuje postavitev pozitivne omejitve na kritični primer v modri skupini in reprezentativni primer zelene skupine.

V četrtem koraku algoritem s pomočjo s strani strokovnjaka podanih argumentov poišče protiprimere, ki so v drugi skupini kot kritičen primer in hkrati sovpadajo s strokovnjakovimi argumenti. Tako par iščemo le po tistih atributih, ki so najbolj relevantni za izbrano skupino. Strokovnjak nato postavi omejitve na par, ki vsebuje tako kritični primer kot dobljeni protiprimer. Slika 4.2 prikazuje postavitev dodatne negativne omejitve na kritični primer iz modre skupine in protiprimer iz zelene skupine, pridobljen z argumenti.



Slika 4.2: Omejitev na kritični primer in protiprimer.

Z dobljenimi omejitvami se vrnemo nazaj na prvi korak, torej k iskanju hipoteze z dobljenimi omejitvami. Nova hipoteza z upoštevanjem omejitev pripelje do nove hipoteze, ki je upoštevala dobljene omejitve. Silka 4.3 prikazuje novo hipotezo, ki smo jo dobili z upoštevanjem omejitev.

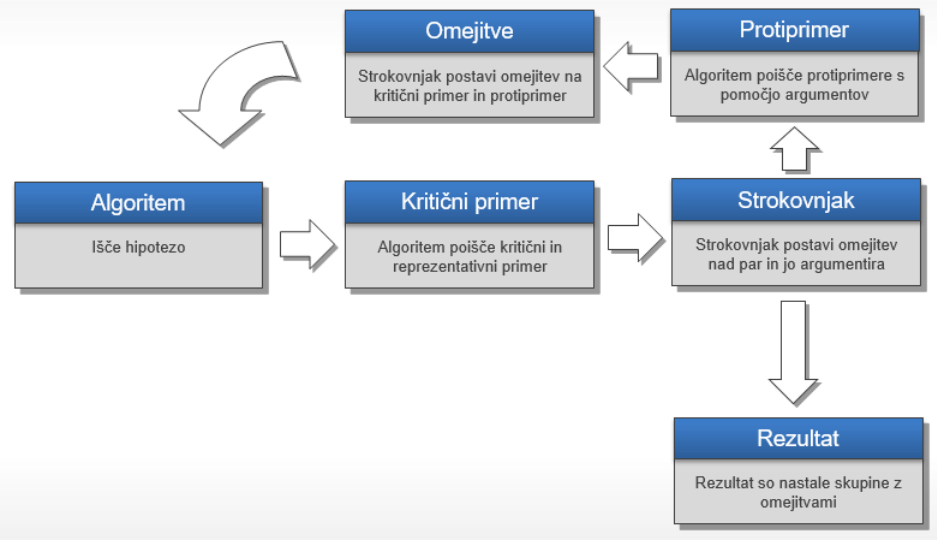


Slika 4.3: Nova hipoteza z omejitvami.

Nato je strokovnjaku predstavljen nov kritični primer. Postopek pona-

vljamo, dokler še imamo kritične primere ali pa se strokovnjak odloči, da je prišel do zadovoljivega rezultata. Takrat se postopek zaključi.

Celoten postopek prikazuje diagram na Sliki 4.4.



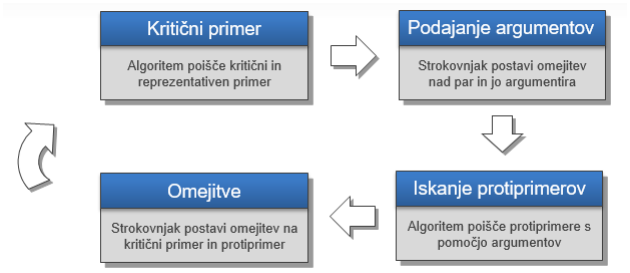
Slika 4.4: Potek algoritma ABKM.

4.2 Zajemanje znanja z AB k-means

Zajemanje znanja je ključna naloga naše metode. Z dodajanjem argumentov k primerom postane strokovnjak bolj prepričan v svojo odločitev. Poda nam informacije, s katerimi lahko poiščemo bolj primerne pare za postavljanje omejitev. Grobi osnutek poteka iterativne zanke za zajemanje znanja smo opisali v poglavju 4.1. V nadaljevanju pa bomo opisali podrobnejše ključne korake za uspešno implementacijo metode, ki so prikazani na diagramu 4.5.

4.2.1 Iskanje kritičnih primerov

Po končani prvi izvedbi metode voditeljev z omejitvami, pri kateri omejitev še nimamo, uporabimo dobljeno hipotezo v 2. koraku iterativne zanke – iskanje kritičnih primerov. Za iskanje kritičnih primerov najprej potrebujemo mero,



Slika 4.5: Potek iterativne zanke.

s katero lahko izračunamo, kako dobro so primeri umeščeni v podano skupino. Za to uporabimo mero silhuetni koeficient (angl. *silhouette coefficient*), ki ga izračunamo na način, kot je opisano v nadaljevanju.

Za vsak učni primer izračunamo povprečno različnost med primeri v isti skupini $a(i)$. Nato izračunamo povprečno različnost med učnim primerom in primeri ostalih skupin. Iz vseh teh meritev izberemo $b(i)$, ki predstavlja najnižjo različnost med primerom in skupino, katere primer ni med člani. S tem definiramo silhuetno vrednost kot:

$$s(i) = \frac{\mathrm{b(i)} - \mathrm{a(i)}}{\max\{\mathrm{a(i)},\mathrm{b(i)}\}} \tag{4.1}$$

Kar lahko tudi napišemo kot

$$s(i) = \begin{cases} 1 - \frac{a(i)}{b(i)}, & \text{če a(i) <b(i)} \\ 0, & \text{če a(i) = b(i)} \\ \frac{b(i)}{a(i)} - 1, & \text{če a(i) >b(i)} \end{cases} \tag{4.2}$$

$s(i)$ ima vrednosti od -1 do 1. Vrednost $a(i)$ nam pove, kako različen je x_i od svoje skupine. Nizek $a(i)$ torej pomeni, da je učni primer x_i dobro postavljen v skupino. In nasprotno, visok $b(i)$ pomeni, da je slabo postavljen. To pomeni, da mora veljati $a(i) \ll b(i)$, če je primer x_i dobro postavljen v skupino. Takrat je vrednost $s(i)$ v bližini 1.

Ko imamo izračunano silhuetno vrednost, za vsak primer najprej razvrstimo primere z najnižjo silhuetno vrednostjo. Uporabniku najprej ponudimo

primer z najnižjo silhuetno vrednostjo, saj nam prav ta mera pove, da so izbrani primeri bolj kritični. Strokovnjak nato izbere primeren kritični primer za argumentacijo.

Mejo kritičnosti nastavimo tako, da določimo maksimalno silhuetno vrednost, do katere želimo izboljšati primere v skupinah.

Časovna kompleksnost silhuetnih vrednosti je kvadratna s številom primerov v odkrivanju skupin $O(n^2)$. Namesto silhuetnih vrednosti bi bila možna rešitev tudi uporaba Davies–Bouldin indeksa (angl. *Davies–Bouldin index*, DBI), katerega časovna kompleksnost je linearna s številom primerov v odkrivanju skupin, vendar slabše izračuna kritičnost primerov.

4.2.2 Podajanje argumentov

Kot smo že omenili v poglavju 3.3, je postopek argumentiranja iz navedenih razlogov prilagojen odkrivanju skupin. Strokovnjak ne argumentira več z razlogi, zakaj kritični primer spada v neko skupino, kot to počne pri "klasičnem" ABML 3. V primeru odkrivanja skupin, kjer ne poznamo resničnih skupin primerov, je to bistveno težje, zato strokovnjaku predstavimo kritični in reprezentativen primer in strokovnjak argumentira, zakaj predstavljen par spada skupaj oz. ne spada skupaj.

Argumentiranje razdelimo na dva dodatna koraka, podobno kot pri osnovni implementaciji ABML:

Korak 1: Razlaganje omejitve. V tem koraku strokovnjak poda pozitivno ali negativno omejitev in argumentira omejitev z odgovorom na vprašanje: "Zakaj podana primera spadata v isto skupino, oziroma zakaj temu ni tako?" Odgovor je tako lahko nabor argumentov $A_1, ..., A_k$, ki potrjujejo podano omejitev. Kadar je omejitev pretežko argumentirati, lahko strokovnjak izbere drug par. Če pa strokovnjak ni prepričan v svojo trditev, lahko preskoči postavljanje omejitve in le argumentira, zakaj meni, da par spada skupaj ali ne.

Korak 2: Dodajanje argumentov primeru. Argumenti A_i so podani

v naravnem jeziku in jih je treba pretvoriti v domenski jezik (atributi). Vsak argument podpira svojo trditev z več razlogi. Ko je razlog navadna vrednost atributa, je argument preprosto dodan primeru. V primeru, ko razlog vsebuje nek nov koncept, ki ga trenutna domena ne vsebuje, se ta koncept doda v domeno kot nov atribut, še preden se argument doda k primeru.

Primer argumentiranja omejitev lahko predstavimo na primeru podatkov odplačevanja kreditov iz tabele 3.1. Strokovnjak določi pozitivno omejitev na reprezentativni primer Gdč. Bevk in kritični primer G. Bond, ki je slabše umeščen v skupino, ker ima atribut *RedniPlačnik* vrednost ne. Nato argumentira:

Če *StatusRačuna* = *pozitiven* POTEM *pozitivna omejitev*

Tako je strokovnjak navedel razlog, v obliki atributa *StatusRačuna*, zakaj primera spadata v isto skupino.

4.2.3 Iskanje protiprimerov

V klasičnem ABML 3 se strokovnjakove argumente uporablja za dopolnjevanje argumentov, pridobljenih s klasifikacijo. V primeru odkrivanja skupin uporabimo argumente za dopolnevanje omejitev na kritični primer. Iskanje protiprimerov lahko razdelimo na dva koraka:

Korak 1: Odkrivanje protiprimerov. Za iskanje protiprimerov poiščemo primere, ki sovpadajo s strokovnjakovimi argumenti, vendar hkrati ne spadajo v skupino kritičnega primera. Tako dobimo največ $k - 1$ protiprimerov. Iz vsake skupine prikažemo le en primer. Če je takšnih primerov v skupini več, se izbere le tistega, ki ima najvišjo silhuetno vrednost. Tako dobimo pare, ki so po relevantnih atributih podobni kritičnemu primeru, hkrati pa prepričljivo spadajo v drugo skupino.

Korak 2: Dodajanje omejitev primerom. Strokovnjak poda omejitve na pare, ki vsebujejo kritični primer in protiprimere. Za par kritičnega primera in protiprimera argumenti niso več potrebni, saj smo z obstoječimi argumenti že dobili željene protiprimere v tej iteraciji. Omejitve se nato doda k seznamu drugih omejitev in se jih upošteva ob naslednji iteraciji.

Tako z argumenti iščemo primere, ki so podobni kritičnemu primeru po kriteriju, ki ga poda strokovnjak. V tem smislu strokovnjak vodi izbor protiprimerov. Tako se lahko hitro najde podobne primere, ki so prav tako narobe razvrščeni in jih s pozitivno omejitvijo postavimo v pravilno skupino. Prav tako lahko hitro najdemo pravilno skupino za kritični primer, s tem ko zožamo prostor hipotez preko argumenta.

4.3 Podajanje omejitev

Podajanje omejitev pri algoritmu ABKM smo implementirali podobno kot jih implementira algoritem CKM, ki je opisan v poglavju 2.2.1. Strokovnjaku smo dodali poleg možnosti dodajanja omejitev na par tudi možnost omejitev para s pozitivno omejitvijo na skupino.

4.3.1 Omejitev na par

Omejitve na par smo opisali v poglavju 2.2. Vpeljemo jih tako, da poskušamo vsak primer d_i dodeliti najbližji skupini C_j. V primeru, da je pri tem katera od omejitev kršena, se primera d_i ne vstavi v to skupino. Če obstaja primer $d_=$, ki mora biti v istem razredu kot d_i in je ta primer že v drugi skupini kot C_j ali če obstaja primer $d_\neq$, ki je že v skupini C_j, pomeni, da je bila omejitev kršena. Ko imamo pozitivno omejitev nad primerom d_i in je primer $d_=$ že v skupini, postavimo primer d_i v skupino $d_=$. Ko pa imamo negativno omejitev nad primerom d_i in je primer $d_\neq$ v skupini C_j pa poskušamo umestiti primer v naslednjo najbližjo skupino.

4.3.2 Omejitev na skupino

Poleg omejitve na par lahko strokovnjak postavi tudi omejitev para na skupino. V tem primeru strokovnjak določi par, ki mora biti v isti skupini, hkrati pa določi tudi skupino, v katero spadata primera.

Takšno omejitev je bolj zahtevno postaviti in ni obvezna, vendar pripomore k hitrosti konvergence in natančnosti algoritma. Zaradi interakcije strokovnjaka z algoritmom postane postavljanje takšnih omejitev v kasnejših iteracijah iterativne zanke za zajemanje znanja lažje.

Slika 4.6a prikazuje primer, kjer strokovnjak želi voditi algoritem do rezultata, kjer je v modri skupini več primerov. Na sliki 4.6a je postavljena omejitev na par, kjer nismo določili skupine. Pogosto nezaželjen rezultat takšne omejitve je prikazan na sliki 4.6b.

(a) Omejitev na par učnih primerov. (b) Rezultat omejitve para.

Slika 4.6: Omejitev na par.

Ko pa strokovnjak določi omejitev na skupino, v kateri želi, da je omejeni par (glej Sliko 4.7a), dobimo bolj zaželjen rezultat, ki je prikazan na sliki 4.7b.

(a) Omejitev para na skupino. (b) Rezultat omejitve para na skupino.

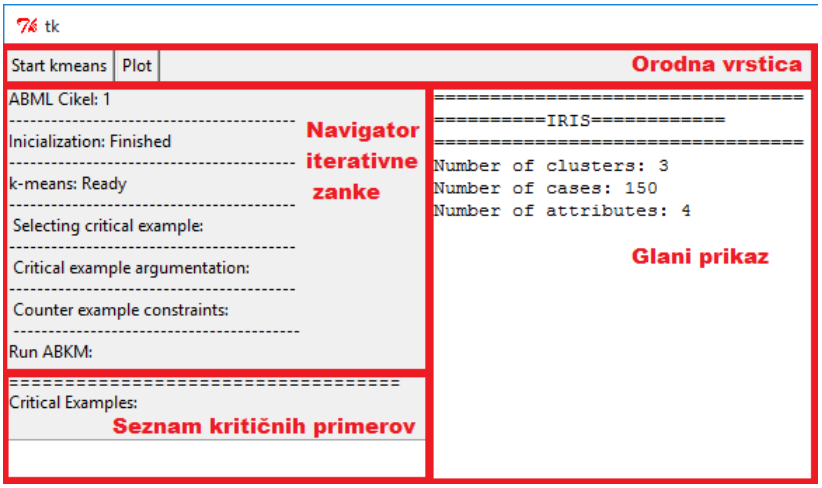
Slika 4.7: Omejitev na skupino.

Poglavje 5

Aplikacija za zajemanje znanja z algoritmom AB k-means

5.1 Predstavitev aplikacije

Za lažjo predstavitev in za lažje testiranje smo ustvarili aplikacijo z grafičnim vmesnikom za upravljanje z algoritmom, ki je prosto dostopna na spletu [41]. Aplikacija je le orodje za lažjo uporabo algoritma.

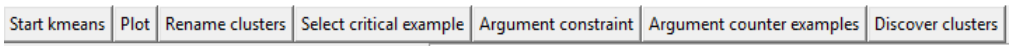


Slika 5.1: Razdelitev grafičnega vmesnika.

Glavno okno smo razdelili na štiri dele, kot so prikazani na sliki 5.1.

Orodna vrstica

orodna vrstica na sliki 5.2 vsebuje gumbe, s katerimi uporabnik sledi korakom algoritma. Aplikacija prikaže le gumbe, ki so aktualni.



Slika 5.2: Navigacija aplikacije.

- **Start k-means** – Zažene osnovno implementacijo k-means metode brez omejitev.
- **Plot** – Izriše trenutne skupine v 2–D prostoru s pomočjo metode *PCA*.
- **Rename clusters** – Uporabnik s tem gumbom preimenuje skupine, kar zelo olajša spremljanje algoritma.
- **Select critical example** – S klikom na ta gumb odpremo novo okno, prikazano na sliki 5.4 kjer prikažemo po pet primerov, razvrščenih po kritičnosti (silhuetne vrednosti) primera. Uporabnik izbere kritični primer, ki je po njegovi presoji primeren za argumentiranje omejitev.
- **Argument constraint** – Ko je uporabnik izbral kritični primer, se prikaže gumb "Argument critical example", ki odpre novo okno, prikazan na sliki 5.5 z izbranim kritičnim primerom in primerom za primerjavo. V oknu je tudi obrazec za izbor argumenta in omejitve.
- **Counter examples** – Ko je kritični primer argumentiran in je algoritem poiskal protiprimere, se pojavi gumb "Counter examples", ki v novem oknu, prikazan na sliki 5.6 prikaže kritični primer in protiprimer ter vpraša uporabnika po omejitvi med njima.
- **Discover clusters** – Ko je ves postopek končan, se prikaže gumb "Get hypothesis". S tem gumbom zaženemo AB k-means, ki upošteva omejitve, dobljene v dosedanjih iteracijah.

Navigator iterativne zanke

Navigator iterativne zanke služi sledenju postopka argumentacije. Zapisani so koraki argumentiranja in v katerem stanju je korak (Ready – pomeni, da smo v tem stanju; Finished – pomeni, da smo opravili ta korak; brez stanja – še moramo opraviti ta korak).

Glavni prikaz

Glavni prikaz izpisuje rezultate algoritma: povprečne vrednosti atributov, oddaljenost voditeljev od ostalih voditeljev, radij voditelja, povprečna razdalja primerov od voditelja, postavljena omejitev, postavljen argument.

Seznam kritičnih primerov

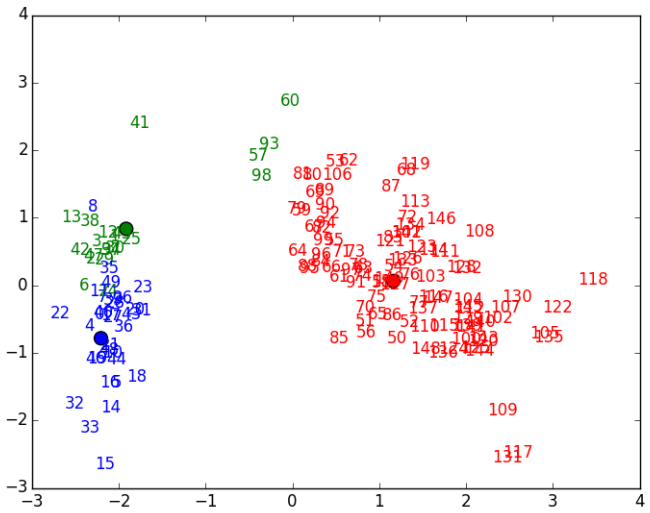
Seznam kritičnih primerov izpisuje seznam izbranih kritičnih primerov. Ko izberemo več kritičnih primerov, lahko na seznamu izberemo kateregakoli od izbranih kritičnih primerov in spremenimo njegove argumente s pritiskom na gumb "Argument critical example".

5.2 Opis postopka argumentiranja

Za lažjo predstavo prikažimo delovanje algoritma na podatkovni zbirki Iris [36]. Zbirka vsebuje tri različne razrede, od katerih sta le dva razločno ločena. Zaradi tega včasih pridemo do napačnih skupin pri odkrivanju skupin. Bolj podrobno to podatkovno zbirko predstavimo v poglavju 6.1.1.

Začetna hipoteza

Tabela 5.1 in Slika 5.3 prikazujeta primer nezadovoljivega k-means rezultata, kjer so primeri v modri skupini zelo slabo razporejeni. V tabeli 5.1 navajamo le centroide, medtem ko slika 5.3 prikazuje celotno podatkovno zbirko. Zaradi slabe inicializacije skupin, dobimo z algoritmom k-means neprime-



Slika 5.3: Nezadovoljiv k-means rezultat.

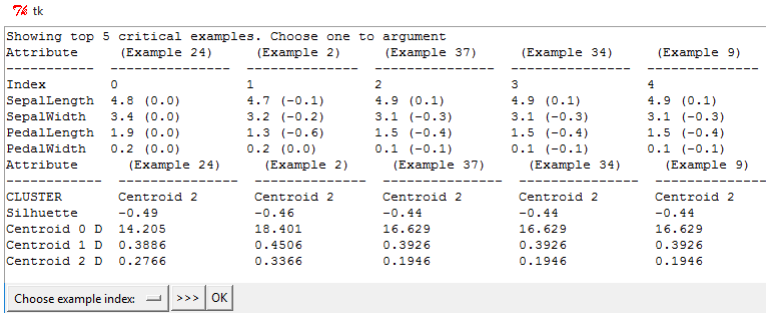
sepal length	6.31	5.18	4.75
sepal width	2.9	3.63	2.91
pedal length	4.97	1.47	1.78
pedal width	1.7	0.28	0.34
barva	rdeča	zelena	modra

Tabela 5.1: Centroidi nezadovoljega rezultata.

ren rezultat. Ta problem se je v preteklosti reševal z večkratnim zagonom algoritma. ABKM reši ta problem z omejitvami.

Izbor kritičnega primera

Ko ima algoritem začetno hipotezo, se začne postopek iterativne zanke za zajemanje znanja iz strokovnjaka. V prvem koraku najdemo kritične primere. Kritične primere najdemo s pomočjo silhuetnega koeficienta. Ta postopek je razložen v poglavju 4.3.2. Strokovnjak izbere en primer izmed petih najbolj kritičnih primerov, ki mu jih predstavimo. Izbiro kritičnega primera mu dovolimo, ker je nad nekatere primere lažje postaviti omejitev in jo argumen-



Slika 5.4: Izbor kritičnih primerov.

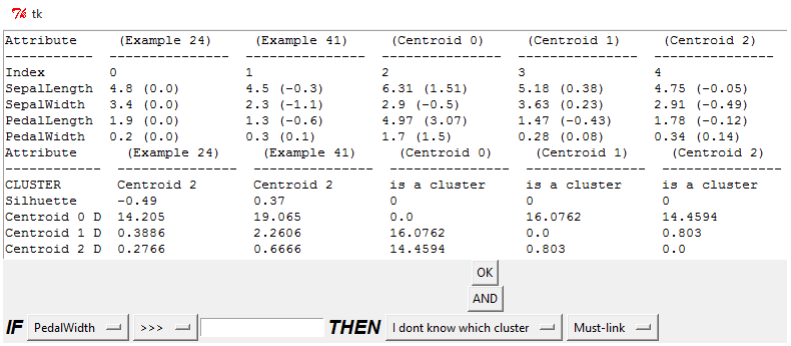
tirati. Slika 5.4 prikazuje ta izbor.

Z izborom kritičnega primera se prestavimo v korak argumentiranja.

Argumentiranje omejitve

S klikom na gumb "Argument constraint" se odpre okno za argumentiranje, kjer strokovnjaku predstavimo kritični primer skupaj z reprezentativnim primerom druge skupine. Na sliki 5.5 vidimo kritični primer z majhno silhuetno vrednostjo in reprezentativni primer z visoko silhuetno vrednostjo. Strokovnjak poda svoje mnenje o omejitvi med primeroma in obrazloži svojo odločitev z argumentom.

Strokovnjak lahko argument poda tudi v obliki "Če je atribut visok, potem...". Tako olajšamo delo strokovnjaku, da se osredotoči na problem in ne na računanje atributov. Operator >>> pomeni "visok" in pomeni višje od

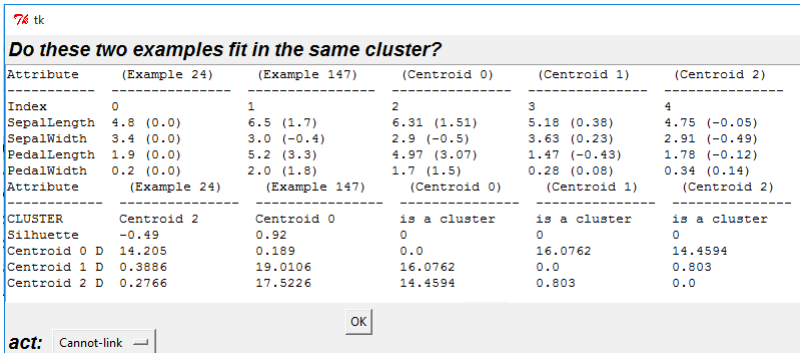


Slika 5.5: Argumentiranje omejitev.

90% vrednosti atributa. Torej, če ima atribut vrednost 100, potem operator >>> pomeni več od 90. Operator <<< pomeni "nizek" in pomeni nižje od 110% vrednosti atributa. Torej, če ima atribut vrednost 100, potem operator <<< pomeni manj od 110. S podanim argumentom se prestavimo v naslednji korak.

Omejitve na protiprimere

S pritiskom na gumb "Counter examples" algoritem poišče protiprimere s podanim argumentom in jih v novem oknu predstavi strokovnjaku. Postopek iskanja protiprimerov smo opisali v poglavju 4.2.3. Strokovnjak poda omejitev ali kritični primer in dobljeni protiprimeri spadajo v isto skupino. Postopek je prikazan na sliki 5.6.

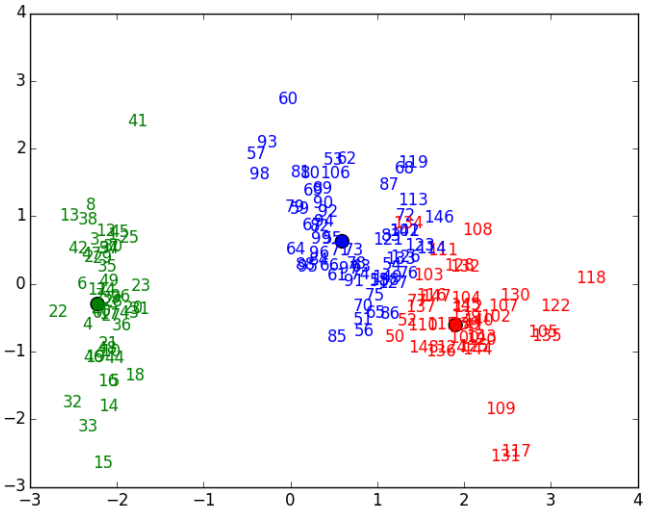


Slika 5.6: Postavljanje omejitev nad kritični primer in protiprimer.

Nova hipoteza

S postavitvijo omejitev na kritični primer in protiprimere smo dobili bolj informativne omejitve. S tem v odkritih skupinah povežemo slabo umeščene primere z bolje umeščenimi primeri. Tako potrebujemo manj omejitev in posledično manj časa za postavitev omejitev.

Slika 5.7 in tabela 5.2 prikazujeta rezultat šestih omejitev, iz katere je razvidno, da že po šestih takšnih omejitvah dobimo dobro izboljšavo odkrivanja skupin na našem primeru podatkovne zbirke rož Iris.



Slika 5.7: Izboljšan Iris rezultat.

sepal length	6.6	5.01	5.66
sepal width	2.99	3.42	2.7
pedal length	5.38	1.46	4.03
pedal width	1.92	0.24	1.24
barva	rdeča	zelena	modra

Tabela 5.2: Centroidi boljšega rezultata.

Poglavje 6

Metodologija evalvacije

Uspešnost naše metode smo preizkusili na različnih podatkovnih zbirkah in z različnimi merami za evalvacijo odkrivanja skupin. Našo metodo smo primerjali z najbolj relevantnim algoritmom – metodo voditeljev z omejitvami (angl. *Constrained k-means*, CKM) 2.2.1. CKM algoritem, ki smo ga uporabili, je implementiral Mateusz Zawiślak in je prosto dostopen na spletu [37]. Metodi smo primerjali po naslednjih kriterijih:

- s številom omejitev,
- vrednost Adjusted Rand Index 6.2.1,
- vrednost Normalized Mutual Information 6.2.2.

Mejo kritičnosti primerov pri metodi ABKM smo nastavili na silhuetno vrednost 1, kar pomeni, da je vsak primer potencialno kritični primer. To smo storili zato, da se algoritem ni ustavil, preden je prišel do željenega števila omejitev.

Vsak poskus smo opravili tridesetkrat in dobljene vrednosti povprečili. Ker ima strokovnjak pri uporabi metode ABKM kar nekaj neobveznih možnosti, smo metodo preizkusili na več različnih načinov.

> **Prvi preizkus** (označen z "ABKM OS"; Omejitve na Skupino): ob vsaki pozitivni ali negativni omejitvi nad parom postavi tudi omejitev na skupino.

Drugi preizkus (označen z "ABKM OS po 10 iteracijah"): postavi ob vsaki pozitivni ali negativni omejitvi nad parom tudi omejitev na skupino, vendar šele po 10. iteraciji iterativne zanke za zajemanje znanja.

Tretji preizkus (označen z "ABKM brez OS"): ne postavlja omejitve na skupino, temveč le pozitivno ali negativno omejitev nad parom.

V primeru, ko poznamo zlato resnico, je postavljanje omejitev na skupino lahka naloga. Ko ne poznamo zlate resnice, pa je postavljanje omejitev na skupino težje, vendar je zaradi interakcije strokovnjaka z algoritmom tudi postavljanje takšnih omejitev realna možnost. Drugi preizkus je bil zato opravljen tako, da pred 10. iteracijo iterativne zanke postavi le pozitivno ali negativno omejitev, brez omejitve na skupino. Ko so skupine bolj jasne in razumljive po 10 iteracijah postavi tudi omejitve na skupine. Zadnji opravljen preizkus je bil izveden brez postavljanja omejitev na skupine. To je primer, ko so skupine tako nerazumljive, da ne znamo nobenega primera umestiti v ustrezno skupino.

6.1 Uporabljene množice podatkov

V tem poglavju predstavimo podatkovne množice, ki so bile uporabljene pri eksperimentu, in njihove lastnosti. Za evalvacijo smo uporabili standardne UCI podatkovne množice [36], ker je dostop do njih enostaven za celotno raziskovalno skupnost. Uporaba teh podatkovnih zbirk ima tudi manj nevarnosti za pristransko evalvacijo, saj so bile zbirke zgrajene neodvisno od možnih pristopov za reševanje problemov. Podatkovne množice so bile velikokrat uporabljene v sorodnih raziskavah in je zato tudi primerjava z drugimi eksperimenti lažja. Rezultate naše metode smo primerjali z najbolj relevantnim algoritmom – metoda voditeljev z omejitvami (angl. *Constrained k-means*, CKM), ki je bil opisan v poglavju 2.2.1.

6.1.1 Iris rože

Zbirka rož Iris (angl. *Iris flower dataset*, Iris) [38] je ena izmed najbolj znanih standardnih podatkovnih zbirk. Zbirka vsebuje 150 primerov, treh različnih vrst rož Iris. Za vsak primer v podatkovni bazi so bili izmerjeni štirje atributi:

1. širina venčnega lista (angl. *petal width*),
2. dolžina venčnega lista (angl. *petal length*),
3. širina čašnega lista (angl. *sepal width*),
4. dolžina čašnega lista (angl. *sepal length*).

Zbirka vsebuje tri različne skupine: *Iris–virginica*, *Iris–versicolor*, *Iris–setosa*. Od treh skupin sta le dve razločno ločeni, zaradi česar pri odkrivanju skupin pogosto dobimo napačne skupine.

6.1.2 Tip vina

Podatkovna baza Tip vina [39] so podatki kemične analize vin, ki so bile vzgojene v Italiji iz treh različnih kultivarjev. Podatkovna baza vsebuje 178 vin treh različnih vrst. Imena vin niso znana, zato so značena s številkami *1*, *2*, *3*. Analiziranih je bilo naslednjih 13 atributov:

1. alkohol (angl.*alcohol*),
2. jabolčna kislina (angl. *malic acid*),
3. pepel (angl. *ash*) ,
4. bazičnost pepela (angl. *alcalinity of ash*),
5. magnezij (angl. *magnesium*),
6. količina fenolov (angl. *total phenols*),
7. flavonoidi (angl. *flavanoids*),

8. neflavaonidni fenoli (angl. *nonflavanoid phenols*),
9. proantocianidin (angl. *proanthocyanins*),
10. intenziteta barve (angl. *color intensity*),
11. odtenek (angl. *hue*),
12. razredčenost vin (angl. *diluted wines*),
13. prolin (angl. *Proline*).

6.1.3 Akutno vnetje

Podatkovna baza Akutno vnetje (angl. *Acute Inflammation*) [40] vsebuje 120 primerov z dvema boleznima urinskega sistema. Vsak primer spada v eno od štirih skupin: *akutno vnetje*, *akutni nefritis*, *brez bolezni*, *z obema boleznima*. Zdravnik je diagnosticiral paciente s šestimi atributi:

1. temperatura pacienta (angl. *temperature of patient*),
2. prisotnost slabosti (angl. *occurrence of nausea*),
3. bolečine v ledvicah (angl. *lumbar pain*),
4. pogosto uriniranje (angl. *urine pushing*),
5. boleče uriniranje (angl. *micturition pains*),
6. pekoča sečnica (angl. *burning of urethra*).

6.2 Mere za evalvacijo

6.2.1 ARI (Adjusted Rand Index)

Za primerjavo našega grupiranja z resničnimi skupinami smo uporabili prilagojen Rand indeks (angl. *Adjusted Rand Index*, ARI). ARI je mera, ki nam

poda informacijo o tem, koliko se resnične skupine primerov razlikujejo od dobljenih skupin algoritma.

ARI meritev izhaja iz mere Rand Indeks (angl. *Rand Index*, RI), ki je matematično enaka natančnosti (angl. *accuracy*). Prednost RI mere je, da se jo lahko uporabi tudi, ko razrednih imen ne poznamo. RI mero izračunamo po enačbi

$$R = \frac{a+b}{a+b+c+d} \tag{6.1}$$

kjer,

- a predstavlja število parov, ki smo jih razvrstili v isto skupino in so hkrati po resničnih skupinah tudi v enaki skupini,
- b predstavlja število parov, ki smo jih razvrstili v različno skupino in so hkrati po resničnih skupinah tudi v različnih skupinah,
- c predstavlja število parov, ki smo jih razvrstili v isto skupino in so hkrati po resničnih skupinah v različnih skupinah,
- d predstavlja število parov, ki smo jih razvrstili v različno skupino in so hkrati po resničnih skupinah v isti skupini.

V tem primeru števec $(a+b)$ pomeni število strinjanj med našimi skupinami in resničnimi skupinami, $c+d$ pa pomeni število nestrinjanj med našimi skupinami in resničnimi skupinami.

ARI mero se iz RI razširi tako, da ob meritvi upoštevamo tudi pričakovan RI. ARI izračunamo po enačbi:

$$ARI = \frac{\text{Index} - \text{PričakovanIndex}}{\text{MaxIndex} - \text{ExpectedIndex}} \tag{6.2}$$

Tako dobimo končno enačbo, kjer se upošteva pričakovan RI:

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]/\binom{N}{2}}{\frac{1}{2}[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}] - [\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}]\binom{N}{2}} \tag{6.3}$$

Medtem, ko ima RI vrednosti med 0 in 1, so ARI vrednosti med -1 in 1. Če

je Rand Index nižji od pričakovanega, potem dobimo ARI vrednost manjšo od 0.

6.2.2 NMI (Normalized Mutual Information)

Normalizirana vzajemna informacija (angl. *Normalized mutual information*, NMI) določi, koliko statistične informacije je skupne med skupinami, ki smo jih določili z našim algoritmom in skupinami, določene z zlato resnico.

NMI izračunamo po enačbi

$$NMI = \frac{I(C;K)}{(H(C) + H(K))/2} \tag{6.4}$$

kjer C predstavlja naključno spremenljivko, ki označuje dodeljeno skupino primeru in K predstavlja naključno spremenljivko, ki označuje resnično skupino dodeljenega primera. $I(X;Y) = H(X) - H(X|Y)$ je vzajemna informacija med naključnima spremenljivkama X in Y, $H(X)$ pa Shannon entropija [34] spremenljivke X in $H(X|Y)$ pogojna entropija spremenljivke X ob podanem Y.

Poglavje 7

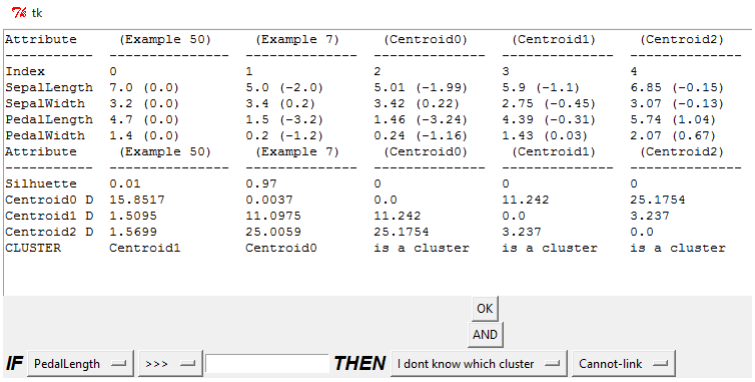
Zajemanje znanja iz eksperta

7.1 Eksperimenti v domeni Iris

Ker je domena Iris [38] relativno enostavna, smo za primerjavo z ostalimi metodami, postopek argumentiranja avtomatizirali. Domena je podrobneje opisana v poglavju 6.1.1 Za vsako skupino v podatkih smo uporabili vnaprej določene argumente in omejitve postavljali z uporabo zlate resnice. Za vsako skupino smo določili en argument, ki je bil skladen s primeri v tej skupini. Argumentirali smo omejitev kritičnega primera na reprezentativen primer. To pomeni, da smo najprej postavili omejitev, ki smo jo dobili z zlato resnico, kot so to storili avtorji v delu [8]. Za dobljeno omejitev smo postavili sledeče argumente:

- Če je bil kritični primer v skupini *Iris–setosa*, smo nastavili argument: "ČE *PedalWidth* >>>"
- Če je bil kritični primer v skupini *Iris–versicolor*, smo nastavili argument: "ČE *PedalWidth* >>>"
- Če je bil kritični primer v skupini *Iris–virginica*, smo nastavili argument: "ČE *PedalWidth* >>>"

Operator >>> pomeni "visok" in pomeni višje od 90% vrednosti atributa kritičnega primera. Operator <<< pomeni "nizek" in pomeni nižje od 110%



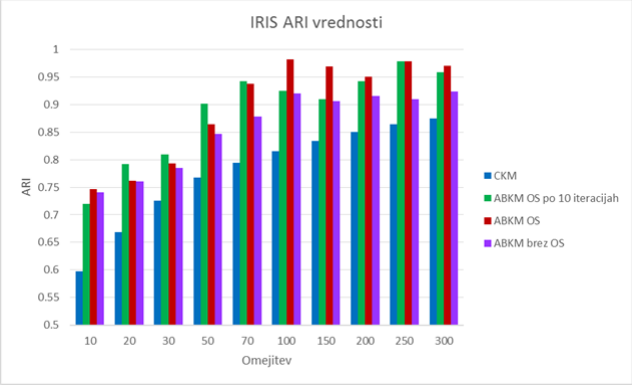
Slika 7.1: Argumentiranje kritičnega primera v podatkovni bazi Iris.

vrednosti atributa kritičnega primera.

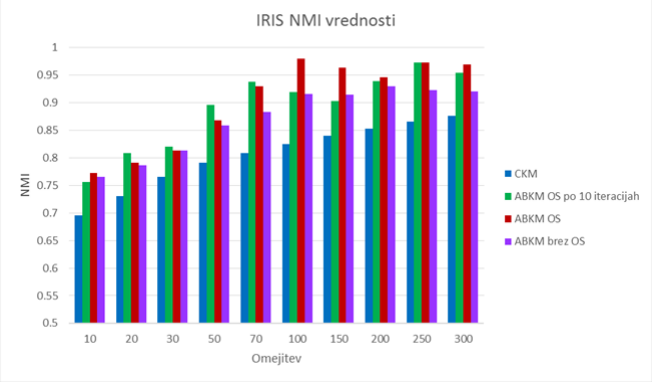
V našem grafičnem vmesniku je prikaz argumenta videti, tako kot je prikazano na sliki 7.1

Dobljene vrednosti mere ARI so predstavljene v grafu na sliki 7.2 in vrednosti mere NMI na sliki 7.3. Za najbolj uspešen način postavljanja omejitev se je izkazal način "ABKM z OS". Primerljive rezultate je dosegel način "ABKM z OS po 10 iteracijah". Tretji rezultat je dosegel način "ABKM brez OS". Pri vsakem številu omejitev smo dobili višje ARI in NMI vrednosti od metode CKM.

To pomeni, da naša metoda natančneje uvrsti primere v skupine pri enakem številu omejitev, ne glede na način uporabe naše metode.



Slika 7.2: Vrednosti ARI in primerjava s CKM algoritmom.



Slika 7.3: Vrednosti NMI in primerjava s CKM algoritmom.

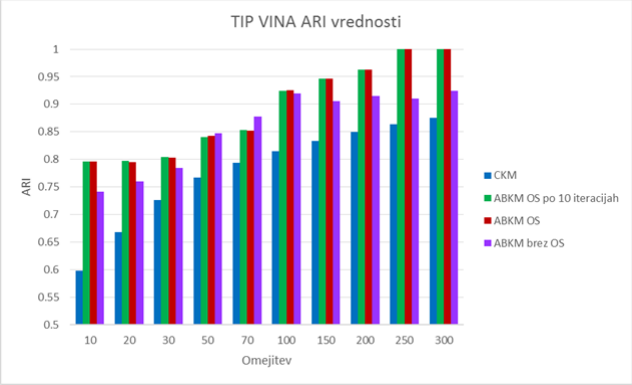
7.2 Eksperimenti v domeni Tip vina

Domeno Tip vin [39], ki je opisana v poglavju 6.1.2, smo prav tako preverjali s pomočjo zlate resnice. Vse atribute smo normalizirali tako pri metodi ABKM kot pri CKM. Omejitve smo postavili z uporabo zlate resnice in z vnaprej določenimi argumenti. Za vsako skupino smo nastavili en argument, ki je bil skladen s primeri v tej skupini. S pomočjo zlate resnice smo argumentirali omejitev kritičnega primera glede na reprezentativen primer. To pomeni, da smo najprej postavili omejitev, ki smo jo dobili z zlato resnico, kot so to storili avtorji v delu [8]. Za dobljeno omejitev smo postavili sledeče argumente:

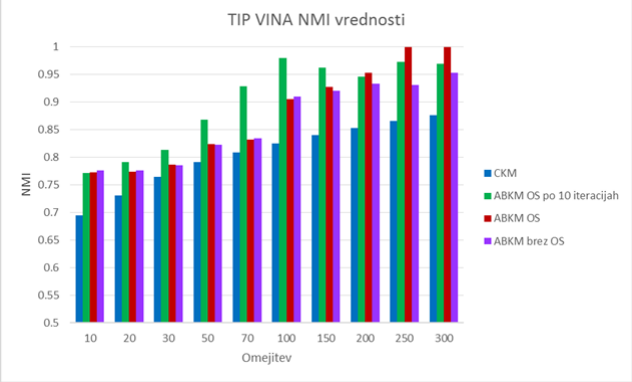
- Če je kritični primer v skupini 1: "ČE *Alcohol* >>>"
- Če je kritični primer v skupini 2: "ČE *MalicAcid* >>>"
- Če je kritični primer v skupini 3: "ČE *ColorIntensity* >>>"

Graf na sliki 7.4 prikazuje rezultate ocene ARI in graf na sliki 7.5 prikazuje rezultate ocene NMI.

Podobno, kot pri prejšnem eksperimentu 7.1 smo z našo metodo v vseh treh primerih dosegli boljše rezultate, kot z metodo CKM.



Slika 7.4: Vrednosti ARI in primerjava s CKM algoritmom.

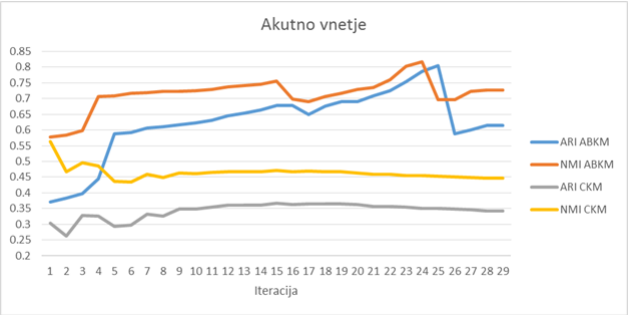


Slika 7.5: Vrednosti NMI in primerjava s CKM algoritmom.

7.3 Eksperimenti v domeni Akutno vnetje

Eksperiment smo izvedli s pomočjo strokovnjaka na domeni akutnega vnetja [40]. Po izvedbi eksperimenta je strokovnjak v 29 korakih iterativne zanke za zajemanje znanja dobil 95 postavljenih omejitev na par in 28 omejitev na skupino. Postopek je vodil do izboljšanja ARI vrednosti z 0.371 na 0.6136 in NMI vrednosti z 0.5775 na 0.728, kljub nekaj napačno postavljenimi omejitvami. Graf na sliki 7.6 prikazuje, rezultate ARI in NMI vrednosti ABKM metode s strokovnjakom in CKM metode z naključnimi omejitvami. ABKM metoda je z uporabo argumentov in omejitev na skupino prikazala boljše rezultate kljub napačnim omejitvam.

Strokovnjak je v prvi iterativni zanki uspel poimenovati dve od štirih. Dveh skupin mu ni uspelo poimenovati zaradi zaradi nerazumljivosti odkrivanja skpupin. Po deveti iteraciji je rezultat postal bolj skladen s strokovnjakovim znanjem in je tako lažje poimenoval še drugi dve skupini. Tako je bilo postavljanje nadaljnjih omejitev lažje.



Slika 7.6: Vrednosti ARI in NMI pridobljene s strokovnjakom.

V iteraciji 14 so se vrednosti ARI in NMI poslabšale. To se je zgodilo, ker je strokovnjak podal napačno negativno omejitev na par v tabeli 7.1

V iteraciji 24 je bila prav tako postavljena napačna negativna omejitev na par v tabeli 7.2.

Zaradi napačnih omejitev je na rezultatih viden padec kvalitete odkrivanja skupin. Ta slabost, ki si jo naša metoda deli z metodo CKM, nastane zaradi postavljanja trdih omejitev, ki se vključijo v odkrivanje skupin kljub

Tabela 7.1: Napačno nastavljena pozitivna omejitev na par.

Temperature	37.1	37.1
Occurrence of nausea	no	no
Lumbar pain	no	no
Urine pushing	yes	yes
Micturition pains	yes	yes
Burning of urethra	yes	yes
Skupina	akutno vnetje	akutno vnetje

Tabela 7.2: Napačno nastavljena negativna omejitev na par.

Temperature	38.1	39.7
Occurrence of nausea	no	no
Lumbar pain	yes	yes
Urine pushing	yes	yes
Micturition pains	no	no
Burning of urethra	yes	yes
Skupina	akutni nefritis	akutni nefritis

njihovi pravilnosti. Pri tem se upošteva tudi omejitve, ki so napačne in poslabšajo rezultat odkrivanja skupin.

Poglavje 8

Sklepne ugotovitve

V magistrskem delu smo izpostavili probleme, ki se pojavijo pri praktični uporabi odkrivanja skupin z omejitvami (angl. *constrained clustering*) in so bile kljub veliki raziskanosti področja zapostavljene. Pri takšnem odkrivanju skupin s postavljanjem pozitivnih (angl. *must-link*) in negativnih (angl. *cannot-link*) omejitev na par učnih primerov vključujemo domensko znanje in tako izboljšamo rezultat odkrivanja skupin. Dosedanji razvoj je bil usmerjen v nove algoritme, ki na kreativne načine uporabijo dobljene omejitve in kažejo dobre rezultate. Praktični problemi, kot je dolgo in naporno podajanje omejitev, za katere potrebujemo vnos domenskega strokovnjaka, pa so ostali nenaslovljeni.

V magistrskem delu smo predlagali novo rešitev za problem zajemanja znanja iz strokovnjaka, prilagojeno odkrivanju skupin. V okviru paradigme argumentiranega strojnega učenja smo razvili metodo voditeljev z argumenti "AB k-means" (angl. *Argument-based k-means*, ABKM), ki z iskanjem slabo in dobro umeščenih primerov v skupinah s pomočjo strokovnjakovih argumentov, pridobi omejitve, ki so ključne za uspešno odkrivanje skupin. Z argumentiranjem lahko domenski strokovnjak podaja razloge, zakaj določeni avtomatsko izbrani primeri po svojih lastnostih pripadajo oz. ne pripadajo v isti skupini. Poleg omejitve ali primera spadata v isto skupino, lahko strokovnjak tudi določi, v katero skupino slabo umeščeni primer spada in s tem

prepreči umestitev primera v napačno skupino. AB k-means nato s pomočjo protiprimerov izpostavlja morebitne pomanjkljivosti strokovnjakovih razlag in mu omogoča izboljševanje podanih omejitev. Tako pridobimo ustreznejše omejitve, hkrati pa se s tem skrajša celoten postopek zajemanja znanja iz strokovnjaka. Tovrstna interakcija med metodo in strokovnjakom vodi do s strokovnjakovim znanjem skladnih omejitev, ki učinkovito pripomorejo k izboljšanju rezultatov odkrivanja skupin, pri tem pa domenskemu strokovnjaku konkretno olajšamo postavljanje omejitev. Za lažje zajemanje znanja iz strokovnjaka smo ustvarili grafični vmesnik, ki je prosto dostopen na spletu [41].

Empirično smo pokazali, da je z vpeljavo argumentov mogoče priti do primernejših omejitev in da nov postopek zajemanja omejitev vodi do boljših rezultatov v primerjavi s sodobnimi algoritmi za odkrivanje skupin z omejitvami. Predlagana metoda se je v vseh treh testnih domenah izkazala za uspešnejšo od primerljive metode za odkrivanje skupin CKM (angl. *constrained k-means*, CKM). K izboljšanim rezultatom so vodili tako s strani metode podani protiprimeri, ki so pripomogli k ustreznejšim omejitvam, kot tudi možnost postavljanja omejitev na skupine. Zaradi interakcije, ki jo omogoča interativna zanka za zajemanja znanja, so dobljeni rezultati bolj razumljivi in skladni s strokovnjakovim znanjem.

Sodobni algoritmi za odkrivanje skupin s pomočjo metode voditeljev (angl. *k-means*) običajno imajo največ težav na večjih podatkovnih bazah, še zlasti zaradi napačno inicializiranih skupin. Kljub temu, da nova metoda zaradi zahtevnih izračunov kakovosti umestitve posameznih primerov v skupine še ni namenjena delu z velikimi učnimi množicami podatkov, pa so njeni rezultati zaradi pridobljenega domenskega znanja s strani domenskega strokovnjaka lahko še posebej koristni prav za uspešno odkrivanje skupin tudi v primeru velike količine podatkov. Tung in avtorji [35] predlagajo pri odkrivanju skupin večjih podatkovnih baz uporabo manjših skupin (angl. *micro-clusters*). Z uporabo ABKM v predhodnem koraku odkrivanja skupin z ustreznim vzorčenjem pridobimo manjše skupine, na katere nato postavimo

primerne omejitve, ki se uporabijo pri odkrivanju skupin celotne podatkovne baze.

Za dodatno zmanjšanje strokovnjakove obremenitve, kot možnost za nadaljne izboljšanje uporabljenega pristopa, predlagamo shranjevanje obstoječih argumentov za omejitve. S tem bi se izognili večkratnemu vnašanju istih argumentov. Prav tako bi bilo treba nasloviti trdnost omejitev, saj so eksperimenti pokazali, da se rezultat odkrivanja skupin močno poslabša, ko pride do napačno podane omejitve. Prav tako je slabost, da bodo vsi učni primeri, ki so povezani s pozitivnimi omejitvami, slepo postavljeni v skupino prvega učnega primera s pozitivno omejitvijo in pri tem ne upošteva podobnosti s centroidi ali drugih omejitev. Iskanje slabo umeščenih primerov je v predlagani metodi časovno zahtevna naloga, zato je še vedno odprta možnost za izboljšave. Na primer namesto silhuetnih vrednosti, za mero ugotavljanja "kritičnosti"umestitev posameznih učnih primerov v skupine, bi bila možna rešitev uporaba Davies–Bouldin indeksa (angl. *Davies–Bouldin index*, DBI).

Literatura

[1] T. Reilly (2005). "What Is Web 2.0", O'Reilly Media, Inc. Dostopno na: `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`, pridobljeno 18. 11. 2014.

[2] M. E. Ares, "Constrained Clustering Algorithms: Practical Issues and Applications", *PhD thesis*, Catedratico de Universidade na area de Ciencias da Computacion e Intelixencia Artificial da Universidade da Coruna, 2013

[3] E. Pampalk, S. Dixon, G. Widmer. "On the evaluation of perceptual similarity measures for music" *Proceedings of the sixth international conference on digital audio effects* , Queen Mary University of London, str. 7–12, 2003.

[4] X. Wang, I. Davidson, "Flexible constrained spectral clustering", *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, str. 563–572, 2010.

[5] M. E. Ares, A. Barreiro, "Constrained text clustering using word trigrams" *Proceedings of the 2nd Spanish Conference on Information Retrieval*, Publicacions de la Universitat Jaume I, str. 13–24, 2012.

[6] R. Yan, J. Zhang, J. Yang, A. Hauptmann, "A discriminative learning framework with pairwise constraints for video object classification" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, str. 578–593, 2006.

[7] P. Arabie, L. Hubert, "Advanced methods in marketing research." *Oxford: Blackwell. Chap. Cluster Analysis in Marketing Research*, str. 160–189, 1994

[8] K. Wagstaff, C. Cardie, S. Rogers, S. Schrodl,"Constrained kmeans clustering with background knowledge" *Proceedings of the Eighteenth International Conference on Machine Learning*, str. 577–584, 2001.

[9] X. Ji, W. Xu, "Document clustering with prior knowledge" *In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* , str. 405–412, 2006.

[10] A. K. Jain, "Data Clustering: 50 Years Beyond K-Means" *Pattern Recognition Letters* vol. 31, no. 8, str. 651–666, 2010.

[11] K. Wagstaff, C. Cardie, "Clustering with instance-level constraints" *Proceedings of the Seventeenth International Conference on Machine Learning*, str. 1103–1110, 2000.

[12] S. Basu, I. Davidson, K. Wagstaff, "Constrained Clustering: Advances in Algorithms" *Theory, and Applications*, 2008.

[13] J. McQueen, "Some methods for classification and analysis of multivariate observations", *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* vol. 1, str. 281–297, 1967.

[14] S. Basu, A. Banerjee, R. Mooney, "Active semi-supervision for pairwise constrained clustering" *In Proceedings of the 2004 SIAM International Conference on Data Mining*, str. 333–344, 2004.

[15] S. Dasgupta, "Performance guarantees for hierarchical clustering" *Journal of Computer and System Sciences*, vol 7, no. 4, str. 555–569, 2002.

[16] C. Ding, "A tutorial on spectral clustering", *Statistics and Computing* vol. 17, no. 4, 2007.

[17] J. Shi, J. Malik, "Normalized cuts and image segmentation", *IEEE transactions on pattern analysis and machine intelligence*, str. 888–905, 2000.

[18] D. Klein, S. D. Kamvar, C. D. Manning, . "From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering", *In Proceedings of the 19th International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc, str. 307–314, 2002.

[19] T. Dalenius, "The problem of optimum stratification I", *Scandinavian Actuarial Journal* vol. 1950, no. 3-4, str. 203-213, 1950.

[20] V. Estivill-Castro, "Why so many clustering algorithms", *A Position Paper, SIGKDD Explorations* vol. 4, no. 1, str. 65-75, 2002.

[21] I. Davidson, K. Wagstaff, S. Basu, "Measuring constraint-set utility for partitional clustering algorithms", *Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases*, str. 115-126, 2006.

[22] I. Davidson, S. S. Ravi, "Identifying and generating easy sets of constraints for clustering", *In Proceedings of the 21st national conference on Artificial intelligence*, vol. 1, str. 336–341, 2006.

[23] P. J. Rousseeuw, "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". *Computational and Applied Mathematics*, vol. 20, str. 53–65, 1987.

[24] S. Petrović, "A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters". *Proceedings of the 11th Nordic Workshop on Secure IT-systems*, str. 53-64, 1987.

[25] M. Možina, J. Žabkar, I. Bratko, "Argument based machine learning", *Artificial Intelligence*, vol. 171, no. 10–15, str. 922–937, 2007.

[26] E. A. Feigenbaum, "Knowledge engineering: the applied side of artificial intelligence", in *Proceedings of a symposium on Computer culture: the scientific, intellectual, and social impact of the computer*, str. 91–107, 1984.

[27] J. H. Boose, "A survey of knowledge acquisition techniques and tools", *Knowledge Acquisition*, vol. 1, no. 1, str. 3–37, 1989.

[28] T. Chklovski, "Using Analogy to Acquire Commonsense Knowledge from Human Contributors", *Ph.D. thesis*, MIT Artificial Intelligence Laboratory, 2003.

[29] P. Clark, R. Boswell, "Rule induction with CN2: Some recent improvements", in *Machine Learning - Proceeding of the Fifth Europen Conference* , str. 151–163, 1991.

[30] E. A. Feigenbaum, "Some challenges and grand challenges for computational intelligence", *Journal of the ACM*, vol. 50, no. 1, str. 32–40, 2003.

[31] M. Guid, M. Možina, V. Groznik, D. Georgiev, A. Sadikov, Z. Pirtošek, I. Bratko. "ABML knowledge refinement loop: a case study", *Foundations of intelligent systems. Lecture notes in artificial intelligence*, vol. 7661, str. 41-50, 2002.

[32] M. Možina, M. Guid, J. Krivec, A. Sadikov, I. Bratko, "Fighting Knowledge Acquisition Bottleneck with Argument Based Machine Learning". ECAI 2008, Patras, Greece, July 21-25, 2008. ECAI 2008 - 18th *European Conference on Artificial Intelligence, Proceedings. Frontiers in Artificial Intelligence and Applications*, vol. 178, str. 234-238, 2008.

[33] M. Pavlič, "Ocenjevanje kvalitete argumentov pri argumentiranem strojnem učenju", *Masters thesis* , Fakulteta za računalništvo in informatiko, 2015.

[34] C. E. Shannon, "A Mathematical Theory of Communication". *Bell System Technical Journal*, vol. 27, str. 379–423, 1948.

[35] A. K. H. Tung, R. T. Ng, L. V. S. Lakshmanan, J. Han. "Constraint-based clustering in large databases", *Proceeding 8th international conference Database Theory*, str. 405–419, 2001.

[36] M. Lichman (2013) "UCI Machine Learning Repository". Dostopno na: `http://archive.ics.uci.edu/ml`, , pridobljeno 18. 6. 2014.

[37] M. Zawiślak (2014) "Constrained K-means Clustering with Background Knowledge". Dostopno na: `https://github.com/mateuszzawislak/k-means-clustering`, pridobljeno 18. 7. 2015.

[38] R. A. Fisher (1988) "Iris Data Set". Dostopno na: `https://archive.ics.uci.edu/ml/datasets/Iris`, pridobljeno 18. 9. 2014.

[39] S. Aeberhard (1991) "Wine Data Set". Dostopno na: `https://archive.ics.uci.edu/ml/datasets/Wine`, pridobljeno 18. 9. 2014.

[40] J. Czerniak (2009) "Acute Inflammations Data Set". Dostopno na: `https://archive.ics.uci.edu/ml/datasets/Acute+Inflammations`, pridobljeno 18. 9. 2014.

[41] P. Šaponja, M. Guid (2015) "argument-based k-means". Dostopno na: `http://www.ailab.si/abkm/`, pridobljeno 28. 9. 2015.