SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 5078

Ostvarenje i analiza algoritama za rudarenje podataka

Borna Bejuk

Zagreb, lipanj 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 8. ožujka 2017.

ZAVRŠNI ZADATAK br. 5078

Pristupnik: **Borna Bejuk (0036477647)**
Studij: Računarstvo
Modul: Računarska znanost

Zadatak: **Ostvarenje i analiza algoritama za rudarenje podataka**

Opis zadatka:

Opisati probleme u području rudarenja podataka i otkrivanja znanja. Opisati, implementirati, usporediti i analizirati primjere algoritama korištenih u rudarenju podataka. Posebnu pažnju posvetiti uporabi algoritama u grupiranju skupova podataka. Ocijeniti i komentirati učinkovitost ostvarenih algoritama s obzirom na vrstu problema i parametre postupka. Radu priložiti izvorne tekstove programa, dobivene rezultate uz potrebna objašnjenja i korištenu literaturu.

Zadatak uručen pristupniku: 10. ožujka 2017.
Rok za predaju rada: 9. lipnja 2017.

Mentor:

Izv. prof. dr. sc. Domagoj Jakobović

Djelovođa:

Doč. dr. sc. Tomislav Hrkać

Predsjednik odbora za završni rad modula:

Prof. dr. sc. Siniša Srbljić

SADRŽAJ

1. Uvod

Živimo u svijetu u kojem se svakodnevno prikupljaju goleme količine podataka. Računala su svuda oko nas i omogućuju nam jednostavno pohranjivanje podataka. Uz pad cijena računala i mogućnosti udaljenih mreža računala koje nam stoje na raspolaganju nije se teško odlučiti na spremanje svih prikupljenih podataka.[6] Naše svakodnevne aktivnosti djelomično su zabilježene kroz korištenje naših računala, pametnih telefona i kreditnih kartica.[6] Nadalje, svjetske kompanije prikupljaju gigantske skupove podataka vezane uz prodajne transakcije, trgovanje dionicama, promotivne akcije i povratne informacije korisnika. Domene poput medicine, znanosti, društvenih mreža i web pretraživanja generiraju petabajte podataka na dnevnoj bazi.[3] Izvora koji generiraju velike količine podataka je mnogo, a alati za otkrivanje korisnih informacija među skupovima podataka nikada nisu bili potrebniji. Iz ove potrebe je proizašlo područje rudarenja podataka.

Jedan primjer izvora podataka jest internet pretraživač, npr. Google. Pretraživač zaprima stotine milijuna upita svakog dana. Svaki upit može predstavljati korisnikovu potrebu za informacijom. Neki od uzoraka poprimljenih iz korisnikovog niza upita mogu otkriti korisne informacije koje se ne mogu otkriti iz samo jednog pojedinog upita. Na primjer, agregiranjem upita vezanih uz bolest poput gripe, Googleov tim je uspješno predviđao periode aktivnosti bolesti. Drugi primjer jest problem neodanih korisnika u visoko konkurentnom tržištu. Baza podataka korisnikovih odabira i profila drži ključ ovog problema. Uzorci ponašanja prijašnjih korisnika se mogu analizirati kako bi se identificirale karakteristične osobine korisnika koji su skloni mijenjanju proizvoda i onih koji su odani proizvodu. Tvrtka može ponuditi korisnicima sklonim mijenjanju proizvoda specijalnu akciju kako bi ih pokušala zadržiti koja inače ne bi bila isplativa kada bi se ponudila svim njihovim korisnicima. Iste tehnike mogu se upotrijebiti za identificiranje korisnika koji bi bili zainteresirani za neku drugu uslugu koju tvrtka nudi.

U prvom poglavlju je opisano područje rudarenja podataka i njegove najbitnije grane. U drugom poglavlju je opisana grana grupiranja podataka i njezini najčešće

korišteni algoritmi. Zatim slijedi poglavlje koje opisuje najjednostavniji i najpoznatiji alogoritam grupiranja podataka - algoritam k-srednjih vrijednosti. Na kraju je opisan algoritam maksimizacije vrijednosti i njegove prednosti.

2. Rudarenje podataka

Rudarenje podataka (engl. *data mining*) je proces otkrivanja uzoraka u velikim skupovima podataka koji uključuje metode koje se nalaze na raskrižju umjetne inteligencije, strojnog učenja, statistike i baza podataka. To je interdisciplinarno područje računarske znanosti. Krajni cilj rudarenja podataka jest izvući informacije iz skupa podataka i transformirati ih u razumljivu strukturu za daljnju upotrebu.

2.1. Podjela zadataka u rudarenju podataka

Četiri zadatka u rudarenju podataka su temelji procesa rudarenja. To su grupiranje, klasifikacija, traženje pravila asocijacija i detekcija stršećih vrijednosti. Ovi zadatci se redovito pojavljuju u kontekstu mnogih primjena rudarenja podataka. Pokušati ćemo ih pobliže objasniti kroz primjer baze podataka $\mathcal{D}$ koju ćemo opisati kao $n \times d$ matricu D u kojoj svaki redak predstavlja jedno opažanje, a stupac jednu značajku. Rudarenje podataka u širem smislu jest traženje uzoraka u odnosima između redaka ili stupaca ovakve matrice.[1]

2.1.1. Grupiranje

Široka i neformalna definicija grupiranja jest sljedeća:

Definicija *Uz danu matricu D, grupiraj njene retke u skupove $C_1, \ldots, C_k$ tako da su retci u svakom skupu "slični" jedni drugima.*

Problem grupiranja se također može definirati kao optimizacijski problem u kojem su varijable optimizacije pripadnosti primjera grupama, a optimiziramo funkciju koja je konkretna matematička kvantifikacija sličnosti primjera unutar jedne grupe. Dozvoljen je široki spektar definicija funkcija sličnosti koje su od velike važnosti u procesu rudarenja. Njih ćemo pobliže objasniti u sljedećim poglavljima.

2.1.2. Klasifikacija

Neformalno, klasifikacija je problem identificiranja člana skupa kategorija kojem određeni primjer pripada na temelju već klasificiranih primjera. Klasični zadatak jest razvrstavanje elektroničke pošte na spam i ne-spam. Na raspolaganju nam stoji sadržaj pošte kao i prethodno klasificirana pošta. Nakon učenja na već klasificiranim primjerima, algoritam bi trebao moći pouzdano odlučiti je li novi primjer pošte spam ili ne-spam. Neformalnu definiciju bismo mogli dati kao:

Definicija *Uz danu $n \times d$ matricu D za treniranje i oznake klasa {1...k} povezane sa svakim od n redova u matrici D, stvori model za treniranje M koji će predviđati oznake za primjere iz Y $\nsubseteq$ D.*

2.1.3. Traženje pravila asocijacija

U primitivnoj formi, traženje pravila asocijacija je definirano u kontekstu *raspršenih baza podataka* u kojima su podatci samo nule i jedinice i većina ih je nula. Većina baza podataka transakcija su ovakvog tipa. Naprimjer, ako svaki redak u matrici podataka odgovara jednoj transakciji, a stupac proizvodu, tada je polje (i, j) jednako jedan ako je u transakciji *i* kupljen proizvod *j*. U kontekstu binarne baze podataka, problem traženja pravila asocijacija mogli bismo neformalno definirati na sljedeći način:

Definicija *Uz danu binarnu $n \times d$ matricu podataka D, odrediti sve podskupove stupaca matrice takve da sve vrijednosti u stupcu imaju vrijednost 1 u barem određenom postotku **s** redaka matrice. Relativna frekvencija nekog uzorka jest njegova potpora. Postotak **s** se naziva minimalna podrška.*

Uzorci čiji postotak pojavljivanja zadovoljava minimalnu podršku se nazivaju *česti uzorci*. Na primjer, ako stupci u matrici *D* koji odgovaraju vrijednostima *Kruh*, *Mlijeko* i *Maslac* često imaju vrijednost 1 u istim redovima matrice, možemo zaključiti da se ovi proizvodi često kupuju zajedno. To može biti od velike važnosti prodavaču iz perspektive smještaja proizvoda u dućanu i povećanju prodaje.

2.1.4. Detekcija stršećih vrijednosti

Stršeća vrijednost je primjer koji značajno odstupa od većine primjera u skupu podataka. Podatke s kojima raspolažemo su obično generirali jedan ili više procesa. Neobično ponašanje procesa rezultira generiranjem stršeće vrijednosti. Takve vrijednosti

nam mogu dati uvid u specifičnosti procesa. U kontekstu matrice podataka, detekciju stršećih vrijednosti možemo definirati na sljedeći način:

Definicija *Uz danu matricu D, odrediti retke matrice koji su vrlo različiti od preostalih redaka u matrici.*

2.2. Proces izvlačenja znanja

Dok neki pojam otkrivanja znanja iz podataka (engl. *knowledge discovery from data, KDD*) smatraju sinonimom za rudarenje podataka, drugi gledaju na rudarenje podataka kao tek jedan od koraka u procesu izvlačenja znanja.[3] Proces izvlačenja znanja se može definirati kao iterativni niz sljedećih koraka:

1. *Prikupljanje podataka:* Od velike je važnosti raspolagati sa podatcima iz kojih možemo izvući pravilne zaključke. Pomni odabiri u ovoj fazi mogu imati velike posljedice na donesene zaključke. Prikupljanje može zahtijevati uporabu specijaliziranog sklopovlja, specifičnih programa ili provođenja anketa. Podatci se zatim najčešće spremaju u bazu podataka na čuvanje prije same obrade.

2. *Definiranje cilja otkrivanja znanja:* Važno je definirati pitanje na koje želimo moći odgovoriti na kraju procesa kako bismo znali koje metode nad kojim podatcima upotrijebiti.

3. *Transformacija podataka:* Nakon prikupljanja, podatci najčešće nisu u obliku koji je prikladan za rudarenje. Jedan od načina transformacije bio bi izvlačenje značajki koje predstavljaju razna izmjerena svojstva s kojima mnogo algoritama strojnog učenja zna raditi. Čišćenje podataka se često radi paralelno s njihovom transformacijom. Pogrešne i podatke koji nedostaju se ispravljuje i procjenjuje.

4. *Primjena algoritama strojnog učenja:* Ključan dio u kojem se primjenjuju inteligentne metode nad prikupljenim podacima. U nekim slučajevima bit će potrebno razdijeliti primjenu u manje dijelove nad kojima ima smisla raditi neke od četiri velikih metoda rudarenja.

5. *Interpretacija rezultata:* Naposljetku nam preostaje interpretirati rezultate. Moguće je korištenje različitih vizualizacija i tehnika za predstavljanje izvučenog znanja.

3. Grupiranje

Grupiranje podataka je proces podjele primjera u grupe (klastere) primjera. Slični primjeri su podijeljeni u istu grupu. Svi primjeri u jednoj grupi su slični po nekom svojstvu, a različiti od primjera koji se nalaze u drugim grupama.[4] Cilj grupiranja jest otkrivanje grupa, ako postoje, u skupu neoznačenih podataka.

Grupiranje podataka se koristi kroz čitav niz domena poput poslovne inteligencije, biologije i internet sigurnosti. Grupiranje također možemo naći u raspoznavanju uzoraka, ali i kod pretraživanja, dohvaćanja i prezentiranja web sadržaja.

Najčešća primjena grupiranja jest istraživanje podataka (engl. *data exploration*) čija je svrha pronalaženje skrivene strukture u podatcima. Nakon grupiranja, otvara nam se mogućnost da grupe ručno označimo te da ustvrdimo značajke tih grupa.[4] Time smo dobili pregledniji uvid u podatke i potencijalno otkrili pravilnosti u podatcima i odnose između grupa.

3.1. Vrste grupiranja

Grupiranje dijelimo u dvije osnovne vrste, particijsko i hijerarhijsko. Prilikom particijskog grupiranja skup primjera se dijeli u grupe sličnih primjera. Hijerarhijsko grupiranje razdjeljuje skup primjera u ugniježđene grupe koje sačinjavaju hijerarhiju grupa.

Drugi način na koji možemo podijeliti grupiranje jest prema čvrstoći granica između grupa. Kod čvrstog grupiranja svaki primjer može pripadati isključivo jednoj grupi. Kod mekog grupiranja jedan primjer može pripadati u više grupa ili može pripadati u više grupa s različitim vjerojatnostima pripadanja.

Grupiranje, također, možemo razdvojiti na temelju kriterija koji se koristi za grupiranje. Kriterij može biti minimizacija kriterijske funkcije, maksimizacija izglednosti ili se za grupiranje može koristiti funkcija udaljenosti.[4]

3.2. Particijsko grupiranje

Particijsko grupiranje je najjednostavnija i najosnovnija vrsta grupiranja koja organizira objekte skupa u nekoliko grupa. Pretpostavljamo da je broj grupa koje želimo naći unaprijed zadan. Taj broj je početna točka za particijsko grupiranje. Formalno, uz dani skup podataka D, u kojem se nalazi n objekata i k grupa koje želimo naći, particijski algoritam organizira objekte u k particija ($k <= n$), gdje svaka particija predstavlja jednu grupu.[3]

Grupe se formiraju tako da se optimizira particijski kriterij poput minimiziranja sume kvadratnih udaljenosti u slučaju algoritma k-srednjih udaljenosti. Time ćemo dobiti grupe unutar kojih su primjeri slični jedni drugima i različiti naspram primjera u drugim grupama.

3.3. Hijerarhijsko grupiranje

U nekim slučajevima htjeli bismo podijeliti podatke u grupe na različitim razinama. Prikaz podataka u obliku hijerarhije je korisno prilikom sažimanja i prikaza podataka. Postoji više metoda pomoću kojih grupiramo podatke poput funkcije udaljenosti ili gustoće. Odluka o dijeljenju ili spajanju podataka je bitna jer nakon te odluke nema povratka, već se ponavlja isti postupak nad novonastalim grupama.[1] Hijerarhijsko grupiranje možemo podijeliti na temelju smjera gradnje hijerarhije grupa:

1. *Bottom-up metode:* Krenuvši od pojedinačnih podataka penjemo se do vrha stabla tako da ih grupiramo u grupe više razine. Glavna razlika između metoda jest odluka spajanja podataka u grupe.

2. *Top-down metode:* Oblikovavši početne grupe rekurzivno ih dijelimo u podgrupe.

4. Algoritam k-srednjih vrijednosti

Najjednostavniji i najpoznatiji algoritam grupiranja je algoritam k-srednjih vrijednosti (engl. *k-means algorithm*). Formalno, algoritmom se primjeri iz neoznačenog skupa primjera $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$ grupiraju u *K* čvrstih grupa. Parametar *K* zadajemo unaprijed. Svaki centroid $\{\mu_k\}_{k=1}^{K}$ predstavlja po jednu grupu.[4] Moramo imati način da objektivno kvantificiramo pogrešku trenutnog stanja algoritma što nam omogućuje ciljna funkcija (engl. *objective function*). Koristimo sumu kvadrata udaljenosti svih točaka od njihovih centroida:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} b_k^{(i)} \|x^{(i)} - \mu_k\|^2 \tag{4.1}$$

gdje je $\|\cdot\|$ euklidska norma. Vrijednost $b_k^{(i)}$ u (4.1) je varijabla koja nam govori kojoj grupi pripada primjer $x^{(i)}$. Ako je $b_k^{(i)} = 1$, primjer $x^{(i)}$ pripada grupi k. Očito, želimo li minimizirati pogrešku J svaki primjer $x^{(i)}$ trebamo svrstati u grupu čije je središte μ_k tom primjeru najbliže. Želimo pronaći vrijednosti μ_k takve da minimiziraju pogrešku J. Budući da vrijednosti $b_k^{(i)}$ ovise o μ_k, optimalne vrijednosti za μ_k nije moguće izraziti u zatvorenoj formi. Algoritam k-srednjih vrijednosti će iz toga razloga optimizaciju provoditi iterativno. Algoritam započinje sa slučajno odabranim srednjim vrijednostima μ_k. U svakoj iteraciji se za svaki primjer $x^{(i)}$ izračunava pripadnost primjera najbližem centroidu, $b_k^{(i)}$. Nakon toga možemo minimizirati pogrešku (4.1) postavljanjem $\nabla_{\mu_k} J = 0$ i rješavanjem po μ_k. Dobivamo:

$$\mu_k = \frac{\sum_i b_k^{(i)} x^{(i)}}{\sum_i b_k^{(i)}} \tag{4.2}$$

. Budući da smo ovime promijenili vrijednosti vektora μ_k, treba ponovno izračunati vrijednosti $b_k^{(i)}$. Ova dva koraka ponavljaju se dok se ne postigne konvergencija vrijednosti μ_k. Konzistentnost pri razrješavanju izjednačenja udaljenosti između točke i centroida je ključna kako algoritam ne bi ušao u beskonačnu petlju.[4] Pseudokod opisanog algoritma bio bi sljedeći:

Algoritam 1. Algoritam k-srednjih vrijednosti

1: **inicijaliziraj** centroide $\mu_k, k = 1, ..., K$

2: **ponavljaj**

3: za svaki $x^{(i)} \subseteq D$

4: $$b_k^{(i)} = \begin{cases} 1, & \text{ako } k = argmin_j \|x^{(i)} - \mu_j\| \\ 0, & \text{inače} \end{cases}$$

5: za svaki $\mu_k, k = 1, ..., K$

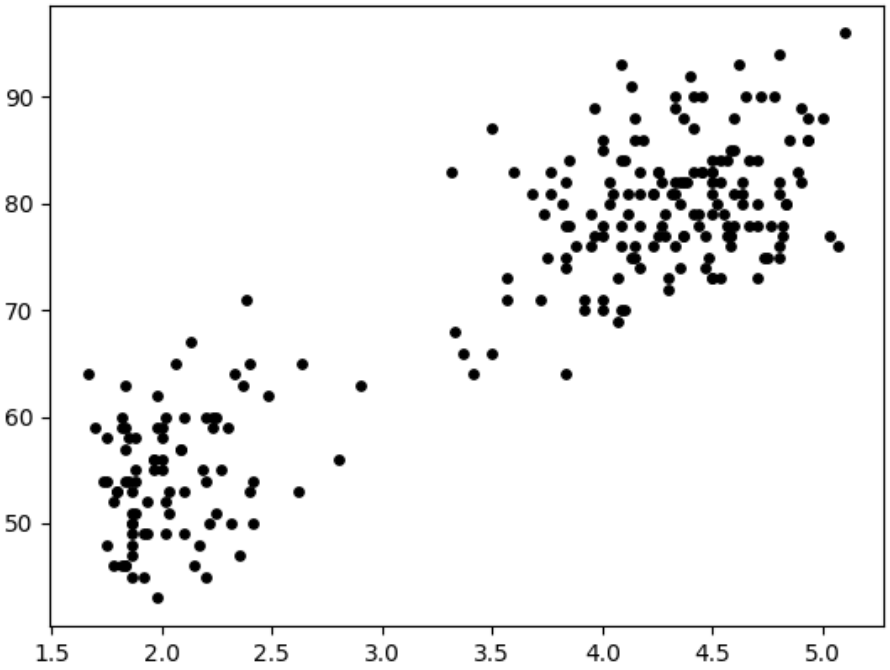
6: $\mu_k = \sum_{i=1}^{N} b_k^{(i)} x^{(i)} / \sum_{i=1}^{N} b_k^{(i)}$

7: **dok** μ_k ne konvergiraju

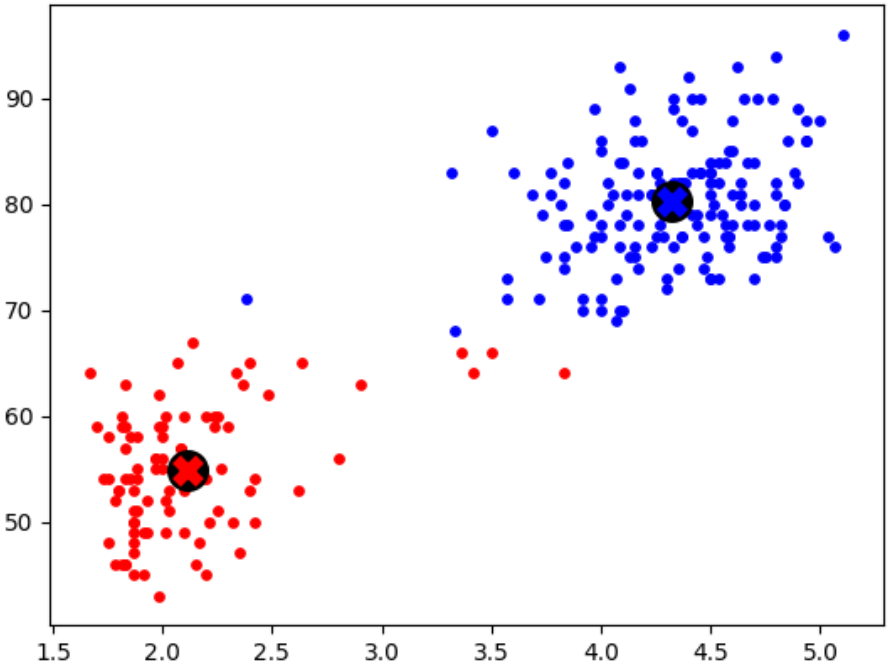
Algoritam k-srednjih vrijednosti je pohlepan algoritam i ne nalazi uvijek optimalno grupiranje. Je li rješenje koje je algoritam pronašao ujedno i globalno optimalno rješenje uvelike ovisi o izboru početnih srednjih vrijednosti μ_k. Mogućnosti odabira početnih srednjih vrijednosti je više:

- Nasumično odabrati K primjera kao početne vrijednosti μ_k. Ovime nećemo eliminirati mogućnost ulaska u lokalni optimum, ali ćemo izbjeći postavljanje centroida u prostor u kojem uopće nema primjera. Problem mogu biti primjeri koji odskaču (engl. *outliers*) koji lako mogu završiti u zasebnim grupama što želimo izbjeći jer je broj grupa ograničen i htjeli bismo da se naše grupe što više podudaraju sa stvarnim grupama koje postoje u podatcima.[4]
- Izračunati srednju vrijednost svih primjera i pomoću nje odrediti početne centroide.
- Slučajno odabrati jedno početno središte μ_k, a zatim svako iduće odabrati tako da je što dalje od ostalih središta. Ovaj pristup se naziva **k-means++**.

U nastavku ćemo proučiti jednu od tih implementacija, **k-means++**, koja će nam u većini slučajeva omogućiti da izbjegnemo loša grupiranja. Ilustrirat ćemo rad algoritma k-srednjih vrijednosti nad poznatim skupom podataka koji opisuje vremensko trajanje erupcije gejzira *Old Faithful* u relaciji s vremenom čekanja do sljedeće erupcije. Na slici 4.1 je prikazan graf na kojem smo označili mjerenja s kojima raspolažemo. Na x osi nalazimo vrijeme trajanja erupcije dok je na y osi vrijeme čekanja do sljedeće erupcije. Oba vremena su izražena u minutama. Na slici 4.2 je prikazano grupiranje nakon provedbe algoritma. Možemo primjetiti da je algoritam pronašao originalne grupe u podatcima.



Slika 4.1: Podatci s kojima raspolažemo iscrtani na grafu.



Slika 4.2: Grupirani podatci i njihovi centroidi nakon provedbe algoritma.

4.1. Algoritam k-means++

Algoritam k-means++ je algoritam za odabir početnih srednjih vrijednosti, μ_k, u algoritmu k-srednjih vrijednosti. Predložen 2007. godine kao poboljšanje algoritma k-srednjih vrijednosti nudi rješenje za ponekad njegove vrlo slabe rezultate. Algoritam će najprije slučajno odabrati jedan početni centroid iz skupa svih primjera te zatim izračunati vektor udaljenosti svih točaka od tog centroida. Sljedeći centroid će biti izabran tako da udaljeniji primjeri od prije odabranih centroida imaju veće šanse da budu odabrani.[2] Algoritam bi u pseudokodu glasio ovako:

Algoritam 2. Algoritam k-means++

1: slučajno izabrati prvi centroid μ_1 među primjerima u skupu D
2: za svaki $x^{(i)} \subseteq D$
3: izračunati vektor udaljenosti od početnog centroida $D_i^2 = \|x^{(i)} - \mu_1\|^2$
4: **ponavljaj**
5: izabrati sljedeći centroid μ_l slučajno prema distribuciji $D_i^2 / \sum_{j=1}^{N} D_j^2$ ▷ udaljenost razmatranog primjera od njemu najbližeg centroida kroz sumu svih udaljenosti primjera od njima najbližeg centroida
6: ponovno izračunati vektor udaljenosti $D_i^2 = min(\|x^{(i)} - \mu_1\|^2, ..., \|x^{(i)} - \mu_l\|^2)$
7: **dok** svi centroidi nisu odabrani

5. Algoritam maksimizacije vrijednosti

Algoritam maksimizacije vrijednosti je algoritam koji pripada području mekog grupiranja. Primjeri više ne pripadaju isključivo jednoj grupi, već mogu biti grupirani u više grupa ili u više grupa s različitim vjerojatnostima pripadanja u slučaju probabilističkog pristupa. Formalno, algoritam maksimizacije očekivanja (engl. *expectation maximization algorithm*) ili EM-algoritam je iterativan optimizacijski postupak za rješavanje problema najveće izglednosti kod modela s latentnim varijablama.[4] Latentna varijabla je slučajna varijabla koju ne opažamo izravno u podatcima već o njoj zaključujemo na temelju drugih opaženih varijabli. Latentna varijabla može biti uvedena kao apstrakcija s ciljem pojednostavljenja modela. Latentnu varijablu koja modelira nešto stvarno, ali nedostupno, nazivamo skrivena varijabla. U kontekstu probabilističkog grupiranja, algoritam maksimizacije vrijednosti kreće od inicijalnog skupa parametara i iterira dok grupiranje ne konvergira ili je promjena parametara dovoljno mala, ispod određene granice. Svaka iteracija se sastoji od dva koraka:

- **E korak** grupira primjere na temelju trenutnih parametara.
- **M korak** pronalazi nove parametre na temelju trenutnog grupiranja primjera u grupe.

5.1. Neizrazito grupiranje

Ako umjesto pripadnosti ili nepripadnosti nekog elementa skupa govorimo o mjeri u kojoj neki element pripada nekom skupu tada govorimo o neizrazitom skupu. U kontekstu grupiranja, omogućuje nam da svakom primjeru pridružimo vjerojatnost pripadanja grupi. Formalno, uz dani skup objekata $o_1, \ldots, o_n$, neizrazito grupiranje u k grupa, $C_1, \ldots, C_k$, se može predstaviti matricom grupiranja, $M = [w_{ij}](1 \leq i \leq n, 1 \leq j \leq k)$ gdje je w_{ij} vjerojatnost pripadanja primjera o_i grupi C_j. Jednostavan vizualan primjer neizrazitog grupiranja koristeći EM-algoritam može biti grupiranje točaka na grafu u dvije grupe.

Slučajno odaberemo dvije točke koje će biti početni centroidi. Prva iteracija provodi E i M korak na sljedeći način: U E koraku za svaki primjer izračunamo njegovu pripadnost grupama c_1 i c_2 tako da izračunamo euklidsku udaljenost primjera do centroida prve i druge grupe i podijelimo s zbrojem tih udaljenosti kako bi normalizirali vjerojatnosti. Izraz $udaljenost(p_i, c_j)$ predstavlja udaljenost primjera p_i do centroida c_j, a w_{ij} vjerojatnost pripadanja primjera i grupi j.[3] U općenitom slučaju s K grupa i N primjera formule su sljedeće:

$$w_{ij} = 1 - \frac{udaljenost(p_i, c_j)}{\sum_{j=1}^{K} udaljenost(p_i, c_j)} \tag{5.1}$$

za svaki $i = 1, ..., N$ u E koraku, i

$$c_j = \frac{\sum_{i=1}^{N} w_{ij}^2 p}{\sum_{i=1}^{N} w_{ij}^2} \tag{5.2}$$

za svaki $j = 1, ..., K$ u M koraku. Jednadžbu (5.2) bi koristili za izračun svakog parametra grupe koliko god ih ona imala. Ponavljamo iteracije dok centroidi ne konvergiraju.

5.2. Primjena na modelu Gaussove miješane gustoće

Pretpostavimo da želimo pronaći K grupa, $C_1, ..., C_K$ u skupu D koji se sastoji od n primjera. Možemo pretpostaviti da je D formiran na sljedeći način. Svaka grupa $C_j (1 \leq j \leq K)$ je povezana s vjerojatnošću, π_k, da je generirala promatrani primjer. Vrijedi $\sum_{k=1}^{K} \pi_k = 1$ što nam osigurava da su svi primjeri generirani od strane K grupa. Zatim su dva koraka izvršena n puta kako bismo dobili N primjera.[3] To su:

1. Odabrati grupu C_j na temelju vjerojatnosti $\pi_1, ..., \pi_K$.

2. Generirati primjer grupe C_j na temelju njezine funkcije gustoće f_j.

Ovaj proces je temeljna pretpostavka modela miješanih gustoća. Formalno, model miješane gustoće pretpostavlja da je skup opaženih objekata mješavina primjera iz više različitih distribucija. Uz dani skup D i grupe $C_1, ..., C_k$ i njihove parametre $\theta_1, ..., \theta_K$ možemo definirati vjerojatnost:

$$P(D|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k p(x^{(i)}|\theta_k). \tag{5.3}$$

Zadaća algoritma jest pronaći skup parametara takve da maksimiziraju vrijednost jednadžbe (5.3). Drugim riječima, cilj je pronaći parametre grupa tako da je vjerojatnost

da su te grupe generirale opaženi skup primjera maksimalna.[3] Miješana gustoća je linearna kombinacija K funkcija gustoća vjerojatnosti:

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\theta_k) \tag{5.4}$$

gdje su $p(x|\theta_k)$ komponente mješavine svaka s parametrima θ_k koje predstavljaju funkcije gustoće.[4] Ako se za komponente koriste Gaussove gustoće vjerojatnosti, $\mathcal{N}(\mu_k, \Sigma_k)$, miješanu gustoću nazivamo mješavina Gaussovih gustoća. Možemo primjeniti Bayesovo pravilo kako bismo izračunali aposteriorne vjerojatnosti $P(\theta_k|x)$:

$$P(\theta_k|x) = \frac{P(\theta_k)p(x|\theta_k)}{p(x)} = \frac{\pi_k p(x|\theta_k)}{\sum_j \pi_j p(x|\theta_j)} \equiv h_k. \tag{5.5}$$

Ovu veličinu označavamo s h_k i nazivamo odgovornost. Odgovornost h_k iskazuje koliko je vjerojatno da primjer x pripada grupi C_k. U općenitom slučaju viševarijatne Gaussove gustoće parametre koje želimo odrediti su:

$$\theta = \{P(\theta_k), \mu_k, \Sigma_k\}_{k=1}^{K}$$

gdje je Σ kovarijacijska matrica. Kovarijacijska matrica je kvadratna simetrična matrica koja na dijagonali ima varijance varijabli, a izvan dijagonale kovarijance svih parova varijabli.[4] Kovarijacijska matrica poopćuje pojam varijance u više dimenzija. Na primjer, varijanca u skupu slučajnih točaka u dvodimenzijskom prostoru se ne može okarakterizirati jednim brojem ili vektorima u x i y smjeru već je potrebna 2×2 matrica. Algoritam iterira kroz E i M korak dok parametri grupa ne konvergiraju. U E koraku za svaki primjer $x^{(i)} \in D$ i svaku komponentu $k = 1, ..., K$ izračunavamo odgovornosti temeljem trenutnih vrijednosti parametara. U M koraku za svaku komponentu $k = 1, ..., K$ izračunavamo procjene parametara temeljem trenutnih odgovornosti. Pseudokod algoritma dan je u nastavku.

Algoritam 3. Algoritam maksimizacije vrijednosti za Gaussovu mješavinu

1: **inicijaliziraj** parametre Gaussove mješavine, $\theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$

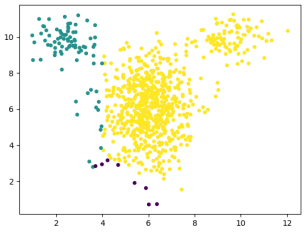
2: **ponavljaj**

3: **E korak**: Za primjere $x^{(i)} \in D$ i komponente $k = 1, ..., K$ izračunati:
$h_k^{(i)} = \pi_k p(x|\mu_k, \Sigma_k) / \sum_{j=1}^{K} \pi_j p(x|\mu_j, \Sigma_j)$

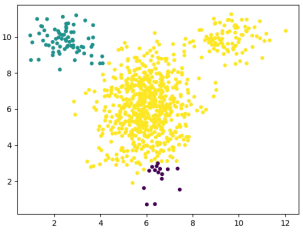
4: **M korak**: Za svaku komponentu $k = 1, ..., K$ izračunati:
$\mu_k = \sum_i h_k^{(i)} x^{(i)} / \sum_i h_k^{(i)}$
$\Sigma_k = \sum_i h_k^{(i)} (x^{(i)} - \mu_k)(x^{(i)} - \mu_k)^\mathsf{T} / \sum_i h_k^{(i)}$
$\pi_k = \frac{1}{N} \sum_{i=1}^{N} h_k^{(i)}$

5: **do** konvergencije parametara

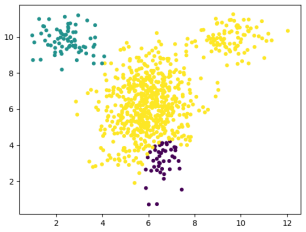
Za ilustraciju rada algoritma generirali smo jednostavan primjer gdje su podatci točke na dvodimenzijskom grafu. U podatcima se nalaze tri grupe koje se ravnaju po normalnoj distribuciji. Iz slika 5.1 do 5.9 možemo vidjeti kako algoritam napreduje i postepeno nalazi originalne grupe u podatcima. Možemo reći da je algoritam k-



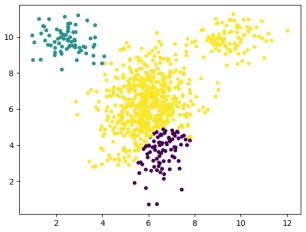
Slika 5.1: 1. iteracija



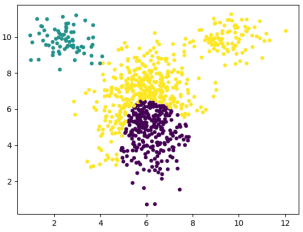
Slika 5.2: 5. iteracija



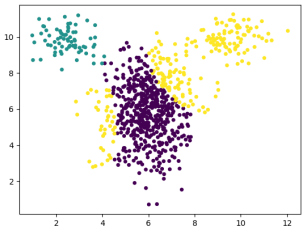
Slika 5.3: 10. iteracija



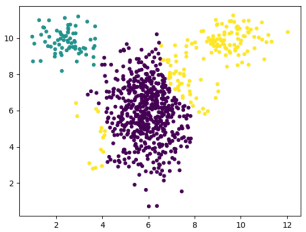
Slika 5.4: 15. iteracija



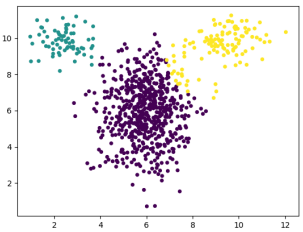
Slika 5.5: 25. iteracija



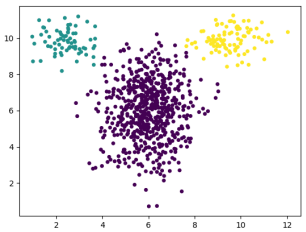
Slika 5.6: 30. iteracija



Slika 5.7: 35. iteracija

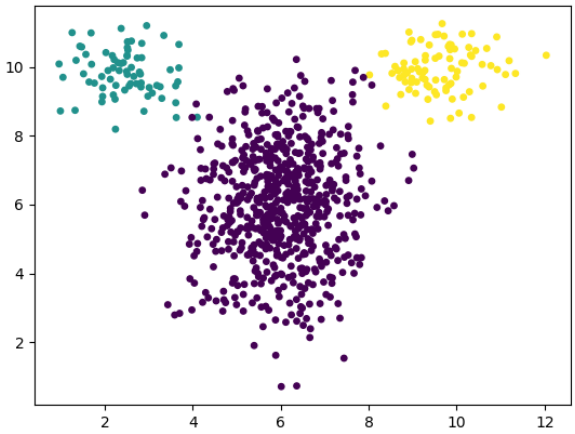


Slika 5.8: 45. iteracija

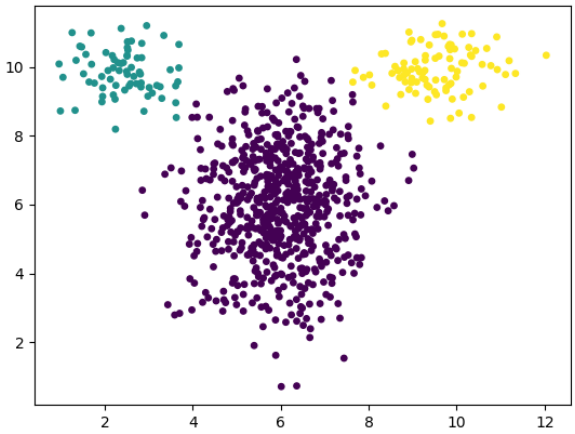


Slika 5.9: 65. iteracija

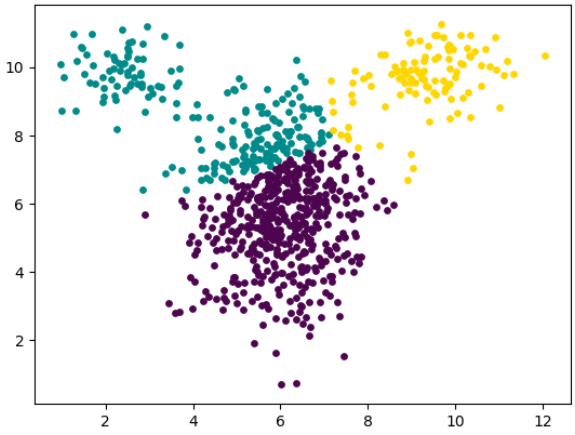
srednjih vrijednosti poseban slučaj algoritma maksimizacije vrijednosti. Razlikuju se po tome što algoritam k-srednjih vrijednosti provodi čvrsto grupiranje za razliku od mekog grupiranja EM-algoritma. Također, algoritam k-srednjih vrijednosti ne uzima u obzir kovarijance podataka. Taj nedostatak možemo ilustrirati na skupu podataka nad kojim ćemo provesti grupiranje jednim i drugim algoritmom. Na slici 5.10 su označene originalne grupe u podatcima. Na slici 5.11 možemo vidjeti da je algoritam maksimizacije vrijednosti u ovom primjeru uspješno pronašao originalne grupe dok na slici 5.12 možemo primjetiti da je algoritam k-srednjih vrijednosti zakazao. Algoritam k-srednjih vrijednosti pretpostavlja da su kovarijance podataka jednake nuli što mu onemogućuje da pronađe originalne grupe u ovom primjeru.



Slika 5.10: Originalne grupe.

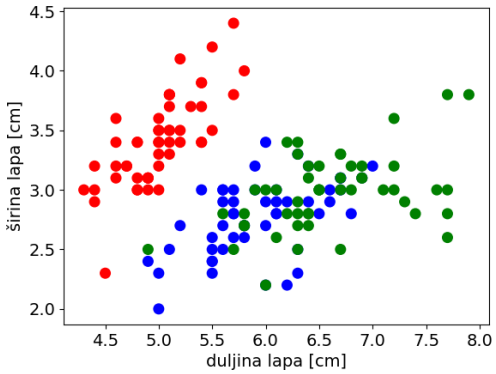


Slika 5.11: Grupe dobivene algoritmom maksimizacije vrijednosti.

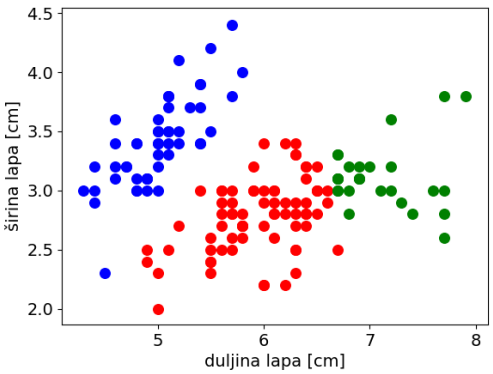


Slika 5.12: Grupe dobivene algoritmom srednjih vrijednosti.

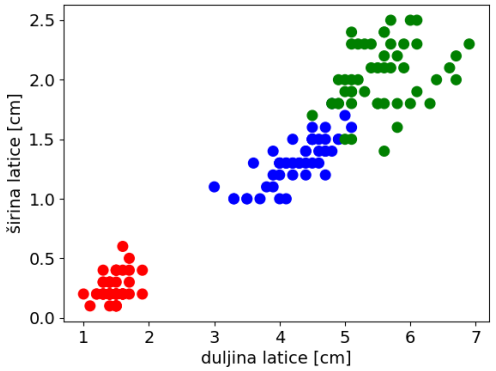
Sljedeći skup podataka nad kojim su algoritmi isprobani je skup podataka o cvijetu perunike koji je jedan od najpoznatijih u području strojnog učenja preuzet sa [5]. U skupu se nalaze podatci o trima vrstama tog cvijeta poput duljine i širine latice cvijeta. Za potrebe testiranja algoritama odabrano je nekoliko relacija tih podataka te su uspoređivani dobiveni rezultati sa stvarnim grupama. Na slici 5.13 prikazane su



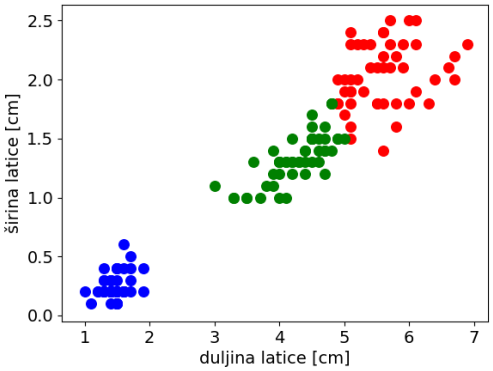
Slika 5.13: Originalne grupe.



Slika 5.14: Grupe dobivene algoritmom k-srednjih vrijednosti.



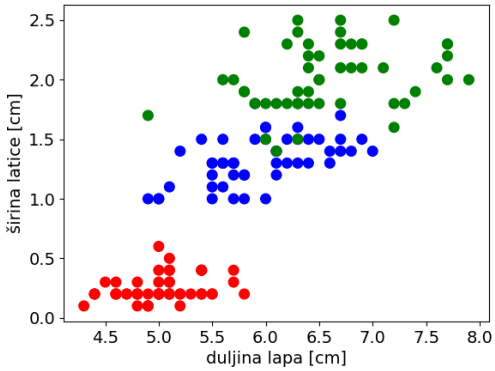
Slika 5.15: Originalne grupe.



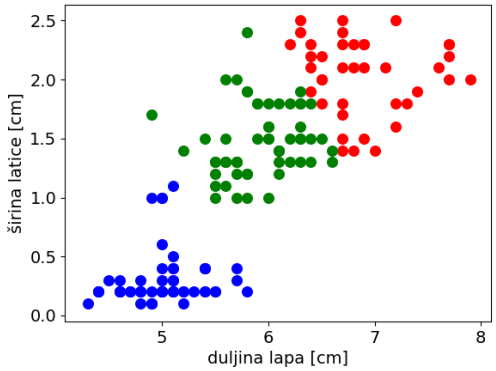
Slika 5.16: Grupe dobivene algoritmom k-srednjih vrijednosti.

originalne grupe dobivene usporedbom duljine lapa i širine lapa triju vrsta perunika. Podatci grupa označenih plavom i zelenom bojom su ispremješani te to predstavlja problem za algoritam k-srednjih vrijednosti što je primjetno na slici 5.14. Algoritam k-srednjih vrijednosti u ovom slučaju ne može primjerima ispravno odrediti pripadnosti grupama. Na slici 5.15 prikazane su grupe dobivene usporedbom duljine i širine latica. U ovom slučaju algoritam k-srednjih vrijednosti zadovoljavajuće dobro pronalazi originalne grupe što se vidi na slici 5.16. Na slici 5.17 prikazane su grupe dobivene usporedbom duljine lapa i širine latice. Na slici 5.18 je primjetno da algoritam k-srednjih

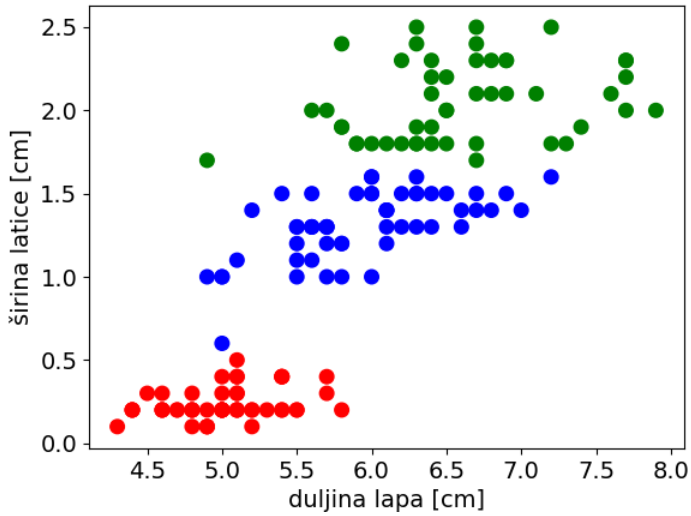
vrijednosti neće uspjeti pronaći originalne grupe, ali zato algoritam maksimizacije vrijednosti u tome uspijeva jer uzima u obzir moguću izduženost položaja primjera svake grupe što se vidi na slici 5.19.



Slika 5.17: Originalne grupe.



Slika 5.18: Grupe dobivene algoritmom k-srednjih vrijednosti.



Slika 5.19: Grupe dobivene algoritmom maksimizacije vrijednosti.

6. Zaključak

Rudarenje podataka i srodna područja u velikoj su potražnji s obzirom na goleme količine podataka koje se svakodnevno prikupljaju. Rezultati dobiveni primjenom algoritama nad podatcima daju zadovoljavajuće rezultate koji su korisni pri izvlačenju zaključaka iz podataka. Pokazano je da je algoritam maksimizacije očekivanja superiorniji od algoritma k-srednjih vrijednost u nalažanju originalnih grupa podataka u skupu podataka. Vizualizacija rezultata algoritama je od posebne važnosti kod prezentiranja rezultata grupiranja podataka jer nam olakšava uočavanje novih saznanja.

Rezultati se mogu značajno razlikovati u ovisnosti o odabiru maksimalnog broja iteracija i preciznosti koje zadamo algoritmima. Također je potrebno obratiti pozornost pri odabiru broja grupa koje tražimo u podatcima i odabir optimalnog broja bi mogao biti zasebna tema budućeg rada. Nadalje, od velike je važnosti i odluka o početnim srednjim vrijednostima o kojima će naše zadovoljstvo rezultatima algoritama često ovisiti. Moramo biti svjesni da algoritmi kreću od nekih pretpostavki te bismo najprije morali provjeriti zadovoljava li ih naš problem. Postoje razna ograničenja i negativne strane algoritama s kojima se vrijedi upoznati prije korištenja algoritma.

LITERATURA

[1] Charu C Aggarwal. *Data mining: the textbook*. Springer, 2015.

[2] DataScienceLab. Improved Seeding For Clustering With K-Means++. `https://datasciencelab.wordpress.com/2014/01/15/improved-seeding-for-clustering-with-k-means/`, 2014. [].

[3] Jiawei Han, Jian Pei, i Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[4] Bojana Dalbelo Bašić Jan Šnajder. *Strojno učenje*. Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva, 2012.

[5] M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[6] Ian H Witten, Eibe Frank, Mark A Hall, i Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

Ostvarenje i analiza algoritama za rudarenje podataka

Sažetak

Rad opisuje područje rudarenja podataka i grupiranja podataka. Ostvareni su i analizirani najčešći algoritmi u području grupiranja podataka poput algoritma k-srednjih vrijednosti i algoritma maksimizacije vrijednosti na nekoliko problema. Rezultati algoritama s obzirom na vrstu problema su uspoređeni i analizirani. Priložene su slike rezultata algoritama nad podatcima.

Ključne riječi: Algoritam srednjih vrijednosti, algoritam maksimizacije očekivanja, k-means++, čvrsto grupiranje, meko grupiranje, neizrazito grupiranje

Implementation and analysis of data mining algorithms

Abstract

This thesis describes the field of data mining and data clustering. Most common algorithms in data clustering such as k-means algorithm and EM algorithm have been analyzed and implemented on several problems. The results of algorithms considering the nature of problems have been compared and analyzed. Pictures of results of the algorithms have been enclosed.

Keywords: K-means algorithm, EM algorithm, k-means++, hard clustering, soft clustering, fuzzy clustering