Istraživanje podataka u bioinformatici

Klasterovanje podataka

Matematički fakultet, Univerzitet u Beogradu

May 29, 2026

Agenda

Problemi pri klasterovanju visokodimenzionih podataka

- Objekti su obično predstavljeni kao tačka u višedimenzionalnom prostoru.
- Skupovi podataka koji se obrađuju postaju sve složeniji sa velikim brojem objekata koji imaju veliki broj dimenzija.
- Javlja se problem skalabilnosti postojećih algoritama – potrebna je ili njihova modifikacija ili razvoj novih algoritama.
- Problem kvaliteta i brzine klasterovanja.
- Postojeće (klasične) mere sličnosti postaju neupotrebljive. Veliki broj dimenzija može da dovede do toga da su svi objekti međusobno jednako udaljeni.

Problemi pri klasterovanju visokodimenzionih podataka

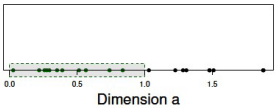
Pri klasterovanju visokodimenzionih podataka mogu da se jave sledeći problemi:

- Prokletstvo dimenzionalnosti
- Efekat koncentracije normi
- Problem relevantnosti (lokalnih) osobina/atributa

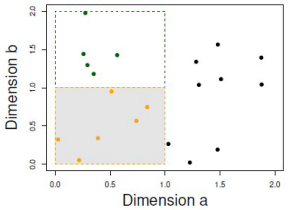
Prokletstvo dimenzionalnosti

- Veći broj atributa – više informacija, ali i više problema u obradi.
- Problem je i što objekti sadrže veliki broj „praznih“ atributa.
- Podaci postaju retki, a rastojanja između tačaka postaju slična.

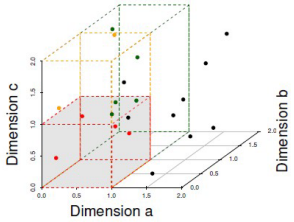
Prokletstvo dimenzionalnosti



(a) 11 Objects in One Unit Bin



(b) 6 Objects in One Unit Bin



(c) 4 Objects in One Unit Bin

Podaci u jednoj dimenziji su relativno gusto spakovani. Povećanje broja dimenzija razvlači i udaljava tačke, dovodeći do toga da su tačke u takvom prostoru relativno retke.

Efekat koncentracije normi

- Na primer, ako se koristi Euklidsko rastojanje kao mera, pokazuje se da takva mera ima druge karakteristike u odnosu na njenu primenu na objekte u prostorima male dimenzije. Pod određenim preduslovima relativna rastojanja od bilo koje tačke do njenog najbližeg i najdaljeg suseda imaju tendenciju da budu skoro identična.
- Kada relativne varijanse rastojanja (relativne u odnosu na srednju vrednost rastojanja) tačaka konvergiraju ka nuli za veće dimenzije, tada zajedno sa povećanjem broja dimenzija relativna razlika rastojanja do najbliže i najdalje tačke takođe teži ka nuli.

Problem relevantnosti atributa

- Čest slučaj je da se u velikom broju dimenzija javljaju atributi čija vrednost ima samo lokalno značenje za jedan (manji) broj objekata.
- Takvi atributi su nekorisni i mogu da smetaju npr. u slučaju dimenzione redukcije.
- Moguće je da su u nekim slučajevima pokazatelji nekih skrivenih klastera.

Dimenziona redukcija

- Tehnike dimenzione redukcije nisu efikasne u slučaju većeg broja irelevantnih atributa.
- Problem nastaje kod interpretacije originalnih domena posle redukcije i kada su objekti povezani na različite načine u različitim podskupovima dimenzija.
- Izbor karakteristika (atributa) je problematičan kada su klasteri u pojedinim potprostorima.

Algoritmi klasterovanja

Algoritmi klasterovanja

Algoritmi se mogu grupisati prema metodama koje koriste.

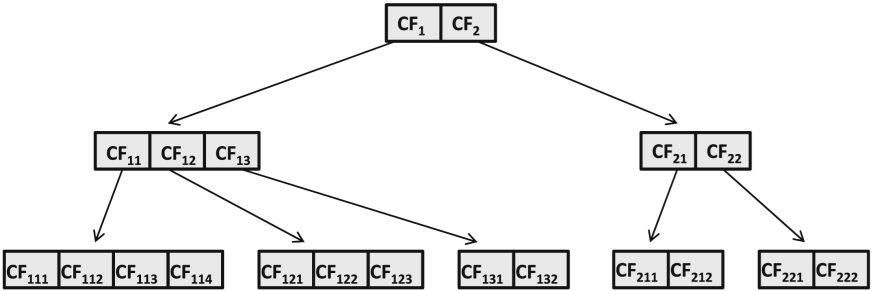
- **Zasnovani na hijerarhiji** (npr. BIRCH)
- **Zasnovani na podeli** (npr. K-sredina i varijante)
- **Zasnovani na gustini** (npr. DBSCAN)
- **Zasnovani na mreži** (npr. CLIQUE)
- **Zasnovani na modelu** (npr. EM)

Algoritmi zasnovani na hijerarhiji

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)

- Kombinacija hijerarhijskog klasterovanja tipa *odozgo nadole* i algoritma K-sredina.
- Koristi se hijerarhijska struktura podataka poznata kao **CF-stablo** (*CF-Tree*).
- CF-stablo predstavlja po visini balansiranu strukturu podataka koja hijerarhijski organizuje klastere.
- Svaki čvor ima faktor grananja najviše B, što odgovara njegovim B potklasterima, tj. deci.

BIRCH


CF1 CF2
CF11 CF12 CF13
CF21 CF22
CF111 CF112 CF113 CF114
CF121 CF122 CF123
CF131 CF132
CF211 CF212
CF221 CF222

BIRCH – CF-stablo

- Svaki čvor sadrži sažet prikaz svakog od najviše B potklastera na koje pokazuje.
- Sažetak klastera naziva se **klasterska karakteristika** (*cluster feature* – *CF*), odnosno **vektor klasterskih karakteristika**.
- Sažetak sadrži trojku (SS, LS, m), gde je:
  - SS – vektor koji sadrži zbir kvadrata tačaka u klasteru,
  - LS – vektor koji sadrži linearni zbir tačaka u klasteru,
  - m – broj tačaka u klasteru.
- Veličina sažetka iznosi $(2 \cdot d + 1)$ za skup podataka dimenzionalnosti d, i često se naziva **CF-vektor**.

BIRCH – osobine klasterske karakteristike

Klasterska karakteristika ima dve osobine:

1. Svaka klasterska karakteristika može se predstaviti kao zbir klasterskih karakteristika pojedinačnih tačaka podataka.
   - Klasterska karakteristika roditeljskog čvora u CF-stablu jednaka je zbiru klasterskih karakteristika njegove dece.
   - Ažuriranje CF-vektora može se lako izvršiti dodavanjem CF-vektora nove instance CF-vektoru klastera.
2. Klasterske karakteristike mogu se koristiti za izračunavanje korisnih svojstava klastera – varijanse i centroida.

BIRCH – centroid i varijansa

- Centroid klastera je:

$$\frac{LS}{m}$$

- Varijanse duž i-te dimenzije mogu se izraziti kao:

$$\sigma_i^2 = \frac{SS_i}{m} - \left(\frac{LS_i}{m}\right)^2$$

- SS_i i LS_i predstavljaju odgovarajuće komponente vektora za i-tu dimenziju.
- Zbir ovih dimenziono-specifičnih varijansi daje varijansu celog klastera.
- Rastojanje bilo koje tačke do centroida može se izračunati korišćenjem klasterske karakteristike, upotrebom već izračunatog centroida LS/m.
- CF-vektor sadrži sve informacije potrebne za ubacivanje tačke podataka u CF-stablo.

BIRCH – prag varijanse

- Svaki list u CF-stablu ima prag varijanse T.
- Vrednost T reguliše granularnost klasterovanja, visinu stabla i ukupan broj klastera u listovima.
- Manje vrednosti T dovode do većeg broja finijih klastera.
- Pošto se pretpostavlja da se CF-stablo nalazi u glavnoj memoriji, veličina skupa podataka ima presudan uticaj na izbor vrednosti T.

BIRCH – prednosti i nedostaci

- Algoritam BIRCH je veoma brz.
- Osnovni pristup (bez faze poboljšanja) zahteva jedan prolaz kroz podatke.
- Svako dodavanje predstavlja efikasnu operaciju.
- Nije projektovan za proizvoljne tipove podataka.
- Nije pogodan za klastere proizvoljnog oblika.

CURE – Clustering Using REpresentatives

- Hijerarhijski algoritam koji koristi najbližu vezu
- Svaki klaster se predstavlja određenim brojem fiksnih tačaka koje su generisane izdvajanjem dobro rasutih tačaka iz klastera, skupljajući ih prema centru klastera za određenu veličinu skaliranja.
- Postojanje više od jedne reprezentativne tačke omogućava CURE algoritmu dobro prilagođavanje skupovima objekata nesferičnog oblika, a skupljanje pomaže da se priguše efekti postojanja elemenata van granica.
- CURE koristi kombinaciju **slučajnog uzorkovanja i particionisanja za proširenje na velike baze podataka**.
- Izvučen slučajni uzorak iz skupa podataka se particioniše u prvom koraku, pri čemu je svaka particija delimično klasterovana.
- U narednom koraku se delimični klasteri grupišu u finalne klastere.

CURE

- CURE može da rukuje elementima van granica tako što periodično eliminiše male klastere tokom procesa spajanja. Ideja je da takvi klasteri ostaju mali zato što uglavnom sadrže elemente van granica.
- Algoritam CURE može da otkriva klastere proizvoljnog oblika

ROCK

- **ROCK** (*RObust Clustering using linKs*) zasniva se na aglomerativnom pristupu *odozdo nagore*
- Klasterovanje kategoričkih podataka
- Klasteri se spajaju na osnovu kriterijuma sličnosti
- Koristi kriterijum koji je zasnovan na meri zajedničkih najbližih suseda

- Pošto su aglomerativne metode relativno skupe u pogledu računanja, ROCK primenjuje ovaj pristup samo na uzorak tačaka podataka kako bi otkrio prototipske klastere.
- Preostale tačke podataka se u završnom prolazu dodeljuju jednom od ovih prototipskih klastera.

ROCK

- Prvi korak ROCK algoritma je transformacija kategoričkih podataka u binarnu reprezentaciju korišćenjem pristupa binarizacije.
- Svaki objekat može se tretirati kao binarna transakcija, odnosno kao skup stavki.
- Sličnost između dve transakcije računa se korišćenjem Žakardovog koeficijenta između odgovarajućih skupova:

$$Sim(T_i, T_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}$$

- Dve instance T_i i T_j definišu se kao susedi ako je sličnost:

$$Sim(T_i, T_j) \geq \theta$$

Hijerarhijski algoritam ROCK

- $link(T_i, T_j)$ označava funkciju sličnosti zasnovanu na zajedničkim najbližim susedima, koja je jednaka broju zajedničkih najbližih suseda između T_i i T_j.
- Funkcija sličnosti zasnovana na vezama obezbeđuje kriterijum spajanja za aglomerativne algoritme.
- Algoritam započinje tako što se svaka tačka podataka (iz inicijalno izabranog uzorka) nalazi u sopstvenom klasteru.
- Klasteri se zatim hijerarhijski spajaju na osnovu kriterijuma sličnosti između klastera.

Sličnost između klastera u ROCK algoritmu

Sličnost između klastera računa se kao:

$$GroupLink(C_i, C_j) = \sum_{T_u \in C_i, T_v \in C_j} Link(T_u, T_v)$$

Završne faze ROCK algoritma

- Spajanja se izvode dok u podacima ne ostane ukupno k klastera.
- Preostale tačke podataka dodeljuju se jednom od klastera.
- Svaka tačka podataka dodeljuje se klasteru sa kojim ima najveću sličnost.

Algoritmi zasnovani na podeli

Varijante K-sredina

Varijante K-sredina se razlikuju po:

- načinu izbora različitih inicijalnih reprezentativnih tačaka (K-medoid, K-medijana, K-modalna vrednost)
- izboru bolje procene inicijalnih centroida (inteligentne K-sredine, genetske K-sredine)
- primeni nekog oblika transformacije (težinske K-sredine, kernel K-sredine).

K-medoid

- Otporniji na elemente van granica i šum u odnosu na K-sredina algoritam.
- Ukupna greška – zbir svih različitosti između svakog objekta p i njegovog reprezentativnog objekta.

```
/* K-medoid algoritam */

Bira se K reprezentativnih tačaka za početne medoide
repeat
  Dodeli svaku tačku klasteru sa najbližim reprezentativnim objektom
  Izabrati slučajnu nereprezentativnu tačku x
  Odrediti promenu ukupne greške S pri zameni  reprezentativnog objekta m sa x
  Ako je S < 0 zameniti m sa x i formirati  novi skup reprezentativnih objekata

until dok se ne dostigne kriterijum konvergencije
```

K-medijana

- Bira se K reprezentativnih tačaka za početne predstavnike.
- Svaka tačka se pridružuje najbližem (u odnosu na korišćenu meru) predstavniku.
- Kada se klaster formira, ažuriraju se vrednosti predstavnika.
- Kao predstavnik se bira medijana – cilj je minimizacija zbira rastojanja između svake tačke i njoj najbližeg centra.
- Kao mera se koristi L_1 norma (Menhetn rastojanje).
- Iterativno se ponavljaju prethodna dva koraka sve dok ima promena ili se zadovolji kriterijum konvergencije.

K-modalno klasterovanje

- Koristi se za klasterovanje podataka sa kategoričkim atributima.
- Svaka vrednost atributa za predstavnika bira se kao mod kategoričkih vrednosti tog atributa u klasteru
- Mod skupa kategoričkih vrednosti je vrednost koja ima najveću učestalost u skupu
- Mod je kategorička vrednost v_j za svaki atribut i za koju histogram učestalosti ima najveću vrednost p_{ij}
- Na primer, funkcija sličnosti zasnovana na **inverznoj učestalosti pojavljivanja** (*inverse occurrence frequency*):

$$sim(x, y) = \sum_{k=1}^{d} \frac{\delta(x_k, y_k)}{f_k(x_k)^2}$$

gde je:

- $\delta(x_k, y_k)$ indikator poklapanja vrednosti atributa,
- $f_k(x)$ udeo instanci u kojima k-ti atribut u skupu podataka ima vrednost x.

Inteligentne K-sredine

- Osnovna ideja algoritma zasniva se na pretpostavci da su interesantne tačke obično udaljene od centra gravitacije skupa podataka (srednja vrednost svih tačaka.).
- Za razliku od klasičnog K-sredina algoritma, broj klastera ne mora unapred biti precizno određen.
- Algoritam iterativno pronalazi tačke koje su najudaljenije od centra gravitacije i formira potencijalne klastere oko njih.
- Mali klasteri se mogu odbaciti korišćenjem unapred definisanog praga.
- Dobijeni centroidi koriste se kao kvalitetnija inicijalizacija za standardni K-sredina algoritam.
- Cilj algoritma je smanjenje zavisnosti od slučajnog izbora početnih centroida i izbegavanje loših lokalnih minimuma.

Inteligentne K-sredine – pseudokod

```
/* Inteligentne K-sredine */

Izračunati centar gravitacije c_g svih tačaka skupa podataka

repeat

    Odrediti centroid c koji je  najudaljeniji od c_g;
    Formirati klaster S_iter od tačaka koje su bliže c nego c_g;
    Izračunati novi centroid s_g  klastera S_iter;
    Sačuvati centroid s_g;
    Ukloniti tačke klastera S_iter iz daljeg razmatranja;
    Ponovo izračunati centar gravitacije preostalih tačaka;

    Odbaciti male klastere  korišćenjem zadatog praga

until dok se ne dostigne kriterijum konvergencije
```

Genetsko K-sredine klasterovanje

- Za izbegavanje konvergencije ka lokalnom minimumu koristi se genetski algoritam.
- Razvijen je Genetski K-sredina algoritam koji predstavlja kombinaciju algoritma K-sredina i genetskih algoritama.

Genetski algoritam

- Genetski algoritam predstavlja optimizacioni pristup inspirisan principima biološke evolucije i prirodne selekcije.
- Skup mogućih rešenja naziva se **populacija**, dok se pojedinačno rešenje naziva **hromozom**.
- Kvalitet svakog rešenja određuje se pomoću **funkcije prilagođenosti**.
- Algoritam iterativno poboljšava populaciju korišćenjem:
  - selekcije
  - ukrštanja
  - mutacije
- Selekcija favorizuje kvalitetnija rešenja.
- Ukrštanje kombinuje delove dva roditeljska rešenja radi generisanja novih rešenja.
- Mutacija uvodi slučajne izmene radi očuvanja raznovrsnosti populacije i izbegavanja lokalnih minimuma.
- Proces se ponavlja kroz više generacija dok se ne dostigne kriterijum zaustavljanja.

Genetski algoritam – pseudokod

```
Generisati početnu populaciju

repeat

    Izračunati funkciju prilagođenosti za svako rešenje

    Izabrati najbolja rešenja (selekcija)
    Formirati nova rešenja ukrštanjem
    Izvršiti mutaciju pojedinih rešenja

    Formirati novu populaciju

until dok se ne dostigne kriterijum zaustavljanja

Vratiti najbolje rešenje
```

Genetsko K-sredine klasterovanje

- Svako potencijalno rešenje predstavlja se kao hromozom koji kodira skup centroida klastera.
- Populacija različitih rešenja evoluira kroz više generacija korišćenjem operatora genetskih algoritama:
  - selekcije,
  - ukrštanja,
  - mutacije.
- Kvalitet svakog rešenja ocenjuje se funkcijom prilagođenosti, koja najčešće predstavlja ukupnu sumu rastojanja tačaka do centroida.
- Najbolja rešenja imaju veću verovatnoću da budu izabrana za generisanje novih rešenja.
- Nakon završetka evolutivnog procesa, najbolje pronađeno rešenje koristi se kao inicijalizacija ili konačno rešenje klasterovanja.
- Prednost pristupa je bolja globalna optimizacija i manja zavisnost od slučajnog izbora početnih centroida.

Kernel K-sredine klasterovanje

- Kernel K-sredine (*Kernel K-Means*) predstavlja proširenje klasičnog K-sredina algoritma.
- Osnovna ideja algoritma je preslikavanje podataka iz originalnog prostora u višedimenzioni kernel prostor korišćenjem kernel funkcije.
- Nakon transformacije, klasterovanje se izvršava u novom prostoru primenom K-sredina algoritma.
- Na taj način moguće je razdvojiti klastere koji nisu linearno razdvojivi u originalnom prostoru podataka.
- Umesto direktnog računanja koordinata u višedimenzionom prostoru koristi se **kernel trik** (*kernel trick*).
- Kernel funkcija omogućava izračunavanje sličnosti između objekata bez eksplicitnog preslikavanja u novi prostor.
- Često korišćene kernel funkcije su:
  - polinomijalni kernel,
  - Gausov (RBF) kernel,
  - sigmoidni kernel.

Težinsko K-sredine klasterovanje

- Osnovna ideja algoritma je da različiti atributi nemaju isti značaj pri određivanju sličnosti između objekata.
- Svakom atributu dodeljuje se težina koja određuje njegov uticaj na postupak klasterovanja.
- Kao mera rastojanja koristi se modifikovana varijanta Euklidskog rastojanja sa težinama:

$$d(x, y) = \sqrt{\sum_{i=1}^{d} w_i (x_i - y_i)^2}$$

gde je:

- w_i težina i-tog atributa,
- x_i i y_i vrednosti i-te dimenzije objekata x i y.

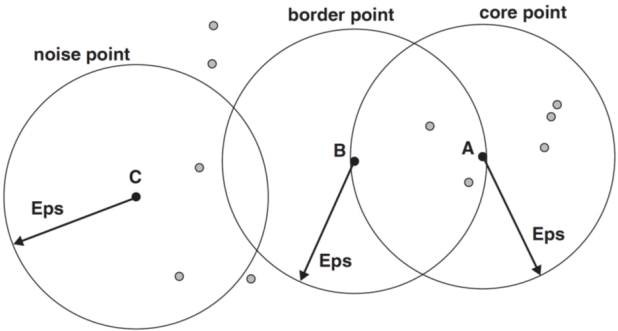
- Veće težine povećavaju uticaj odgovarajućih atributa na formiranje klastera.
- Algoritam je posebno koristan kod visokodimenzionih podataka i skupova sa irelevantnim atributima.

Algoritmi zasnovani na gustini

DBSCAN

Za zadatu vrednost ε i broj t:

- Tačka pripada *jezgru* ako se u krugu poluprečnika ε nalazi bar t drugih tačaka.
- Tačka je *na granici* ako se u krugu poluprečnika ε nalazi manje od t drugih tačaka, ali se nalazi bar jedna tačka jezgra.
- Tačka je *šum* ako nije niti u jezgru niti na granici.



DBSCAN – pseudokod

```
DBSCAN(Podaci: D, Poluprečnik: Eps, Gustina: tau)

begin

    Odredi tačke jezgra, granične i šum tačke skupa D za (Eps, tau);

    Napravi graf u kome su tačke jezgra povezane ako se nalaze na rastojanju
    manjem ili jednakom od Eps;

    Odredi povezane komponente u grafu;

    Dodeli svaku graničnu tačku onoj povezanoj komponenti sa kojom ima
    najjaču povezanost;

    Vrati tačke u svakoj povezanoj komponenti kao klaster;
end
```

DBSCAN

- DBSCAN može da otkriva klastere proizvoljnog oblika i ne zahteva zadavanje broja klastera kao ulaznog parametra.
- DBSCAN je osetljiv na varijacije u lokalnoj gustini klastera.

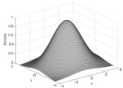
DENCLUE

DENCLUE (*DENsiti CLUstEring*) je pristup klasterovanju zasnovan na gustini koji modeluje ukupnu gustinu skupa tačaka kao zbir funkcija uticaja pridruženih svakoj tački.

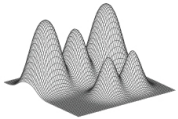
- Rezultujuća funkcija ukupne gustine imaće lokalne maksimume gustine koji se koriste za direktno definisanje klastera.
- Za svaku tačku se nalazi najbliži maksimum funkcije, a skup svih tačaka povezanih sa tako određenim maksimumom postaje klaster.
- Ako je gustina na lokalnom maksimumu manja od praga minimalne gustine, tačka je šum i odbacuje se.
- Ako se lokalni maksimum može povezati sa drugim lokalnim maksimumom preko tačaka podataka, a gustina u svakoj tački putanje je iznad praga minimalne gustine, tada se klasteri koji su dodeljeni ovim lokalnim maksimumima spajaju.

DENCLUE – ilustracija

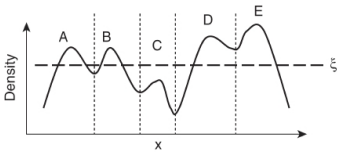
$$K(y) = e^{-distance(\mathbf{x},\mathbf{y})^2/2\sigma^2}$$



Set of 12 points.



Overall density—surface plot.



DENCLUE – pseudokod

```
begin

    Izvesti funkciju gustine za prostor koji zauzimaju tačke podataka;
    Identifikovati tačke koje su lokalni maksimumi gustine;

    Pridružiti svaku tačku podataka odgovarajućem lokalnom maksimumu;

    Definisati klastere koji se sastoje od tačaka pridruženih istom lokalnom
    maksimumu;

    Odbaciti klastere čiji lokalni  maksimum ima gustinu manju od korisnički
    definisanog praga eps;

    Spojiti klastere koji su povezani putanjom tačaka čija je gustina jednaka
    eps ili veća;

end
```

BRIDGE algoritam

BRIDGE (most) predstavlja hibridni algoritam koji kombinuje:

- K-sredina algoritam,
- pristup zasnovan na gustini (DBSCAN).

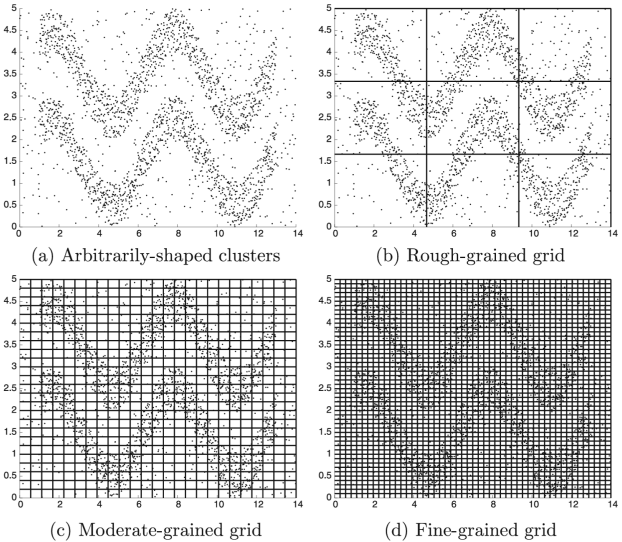
- Cilj algoritma je kombinovanje brzine K-sredina algoritma i otpornosti DBSCAN algoritma na šum i elemente van granica.
- U prvom koraku primenjuje se K-sredina algoritam radi dobijanja početnih klastera.
- Zatim se nad svakim klasterom posebno primenjuje DBSCAN algoritam.
- Parametri DBSCAN algoritma nasleđuju se iz rezultata K-sredina algoritma.
- DBSCAN omogućava:
  - identifikaciju gustih regiona – tačaka koje nisu šum,
  - identifikaciju šuma
- Nakon uklanjanja šuma ponovo se izvršava K-sredina algoritam nad preostalim tačkama.
- BRIDGE poboljšava kvalitet klasterovanja u odnosu na pojedinačnu primenu K-sredina algoritma.

Algoritmi zasnovani na mreži

Algoritmi zasnovani na mreži

- Podaci se diskretizuju na p intervala koji su najčešće jednake širine.
- Za skup podataka dimenzionalnosti d, prostor se deli na p^d hiperkocki u prostoru podataka.
- Hiperkocke predstavljaju gradivne blokove na kojima se zasniva algoritam.
- Svaka hiperkocka može sadržati:
  - veliki broj tačaka,
  - mali broj tačaka,
  - ili nijednu tačku.
- Prag gustine τ koristi se za određivanje podskupa hiperkocki koje se smatraju gustim.
- Klasteri se formiraju povezivanjem susednih gustih hiperkocki.
- Algoritam GRIDCLUS

Algoritmi zasnovani na mreži



(a) Arbitrarily-shaped clusters

(b) Rough-grained grid

(c) Moderate-grained grid

(d) Fine-grained grid

Generički algoritam zasnovan na mreži

Algoritam GenerickaMreza(*Podaci*: D, *Opsezi*: p, *Gustina*: τ)

```
begin
    Diskretizuj svaku dimenziju podataka D na p opsega;
    Odredi guste ćelije mreže na nivou gustine  (tau);
    Kreiraj graf u kome su guste ćelije povezane ako su susedne;
    Odredi povezane komponente grafa;
return tačke u svakoj povezanoj komponenti kao klaster;
end
```

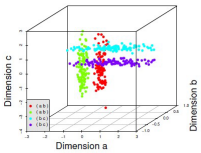
Parametri algoritama zasnovanih na mreži

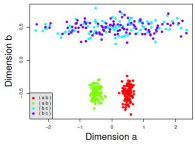
- Cilj algoritama jeste pronalaženje prirodnih klastera zajedno sa njihovim oblikom.
- Broj klastera se ne definiše unapred.
- Potrebno je definisati dva ključna parametra:
  1. broj intervala mreže p,
  2. prag gustine τ.

Prednosti i mane

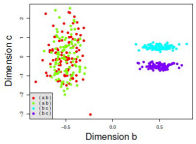
- Metode zasnovane na mreži:
  - ne zahtevaju unapred zadat broj klastera,
  - ne pretpostavljaju specifičan oblik klastera.
- Nedostaci:
  - potrebno je definisati prag gustine τ,
  - izbor rezolucije mreže nije trivijalan.
- Mnogi algoritmi koriste jedan globalni prag gustine: τ
- U realnim podacima različiti klasteri mogu imati različite gustine.

Klasterovanje po podprostorima

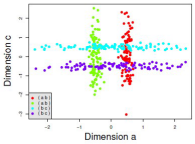




(a) Dims a & b



(b) Dims b & c



(c) Dims a & c

CLIQUE

- CLustering In QUEst (CLIQUE)
- Ulazni parametri metode su:
  - broj mrežnih intervala p za svaku dimenziju,
  - gustina τ.
- Gustina τ predstavlja minimalan broj tačaka podataka u gustoj mrežnoj ćeliji.
- Parametar τ može se posmatrati i kao minimalni zahtev podrške za mrežnu ćeliju.

CLIQUE - diskretizacija i potprostori

- U prvoj fazi koristi se diskretizacija kako bi se formirala mrežna struktura.
- Glavna razlika između algoritma CLIQUE i metoda zasnovanih na mrežama u punoj dimenzionalnosti:
  - intervali se određuju **samo nad relevantnim podskupom dimenzija**,
  - gustina u tim potprostorima mora biti veća od τ.

CLIQUE

- Problem određivanja gustih potprostora ekvivalentan je problemu otkrivanja čestih obrazaca.
- Svaki diskretizovani interval tretira se kao stavka (item), a prag podrške postavlja se na τ
- U originalnom CLIQUE algoritmu koristi se Apriori metod, a može se koristiti bilo koji metod za otkrivanje čestih obrazaca.

Grupisanje mrežnih ćelija

- Kao i kod opštih metoda zasnovanih na mrežama susedne mrežne ćelije grupišu se zajedno.
- Razlika kod algoritma CLIQUE: **dve mrežne ćelije mogu biti susedne samo ako su definisane nad istim potprostorom**.
- Svi pronađeni obrasci vraćaju se zajedno sa tačkama podataka koje im pripadaju.

PROCLUS

- PROCLUS (*PROjected CLUStering*)
- Algoritam za projektovano klasterovanje visokodimenzionih podataka zasnovan na medoidima.
- Svaki klaster ima sopstveni podprostor dimenzija koje su za njega relevantne.
- Skup dimenzija klastera se određuje na osnovu statističke raspodele instanci u lokalnoj okolini medoida
- Robustan je na:
  - šum,
  - irelevantne dimenzije.

Faze algoritma PROCLUS

1. **Inicijalizacija**
   - bira se mali skup kandidata medoida M.
2. **Iterativna optimizacija**
   - postupno poboljšavanje rešenja korišćenjem klasterovanja zasnovanog na medoidima.
3. **Unapređenje klastera**
   - instance se dodeljuju optimalnim medoidima,
   - uklanjaju se elementi van granica.

Algoritam PROCLUS

Algoritam PROCLUS(*Baza podataka*: D, *Broj klastera*: k, *Broj dimenzija*: l)

```
begin
    Izaberi kandidatski skup medoida M iz D primenom pristupa najudaljenije tačke;
    S = nasumično izabran podskup skupa M veličine k;
    NajboljaVredCiljneFunkcije = beskonačno;

    repeat
        Izračunaj dimenzije (podprostor) pridružene svakom medoidu u skupu S;
        Dodeli instance iz skupa D najbližim medoidima iz skupa S koristeći projektovanu udaljenost;
        TrenVredCiljneFunkcije =
            prosečna projektovana udaljenost instanci do centara njihovih klastera;
        if (TrenVredCiljneFunkcije < NajboljaVredCiljneFunkcije)
            Sbest = S;
            NajboljaVredtCiljneFunkcije = TrenVredCiljneFunkcije;
        end;
         Ponovo izračunaj skup S zamenom loših medoida u skupu Sbest
         nasumično izabranim tačkama iz skupa M;
    until nije ispunjen kriterijum zaustavljanja;

    Dodeli instance skupa medoidima iz skupa Sbest  koristeći unapređene podprostore;
    Vrati sve parove (klaster, podprostor);
end
```

PROCLUS – određivanje projektovanih dimenzija za medoid

- Za određivanje kvaliteta skupa medoida potrebno je dodeliti instance skupa odgovarajućim medoidima.
- Lokalnost medoida predstavlja skup instanci koje se nalaze unutar sfere čiji je poluprečnik jednak rastojanju do najbližeg drugog medoida.
- Rastojanje instance do svakog medoida računa se u podprostoru relevantnom za i-ti medoid.

PROCLUS – određivanje projektovanih dimenzija za medoid

- Neka je r_{ij} prosečno rastojanje instanci u lokalnosti medoida i do tog medoida duž dimenzije j.
- Za svaku lokalnost računaju se:
  - srednja vrednost $\mu_i = \sum_{j=1}^{d} \frac{r_{ij}}{d}$
  - standardna devijacija $\sigma_i = \sqrt{\frac{\sum_{j=1}^{d}(r_{ij}-\mu_i)^2}{d-1}}$

PROCLUS – određivanje projektovanih dimenzija za medoid

- Računa se statistički normalizovana vrednost z_{ij}

$$z_{ij} = \frac{r_{ij} - \mu_i}{\sigma_i}$$

- Negativne vrednosti z_{ij} posebno su poželjne
  - ukazuju na manja prosečna rastojanja od očekivanih za dati par medoid–dimenzija.

PROCLUS – izbor relevantnih dimenzija

- Osnovna ideja jeste izbor $k \cdot l$ najmanjih (najnegativnijih) vrednosti z_{ij} radi određivanja bitnih dimenzija specifičnih za klastere.
- Različiti klasteri mogu imati različit broj dimenzija.
- Ukupan broj dimenzija pridruženih svim medoidima mora biti jednak $k \cdot l$
- Broj dimenzija pridruženih jednom medoidu mora biti najmanje 2.
- Za svaki medoid:
  - vrednosti z_{ij} sortiraju se rastuće,
  - biraju se dve najmanje vrednosti,
  - preostalih $k \cdot (l - 2)$ parova medoid–dimenzija bira se pohlepno iz preostalih vrednosti.

PROCLUS – dodela instanci klasterima

- Na osnovu medoida i njima pridruženih skupova dimenzija instance se dodeljuju medoidima jednim prolazom kroz bazu podataka.
- Rastojanje između instanci i medoida računa se korišćenjem *segmentalne Menhetenove udaljenosti*
- Segmentalna Menhetenova udaljenost predstavlja Menhetenovu udaljenost normalizovanu u odnosu na promenljiv broj dimenzija pridruženih svakom medoidu.
- Rastojanje se računa:
  - samo nad relevantnim dimenzijama,
  - a zatim deli brojem relevantnih dimenzija.

Algoritmi zasnovani na modelu

Algoritmi zasnovani na modelu

- Algoritmi ove grupe pokušavaju da optimizuju poklapanje između posmatranih podataka i nekog matematičkog modela izračunavanjem verovatnoće.
- Ove metode zasnivaju se na pretpostavci da podaci zadovoljavaju mešavinu osnovnih distribucija verovatnoće.
- U praksi, svaki klaster može se matematički predstaviti parametarskom distribucijom verovatnoće:
  - Gausova raspodela,
  - Poasonova raspodela,
  - i druge distribucije.

Algoritmi zasnovani na modelu

- Problem klasterovanja prevodi se u problem procene parametara modela.
- Cilj je da se grupišu tačke koje najverovatnije imaju istu distribuciju

EM algoritam

- EM (*Expectation Maximization*
- U opštem slučaju nije poznato koje tačke su generisane kojom raspodelom.
- Zbog toga se izračunava verovatnoća da svaka tačka pripada svakoj distribuciji.
- Dobijene verovatnoće koriste se za računanje nove procene parametara:
  - parametara koji maksimalizuju verovatnoću podataka.
- Iterativni postupak se nastavlja:
  - dok se parametri više ne menjaju,
  - ili dok promene ne postanu veoma male.

EM i raspodele verovatnoće

- EM se najčešće primenjuje kada je raspodela:
  - Gausova (normalna),
  - ili Poasonova.
- EM algoritam konvergira ka *lokalnom maksimumu* funkcije verovatnoće.
- Krajnje rešenje značajno zavisi od početnog izbora parametara.
- Zbog toga EM može konvergirati ka suboptimalnom maksimumu verovatnoće

EM algoritam

```
Izdvojiti inicijalni skup parametara modela
/* Način izbora sličan kao kod K-sredina (slučajno ili na neki drugi način) */

repeat

    Korak očekivanja:
        za svaki objekat izračunati verovatnoću da objekat pripada svakoj distribuciji;

    Korak maksimizacije:
        s obzirom na dobijene verovatnoće u koraku očekivanja odrediti novu
        procenu parametara koji maksimalizuju  očekivanu verovatnoću

until parametri se ne menjaju

/* Alternativno, algoritam se zaustavlja  ako je promena parametara manja od  zadatog praga */
```