Klasteriranje podataka i konceptualno klasteriranje

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

17. svibnja, 2026.

Klasteriranje bazirano na rešetci

- Algoritmi klasteriranja bazirani na rešetci **particioniraju** prostor podataka u konačan broj **regija** koje tvore **strukturu rešetke**, a potom stvaraju klastere iz tih regija u strukturi rešetke.
- Klasteri odgovaraju regijama koje su gušće od njihovog okruženja.
- Algoritmi klasteriranja bazirani na rešetci postižu značajne uštede u vremenskoj složenosti pošto ne klasteriraju direktno točke iz skupa podataka, već klasteriraju susjedstva oko točaka reprezentiranih regijama (regija je puno manje od točaka).
- Algoritmi bazirani na rešetci uglavnom uključuju glavnih pet koraka:
  - Kreiranje strukture rešetke, tj. particioniranje prostora skupa podataka u konačan broj regija.
  - Računanje gustoće svake regije.
  - Sortiranje regija prema njihovoj gustoći.
  - Identificiranje centara klastera.
  - Obilazak susjednih regija.

Klasteriranje bazirano na rešetci

- Pošto često moramo računati gustoće regija da bi ih mogli sortirati, algoritme bazirane na rešetci možemo smatrati i baziranima na gustoći.
- Neki algoritmi bazirani na rešetci koombiniraju hijerarhijsko klasteriranje ili potprostorno klasteriranje da bi organizirali regije prema gustoći.
- Najveći izazovi klasteriranja baziranog na rešetci jesu:
  - **Ne-uniformnost** - korištenje jedne nefleksibilne, uniformne rešetke potencijalno nije dovoljno za postizanje kvalitetnog klasteriranja za skupove podataka s iznimno nepravilnim distribucijama.
  - **Lokalnost** - ukoliko postoje varijacije u obliku i gustoći distribucije točaka u podacima, učinkovitost metoda baziranih na rešetci je ograničena predefiniranim veličinama regija, njihovim granicama i pragovima gustoće značajnih regija.
  - **Dimenzionalnost** - performanse ovise o veličini strukture rešetke, koja se može znatno povećati uz povećanje dimenzionalnosti. Dodatni problemi kod visokodimenzionalnih skupova podataka uključuju filtriranje šuma i selektiranje relevantnih atributa.

Klasteriranje bazirano na rešetci

- Ne-uniformnost podataka se uglavnom adresira algoritmima klasteriranja koji rade adaptivno klasteriranje bazirano na rešetci. Ti algoritmi dijele prostor značajki u više rezolucija. Na taj način učinkovito klasteriraju podatke s neuniformnim distribucijama.
- Lokalnost se rješava uvođenjem algoritama koji mogu pomicati osi. Particioniranjem prostora koristeći strategiju pomaka osi, ti algoritmi pronalaze područja visoke gustoće u prostoru značajki.
- Visokodimenzionalni podaci se klasteriraju ili selekcijom odgovarajućih podprostora za pronalazak gustih regija ili korištenjem procjene gustoće.

GRIDCLUS algoritam

- GRIDCLUS je prvi hijerarhijski algoritam klasteriranja baziran na rešetci.
- Algoritam particionira prostor podataka u strukturu rešetke koja se sastoji od disjunktnih d-dimenzionalnih hiper pravokutnika, odnosno blokova.
- Podatke reprezentiramo kao točke u d-dimenzionalnom prostoru i distribuiramo blokovima rešetke tako da održimo njihovu topološku distribuciju.
- Nakon što se podaci dodijele blokovima, klasteriranje vršimo algoritmima koji pretražuju susjedstvo.
- GRIDCLUS raspoređuje točke u blokove svoje rešetkaste strukture, računa gustoće blokova, sortira blokove prema njihovoj gustoći, prepoznaje najgušće blokove kao centre klastera i konstruira ostatak klastera koristeći pretragu susjeda u bloku.
- Struktura rešetke ima mjerilo za svaku dimenziju, direktorij rešetke i skup podatkovnih blokova.
- Mjere se koriste za particioniranje d-dimenzionalnog prostora, koje se sprema u direktorij rešetke.

GRIDCLUS algoritam

- Podatkovni blok sadrži entitete (točke). Postoji gornja ograda na broj točka u bloku.
- Blok mora biti neprazan, pokrivati sve podatkovne točke, a dva bloka ne smiju imati presjek.
- Indeks gustoće se definira kao broj točaka u bloku podijeljen sa prostornim volumenom bloka, $G_B = \frac{|B|}{V_B}$.
- GRIDCLUS sortira blokove prema njihovoj gustoći, a blokovi s najvećom gustoćom postaju centri klastera.
- Blokovi se dalje klasteriraju u silaznom poretku prema gustoći iterativno, time se stvara ugnježđeni niz nepraznih, disjunktnih klastera.
- Konstrukcija klastera kreće iz centra, provjeravaju se samo njegovi susjedni blokovi i spajaju u klaster.
- Pretraga susjeda se odvija rekurzivno. Tražimo susjede samo onih susjeda koji su spojeni u klaster.

Algoritam GRIDCLUS

$u \leftarrow 0, W[] \leftarrow \{\}, C[][] \leftarrow \{\}$ {inicijalizacija};
Kreiraj strukturu rešetke i izračunaj indekse gustoća blokova;
Generiraj sortirani niz blokova $B_{1'}, B_{2'}, \ldots, B_{b'}$ i označi sve blokove neaktivnim i ne klasteriranim.
while postoji ne aktivan blok **do**
  $u \leftarrow u + 1$;
  označi prvi $B_{1'}, B_{2'}, \ldots, B_{j'}$ s jednakim indeksom gustoće aktivnim;
  for svaki ne klasterirani blok $B_{l'} := B_{1'}, B_{2'}, \ldots, B_{j'}$ **do**
    Stvori novi klaster $C[u]$;
    $W[u] \leftarrow W[u] + 1, C[u, W[u]] \leftarrow B_{l'}$;
    Označi blok $B_{l'}$ klasteriranim;
    $PRETRAGA\_SUSJEDA(B_{l'}, C[u, W[u]])$;
  end for
  for svaki ne aktivni blok B **do**
    $W[u] \leftarrow W[u] + 1, C[u, W[u]] \leftarrow B$;
  end for
  Označi sve blokove ne klasteriranim;
end while

GRIDCLUS algoritam

Algoritam Procedure PRETRAGA_SUSJEDA(B,C)

1: **for** svaki aktivan i ne klasterirani susjed B' od B **do**
2: $C \leftarrow B'$;
3: Označi blok B' klasteriranim;
4: $PRETRAGA\_SUSJEDA(B', C)$;
5: **end for**

STING algoritam

- STING (eng. Statistical Information Grid-based clustering) je metoda klasteriranja bazirana na rešetci koja klasterira prostorne baze podataka i olakšava upite vezane uz regije.
- STING dijeli prostor u pravokutne regije i sprema ih u hijerarhijsku strukturu stabla koje sadrži rešetke.
- Svaka regija (osim listova u stablu) se particionira u 4 regije djece, gdje svako dijete odgovara kvadrantu regije roditelja.
- Regija roditelja je unija regija djece, gdje korjen odgovara cijelom prostoru.
- Regije djece su uniformne veličine, koju određujemo globalno prema prosječnoj gustoći objekata.
- Za svaku regiju pamtimo parametre ovisno o atributima i parametre neovisne o atributima. U te parametre spadaju: broj točaka, srednja vrijednost, devijacija, minimum, maksimum i distribucija.
- Statistiku čvorova roditelja jednostavno računamo iz statistike čvorova djece.

STING algoritam

- Tipovi podržanih distribucija su normalna, uniformna, eksponencijalna ili `none`. Tip distribucije zadaje korisnik ili se određuje testovima, npr. χ^2.
- STING provodi top-down pristup za klasteriranje i upite, dok se statistički parametri računaju bottom-up.
- Algoritam je neovisan o upitima, jednostavan za paralelizaciju i omogućuje pretragu po više rezolucija (pošto gradi hijerarhiju regija rešetkaste strukture). Također, podržava inkrementalna ažuriranja, što omogućava brzu i učinkovitu ugradnju novih podataka.

STING algoritam

Algoritam STING

1: Odredi početnu razinu.
2: Za svaku regiju dane razine, računamo interval pouzdanosti (ili procijenjeni raspon) vjerojatnosti da je ta regija relevantna za upit.
3: Prema izračunatom intervalu označi regiju relevantnom ili irelevantnom.
4: Ukoliko je dana razina razina lista, idi na korak 6, inače idi na korak 5.
5: Spusti se u hijerarhiji za jednu razinu. Idi na korak 2 za regije koje tvore relevantne regije više razine.
6: Ukoliko je zadovoljena specifikacija upita, idi na korak 8, inače idi na korak 7.
7: Dohvati podatke koji pripadaju relevantnim regijama i vrši daljnje računanje. Vrati rezultat koji zadovoljava uvjete upita. Idi na korak 9.
8: Pronađi skupine relevantnih regija. Vrati skupine koje zadovoljavaju uvjete upita. Idi na korak 9.
9: Stop.

- Primjer STING upita: *Pronađi sve regije gdje je gustoća kuća veća od* 50 *po* m^2.

CLIQUE algoritam

- CLIQUE je hibridni algoritam klasteriranja baziran na gustoći i rešetki.
- Algoritam automatski pronalazi klasteriranje u potprostoru visokodimenzionalnih numeričkih podataka.
- Rezultati klasteriranja se predstavljaju u interpretabilnom formatu tako što se njegov minimalni opis prezentira u obliku disjunktivne normalne forme (DNF). Zbog te činjenice, algoritam spada i u kategoriju algoritama **konceptualnog klasteriranja**.
- CLIQUE prvo particionira numerički prostor u jedinice za svoju strukturu rešetke.
- Neka je $A = (A_1, A_2, \ldots, A_d)$ skup ograničenih, potpuno uređenih domena (atributa) i $S = A_1 \times A_2 \times \cdots \times A_d$ d-dimenzionalni numerički prostor.
- Particioniranjem svake dimenzije A_i $(1 \leq i \leq d)$ u m intervala jednake duljine, CLIQUE dijeli d-dimenzionalni prostor podataka u m^d ne preklapajućih pravokutnih jedinica.

CLIQUE algoritam

- d-dimenzionalna točka iz skupa podataka (entitet) v pripada jedinici u, ukoliko su vrijednosti svakog atributa entiteta v veće ili jednake lijevoj granici tog atributa u u i manje od desne granice tog atributa u u.
- Selektivnost jedinice se definira kao udio svih točaka skupa podataka koji pripadaju toj jedinici.
- Jedinice čija selektivnost je veća od parametra τ se smatraju gustima, te se stoga zadržavaju.
- Definicija gustih jedinica se primjenjuje na sve potprostore originalnog d-dimenzionalnog prostora.
- Da bi identificirao guste jedinice koje treba zadržati i potprostore koji sadrže klastere, CLIQUE istražuje projekcije potprostora u bottom-up poretku, prvo potprostore najmanje dimenzionalnosti, zatim sve veće.
- Za dani projekcijski potprostor $A_{t_1} \times A_{t_2} \times \cdots \times A_{t_p}$, gdje $p < d$ i $t_i < t_j$ ako $i < j$. Jedinica je presjek intervala svake dimenzije.

CLIQUE algoritam

- Koristeći Apriori algoritam, CLIQUE pretražuje u bottom-up poretku. Vrijedi sljedeće svojstvo monotonosti: ako je kolecija točaka klaster u p-dimenzionalnom prostoru, tada je ta kolekcija točaka također dio klastera u proizvoljnoj $p-1$-dimenzionalnoj projekciji toga prostora.
- CLIQUE algoritam radi rekurzivni korak izgradnje p-dimenzionalnih jedinica iz $p-1$-dimenzionalnih jedinica koristeći spajanje $p-1$-dimenzionalnih jedinica koje dijele prvih $p-2$ dimenzije.
- Da bi se reducirala vremenska složenst Apriori postupka, CLIQUE vrši rezanje skupa kandidata, zadržavajući samo guste jedinice. Iz gustih jedinica trenutne razine se stvaraju jedinice kandidati za sljedeću razinu generiranja gustih jedinica.
- Izrezivanje kandidata se provodi tako da se svi potprostori sortiraju prema pokrivenosti (postotku baze podataka koja je pokrivena gustim jedinicama koje sadržava).
- Potprostori koji imaju premalu pokrivenost se režu.

CLIQUE algoritam

- Od preostalih jedinica se stvaraju klasteri.
- Dvije p-dimenzionalne jedinice u_1 i u_2 su povezane ako dijele lice ili ako postoji neka druga p-dimenzionalna jedinica u_s takva da je u_1 povezan s u_s i u_2 je povezan s u_s.
- Klaster je maksimalan skup povezanih gustih jedinica u p dimenzija.
- Pronalaženje klastera je ekvivalentno pronalasku povezanih komponenti u grafu koji reprezentira guste jedinice gdje vrhovi i bridovi između vrhova postoje ako i samo ako jedinice dijele lice.
- Nakon pronalaska svih klastera, CLIQUE koristi DNF izraze da bi definirao konačan skup maksimalnih segmenata (regija) čija unija definira klaster.
- Pošto je pronalazak takvih opisa $\mathcal{NP}$-težak, CLIQUE koristi pohlepnu strategiju pokrivanja klastera u regijama nakon čega odbacuje redundantne regije.

CLIQUE algoritam

- Kombinacijom klasteriranja baziranog na gustoći, rešetci i klasteriranja potprostora, CLIQUE otkriva klastere uložene u potprostore visoko dimenzionalnih podataka bez potrebe korisnika za selekcijom potprostora od interesa.
- DNF izrazi pružaju interpretaciju rezultata klasteriranja.
- Vremenska složenost algoritma je $\mathcal{O}(c^p + pN)$, gdje je p najveći izabrani potprostor, c je konstanta, a N broj ulazni točaka.
- CLIQUE može propustiti neke klastere u potprostorima s manjom pokrivenosti. Ovisnost o šumu i sposobnost identifikacije relevantnih atributa je jako ovisna o korisničkom izboru intervala jedinica i definiciji praga osjetljivosti τ.

MAFIA

- MAFIA (eng. Merging of Adaptive Finite Intervals) je nasljednik algoritma CLIQUE.
- Umjesto korištenja strukture rešetke s regijama fiksne veličine i jednakim brojem pretinaca u svakoj dimenziji, MAFIA kreira adaptivne rešetke s ciljem poboljšavanja potprostornog klasteriranja.
- MAFIA uvodi adaptivnu rešetku pretinaca u svakoj dimenziji.
- Koristi Apriori algoritam za spajanje gustih intervala u klastere u višedimenzionalnim prostorima.
- Adaptivna rešetka se kreira particioniranjem svake dimenzije nezavisno bazirano na distribuciji (histogramima) svake dimenzije, te spajanjem intervala koji imaju identičnu distirbuciju. Intervali niske gustoće se režu.
- Vremenska složenost algoritma je $\mathcal{O}(m^p + pN)$, gdje je m konstanta, a p najveća dimenzionalnost guste jedinice kandidata.
- U praksi je znatno brži od algoritma CLIQUE.

MAFIA

- MAFIA točnije detektira granice klastera koje su vrlo blizu granicama rešetke, te ih opisuje DNF izrazima.

Algoritam MAFIA

Iteriraj po cijelom skupu podataka i konstruiraj adaptivnu rešetku u svakoj dimenziji.
Izračunaj histograme čitajući blokove podataka u memoriju koristeći pretince.
Koristeći histograme za spajanje pretinaca u manji broj adaptivnih pretinaca varijabilne veličine, spoji susjedne pretince sa sličnim vrijednostima histograma u veće pretince. Pretinci koji imaju manju gustoću podataka se režu.
Izaberi pretince koji su α-puta (α je parametar koji se zove faktor dominacije klastera) gušći od prosjeka kao p-jedinice kandidate (početno $p = 1$).
Iterativno traži podatke viših dimenzija, konstruiraj nove p-jedinice kandidate iz dvije $(p-1)$-jedinice kandidata ako dijele proizvoljnu $(p-2)$-stranu, i spoji susjedne jedinice kandidate u klastere.
Generiraj minimalni DNF izraz za svaki klaster.

Klasteriranje bazirano na ne-negativnoj matričnoj faktorizaciji

- Ne negativna matrična faktorizacija kao ulaz prima matricu ne-negativnih brojeva (ulazni skup koji sadrži atribute s ne-negativnim vrijednostima) i stvara dvije ne-negativne matrice nižeg ranga koje u umnošku aproksimiraju ulaznu matricu.
- Za nenegativan ulaz $X = (x_1, \ldots, x_n)$, gdje x_i reprezentira stupčani vektor dimenzije p, računamo $X \approx FG^T$, gdje $X \in \mathbb{R}^{p \times n}$, $F \in \mathbb{R}^{p \times k}$ i $G \in \mathbb{R}^{n \times k}$. Generalno je $p < n$ i rang matrica F i G je puno manji od ranga matrice X, dakle $k \ll \min(p, n)$.
- F i G dobivamo minimiziranjem funkcije cilja (najčešće sumu kvadrata pogrešaka). $\min_{F,G \geq 0} J_{SKP} = \|X - FG^T\|^2$.

Klasteriranje bazirano na ne-negativnoj matričnoj faktorizaciji

Teorem

G ortogonalan *NMF* je ekvivalentan *K*-means klasteringu.

- *G* ortogonalan *NMF* definiramo kao $\min_{F\geq 0, G\geq 0} \|X - FG^T\|^2, t.d. G^T G = I.$
- Pretpostavimo da su $C = \{c_1, c_2, \ldots, c_k\}$ centroidi klastera dobiveni *K*-means algoritmom klasteriranja.
- Označimo s *H* indikator klasteriranja, $h_{ki} = 1$ ako x_i pripada klasteru k, inače $h_{ki} = 0$.
- Funkciju cilja *K*-means klasteriranja možemo zapisati kao $J = \sum_{i=1}^{n}\sum_{k=1}^{K} h_{ik}\|x_i - c_k\|^2 = \|X - CH^T\|.$

Spektralno klasteriranje

- Algoritmi spektralnog klasteriranja provode tri glavna koraka:
  - Izgradnja grafa sličnosti između točaka sadržanih u skupu podataka.
  - Točke ulažemo u potprostor u kojem su klasteri lakše uočivi. Postupak se provodi računanjem svojstvenih vrijednosti Laplaciana grafa.
  - Klasičan algoritam sortiranja (npr. K-means) se koristi za particioniranje točaka u uloženom prostoru.
- Neka $X = \{x_1, \ldots, x_n\}$ reprezentira n točaka iz $\mathbb{R}^m$.
- Za potrebe spektralnog klasteriranja, prvo moramo te točke reprezentirati u obliku neusmjerenog grafa sličnosti $G = (V, E)$.
- Svaka točka x_i je reprezentirana vrhom v_i, a E označava bridove između vrhova.
- G možemo opisati ne-negativnom težinskom matricom susjedstva W dimenzija $n \times n$.
- Spektralno klasteriranje particionira vrhove tako da vrhovi koji pripadaju istim klasterima imaju visoku sličnost, vrhovi koji pripadaju različitim klasterima imaju nisku sličnost.

Spektralno klasteriranje

- Matricu W možemo konstruirati:
  - Grafovi K najbližih susjeda. v_i je povezan s v_j ukoliko je v_j među K najbližih susjeda od v_i ili je v_i unutar K najbližih susjeda od v_j. Udaljenost računamo koristeći originalnu reprezentaciju točaka u skupu podataka. Težine koje označavaju sličnost možemo izračunati iz sličnosti točaka, možemo dodati $0-1$ težine ili računati inverzno proporcionalne vrijednosti izračunatih udaljenosti.
  - Graf ε susjedstva. Kod takvih grafova, vrhovi su spojeni samo ako je međusobna udaljenost $\|x_i - x_j\|^2$ manja od ε. Ova metoda može dovesti do nepovezanih grafova ukoliko nepažljivo odaberemo vrijednost ε.
  - Potpuno povzani graf. Svi bridovi su povezani pozitivnim sličnostima. Kao mjeru sličnosti možemo npr. koristiti funkciju $W_{i,j} = e^{-\frac{\|x_i - x_j\|^2}{\sigma^2}}$. σ kontrolira širinu susjedstva.

Spektralno klasteriranje

- Uz dani graf sličnosti G, glavni korak spektralnog klasteriranja je računanje matrice Laplaciana grafa.
- Za svaki vrh grafa G, njegov stupanj se definira kao $d_i = \sum_{j=1}^{n} W_{ij}$.

  Matrica stupnjeva D se definira kao dijagonalna matrica $D_{i,i} = d_i$.
- Nenormalizirani Laplacian grafa se definira kao $L = D - W$. On zadovoljava sljedeća svojstva:
  - Za proizvoljan vektor $f \in \mathbb{R}^n$ imamo $f^T L f = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij}(f_i - f_j)^2$.
  - L je simetričan i pozitivno semidefinitan.
  - Najmanja svojstvena vrijednost od L je 0, uz svojstveni vektor $\mathbb{1}$.
  - L ima n ne-negativnih realnih svojstvenih vrijednosti $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.

Spektralno klasteriranje

- L možemo reprezentirati kao blok dijagonalnu matricu:

$$L = \begin{bmatrix} L_1 & & & \\ & L_2 & & \\ & & \ddots & \\ & & & L_k \end{bmatrix}$$

- Pošto je L blok dijagonalna matrica, njezine svojstvene vrijednosti i svojstveni vektori su unija svojstvenih vrijednosti svojstvenih vektora blokova $L_1, L_2, \ldots, L_k$. Stoga je multiplicitet svojstvene vrijednosti 0 najmanje k.
- Pretpostavimo da je $F \in \mathbb{R}^{n \times k}$ matrica koja sadrži k značajnih ortonormalnih vektora $f_1, f_2, \ldots, f_k$.
- Navedene vektore određujemo optimizirajući funkciju: $\min_F Tr(F^T L F)$, t.d. $F^T F = I$.

Spektralno klasteriranje

Algoritam Nenormalizirano spektralno klasteriranje

Konstruiraj graf sličnosti koristeći jednu od definiranih metoda. Izračunaj D i W.
Računaj nenormalizirani graf Laplacian L, gdje $L = D - W$.
Odredi $f_1, f_2, \ldots, f_k$, top k svojstvenih vektora od L.
Konstruiraj matricu $F \in \mathbb{R}^{n \times k}$ iz $f_1, f_2, \ldots, f_k$.
Promatraj svaki redak od F kao vrh u $\mathbb{R}^k$, particioniraj te vrhove u k klastera koristeći neki od dostupnih algoritama za klasteriranje (npr. K-means).

Spektralno klasteriranje

- Dvije varijante normaliziranog Laplaciana grafa su:

$$L_1 = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

$$L_2 = D^{-1} L = I - D^{-1} W$$

- Navedeni normalizirani Laplaciani zadovoljavaju svojstva:
  - Za proizvoljan vektor $f \in \mathbb{R}^n$:

$$f^T L_1 f = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2$$

  - λ je svojstvena vrijednost od L_1 sa svojstvenim vektorom u ako i samo ako je λ ujedno svojstvena vrijednost i od L_2 sa svojstvenim vektorom w, tako da vrijedi $u = D^{\frac{1}{2}} w$.
  - λ je svojstvena vrijednost od L_2 sa svojstvenim vektorom w ako i samo ako su λ i w rješenje jednadžbe $Lw = \lambda Dw$.
  - I L_1 i L_2 su pozitivno semidefinitne matrice.
  - Najmanja svojstvena vrijednost od L_1 i L_2 je 0, sa svojstvenim vektorom $D^{\frac{1}{2}} 1$ za L_1 i 1 za L_2.

Spektralno klasteriranje

- I L_1 i L_2 imaju n ne negativnih, realnih svojstvenih vrijednosti $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$.
- Ortonormalne vektore $f_1, f_2, \ldots, f_k$ tražimo optimiziranjem funkcija:
  $\min_F Tr(F^T L_1 F)$, t.d. $F^T F = I$ i
  $\min_F Tr(F^T L_2 F)$, t.d. $F^T D F = I$.

Algoritam Normalizirano spektralno klasteriranje (simetrična verzija)

Konstruiraj graf sličnosti koristeći jednu od definiranih metoda. Kreiraj W i D.
Kreiraj simetričan, normaliziran Laplacian grafa L_1 gdje $L_1 = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.
Odredi $f_1, f_2, \ldots, f_k$, top k svojstvenih vektora od L_1.
Konstruiraj matricu $F \in \mathbb{R}^{n \times k}$ iz $f_1, f_2, \ldots, f_k$.
Normaliziraj retke od F tako da $\forall i \leq n, \sum_j F_{ij}^2 = 1$.
Promatraj svaki redak od F kao vrh iz $\mathbb{R}^k$, particioniraj vrhove u k klastera koristeći K-means algoritam.

Konceptualno klasteriranje

- Tradicionalni pristupi klasteriranju grupiraju entitete isključivo na temelju sličnosti parova entiteta. Zbog toga dobiveni klasteri nemaju jednostavno konceptualno objašnjenje.
- Konceptualno klasteriranje organizira entitete tako da odgovaraju razumljivim konceptima.
- Koncepti mogu biti različitih vrsta. Najčešće su pravila (konjunkcije ili disjunktivne normalne forme), međutim mogu biti i distribucije vrijednosti atributa u svim klasterima itd.
- Algoritmi konceptualnog klasteriranja uz pronalazak klastera pružaju i razumljive koncepte koji ih opisuju.
- Neka su $a_1, a_2, \ldots, a_n$ diskretne varijable koje opisuju entitete u skupu podataka. Svaka varijabla ima domenu (skup mogućih vrijednosti). Pretpostavljamo da za svaku varijablu a_i, domena $D_{a_i} = \{0, 1, \ldots, d_i\}$.

Konceptualno klasteriranje

Definicija

Za dva entiteta e_1, e_2 iz E, sintaktička udaljenost $\delta(e_1, e_2)$ između e_1 i e_2 se definira kao broj varijabli koje imaju različite vrijednosti u e_1 i e_2.

Definicija

Relacijski izraz $[a_i \# R_i]$, gdje se R_i zove **referentni skup**, i odgovara jednom ili više elemenata skupa D_{a_i}, a $\#$ odgovara jednom od relacijskih operatora $=$, $\neq$, $\geq$, $\leq$ se zove **selektor**.

- `[height = tall]`, `[color = blue, red]`, `[length` $\geq$ `2]` su primjeri selektora.

Definicija

Logički umnožak selektora se zove **koncept**. $\wedge_{i \in I}[x_i \# R_i]$.
$I \subseteq \{1, 2, \ldots, n\}$, $R_i \subseteq D_i$. Skup entiteta koji zadovoljavaju koncept se zove kompleks.

Konceptualno klasteriranje

- Npr. entitet $e = (2, 7, 0, 1, 5, 4, 6)$ zadovoljava koncept $[a_1 = 2, 3][a_3 \leq 3][a_5 = 3..8]$.
- Neka je $E \in \mathcal{E}$ skup entiteta iz skupa podataka, a $\mathcal{E}$ skup svih mogućih entiteta. Skup E zovemo skupom obzerviranih entiteta (entiteta skupa podataka), a skup $\mathcal{E} \setminus E$ zovemo skupom neobzerviranih entiteta (praznih entiteta).
- Pretpostavimo da je α koncept koji opisuje neke entitete iz skupa podataka i neke entitete koji nisu obzervirani.

Definicija

Broj praznih entiteta pokrivenih od strane koncepta α se zove **rijetkost** od α, u oznaci $s(\alpha)$.

- Neka je $p(\alpha)$ broj obzerviranih entiteta pokrivenih od strane koncepta α, a $t(\alpha)$ ukupan broj entiteta pokrivenih od strane koncepta α.

Konceptualno klasteriranje

- $t(\alpha) = p(\alpha) + s(\alpha)$. Ukupan broj entiteta koji zadovoljava koncept $\alpha = \wedge_{i \in I}[a_i \# R_i]$ je $t(\alpha) = \prod_{i \in I} c(R_i) \cdot \prod_{i \notin I} d_i$. $c(R_i)$ je kardinalitet od R_i, d_i je kardinalitet skupa vrijednosti atributa a_i.

Definicija

Stupanj generalnosti $g(\alpha)$ koncepta α se definira kao:

$$g(\alpha) = \log \frac{t(\alpha)}{p(\alpha)} = \log(1 + \frac{s(\alpha)}{p(\alpha)}).$$

- Neka je L skup koncepata, a R_i skup svih različitih vrijednosti koje atribut a_i poprima u tim konceptima.

Definicija

Operacija koja transformira L u koncept $\wedge_{i=1}^{n}[x_i = R_i]$ se zove **referentna unija**. Rezultantni koncept se zove koncept minimalnog pokrivanja ili *mc*-koncept od L, u oznaci $RU(L)$.

Konceptualno klasteriranje

- *mc*-koncept koji pokriva skup entiteta S ima najmanju rijetkost od svih drugih koncepata koji pokrivaju isti skup.
- Za dva disjunktna skupa entiteta E_1 i E_2 vrijedi $s(RU(E_1)) + s(RU(E_2)) \leq s(RU(E_1 \cup E_2))$.
- Neka su α_1 i α_2 dva preklapajuća koncepta, čija unija pokrivanja je skup entiteta S. Neka S_1 (S_2) označavaju skupove entiteta koje pokrivaju redom samo koncept α_1 i α_2. Neka su α_1' i α_2' proizvoljna dva disjunktna koncepta koji pokrivaju isti skup entiteta S. Ako su $RU(E_1)$ i $RU(E_2)$ disjunktni koncepti, tada $s(RU(E_1)) + s(RU(E_2)) \leq s(\alpha_1') + s(\alpha_2')$.

Definicija

Koncept α je maksimalan uz operaciju $\subset$ s obzirom na svojstvo P, ukoliko ne postoji koncept $\alpha*$ sa svojstvom P, takav da $\alpha \subset \alpha*$. Operator zvijezda $G(e|F)$, entiteta e nasuprot skupa entiteta F, je skup svih maksimalnih koncepata uz operaciju $\subset$ koji pokrivaju entitet e, a ne pokrivaju niti jedan entitet iz F.

Konceptualno klasteriranje

- Neka su E_1 i E_2 dva disjunktna skupa entiteta:

Definicija

Pokrivanje $COV(E_1|E_2)$ skupa entiteta E_1 nasuprot E_2 je proizvoljan skup koncepata $\{a_j\}_{j\in J}$, takav da za svaki entitet $e \in E_1$ postoji koncept $\alpha_j,\ j \in J$ koji ga pokriva, a niti jedan koncept α_j ne pokriva niti jedan entitet iz E_2. Vrijedi $E_1 \subseteq \cup_{j\in J}\alpha_j \subseteq \mathcal{E} \setminus E_2$.

Definicija

Rijetkost pokrivanja se definira kao suma rijetkosti koncepata koji čine pokrivanje.

Konceptualno klasteriranje

Teorem

Za proizvoljni prostor entiteta $\mathcal{E}$ i cijeli broj $k \leq d_1 \cdot d_2 \cdot \cdots \cdot d_n$, gdje je d_i kardinalitet skupa vrijednosti varijable a_i, postoji k po parovima disjunktnih koncepata $\alpha_1, \alpha_2, \ldots, \alpha_k$ koji pokrivaju cijeli prostor: $\cup_{j=1}^{K} \alpha_j = \mathcal{E}$.

Teorem

Princip dovoljnosti: za prostor entiteta $\mathcal{E}$ i proizvoljan skup entiteta $E = \{e_1, e_2, \ldots, e_k\}$, $E \subseteq \mathcal{E}$, postoji najmanje jedan skup od k po parovima disjunktnih koncepata $\alpha_1, \alpha_2, \ldots, \alpha_k$ takvih da svaki koncept pokriva jedan entitet, $e_j \in \alpha_j$, $j = 1, 2, \ldots, k$, a unija koncepata pokriva cijeli prostor $\mathcal{E}$, $\cup_{j=1}^{k} \alpha_j = \mathcal{E}$.

Konceptualno klasteriranje

- Leksikografski evaluacijski funkcional (LEF) se definira kao par lista $A =< a - lista, \tau - lista >$, gdje a-list $(a_1, a_2, \ldots, a_l)$ je lista atributa kojom se evaluira pokrivanje. τ-lista $(\tau_1, \tau_2, \ldots, \tau_l)$ je lista tolerancija dodijeljenih atributima a_i $(0 \leq \tau_i \leq 1)$.
- Neka $V_j, j = 1, 2, \ldots$ reprezentira sva moguća pokrivanja skupa entiteta E. Kažemo da je pokrivanje V optimalno po funkcionalu A ako $A(V) \overset{\tau}{<} A(V_j)$.
- $A(V) = (a_1(V), a_2(V), \ldots, a_l(V))$,
  $A(V_j) = (a_1(V_j), a_2(V_j), \ldots, a_l(V_j)), \quad j = 1, 2, \ldots,$ a $\overset{\tau}{<}$ je relacija koja se zove *leksikografski uređaj s tolerancijama*, koji vrijedi ako:

$$
\begin{aligned}
& a_1(V_j) - a_1(V) > Y_1 \\
\text{ili } & |a_1(V_j) - a_1(V)| \leq Y_1 \quad \text{i} \quad a_2(V_j) - a_2(V) > Y_2 \\
\text{ili } & \ldots \\
& \vdots \\
\text{ili } & \ldots \quad \text{i} \quad a_l(V_j) - a_l(V) \geq 0
\end{aligned}
$$

Konceptualno klasteriranje

$$Y_i = \tau_i \cdot (a_{i\,\max} - a_{i\,\min}), \quad i = 1, 2, \ldots, l-1$$
$$a_{i\,\max} = \max_j \{a_i(V_j)\}$$
$$a_{i\,\min} = \min_j \{a_i(V_j)\}$$

- Relacija $\overset{\tau}{<}$ particionira pokrivanja u klase ekvivalencije i uređuje klase linearno, gdje prva klasa sadrži jedno ili više optimalnih pokrivanja, dok sljedeće klase sadrže pokrivanja manje kvalitete.
- A-listu možemo dobiti koristeći kriterije kao:
  - Rijetkost (ili generalnost) pokrivanja.
  - Presjek - definiran kao prosječan stupanj presjeka između proizvoljna dva kompleksa u pokrivanju. Prosječan supanj presjeka je ukupan broj selektora koji ostaju nakon što se iz para kompleksa izbace disjunktni selektori.
  - Neravnoteža, definirana kao $1/k \sum_{i=1}^{k} |1/k \cdot c(E) - c(E \cap \alpha_i)|$.

Konceptualno klasteriranje

- A-listu možemo dobiti koristeći kriterije kao (nastavak):
  - $c(E)$ je kardinalitet skupa entiteta, a $c(E \cap \alpha)$ je broj entiteta pokrivenih od strane kompleksa α.
  - Dimenzionalnost je ukupan broj različitih varijabli uključenih u komplekse pokrivanja. Dimenzionalnost nam govori koliko varijabli koristimo za opisivanje klastera, odnosno koliko varijabli treba izmjeriti da klasificiramo objekt u klaster.

Konceptualno klasteriranje

- Neka je e_0 entitet i α koncept. Operacija $e_0 \vdash \alpha$ je definirana kao:
$$e_0 \vdash \alpha = \begin{cases} \alpha, & \text{ako } e_0 \in \alpha \\ \varnothing, & \text{inače} \end{cases}$$
- Neka je $e_1 = (r_1, r_2, \ldots, r_n)$ i $e_1 \neq e_0$. Operaciju $e_0 \dashv e_1$ definiramo kao: $e_0 \dashv e_1 = \bigcap_{i \in I} (e_0 \vdash [x_i \neq r_i])$.
- Neka $G^u(e|E)$ označava uniju koncepata operatora zvijezda $G(e|E)$. $G^u(e|E) = \cap_{e_i \in E}(e \dashv e_i)$.
- Operator zvijezda dobijemo pretvorbom presjeka u uniju maksimalnih uz operator $\subset$ koncepata. To možemo postići upotrebom pravila distribucije na dobivene logičke izraze, te pravila apsorpcije.

Konceptualno klasteriranje

- `NID` pretvara ne-disjunktno pokrivanje u disjunktno pokrivanje.

Algoritam Procedura NID

Ulaz: $\{\alpha_1, \alpha_2, \ldots, \alpha_l\}$, ne disjunktno pokrivanje skupa entiteta F
Izlaz: Disjunktno pokrivanje $\{\alpha_1^0, \alpha_2^0, \ldots, \alpha_l^0\}$ ili `NEUSPJEH`

Izračunaj sumu kardinaliteta: $sc \leftarrow \sum_{i=1}^{l} c(\alpha_i)$
Izračunaj kardinalitet unije koncepata (algebarski gledano suma, logički disjunkcija koncepata): $cs \leftarrow c\left(\bigcup_{i=1}^{l} \alpha_i\right)$
if $sc = cs$ **then**
    STOP: L je već disjunktno pokrivanje.
end if
for $i = 1, 2, \ldots, l$ **do**
    Odredi relativnu jezgru: $\text{JEZGRA}_i \leftarrow$ entiteti koji su jedinstveno pokriveni od α_i
end for
$\text{Rezidual} \leftarrow F \setminus \bigcup_{i=1}^{l} \text{CORE}_i$
for each JEZGRA_i **do**
    Odredi mc-koncept: $\alpha_i^0 \leftarrow RU(\text{JEZGRA}_i)$, za $i = 1, 2, \ldots, l$
end for
if proizvoljna dva koncepta α_i^0 imaju presjek **then**
    STOP: Ne možemo dobiti disjunktno pokrivanje.
end if

Algoritam Procedura NID - nastavak

while Rezidual $\neq \varnothing$ **do**
  Izaberi $e \in$ Rezidual i izbaci ga: Rezidual $\leftarrow$ Rezidual $\setminus \{e\}$
  for $i = 1, 2, \ldots, l$ **do**
    Odredi koncept pokrivanja: $\alpha_i^1 \leftarrow RU(\{e\} \cup \alpha_i^0)$
  end for
  Briši proizvoljan α_i^1 koji ima presjek s proizvoljnim α_j^0 $(j \neq i)$
  if svi α_i^1 obrisani **then**
    STOP: Ne možemo dobiti disjunktno pokrivanje.
  end if
  Izaberi Najbolji-α među preostalim α_i^1 koristeći LEF:
  $\langle(\Delta\text{rijet}, -\text{rez}, -\Delta\text{sel})(\tau_1, \tau_2, \tau_3)\rangle$
  Neka je α_j^0 koncept koji je spajanjem s e stvorio Najbolji-α
  Ažuriraj koncept: $\alpha_j^0 \leftarrow$ Najbolji-α
end while
return $\{\alpha_1^0, \alpha_2^0, \ldots, \alpha_l^0\}$

- Δrijet - razlika u rijetkosti između α_i^1 i α_i^0.
- res - broj entiteta u Rezidualu pokrivenih od strane α_i^1.
- Δsel - razlika u broju selektora između α_i^0 i α_i^1.

Algoritam PAF algoritam

Ulaz:

E – skup entiteta
k – željeni broj klastera
A – evaluacijski funkcional (LEF)
Izaberi k entiteta (sjemena) iz E
repeat
  koristeći operator zvijezda , odredi zvijezdu svakog sjemena nasuprot drugih sjemena.
  iz svake zvijezde izaberi jedan koncept od k koncepata, tako da je dobivena kolekcija, P, najbolji disjunktni pokrivac od E (pomoću NID postupka).
  if uvjet zaustavljanja primijenjen na P je zadovoljen **then**
    return P
  end if
  if iteracija je neparna **then**
    Izaberi k novih entiteta (sjemena) koji su centralni u kompleksima iz P
  else
    Izaberi k novih entiteta (sjemena) koji su ekstremni u kompleksima iz P
  end if
until False

Softver

Softver za selekciju značajki:

- Python - **scikit-feature** (`SPEC`)

Softver za klasteriranje:

- Java - **WEKA** (`K-means`, `Aglomerativno klasteriranje` - jedna i potpuna veza, prosjek, centroid, Wardov kriterij, `Maksimizacija očekivanja` (normalna distribucija za kontinuirane atribute)).
- Java - **SPMF** (`K-means`, `Bisekcijski K-means`, `DBSCAN`, `Optics`, itd.).
- Python - **scikit-learn** (`Bisekcijski K-means`, `DBSCAN`, `SpectralClustering`).
- Python - **pyclustering** (`CLIQUE`)

Softver

Softver za klasteriranje (nastavak):

- Java - **Elki**, `https://elki-project.github.io/algorithms/` (razni algoritmi bazirani na gustoći, hijerarhijski i razne vrste K-means algoritma).
- C/C++ **gpumafia**, `https://github.com/canonizer/gpumafia` (MAFIA).

Softver za konceptualno klasteriranje:

- Java - **WEKA** (`COBWEB` - koncept u obliku informacija o distribucijama vrijednosti atributa za pripadnike svakog klastera).