

TARTU ÜLIKOOLI
MATEMAATIKA-INFORMAATIKA TEADUSKOND

Klasterdamine andmekaevanduses

Mihhail Juhkam

Referaat

Andmekaevanduse uurimisseminaar
Arvutiteaduse instituut, Tartu Ülikool

Tartu 2003

Abstrakt

Klasterdamine on andmete gruppimine **klasterite** klassidesse, nii et iga klasteri objektid on omavahel väga sarnased ning samal ajal on väga erinevad teiste klasterite objektidest. Objektidevahelised erinevused põhinevad atribuutide, või tunnuste väärtustele. Tihtipeale erinevused avaldatakse kauguse mõõtudena. Klasterdamisel on rakedusi paljudes alades, näiteks andmekaevanduses, statistikas, bioloogias ja masinõppimises.

Andmekaevandus on suurtes andmekogudes huvitavate seoste ja omaduste avastamine. Käesolevas artiklis vaadatakse üle erinevaid klasterdamismeetodeid ning uuritakse, millised klasterdamise meetodid võivad andmekaevanduse protsessile kasu tuua.

Sisukord

Sissejuhatus	3
1 Andmetüübid klastranalüüsis	5
1.1 Intervalltunnused	5
1.2 Binaarsed tunnused	6
1.3 Nominaal- ning järjestustunnused	7
1.4 Erinevat tüüpi tunnused	8
2 Ülevaade klasterdamismeetoditest	9
2.1 Eraldusmeetodid	10
2.1.1 k -Keskmise meetod	10
2.1.2 k -Medoidi meetod	11
2.1.3 Eraldusmeetodid suurtel andmehulkadel: CLARA ja CLARANS	12
2.2 Hierarhilised meetodid	14
2.2.1 Aglomeratiivne ning divisiivne hierarhiline klasterdamine	14
2.3 Tihedusel põhinevad meetodid	16
2.4 Võrestiku meetodid	16
2.5 Mudelitel põhinevad meetodid	16

Sissejuhatus

Objektide gruppeerimine sarnaste objektide klassideks nimetatakse **klasterdamiseks** (ingl. k. *clustering*). **Klaster** (ingl. k. *cluster*) on objektide kogum, mis on sarnased antud klasteri sees ning erinevad teiste klasterite objektidest. Klasteri objekte võib töödelda ühesuguselt paljudes rakendustes.

Klasterdamist rakendatakse paljudes alades: turunduses, bioloogias, kartograafias. Tänu andmebaasidesse kogutud tohutule andmehulgale, klasteranalüüs sai väga populaarseks ja kiiresti arenevaks meetodiks andmekaevanduses. Andmekaevanduses võib klasterdamist kasutada kui iseseisvaks tööriistaks andmete jaotuse uurimiseks ja erinevate klasterite karakteristikute uurimiseks. Samuti klasterdamine võib olla ka ettevalmistussammuks teiste meetodite kasutamiseks, näiteks klassifitseerimise jaoks.

Andmekaevanduses on uurimiste teemaks selliste klasterdamismeetodite leidmine, mis oleksid efektiivsed suurtel andmebaasidel. Põhilisteks nõueteks klasterdamise meetoditele andmekaevadunduse poolt on

- **Võime töötada suure andmehulga korral:** Suurtel andmebaasides on miljardeid kirjeid, mis nõuab klasterdamismeetoditest suurt võimekust.
- **Võime töötada erinevat tüüpi tunnustega:** Paljud algoritmid on loodud intervalltunnuste klasterdamiseks. Samal ajal andmebaasis esinevad ka binaarsed, nominaalsed ning järjestustunnused või nende tüüpide segu.
- **Suvalise kuju klasterite avastamine:** Paljud klasterdamisalgoritmid avastavad klustereid, mis põhinevad Euklidilise või Manhatteni kauguse mõõtmisel. Sellised algoritmid kalduvad avastama sfäärilisi klustereid ühesuguse suurusega ja tihedusega, kuid klasterid võivad olla igasuguse kujuga.
- **Minimaalne nõutav sisendparameetrite arv:** paljud algoritmid nõuavad kasutajatest sisestada vajalikud parameetrid (näiteks klasterite arv). Klasterdamise tulemused võivad olla tundlikud sisendi vastu, mille tõttu tulemuse headust on raske kontrollida.
- **Võime töötada “müraga”:** Paljud andmebaasid sisaldavad erandeid, puuduvaid ning vigaseid andmeid. Mõned klasterdamisalgoritmid on tundlikud sellise müra vastu.
- **Tundetud andmete järjestamise vastu:** Mõned klasterdamisalgoritmid on tundlikud andmete järjestamise vastu ja võivad anda täiesti erinevaid tulemusi erineva järjestamise puhul.
- **Võime töötada paljudimensionaalsete andmetega:** Andmebaas võib sisaldada palju atribuute, kuid paljud klasterdamise algoritmid töötavad hästi ainult väikese mõõtmiste arvuga.

Pidades neid nõudeid meeles, vaatleme lähemalt klasteranalüüsi meetodeid. Esialgu tutvustame erinevaid andmetüüpe ja näitame, kuidas on neid võimalik kasutada klsterdamismeetodites. Järgnevalt kirjeldame erinevaid klsterdamismeetodite liike, sealhulgas eraldamisalgoritmid, hierarhilised algoritmid, tihedusel põhinevad algoritmid ning skaalal põhinevad meetodid. Samuti uurime klsterdamist kõrgedimensionaalses ruumis ning erindite analüüsi.

1. Arvutama *keskmise absoluuthälbe* s_f

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + \dots + |x_{nf} - m_f|), \quad (1.3)$$

kus m_f on tunnuse f aritmeetiline keskmine, st $m_f = \frac{1}{n}(x_{1f} + \dots + x_{nf})$.

2. Arvutama standardiseeritud väärtuse või *z-väärtuse*:

$$z_{if} = \frac{x_{if} - m_f}{s_f}. \quad (1.4)$$

Pärast andmete standardiseerimise (kui standardiseerimine on õigustatud) rehkendatakse kaugused objektide vahel. Kõige rohkem kasutatav kauguse mõõt on ***Eukliidiline kaugus***

$$d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + \dots + (x_{ip} - x_{jp})^2}, \quad (1.5)$$

kus $x_i = (x_{i1}, \dots, x_{ip})$ ja $x_j = (x_{j1}, \dots, x_{jp})$ on kaks objekti p -mõõtmilises ruumis. Teine tuntud kauguse mõõt on ***Manhattani*** või ***linnakvartalide kaugus***

$$d(i, j) = |x_{i1} - x_{j1}| + \dots + |x_{ip} - x_{jp}|. \quad (1.6)$$

Nii Eukliidiline, kui ka Manhattani kauguse mõõt rahuldab järgmisi kaugusfunktsiooni omadusi

1. $d(i, j) \geq 0$: Kaugus on mittenegatiivne arv.
2. $d(i, i) = 0$: Kaugus objektist iseendani on v rdne 0.
3. $d(i, j) = d(j, i)$: Kaugus on sümmeetriline funktsioon.
4. $d(i, j) \leq d(i, h) + d(h, j)$: Otsene tee i -st j -sse on lühem, kui suvaline umbtee, mis läbib suvalise objekti h (*kolmnurga võrratus*).

1.2 Binaarsed tunnused

Binaarne tunnus on selline tunnus mille väärtuste hulgas on kaks elemendi. Tavaliselt neid elemente tähistatakse 1 ja 0, kus 1 tähendab mingisuguse omaduse olemasolut ning 0 - selle omaduse puudumist. Binnarsete andmete korral objektidevahelise erinevuse mõõdu arvutamiseks on vaja erilisi meetodeid. Üks meetod seisneb *kontingentsustabeli* (ingl. k. *contingency table*) arvutamises. Kui kõik binaarsed tunnused eeldatakse võrdseid kaalusid omavat, siis arvutame järgmist tabelit:

	1	0	Σ
1	a	b	$a + b$
0	c	d	$c + d$
Σ	$a + c$	$b + d$	$a + b + c + d$

kus a on tunnuste arv, mille väärtused on võrdsed 1-ga mõlema objekti i ja j korral, b on tunnuste arv, mille väärtused on võrdsed 1-ga objekti i korral kuid on 0 objekti j korral jne. Tunnuste koguarv on seega $a+b+c+d$. Binnarne tunnus on *sümmeetriline*, kui mõlemad väärtused on sama tähtsusega, näiteks sugu. Selliste tunnuste korral objektidevahelist kaugust arvutatakse järgmiselt.

$$d(i, j) = \frac{b + d}{a + b + c + d} \quad (1.7)$$

Binaarset tunnust nimetatakse *asümmeetriliseks*, kui väärtused ei ole sama tähtsusega, näiteks haiguse esinemise indikaator. Kokkuleppeliselt tähistame haruldasema väärtuse 1-ga (näiteks haiguse esinemine). Üks enamlevinutest kauguse mõõtudest on *Jaccardi koeffitsient*. Selle korral ignoreeritakse tunnuseid, mille korral on mõlemate objektide väärtused nullid (meid huvitava sündmuse puudumine):

$$d(i, j) = \frac{b + d}{a + b + c}. \quad (1.8)$$

Kui andmebaasis esinevad nii sümmeetrilised kui ka asümmeetrilised tunnused, siis kasutatakse tunnuste segu meetodikat, mida käsitletakse allpool.

1.3 Nominaal- ning järjestustunnused

Nominaalne tunnus on binaarse tunnuse üldistus, selline tunnus võib omada rohkem kui kaht väärtust. Nominaalse tunnuse on näiteks värv. Olgu nominaalse tunnuse väärtuste arv M . Väärtuseid võib tähistada arvudega $1, 2, \dots, M$, kuid seda tähistust kasutatakse ainult andmetöötamiseks, see ei kajastu väärtuste järjestust. Erinevus, või kaugust kahe objektide vahel võib arvutada järgnevalt:

$$d(i, j) = \frac{p - m}{p}, \quad (1.9)$$

kus m on nende tunnuste arv, mille korral objektidel on samad väärtused ning p on tunnuste koguarv.

Järjestustunnused on tunnused, mille väärtuste korral võib öelda kumb kahest väärtusest järgneb teisele. Näidena võib tuua koolihindeid või auastmeid. Järjestustunnuse väärtused võib teisendada nende *astakuteks*. Oletame, näiteks, et tunnusel f on M_f väärtust. Tähistame järjestatud väärtuste jada $1, \dots, M_f$. Erinevus kahe objektide vahel järjestustunnuse f suhtes võib arvutada järgmiselt.

1. Asendatakse tunnuse f väärtused x_{if} i -nda objekti korral nende väärtuste astakutega $r_{if} \in \{1, \dots, M_f\}$.
2. Kujutame tunnuse f väärtusi vahemikule $[0, 1]$. Seda võib teha teisendades i -nda objekti astakud r_{if} normeeritud arvudega

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}. \quad (1.10)$$

- Objektidevahelise kauguse võib arvutada samamoodi nagu intervalltunnuste korral.

1.4 Erinevat tüüpi tunnused

Reaalelu andmebaasides on objektid esindatud mitme tüüpi tunnustega. Võib juhtuda, et andmebaas sisaldab kõike 5 ülalmainitud tüüpi tunnuseid: intervall-, binaarseid sümmeetrilisi ning asümmeetrilisi, nominaalseid ning järjestustunnuseid. Kuidas võib arvutada kõikide nende tunnuste põhjal kauguste maatriksit? Üks võimalus on teha eraldi klasteranalüüsi eri tüüpide tunnuste jaoks, kuid niimoodi saadud analüüside tulemused ei ole omavahel võrreldavad.

Teine võimalus on arvutada kauguste maatriksi elemente kõikide tunnuste pealt, kujutades kõik tunnused ühise intervalli $[0, 1]$ peale. Oletame, et andmestik sisaldab p erinevat tüüpi tunnust. Siis kahe objekti vaheline kaugus $d(i, j)$ on defineeritud valemiga

$$d(i, j) = \frac{\sum_{f=1}^p \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^p \delta_{ij}^{(f)}},$$

kus indikaatortunnus $\delta_{ij}^{(f)} = 0$ kui kasvõi

- vähemalt ühte vaatlustest x_{if} ja x_{jf} ei ole andmestikus (st. ei ole mõõdetud tunnuse f väärtus kas objektil i või objektil j)
- $x_{if} = x_{jf} = 0$ ja f on asümmeetriline binaarne tunnus

vastasel juhul $\delta_{ij}^{(f)} = 1$. Tunnuse f panust $d_{ij}^{(f)}$ objektide i ja j erinevusse arvutatakse sõltuvalt tunnuse tüübist:

- Kui f on binaarne või nominaalne tunnus, $d_{ij}^{(f)} = 0$ kui $x_{if} = x_{jf}$ ning $d_{ij}^{(f)} = 1$ vastasel juhul.
- Kui f on intervalltunnus,

$$d_{ij}^{(f)} = \frac{|x_{if} - x_{jf}|}{\max_h x_{hf} - \min_h x_{hf}},$$

kus maksimum ja minimum arvutatakse üle kõikide objektide h , mille jaoks on tunnus f vaadeldud.

- Kui f on järjestustunnus, siis viime läbi teisendust, mis on kirjeldatud alampeatükis (1.3). St arvutame väärtuste x_{if} astakuid r_{if} ning käsitleme väärtuseid $z_{if} = \frac{z_{if}-1}{M_f-1}$ intervalltunnuse omadena.

2 Ülevaade klasterdamismeetoditest

On olemas palju klasterdamisalgoritme. Vajaluiku algoritmi peab valima sõltuvalt andmete iseloomust ning püstitatud ülesandest. Klasterdamisalgoritmide kõige tuntum klassifikatsioon on nende jaotamine *eraldusmeetoditeks* (ingl. k. *partitionial methods*) ning *hierarhilisteks meetoditeks* (ingl. k. *hierarchical methods*). Hierarhiliste meetodite tulemuseks saame mitu hierarhilist andmete jaotust klasteriteks, eraldusmeetodite tulemuseks aga vaid ühte jaotust.

Eraldusmeetodid: Eraldusmeetod jagab andmebaasi n objektidest k gruppideks ($k \leq n$) nii, et: (1) iga grupp sisaldab vähemalt ühte objekti ning (2) iga objekt kuulub ainult ühte gruppi. Viimane nõue ei ole rahuldatud nn. *fuzzy* algoritmide puhul, aga neid käsitleme eespool. Etteantud klasterite arvu k jaoks algoritm tekitab esialgse andmete jaotuse. Seejärel algoritm püüab parandada jaotust vahetades punktid klasterite vahel. Hea jaotuse kriteerium on see, et ühe klasteri objektid on üks teisele “lähedased” ning erinevate klasterite objektid on teine teisest “kaugel”. Eraldusmeetodid on vaadeldud alamosas (2.1).

Hierarhilised meetodid: Hierarhiline meetod tekitab mitu teine teisest saadud objektide jaotust. Hierarhilised meetodid jagunevad *jagavateks* (ingl. k. *divisive*) ning *ühendavateks* (ingl. k. *agglomerative*). *Jagav* meetod alustab ühest klasterist, mis koosneb kõikidest andmebaasi objektidest. Igal järgneval sammul mingi klaster jaotatakse kaheks väiksemateks klasteriteks, iga kord tekib üks klaster juurde. Protsess kordub kuni peatumiskriteerium on rahuldatud, või kuni igas klasteris on ainult üks objekt. Ühendav meetod alustab jaotusega, kus iga objekt moodustab iseseisva klasteri. Igal sammul algoritm ühendab kaht kõige lähedaset klasterit, kuni peatumiskriteerium on täidetud või kuni kõik objektid on ühendatud ühte klasterisse. Eraldusmeetodid on vaadeldud lähemalt alamosas (2.2).

On olemas ka teised meetodid:

Tihedusel põhinevad meetodid: (ingl. k. *Density-based methods*) Paljud eraldusmeetodid gruppeerivad objekte klasteritesse objektidevahelise kauguse põhjal. Sellised meetodid on võimelised üles otsida ainult sfäärilisele kujule lähedased klasterid. Suvalise kuju klasterite avastamine ei ole nende meetoditega võimalik. Selle probleemi võib lahendada kasutades *tiheduse* mõistet. Nende meetodite korral võetakse klasterisse uued punktid, niikaua kuni punktide tihedus (punktide või objektide arv ühiku pindala kohta) klasteri “ümbruses” ületab mingit piiri. See tähendab, et iga klasteri punkti antud ümbruses peab sisalduma vähemalt mingi kindel arv punkte. Tihedusel põhinevad meetodid aitavad filtreerida välja andmebaasi erindeid või “müra” ning avastada suvalise kujuga klasterid. Meetod DBSCAN on üks tüüpiline tihedusel põhinev meetod, mida uurime peatükkis (2.3).

Võrestiku meetodid: (ingl. k. *Grid-based methods*) Võrestiku meetodid jagavad objektide ruumi lõplikuks ruutude arvuks moodustades võrestiku struktuuri. Kõik klasterdamise operatsioonid toimuvad sellel võrestikul. Kõige suuremaks meetodi eeliseks on tööaja vähenemine, sest tööaeg ei sõltu tavaliselt objektide arvust, vaid ainult ruutude arvust igas ruumi dimensioonis. Tüüpiliseks võrestiku meetodiks on STING, mida vaatleme peatükis (2.4).

Mudelil põhinevad meetodid: (ingl. k. *Model-based methods*) Mudelil põhinevad meetodid eeldavad mingisuguse parameetrilise mudeli kehtivust andmete kohta ning leiavad paremat mudelit, mis sobib andmetega kokku. Sellised algoritmid jaotavad objekte klasteriteks konstrueerides tihedusfunktsiooni, mis kajastab andmete ruumilist paiknemist. Mudelil põhinevate meetodite abil võib teha kindlaks klasterite arvu, põhinedes standardsele statistikale ning võtta arvesse erindeid. Neid meetodeid vaatleme peatükis (2.5).

Mõned klasterdamisalgoritmid ühendavad endas mitu loetletutest meetoditest, nii et mõnikord on raske otsustada millesse kategooriasse kuulub antud meetod. Järgnevates osatükkides uurime iga kategooria meetodeid lähemalt.

2.1 Eraldusmeetodid

Kõige populaarsemad eraldusmeetodid on **k-keskmise** (ingl. k. *k-means*) ning **k-medoidi** (ingl. k. *k-medoids*) meetodid.

2.1.1 k-Keskmise meetod

Kõige lihtsam ja levinum eraldusalgoritmidest **k-keskmise** algoritm nõuab sisendiks klastrite arvu k ning jaotab n objekte k klastriteks nii, et klastrisisene sarnasus on suur, aga klastritevaheline sarnasus väike. Klasterite sarnasust mõõdetakse lähtuvalt klasterisiseste objektide aritmeetilisest *keskmisest*, mida vaadeldakse klasteri keskusena või tsentroidina. Algoritm töötab järgmiselt. Juhuslikult valitakse k objekti, mis esindavad esialgsete klasterite keskmisi või tsentroide. Ülejäänud objektid jaotatakse klasterite vahel nii, et sarnasus objekti ja vastava klasteri tsentroidi vahel oleks maksimaalne. Sarnasuse all mõistetakse objekti kaugust klasteri tsentroidist. Seejärel arvutatakse klasterite tsentroidid ümber ning protsess kordub, kuni kriteeriumfunktsioon koondub (st kuni kriteeriumi vähenemine on piisavalt väike). Tavaliselt kasutatakse **ruutvea kriteeriumi**

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - m_i|^2,$$

kus p on suvalise objekti esindatav punkt, C_i on klaster keskpunktiga m_i . Nii m_i kui ka p on multidimensionaalsed punktid. Kriteeriumi väärtus on seda väiksem, mida lähedasemad on klasterite objektid oma klastrite keskpunktidele. Algoritmi kirjeldus on toodud Joonisel 1.

- **Algoritm:** k -Keskmise eraldusalgoritm.
- **Sisend:** Klasterite arv k , andmestik n objektiga.
- **Väljund:** Klasterid (k tükki), mis minimeerivad ruutvea kriteeriumi.
- **Meetod:**
 1. Valida juhuslikult k objekti;
 2. korrata
 3. panna iga objekti kõige lähedasema tsentroidiga klasterisse;
 4. arvutada klasterite tsentroidide väärtused ümber;
 5. kuni ruutvea kriteerium koondub;

Joonis 1: k -keskmise algoritm

Algoritm annab häid tulemusi, kui klasterid on eraldatud teine teisest sfäärilised objektide hulgad. Meetodi eelis on see, et ta on efektiivne suurte andmemahude korral. Selle keerukus on $O(nkt)$, kus n on objektide arv, k – klasterite arv ning t – iteratsioonide arv. Meetodi puuduseks on see, et klasterite arv k peab olema määratud. Selle arvu kindlaks tegemiseks kasutatakse mõnikord hierarhilist algoritmit. Probleemiks on ka esialgsete klasterite keskpunktide valik, sest algoritm annab k punktide erineva valiku korral erinevad tulemused. Tavaliselt jooksutatakse algoritmi mitmu korda erineva algseisundi korral, ning valitakse paremat ruutvea kriteeriumi mõttes tulemust. Meetodi puuduseks on ka see, et ta “ei tunne ära” klastereid, mis on sfäärilisest väga erinevad. Samas ta on tundlik erindite ning vigaste andmete vastu, kuna üksik erakordselt suur väärtus mõjutab klasteri keskvärtust, ning seeläbi ka klasterite struktuuri. Tundlikkuse vähendamiseks kasutatakse k -keskmise algoritmi asemel ***k-medoidi*** algoritmi. Selle meetodi korral klasterite tsentroidideks võetakse *medoide* – klasterite kõige keskmisemaid objekte. Sel juhul klasterdamise kaviteet paraneb, aga algoritmi kiirus langeb, sest medoidide valik nõuab rohkem aega ja resursse kui keskmiste arvutamine.

2.1.2 k -Medoidi meetod

Selle meetodi korral valitakse juhuslikult k objekte (*medoide*), mis esindavad klastereid. Igaühe ülejäänutest objektidest pannakse kõige lähedasema medoidiga klasterisse. Seejärel asendatakse igal saamul ühte medoidi mittemedoidiga, kui see asendus põhjustab ruutvea kriteeriumi vähenemist. Ruutvea kriteeriumiks on antud juhul

$$E = \sum_{i=1}^k \sum_{p \in C_i} |p - med_i|^2,$$

kus p on suvaline objekt, C_i on klaster medoidiga med_i .

PAM (Partitioning around Medoids) on üks esimestest k -medoidi meetoditest. Ta rühmitab n objekti k klastridesse. Pärast medoidide esialgset valikut algoritm püüab leida paremat medoidide valikut. Vaadeldakse kõikvõimalikuid objektide paare, kus esimene objekt on medoid aga teine seda pole. Iga sellise paari korral arvutatakse klasterite ruutvea väärtust. Medoid med_j asendatakse sellise objektiga, mille korral ruutviga on minimaalne. Iga klasteri parimad objektid moodustavad medoidide hulka järgmise iteratsioonisammu jaoks. Medoidi PAM algoritm on toodud Joonisel 2. Ehkki k -medoidi meetod on vähetundlik erindite ning andmebaasi vigade vastu, on

- **Algoritm:** PAM eraldusalgoritm.
- **Sisend:** Klasterite arv k , andmestik n objektiga.
- **Väljund:** Klasterid (k tükki), mis minimeerivad ruutvea kriteeriumi.
- **Meetod:**
 1. Valida juhuslikult k klasterite medoidi;
 2. korrata iga valitud medoidi med_j jaoks
 3. korrata iga mittemedoidilise objekti p jaoks
 4. panna iga objekti kõige lähedasema medoidiga klasterisse;
 5. kontrollida, kas medoidi med_j asendamisel p -ga ruutviga väheneb;
 6. kui väheneb, siis valida objekti p j -ndaks medoidiks;
 7. kuni ruutvea kriteerium enam ei vähene;

Joonis 2: PAM algoritm

see meetod väga ajanõudlik võrreldes k -keskmise meetodiga. Suurte andmehulkade klasterdamiseks on k -medoidi meetod kõlbmatu. Samamoodi nagu k -keskmise meetodi puhul, nõuab k -medoidi meetod sisendparameetiks klasterite arvu k , mida ei ole alati ette teada.

2.1.3 Eraldusmeetodid suurtel andmehulkadel: CLARA ja CLARANS

Tavaline k -medoidi algoritm on efektiivne ainult väikestel andmehulkadel. Suurte andmemahtudega tegelemiseks võib kasutada meetodit, mille nimi on CLARA (Clustering LARge Applications). Selle meetodi puhul kasutatakse kogu andmebaasi asemel väiksemat andmete osat või *valimit*. See valim esindab, piisavalt juhusliku valiku korral, kogu andmebaasi. Valimile rakendatakse PAM meetodit, leitakse valimi medoidide ning võetakse neid kogu andmestiku medoidideks. Et tulemust võimalikult paremaks teha, võetakse mitu valimit ning paremaks loetakse selle valimi medoide, mille puhul on kogu andmestiku ruutviga minimaalne. Raamatus ?? näiteks uuri-

ti välja, et rahuldava tulemuse saamiseks piisab 5-st valimist mahuga ¹ $40 + 2k$. Sellele tulemusele põhinedes paneme CLARA algoritmi kirja (Joonis 3). Algoritmi

- **Algoritm:** CLARA eraldusalgoritm.
- **Sisend:** Klasterite arv k , andmestik n objektiga.
- **Väljund:** Klasterid (k tükki), mis minimeerivad ruutvea kriteeriumi (lokaalselt).
- **Meetod:**
 1. Valida juhuslikult k klasterite medoidi;
 2. $E_{min} := \infty$;
 3. korrata $i=1$ kuni 5
 4. võtta andmebaasist juhusliku valimi mahuga $40 + 2k$;
 5. algoritmi PAM abil leida valimi medoide (k tükki);
 6. panna iga andmebaasi objekti lähedaima medoidiga klasterisse;
 7. arvutada kogu andmestiku põhjal ruutvea E ;
 8. kui $E < E_{min}$, siis $E_{min} := E$, lugeda leitud medoidid paremateks;

Joonis 3: CLARA algoritm

CLARA ühe iteratsiooni keerukus on lineaarne objektide arvu n suhtes. Suure n korral see asjaolu teeb meetodit paremaks võrreldes PAM meetodiga, mille korral ühe iteratsioonisammu keerukus on $O(n^2)$. Kuid meetodi CLARA kvaliteet sõltub võetava valimi mahust. Võivad esineda olukordi, kus parima kvaliteediga klasterite medoidid ei sattu ühtegi valimisse. Sel juhul ei ole võimalik saavutada kogu andmebaasi parimat klasterdamist. Meetodi CLARA perandamiseks oli pakutud teine meetod nimega CLARANS (Clustering Large Applications based upon Randomized Search). Meetod CLARANS põhineb selles, et kõik võimalikud k objektidest hulgad esitatakse graafi tippudena, sealjuures naabertippudeks loetakse neid objektide hulga, mis on saadud teine teisest ühe objekti vahetamisega. Ülesanne on leida sellises graafis $G_{n,k}$ tippu, mis koosneb parimatest medoididest (mis minimeerivad ruutvea). Kuna ühe graafi tippu naabrite arv on $k(n - k)$, mis on suur, kui n on suur arv, siis piiratakse üle kontrollitavate naabrite hulka. Olgu maksimaalne üle kontrollitavate naabrite arv m .

Alustatakse otsingut ühe k -elemendilise juhusliku valimiga või ühe $G_{n,k}$ tippuga S_0 . Otsing toimub nii, et vaadatakse üle m juhuslikult valitud S_0 naabreid (objektide hulki). Seda naabrit S , mis minimeerib ruutvea, loetakse paremaks jooksvaks

¹Alustades teisest valimist valitakse k hetkel parimaid medoide ning valitakse juhuslikult juurde $40 + k$ objekte

medoidide hulgaks ning otsimise protsess kordub tippu S jaoks. Peatatakse siis, kui ruutvea vähenemist enam ei toimu. Tulemuseks saadakse k medoide, mis minimeerivad ruutvea, kuid miinimum on ainult lokaalne. Võib korrata meetodi mitme algustippude S_0 korral ning valida seda tulemust, mis annab ruutvea väiksemat lokaalset miinimumi. Joonisel 4 on näidatud CLARANS algoritm.

- **Algoritm:** CLARANS eraldusalgoritm.
- **Sisend:** Klasterite arv k , andmestik n objektiga, maksimaalne kontrollitavate naabrite arv m , lokaalse miinimumi otsingute arv l .
- **Väljund:** Klasterid (k tükki), mis minimeerivad ruutvea kriteeriumi (lokaalselt).
- **Meetod:**
 1. $E_{min} := \infty$; $i := 1$;
 2. Valida juhuslikult graafi $G_{n,k}$ tippu S_0 ;
 3. $j := 1$;
 4. Valida juhuslikult S_0 naabertippu S ;
 5. Kui $E(S) < E(S_0)$, siis $S_0 := S$, minna üle sammule (3);
 6. Vastasel juhul $j := j + 1$. Kui $j \leq m$ siis minna üle sammule (4);
 7. Kui $j > m$ ning $E(S_0) < E_{min}$, siis $E_{min} := E(S_0)$, lugeda tippu S_0 paremaks;
 8. $i := i + 1$, ning kui $i > l$, väljastada S_0 ;

Joonis 4: CLARANS algoritm

On näidatud, et CLARANS on efektiivsem nii PAM, kui ka CLARA meetoditest. Meetodid CLARA ning CLARANS sobivad suurte admehulkade klasterdamiseks.

2.2 Hierarhilised meetodid

Hierarhilise meetodid gruppeerivad andmeid klasterite puuks. Hierarhilisi klasterdamismeetodeid võib klassifitseerida *ühendavateks* (*aglomeratiivseteks*) ning *jagavateks* (*divisiivseteks*) meetoditeks, sõltuvalt sellest, kuidas klasterite puu ehitatakse: kas allpoolt ülesse või ülespoolt alla.

2.2.1 Aglomeratiivne ning divisiivne hierarhiline klasterdamine

On olemas kaht tüüpi hierrhilise klasterdamise meetodeid:

Agglomeratiivne (ühendav) klasterdamine: Need meetodid alustavad seisundiga, kus iga objekt on omaette klasteriks ning seejärel ühendavad neid atomaarseid klustereid järjest suurematesse klastridesse. Igal sammul pannakse kokku kaht kõige lähedasemat klasterit. Protsess kestab kuni kõik objektid on ühes klasteris või kuni peatumiskriteerium on rahuldatud. Tavaliselt peatatakse siis, kui kahe kõige lähedasema klasteri vaheline kaugus D ületab etteantud väärtuse D_0 . Valdav osa hierarhilistest meetoditest kuulub sellesse gruppi.

Divisiivne (jagav) klasterdamine: Selle meetodi algseisus on kõik objektid ühes klasteris. See suur klaster jagatakse väiksemateks klasteriteks kuni iga objekt sattub eraldi klasterisse, või kuni peatumisreegel on rahuldatud (näiteks vajalik klasterite arv on saavutanud või kahe kõige lähedasema klasteri vaheline kaugus d jääb etteantud väärtusest d_0 allapoole). Igal algoritmi sammul jagatakse ühe klasteri nii, et saada võimalikult kaugemad klasterid.

Mõlema meetodi puhul klasterite vahelise kauguse mõõtudena kasutatakse tavaliselt ühte järgmistest funktsioonidest:

Minimaalne naabrivaheline kaugus:

$$d_{min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'|$$

Maksimaalne naabrivaheline kaugus:

$$d_{max}(C_i, C_j) = \max_{p \in C_i, p' \in C_j} |p - p'|$$

Keskliste kauguste vahe:

$$d_{mean}(C_i, C_j) = |m_i - m_j|$$

Kauguste vahe keskmine:

$$d_{avg}(C_i, C_j) = \frac{1}{n_i n_j} \sum_{p \in C_i} \sum_{p' \in C_j} |p - p'|$$

kus $|p - p'|$ on punktide p ja p' vaheline kaugus, m_i on klasteri C_i objektide keskmine ning n_i on klasteris C_i asuvate objektide arv.

Kõige levinumad hierarhiliste klasterdamise meetoditest on agglomeratiivne **lähimsideme** (ingl. k. **single-linkage**) ning **täissideme** (ingl. k. **complete-linkage**) algoritmid. Lähimsideme meetodis klasteritevaheline kaugus on minimaalne naabrivaheline kaugus ja täissideme meetodis – maksimaalne naabrivaheline kaugus. Mõlema meetodi korral klasterid ühendatakse siis, kui nendevaheline kaugus on minimaalne. Täissideme meetod produtseerib tavaliselt kompaktseid sfäärilise kujuga

klustereid, seevastu lähimsideme meetodil on tendents genereerida väljaveinitatud või hargneva struktuuriga klustereid. Vaatamata sellele, et lähimsideme meetod on võimeline eristada keerulise struktuuriga klustereid, täissideme meetod annab tavaliselt siiski kasulikumaid tulemusi [?].

Hierarhiliste meetodite peamine puudus on see, et kui objektide ühendamise või jagamise toimunud, ei saa seda otsustust enam muuta. Ei saa mitte taastada elneval sammul tekitatud klasterid, ega nihutada objekte klasterite vahel. Ühendamisel või jagamisel tekkinud ebatäpsus võib tuua kaasa kehvade kvaliteediga klustereid. Vaatamata sellele, et hierarhilised meetodid on lihtsad, nad on aja- ning resurssinõudlikud, sest otsustuse tegemiseks peab arvutama ja võrdlema omavahel palju objekte ning klustereid. On pakutud välja ka teisi hierarhilisi meetodeid, mille efektiivsus ning klasterdamise kvaliteet on suurem.

2.3 Tihedusel põhinevad meetodid

2.4 Võrestiku meetodid

2.5 Mudelitel põhinevad meetodid

[2] [1]

Viited

- [1] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–319, 1999.
- [2] Raymond T. Ng and Jiawei Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference*, Santiago, Chile, 1994.