Tiheyteen perustuva klusterointi

Petteri Hintsanen
petteri.hintsanen@cs.helsinki.fi

Klusterointimenetelmät-seminaari
Helsingin yliopisto, tietojenkäsittelytieteen laitos
Raportti C–2002–54, s. 107–117, marraskuu 2002

Tiivistelmä

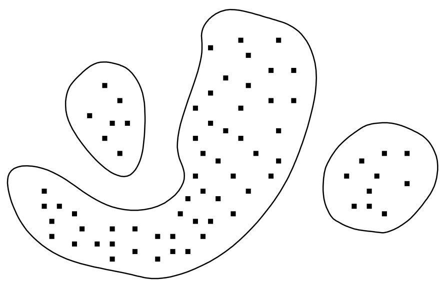
Tiheysperustaisessa klusteroinnissa datapistejoukko metrisessä avaruudessa jaetaan sen yhtenäisiin komponentteihin. Jakaminen vaatii tiheyden, yhtenäisyyden ja rajan käsitteiden määrittelyt. Tiheyteen perustuva klusterointi sopii erityisesti tilaa kuvaavan datan klusterointiin, joka on yleensä esitetty metrisessä avaruudessa. Tässä raportissa tarkastellaan tiheysperustaisessa klusteroinnissa vaadittavien käsitteiden määrittelyä ja soveltamista käyttäen esimerkkinä DBSCAN-algoritmia, joka on eräs kirjallisuudessa esitetyistä tiheyteen perustuvista klusterointialgoritmeista. DBSCAN soveltuu myös tilaa kuvaavan datan klusterointiin. Lisäksi raportissa tarkastellaan DBSCAN-algoritmin erityispiirteitä ja laajennuksia. Havainnot kootaan lopuksi lyhyeksi yhteenvedoksi.

1 Johdanto

Käsillä olevassa raportissa tarkastellaan *tiheyteen perustuvaa klusterointia* (density-based clustering). Tiheyteen perustuvassa klusteroinnissa datajoukon S alkiot ajatellaan pisteiksi koko datajoukon sisältävässä metrisessä avaruudessa. Tämä on tyypillinen oletus useissa klusterointialgoritmeissa. Muita oletuksia datajoukon sisältävän avaruuden rakenteesta ja ominaisuuksista ei tarvitse (välttämättä) tehdä.

Ideana tiheyteen perustuvassa klusteroinnissa on osittaa pistejoukko avaruudessa joukoksi sen *yhtenäisiä* (connected) komponentteja [Ber02]. Tällaisen jakoidean soveltaminen vaatii *tiheyden* (density), *yhtenäisyyden* (connectivity) ja *rajan* (boundary) käsitteiden määrittelyt. Käsitteet liittyvät läheisesti pisteen lähimpien naapureiden käsitteeseen. Intuitiivisesti tiheyteen perustuvien klusterointimenetelmien tuottamat klusterit ovat sellaisia avaruuden alueita, jotka sisältävät useita pistejoukon pisteitä suhteellisen "tiheässä" [Fas99]. Tiheysperustaisissa klusteroinneissa tällaiset alueet voivat olla mielivaltaisen muotoisia ja kokoisia (katso kuvaa 1).

Tiheyteen perustuvat klusterointimenetelmät eroavat niinsanotuista k-optimointialgoritmeista (esimerkiksi k:n keskiarvon menetelmä (K-MEANS), k:n medoidin menetelmä (K-MEDOIDS, sumean k:n keskiarvon menetelmä (FUZZY K-MEANS)) tuottamalla konveksien klustereiden sijaan mielivaltaisen muotoisia ja kokoisia klustereita. Ne ovat myös hyvin neutraaleja kohinan (outliers) suhteen [Fas99, Ber02]: kohina aineistossa ei vaikuta esimerkiksi klustereiden muotoon läheskään yhtä herkästi kuin k-optimointialgoritmeissa. Kolmas merkittävä ero on parametrin k puuttuminen: tiheysperustaiset menetelmät eivät (tavallisesti) tarvitse tietoa klustereiden lukumäärästä etukäteen.



Kuva 1: Tason pistejoukko on jaettu tiheyden perusteella kolmeen klusteriin (ympäröidyt alueet)

Tiheyteen perustuvien klusterointimenetelmien luonnollinen sovelluskohde on *tilaa kuvaavan datan* (spatial data) klusterointi. Data on esitetään tavallisesti metrisessä avaruudessa [Ber02]. Tilan oliot voivat muodostaa luokkia, joita klusterointimenetelmä pyrkii löytämään datasta. Esimerkiksi satelliittikuvat ja paikkatietojärjestelmän data ovat tilaa kuvaavaa tietoa. Edelleen kuvista voidaan yrittää löytää klusterointimenetelmän avulla vaikkapa kaikki talot, jotka sijaitsevat tietyn joen rannalla. Sellaiset talot muodostavat aineiston yhden luokan [EKSX96]. Lisäksi tämänkaltaisessa datassa olevien klusterien määrää voi olla hyvin vaikeaa arvioida etukäteen.

Seuraavassa käsiteltävä Ester et al.:n kehittämä DBSCAN-algoritmi (Density-Based Spatial Clustering of Applications with Noise) on eräs esimerkki lukuisista kirjallisuudessa esitetyistä, tiheyteen perustuvista klusterointialgoritmeista. Esitys perustuu alkuperäisartikkeliin [EKSX96].

2 DBSCAN-algoritmi

Tyypillisesti tilaa kuvaava data-aineisto on koottu *tilatietokantajärjestelmään* (spatial database system, SDBS). Tilaa kuvaava aineisto sekä tietokantajärjestelmä asettavat klusterointimenetelmille erityisiä vaatimuksia [EKSX96]:

1. Valittavien parametrien määrän tulee olla mahdollisimman pieni, sillä usein parametrien arvoja on vaikeaa tai mahdotonta tuntea etukäteen suuria aineistoja käsiteltäessä.

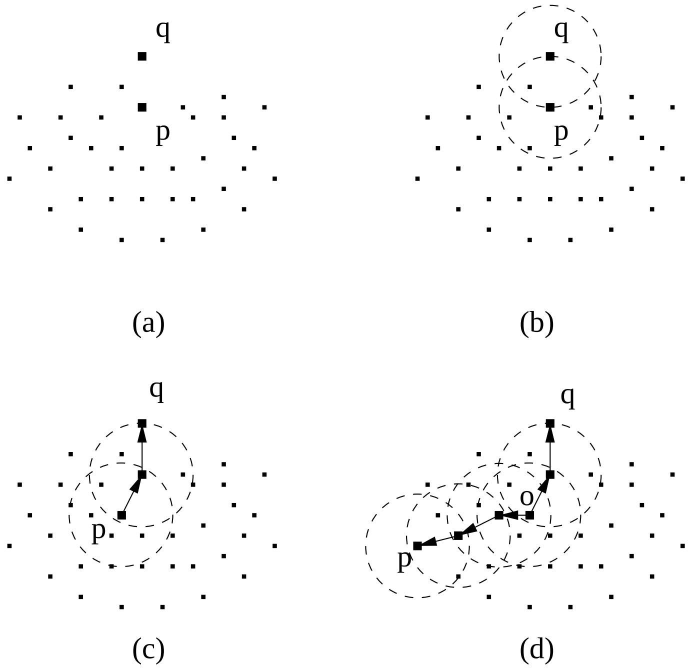
2. Klustereiden muodoille ei saa asettaa rajoituksia, koska tilaa kuvaavassa datassa klusterit voivat olla lähes minkä muotoisia tahansa (esimerkiksi joet satelliittikuvissa).

3. Aineistojen valtava koko (mahdollisesti satoja tuhansia tietueita) asettaa tehokkuudelle erityisiä vaatimuksia. Erityisesti aineiston läpikäyntien määrä tulisi minimoida.

DBSCAN-algoritmi täyttää nämä vaatimukset. Aloitetaan klusterointialgoritmin tarkastelu määrittelemällä täsmällisesti tarvittavat käsitteet. Huomaa, että määritelmät liittyvät ainoastaan DBSCAN-algoritmiin. Ne ovat siis konkreettisia esimerkkejä siitä, kuinka tiheyden käsite voidaan määritellä.

2.1 Määritelmät

Merkitään kirjaimella $D \subset \mathbb{R}^n$ klusteroitavien pisteiden joukkoa. DBSCAN-algoritmi käyttää kahta parametria, ε ja μ, määrittämään relaatioita pisteiden välillä. Käyttäjä antaa nämä



Kuva 2: Esimerkkejä saavutettavuudesta tason pistejoukossa. Kuvaan (a) on merkitty ydinpiste $\mathbf{p}$ ja rajapiste $\mathbf{q}$. Kuvassa (b) piste $\mathbf{q}$ on suoraan saavutettavissa tiheyden perusteella pisteestä $\mathbf{p}$, siis $\mathbf{p} \rightarrow \mathbf{q}$ (huomaa, että kääntäen ei päde). Kuvassa (c) piste $\mathbf{q}$ on saavutettavissa tiheyden perusteella pisteesta $\mathbf{p}$, siis $\mathbf{p} \rightsquigarrow \mathbf{q}$ (tämäkään ei päde kääntäen). Kuvassa (d) pisteet $\mathbf{p}$ ja $\mathbf{q}$ ovat yhteydessä tiheyden perusteella pisteen $\mathbf{o}$ kautta, siis $\mathbf{p} \leftrightarrow \mathbf{q}$.

parametrit. Seuraavissa määritelmissä merkintä $\mathrm{B}(\mathbf{x_0}, \varepsilon) = \{\mathbf{x} \in D : ||\mathbf{x_0} - \mathbf{x}|| < \varepsilon\}$ tarkoittaa pisteen $\mathbf{x_0} \in D$ ε-säteistä (pallo)ympäristöä. Merkinnällä $|\mathrm{B}(\mathbf{x_0}, \varepsilon)|$ tarkoitetaan kyseisen ympäristön sisältämien pisteiden lukumäärää. Pisteiden oletetaan olevan aina joukon D alkioita, jollei toisin mainita.

Määritelmä 1 (Ydinpiste ja rajapiste). Olkoon $\mathbf{p}$ jokin piste. Jos $|\mathrm{B}(\mathbf{p}, \varepsilon)| \geq \mu$, pistettä $\mathbf{p}$ kutsutaan *ydinpisteeksi*. Muuten piste $\mathbf{p}$ on *rajapiste*.

Määritelmä 2 (Suora saavutettavuus tiheyden perusteella). Piste $\mathbf{q}$ on *suoraan saavutettavissa tiheyden perustella* pisteestä $\mathbf{p}$ parametreilla ε ja μ, jos $\mathbf{q} \in \mathrm{B}(\mathbf{p}, \varepsilon)$ ja $|\mathrm{B}(\mathbf{p}, \varepsilon)| \geq \mu$ (eli $\mathbf{p}$ on ydinpiste). Tällöin merkitään $\mathbf{p} \rightarrow \mathbf{q}$.

Määritelmä 3 (Saavutettavuus tiheyden perusteella). Piste $\mathbf{q}$ on *saavutettavissa tiheyden perusteella* pisteestä $\mathbf{p}$, jos on olemassa jono pisteitä $(\mathbf{p}_1 = \mathbf{p}, \mathbf{p}_2, \ldots, \mathbf{p}_{n-1}, \mathbf{p}_n = \mathbf{q})$ siten, että $\mathbf{p_i} \rightarrow \mathbf{p_{i+1}}\ \forall i, 1 \leq i < n$ parametreilla ε ja μ. Tällöin merkitään $\mathbf{p} \rightsquigarrow \mathbf{q}$.

Huomaa, että kumpikaan saavutettavuusrelaatioista ei ole välttämättä symmetrinen (formaalisti $\mathbf{p} \rightarrow \mathbf{q} \not\Rightarrow \mathbf{q} \rightarrow \mathbf{p}$ ja $\mathbf{p} \rightsquigarrow \mathbf{q} \not\Rightarrow \mathbf{q} \rightsquigarrow \mathbf{p}$).

Määritelmä 4 (Yhteys tiheyden perusteella). Piste $\mathbf{p}$ on *yhteydessä tiheyden perusteella* pisteeseen $\mathbf{q}$, jos on olemassa piste $\mathbf{o}$ siten, että $\mathbf{o} \rightsquigarrow \mathbf{p}$ ja $\mathbf{o} \rightsquigarrow \mathbf{q}$ parametreilla ε ja μ. Tällöin merkitään $\mathbf{p} \leftrightarrow \mathbf{q}$.

Kuvassa 2 on esitetty muutama esimerkki ylläolevista määritelmistä tasossa.

Määritelmä 5 (Klusteri). Joukkoa $C \subseteq D$ sanotaan *klusteriksi*, jos seuraavat ehdot ovat voimassa:

1. Jos $\mathbf{p} \in C$ ja $\mathbf{p} \to \mathbf{q}$, niin myös $\mathbf{q} \in C$ (maksimaalisuus).
2. Jos $\mathbf{p}, \mathbf{q} \in C$, niin $\mathbf{p} \leftrightarrow \mathbf{q}$ (yhtenäisyys).

Havaitaan, että minkä tahansa klusterin yksi ydinpiste $\mathbf{p}$ määrittelee koko klusterin: jokainen piste, joka on tiheyden perusteella saavutettavissa pisteestä $\mathbf{p}$, kuuluu samaan klusteriin kuin piste $\mathbf{p}$. Pistettä $\mathbf{p}$ voidaan kutsua vastaavaa klusteria *kuvaavaksi pisteeksi* (representative point). Lisäksi parametreja liu'uttamalla DBSCAN-algoritmia voidaan käyttää hierarkkiseen klusterointiin.

Määritelmä 6 (Kohina). Olkoon $\{C_1, \ldots, C_n\}$ joukon D klustereiden joukko ja $\mathbf{p} \in D$ piste. Jos $\mathbf{p} \notin C_i \; \forall i, 1 \leq i \leq n$, pistettä $\mathbf{p}$ sanotaan *kohinapisteeksi*.

2.2 Algoritmi

Varsinaisen klusterointialgoritmin suoritus aloitetaan mielivaltaisesta pistejoukon pisteestä $\mathbf{p}$. Pisteiden käsittelyjärjestyksellä ei yleensä ole merkitystä. Joissain harvinaisissa tapauksissa tietty rajapiste $\mathbf{r}$ voi olla kahden klusterin välissä siten, että se voisi kuulua määritelmän 5 mukaan kahteen tai useampaan klusteriin. Tällöin lopullinen pisteen $\mathbf{r}$ klusteri riippuu pisteiden käsittelyjärjestyksestä (jokainen piste kuuluu siis korkeintaan yhteen klusteriin).

Jos piste $\mathbf{p}$ on ydinpiste, algoritmi muodostaa määritelmän 5 mukaisen klusterin kaikista pisteistä, jotka ovat saavutettavissa tiheyden perusteella pisteestä $\mathbf{p}$. Toisaalta jos piste $\mathbf{p}$ on rajapiste, yhtään pistettä ei voida saavuttaa tiheyden perusteella pisteestä $\mathbf{p}$, jolloin algoritmi siirtyy käsittelemään seuraavaa syötepistettä.

Alussa kaikki pisteet ovat "merkitsemättömiä" (eli käsittelemättömiä). DBSCAN (algoritmi 1) kutsuu jokaisella tietokannan merkitsemättömällä pisteellä laajenna-aliohjelmaa (algoritmi 2), joka muodostaa pistettä vastaavan klusterin. Jos käsiteltävä piste ei ole ydinpiste, se merkitään kohinapisteeksi ja aliohjelman suoritus lopetetaan. Kyseinen piste voidaan kuitenkin algoritmin suorituksen myöhemmässä vaiheessa merkitä johonkin klusteriin kuuluvaksi.

Algoritmi 1 DBSCAN

Esiehto: Pistejoukko D, ε, μ.
Jälkiehto: Jokainen piste on merkitty joko kohinaksi tai johonkin klusteriin.

$k := 1$
for all $\mathbf{p} \in D$ **do**
    if $\mathbf{p}$ on merkitsemätön **then**
        if laajenna(D, $\mathbf{p}$, k, ε, μ) = TRUE **then**
            $k := k + 1$ {klusteri k on valmis, muodostetaan seuraava klusteri $k + 1$}
        end if
    end if
end for

Jokainen laajenna-aliohjelman kutsu luo siis joko uuden klusterin tai merkitsee käsittelemättömän pisteen kohinapisteeksi. Aliohjelma pitää yllä joukkoa S, joka sisältää kaikki sellaiset pisteet, jotka ovat tiheyden perusteella saavutettavissa aliohjelmalle parametrina annetusta pisteestä $\mathbf{p}$ (määritelmä 3). Tämän joukon kaikki pisteet merkitään kuuluvaksi samaan klusteriin. Aliohjelman suoritusta jatketaan niin kauan kunnes kaikki joukon S pisteet on käsitelty. Aliohjelman suorituksen aikana joukkoon S lisätään uusia pisteitä $\mathbf{p}_i$ siten, että jokainen piste $\mathbf{p}_i$ on saavutettavissa tiheyden perusteella pisteestä $\mathbf{p}$. Tämä varmistaa aliohjelman tuottavan määritelmän 5 mukaisia klustereita ja samalla DBSCAN-algoritmin oikeellisuuden.

Algoritmi 2 laajenna

Esiehto: Pistejoukko D, piste $\mathbf{p}$, klusteritunnus k, ε, μ.

Jälkiehto: Kaikki pisteet, jotka ovat tiheyden perusteella saavutettavissa pisteestä $\mathbf{p}$, on merkitty klusteriin k.

```
 1: S := B(p, ε) {tietokantajärjestelmän funktio}
 2: if |S| < μ then {p on rajapiste}
 3:    merkitse p kohinaksi
 4:    return FALSE
 5: else {joukon S pisteet ovat saavutettavissa tiheyden perusteella pisteestä p}
 6:    merkitse kaikki joukon S pisteet klusteriin k
 7:    S := S \ {p}
 8:    while S ≠ ∅ do
 9:       valitse jokin p0 ∈ S
10:       S0 := B(p, ε)
11:       if |S0| ≥ μ then
12:          for all s ∈ S0 do
13:             if s on merkitsemätön then
14:                S := S ∪ {s}
15:                merkitse s klusteriin k
16:             end if
17:             if s on merkitty kohinaksi then
18:                merkitse s klusteriin k
19:             end if
20:          end for
21:       end if
22:       S := S \ {p0}
23:    end while
24:    return TRUE
25: end if
```

2.3 Analyysi

Ester et al. eivät esitä artikkelissaan kovin täsmällistä aikavaatimusanalyysia algoritmilleen. Algoritmin 1 aikavaatimus on rivin 4 aliohjelmakutsua lukuunottamatta selvästi $\Theta(n)$, jossa n on klusteroitavien pisteiden lukumäärä. Algoritmin 2 aikavaatimukseksi Ester et al. esittävät keskimäärin $O(\log n)$. Algoritmin aikavaativuus riippuu erityisesti kahdesta seikasta: rivin 1 ja 10 aluekyselyistä sekä rivien 12-20 silmukasta.

Rivien 1 ja 10 aluekyselyt voidaan toteuttaa R^*-puilla [BKSS90] SD-tietokantajärjestelmässä siten, että niiden vaativuus on keskimäärin $O(\log n + c)$. Tässä oletetaan, että aluekyselyjen palauttamien pisteiden määrä on rajoitettu ylhäältä vakiolla c ja samalla myös huomattavasti pienempi kuin n (siis $|\mathrm{B}(\mathbf{p}, \varepsilon)| \leq c \; \forall \mathbf{p} \in D$, $c \ll n$ vakio). Silloin myös riveillä 12-20 oleva silmukka vie ajan $O(c)$ ja koko DBSCAN-algoritmi ajan $\Theta(n) \cdot O(\log n + c) = O(n \log n)$.

Tämä ei selvästikään päde ilman mainittuja oletuksia. Samoin R^*-puut ovat tehokkaita (käyttökelpoisia) vain silloin, kun avaruuden dimensio on korkeintaan 16 [CN00]. Toisaalta Weber, Schek ja Blott ovat kokeellisesti todenneet, että yksinkertainen kaikki pisteet läpikäyvä haku on käytännössä tehokkaampi kuin mikään avaruuden osittamiseen perustuva haku, jos dimensio on yli 10 [WSB98].

2.4 Klusteroinnin laatu ja parametrien valinta

DBSCAN-algoritmin tuottamien klustereiden laatu riippuu erityisesti käyttäjän antamista parametreista ε ja μ. Sopivan arvon valitseminen etukäteen on tietenkin vaikeaa, koska yleensä syötedatan "luonnollisten" klustereiden rakennetta ei tunneta. Parametrien tulisi olla pieniä, jotta vältyttäisiin kohinan aiheuttamalta virheeltä, mutta toisaalta tarpeeksi suuria, jotta aineistosta löytyisi "luonnollisia" klustereita.

Toisaalta, kuten klusterointialgoritmeissa yleensä, tiheyteen perustuvan klusteroinnin laadun mittaaminen on hankalaa, koska usein klusterien laatu (tai oikeammin klusterin käsite) riippuu myös tulkitsijasta [EC02]. Toisaalta tulosten tulkitseminen on jo itsessään vaikeaa. Esimerkiksi klustereiden visualisointi korkeammissa ulottuvuuksissa on hankalaa.

Ester et al. eivät esitä artikkelissaan mitään kvantitatiivista mittaa klusteroinnin laadun arvioimiseksi. Tätä perustellaan vertailukohteen puuttumisella. Siten, Estivill-Castron terminologiaa [EC02] käyttäen, DBSCAN-algoritmilla ei ole eksplisiittistä *induktiivista periaatetta* (inductive principle), eikä siinä ole eksplisiittistä *optimointiongelmaa*: pistejoukon klusterointi on lähes yksikäsitteisesti määrätty ja helposti löydettävissä, kun parametrit ε ja μ on kiinnitetty. Voidaan siis todeta, ettei DBSCAN-algoritmissa ole optimointiongelmaa lainkaan, tai että optimointiongelma liittyy sopivan klusteroinnin tuottavien "optimaalisten" parametriarvojen löytämiseen.

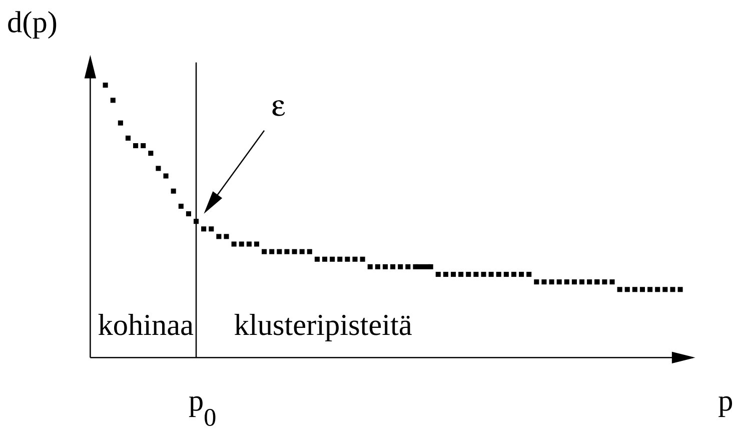
DBSCAN kuitenkin selvästi muodostaa datasta *mallin*. Malli (klustereita edustavat ydinpisteet) kuvaa datan rakennetta ja sen avulla on mahdollista luokitella uusia datapisteitä joko tiettyihin klustereihin tai kohinaksi.

Sen sijaan Ester et al. esittävät artikkelissaan yksinkertaisen heuristiikan parametrien määrittämiseksi. Heuristiikalla voidaan yrittää etsiä pienimpiä mahdollisia parametreja ε ja μ, joilla aineistosta löydetään kaikkein hajanaisin (eli vähiten "tiheä") klusteri ja samalla vältetään kohinan aiheuttama virhe.

Olkoon $\mathbf{q}$ pisteen $\mathbf{p}$ k:nneksi lähin piste ja $d = ||\mathbf{p} - \mathbf{q}||$. Silloin $|\mathrm{B}(\mathbf{p}, d)| \approx k + 1$ ja $|\mathrm{B}(\mathbf{p}, d)| > k + 1$ vain jos yksi tai useampi pisteen $\mathbf{p}$ naapuripisteistä on yhtä etäällä pisteestä $\mathbf{p}$, mikä on harvinaista. Edelleen jos $\mathbf{p}$ on klusterin piste, luvun k vaihtaminen ei tavallisesti kasvata lukua d huomattavasti: jos d kasvaisi nopeasti, klusterin pisteet olisivat suurinpiirtein samalla linjalla, mikä sekin on harvinaista.

Kiinnittämällä luku k voidaan muodostaa kuvaus $d : D \to \mathbb{R}$, $d(\mathbf{p}) = ||\mathbf{p} - \mathbf{q}||$, jossa $\mathbf{q}$ on siis pisteen $\mathbf{p}$ k:nneksi lähin piste. Laskemalla kaikille pisteille $\mathbf{p} \in D$ funktion $d(\mathbf{p})$ arvo, lajittelemalla arvot laskevaan järjestykseen ja piirtämällä arvoista kuvaaja saadaan *lajiteltu k-etäisyyskuvaaja* (etäisyysjakauma). Tavallisesti kohinapisteillä on suuri funktion d arvo. Tämä näkyy kuvaajassa siten, että käyrän alkupuolella d arvo pienenee nopeasti (katso kuvaa 3). Klusteripisteiden kohdalla käyrän kulku on tasaisempaa. Käyrän "tasaisen" osan alkupiste voi olla vaikeaa löytää automaattisesti, mutta käyttäjän on suhteellisen helppoa todeta se kuvaajasta. Olkoon $\mathbf{p}_0$ tämä piste. Parametriksi ε valitaan $d(\mathbf{p}_0)$.

Ester et al. suosittelevat parametrin μ kiinnittämistä arvoon $\mu = 4$ (kaksiulotteisessa avaruudessa). Arvoilla $\mu > 4$ kokeissa ei havaittu merkittävää muutosta lajitellussa k-etäisyyskuvaajassa. Lisäksi suuremmat parametrin μ arvot vaativat enemmän laskenta-aikaa. Pienemmät arvot voivat tulla kyseeseen sovelluksesta riippuen.



Kuva 3: Esimerkki lajitellusta k-etäisyyskuvaajasta. Pystyviiva osoittaa sopivan $\mathbf{p}_0$:n, josta saadaan parametri $\varepsilon = d(\mathbf{p}_0)$.

2.5 Laajennuksia ja muita menetelmiä

DBSCAN-algoritmissa, samoin kuin lähes kaikissa muissakin klusterointimenetelmissä, on eräs puute: se ei ota huomioon data-aineistossa mahdollisesti olevia rajoitteita [ZL02a]. Rajoitteet tulee ottaa huomioon aineiston klusteroinnissa. Esimerkiksi lääketieteellisessä sovelluksessa kaksi aineiston pistettä voivat olla etäisyydeltään hyvin lähellä toisiaan, mutta ne tulee sijoittaa eri klustereihin fyysisten tai biologisten rajoitteiden takia.

Tason pistejoukon klusteroimiseen tarkoitettu DBCluC-algoritmi (Density-Based Clustering with Constraints) lisää DBSCAN-algoritmiin rajoitteiden olemassaolon. DBCluC-algoritmissa rajoitteet kuvataan tason monikulmioina. Tässä raportissa ei kuitenkaan käsitellä DBCluC-algoritmia lähdemateriaalin puutteellisuuden vuoksi: ainakin yhdessä lähdeartikkelissa [ZL02a] on selviä ristiriitoja ja puutteita, jotka näyttävät olevan ratkaistuja vasta viimeisimmässä, vielä julkaisemattomassa versiossa [ZL02b].

Ester et al. ovat myös itse laajentaneet DBSCAN-algoritmiaan. Tuloksena on yleistetty GDBSCAN-algoritmi (Generalised DBSCAN) [SEKX98], jossa datapisteiden ei oleteta olevan ainoastaan pisteitä vaan esimerkiksi monikulmioita. Lisäksi tiheyttä ei käsitellä yksinkertaisena pisteiden lukumääränä tietyn pisteen palloympäristössä. Datapisteiden naapuruus voidaan määritellä mielivaltaisella symmetrisellä predikaatilla (esimerkiksi monikulmioiden voidaan sanoa olevan naapureita, jos ne leikkaavat toisiaan). Lisäksi datapisteille voidaan antaa eri intensiteettejä, joiden perusteella pistejoukon tiheyttä mitataan.

Tiheysperustaisessa klusteroinnissa datapisteen naapureiden hakeminen ja käsittely voi olla laskennallisesti vaativaa. Lisäksi haku joudutaan yleensä tekemään useita kertoja, vaikka varsinaista klusterointialgoritmia iteroitaisiin ainoastaan kerran. Tämän takia tiheysperustaisen klusterointialgoritmin yksi iteraatio voi vaatia huomattavasti enemmän aikaa kuin vastaava iteraatio esimerkiksi k:n keskipisteen menetelmällä. Toisaalta tiheysperustainen klusterointi vaatii tavallisesti ainoastaan yhden iteraation, kun vastaavasti k:n keskipisteen menetelmässä tarvitaan useita iteraatioita kriteeriofunktion (lokaalin) maksimin saavuttamiseen.

Pahimmassa tapauksessa monet tiheysperustaiset klusterointimenetelmät vaativat $\Omega(n^2)$ ajan, jossa n on datapisteiden lukumäärä [CSX02] (tämä tilanne kierrettiin DBSCAN-algoritmin analyysissä sopivilla oletuksilla). Tällainen tilanne ilmaantuu, jos datapisteiden jakauma on epäsuotuisa (esimerkiksi erittäin tiheä). Chen, Smid ja Xu esittävät artikkelissaan [CSX02] approksimointialgoritmin, jonka aikavaatimus $O(n \log n)$ on riippumaton datapisteiden jakaumasta. Approksimointialgoritmi perustuu niinsanottuun ε-sumeaan etäisyysmittaan, jonka avulla osa datapisteen naapureista voidaan karsia käsittelystä. Kirjoittajat ovat testanneet approksimointialgoritmiaan useilla "luonnollisia" klustereita sisältävillä pistejoukoilla ja todenneet sen käytännössä hyvin tehokkaaksi. Approksimointi on järkevää, sillä useissa käytännön tilanteissa syöteaineistot ovat epätarkkoja tai nopeasti laskettavissa oleva approksimoitu klusterointiratkaisu on riittävä [CSX02].

Lisäksi Chen, Smid ja Xu esittävät julkaisussaan myös kaksi determinististä, aikavaatimuksiltaan lievästi alineliöllistä ($o(n^2)$) tiheysklusterointialgoritmia. Kyseiset deterministiset algoritmit ovat kuitenkin mielenkiintoisia lähinnä teoreettisesta näkökulmasta.

Dash, Liu ja Xu tarkastelevat Chenin, Smidin ja Xun tapaan edelläkäsiteltyä DBSCAN-algoritmin potentiaalista vaativuutta. Heidän esittämässään BRIDGE-algoritmissa [DLX01] on yhdistetty k:n keskipisteen menetelmä sekä DBSCAN-algoritmi. Motivaationa yhdistämisessä on käyttää toisen menetelmän hyviä puolia kompensoimaan toisen huonoja puolia. Siinä missä k:n keskipisteen menetelmä on hyvin herkkä kohinalle aineistossa, kohina ei vaikuta oleellisesti DBSCAN-algoritmin toimintaan. Toisaalta DBSCAN-algoritmin potentiaalisesti paljon aikaa vaativa iteraatio voidaan kiertää k:n keskipisteen menetelmän aina lineaarisen ajan (datapisteiden lukumäärän suhteen) vaativalla iteraatiolla.

Lyhyesti kuvailtuna BRIDGE toimii seuraavasti. Aluksi pistejoukko klusteroidaan k:n keskipisteen menetelmällä. Parametrina annettava k saadaan siis käyttäjältä. Dash, Liu ja Xu perustelevat tätä sillä, että heidän mielestään käyttäjän on helpompi arvioida yksittäinen parametri k (klusterien määrä) kuin DBSCAN-algoritmin parametrit ε ja μ. Tämän jälkeen saatua klusterointia parannetaan etsimällä siitä kohinapisteitä DBSCAN-algoritmilla. DBSCAN-algoritmin parametreista ε on kiinnitetty ja μ arvioidaan k:n keskipisteen menetelmän avulla. Lisäksi osa DBSCAN-algoritmin vaatimista naapuruustarkasteluista suoritetaan jo k:n keskipisteen menetelmän suorituksen aikana. Tämä on toinen motivaatio suoritusjärjestykselle. Tehokkuuslisäys DBSCAN-algoritmin suoritukseen tulee siitä, ettei algoritmilla käsitellä kuin osa datapisteistä.

Dash et al.:n mukaan BRIDGE soveltuu samankaltaisten pistejoukkojen klusterointiin kuin k:n keskipisteen menetelmä. DBSCAN-algoritmin käyttäminen vähentää kohinan aiheuttamaa virhettä klusterointiin. Lisäksi tiheysperustaisen klusteroinnin käyttäminen voi korjata väärän k-parametrin käytöstä syntyviä virheitä. Sen sijaan sellaisiin pistejoukkoihin, joista k:n keskipisteen menetelmä ei yleensäkään löydä hyviä klustereita (esimerkiksi ei-konvekseja klustereita sisältävät pistejoukot), BRIDGEkään ei tavallisesti sovellu.

3 Yhteenveto

Tiheyteen perustuva klusterointi on erityisesti tilaa kuvaavan datan klusterointiin sopiva menetelmä, johon perinteiset klusterointialgoritmit sopivat usein huonosti. Tilaa kuvaavan datan klusterit voivat olla mielivaltaisen muotoisia. Käytettävän klusterointimenetelmän tulee ottaa tämä huomioon. Lisäksi klusterointimenetelmän täytyy pystyä käsittelemään kohinaa datassa. Menetelmälle annettavien parametrien määrän tulee olla pieni, sillä datan rakennetta ei tavallisesti tunneta.

Tiheyteen perustuvat klusterointimenetelmät jakavat datajoukon pisteet yhtenäisiin komponentteihin. Yhtenäisyyden määrittely on tietenkin tapauskohtaista. Tämän ominaisuuden avulla tiheyteen perustuvat klusterointialgoritmit pystyvät löytämään mielivaltaisen muotoisia klustereita ja sopivat siten tilaa kuvaavan datan klusterointiin. Ne pystyvät käsittelemään myös kohinaista aineistoa suhteellisen hyvin. Tiheysperustaiset klusterointimenetelmät eivät välttämättä tarvitse tietoa klustereiden lukumäärästä etukäteen. Pääasiassa nämä seikat erottavat tiheysperustaiset menetelmät muista klusterointimenetelmistä, esimerkiksi niinsanotuista k-optimointialgoritmeista.

Tässä raportissa esitelty DBSCAN-algoritmi on yksi lukuisista kirjallisuudessa esitetyistä tiheyteen perustuvista klusterointialgoritmeista. Se täyttää edellä kuvatut vaatimukset ja on laskennallisesti riittävän tehokas käytettäväksi suurten data-aineistojen klusteroimiseen. DBSCAN-algoritmissa ei ole erityistä optimointiongelmaa: klusterit ovat lähes yksikäsitteisesti määriteltyjä ja helposti löydettävissä, kun algoritmin parametrit ovat kiinnitettyjä. Lisäksi DBSCAN-algoritmia voidaan käyttää hierarkkiseen klusterointiin liu'uttamalla käytettäviä parametreja.

DBSCAN-algoritmi on sittemmin laajennettu GDBSCAN-algoritmiksi, jossa datajoukossa voi olla pisteiden lisäksi monikulmioita sekä näillä eri intensiteettejä. Toinen laajennus on hiljattain julkaistu DBCluC, joka lisää DBSCAN-algoritmiin rajoitteiden olemassaolon. BRIDGE-algoritmi yhdistää tunnetun k:n keskipisteen menetelmän ja DBSCAN-algoritmin. Yhdistämisellä pyritään kompensoimaan toisen algoritmin huonoja puolia toisen hyvillä puolilla. Erityisesti yhdistäminen auttaa vähentämään kohinan aiheuttamaa virhettä k:n keskipisteen menetelmässä ja korjaamaan väärän k:n arvon valinnasta johtuvia ongelmia. Muista tiheyteen perustuvista algoritmeista voidaan mainita esimerkiksi OPTICS [ABKS99] ja DBCLASD [XEKS98]. Hanin ja Kamberin kirjasta [HK01] löytyy hyvä yleisesitys tiheysperustaisesta klusteroinnista.

Viitteet

[ABKS99] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander. OPTICS: Ordering points to identify clustering structure. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 49–60, 1999.

[Ber02] P. Berkhin. Survey of clustering data mining techniques. http://www.accrue.com/products/rp_cluster_review.pdf [16.10.2002], 2002.

[BKSS90] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R^*-tree: An effecient and robust access method for points and rectangles. In *Proceedings of the International Conference on Management of Data and Symposium on Principles of Database Systems*, pages 322–331, 1990.

[CN00] N. Colossi and M. Nascimento. Benchmarking access structures for high-dimensional multimedia data. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1215–1218, 2000.

[CSX02] D. Chen, M. Smid, and B. Xu. Geometric algorithms for density-based data clustering. In *Proceedings of the 10th Annual European Symposium on Algorithms*, pages 284–296, 2002.

[DLX01] M. Dash, H. Liu, and X. Xu. '1+1 > 2': Merging distance and density based clustering. In *Proceedings of the 7th International Conference on Database Systems for Advanced Applications*, pages 32–39, 2001.

[EC02] V. Estivill-Castro. Why so many clustering algorithms: a position paper. *ACM SIGKDD Explorations Newsletter*, 4:65–75, 2002.

[EKSX96] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[Fas99] Daniel Fasulo. An analysis of recent work on clustering algorithms. Technical Report 01-03-02, University of Washington, Department of Computer Science and Engineering, 1999.

[HK01] J. Han and M. Kamber. *Data Mining.* Morgan Kaufmann Publishers, San Francisco, 2001.

[SEKX98] J. Sander, M. Ester, H. Kriegel, and X. Xu. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery*, 2:169–194, 1998.

[WSB98] R. Weber, H. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in higher-dimensional spaces. In *Proceedings of the International Conference on Very Large Databases*, pages 194–205, 1998.

[XEKS98] X. Xu, M. Ester, H. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining large spatial databases. In *Proceedings of the 14th International Conference on Data Engineering*, pages 324–331, 1998.

[ZL02a] O. R. Zaïane and C. Lee. Clustering spatial data in the presence of obstacles: a density-based approach. In *Proceedings of the International Database Engineering and Applications Symposium*, pages 214–223, 2002.

[ZL02b] O. R. Zaïane and C. Lee. Clustering spatial data when facing physical constraints. In *Proceedings of the IEEE International Conference on Data Mining (in press)*, 2002.