BIRCH: suurten tietojoukkojen klusterointi

Kari Laasonen
Kari.Laasonen@cs.helsinki.fi

Klusterointimenetelmät-seminaari
Helsingin yliopisto, tietojenkäsittelytieteen laitos
Raportti C–2002–54, s. 118–126, marraskuu 2002

Tiivistelmä

BIRCH on klusterointialgoritmi, joka on tarkoitettu suurten tietomäärien käsittelyyn. Se pyrkii minimoimaan klusteroinnista aiheutuvat tiedonsiirräntäkustannukset ja ottamaan huomioon muistirajoitukset. BIRCH lukee aineiston läpi vain kerran ja muodostaa siitä piirrepuun, joka sisältää tiiviissä muodossa yhteenvedon aineistosta. Tämän jälkeen BIRCH suorittaa klusteroinnin soveltamalla hierarkkista klusterointia piirrepuun lehtiin. Raportissa käsitellään itse BIRCH-algoritmin lisäksi sen erilaisia muunnoksia ja yleistyksiä.

1 Johdanto

Moniin klusterointimenetelmiin liittyy oletuksia käytettävissä olevasta muistista ja suoritusajasta. Tavallisesti oletetaan, että käsiteltävä aineisto mahtuu muistiin ja että aineiston pisteet voidaan käydä läpi niin monta kertaa kuin tarvitaan.

Tarkastellaan esimerkiksi k:n keskiarvon menetelmää (k-MEANS). Siinä valitaan k klusterikeskusta ja jaetaan datapisteet sitten sellaisille klusterikeskuksille, että etäisyys pisteestä keskukseen minimoituu. Tätä prosessia iteroidaan, kunnes klusterikeskukset eivät enää muutu. Aineisto käydään siis läpi useita kertoja, sitä useammin mitä parempi klusterointi halutaan. Jos aineiston koko on suurempi kuin käytettävissä oleva muisti, voidaan olettaa, että prosessi kuluttaa runsaasti aikaa.

Suuria tietomääriä varten tarvitaan ilmeisesti uudentyyppinen klusterointialgoritmi, jossa tiedonsiirräntäkustannukset (I/O costs) ovat mahdollisimman pienet ja joka huomioi mahdolliset muistirajoitukset. Pieniin siirräntäkustannuksiin päästään lukemalla aineisto tietokannasta vain kerran tai ottamalla aineistosta satunnaisotos. Muistirajoitusten vuoksi koko aineiston ei oleteta mahtuvan kerralla muistiin. Näiden perusvaatimusten lisäksi algoritmien tulisi pyrkiä käyttämään tietokantajärjestelmiä mahdollisimman tehokkaasti [Ber02, s. 30].

Tämä raportti käsittelee BIRCH-algoritmia [ZRL96, ZRL97], joka täyttää nämä vaatimukset. BIRCH on lyhenne sanoista ”Balanced Iterative Reducing and Clustering using Hierarchies”, jotka kuvaavat hyvin sen toimintaa. BIRCH perustuu aineiston tiivistykseen, jossa alkuperäisestä aineistosta muodostetaan kompakteja yhteenvetoja, klusterointipiirteitä. Jokainen klusterointipiirre edustaa useita aineiston pisteitä. Varsinainen klusterointi tehdään (hierarkkisesti) näille piirteille, joiden määrä on rajoitettu niin, että ne mahtuvat annettuun muistiin.

Luvussa 2 esitellään ensin tarvittavaa käsitteistöä. Klusterointipiirteet ja niiden tallennukseen käytettävä piirrepuu määritellään luvussa 3. Tämän jälkeen luvussa 4 käydään läpi itse BIRCH-algoritmin toiminta. Aineiston tiivistämisen ja suurten tietojoukkojen klusteroinnin kehitystä BIRCHin jälkeen käsitellään luvussa 5. Luku 6 on yhteenveto.

2 Klusterointiongelma

Aineisto, jota halutaan klusteroida, muodostuu riveistä, jotka sisältävät d attribuuttia. Kaikkien attribuuttien oletetaan olevan *metrisiä*, jolloin niiden välisten etäisyyksien laskeminen on mielekästä. (On mahdollista klusteroida myös kategorisia attribuutteja, mutta siihen tarvitaan muita algoritmeja.) Rivien voi ajatella olevan d-ulotteisen vektoriavaruuden *pisteitä*, jota termiä jatkossa käytetäänkin "rivin" asemesta.

Klusterointiongelma määritellään seuraavasti [ZRL97]:

> On annettu klustereiden lukumäärä k, aineisto (pisteiden lukumäärä on N) ja etäisyysmitta d. Halutaan löytää sellainen aineiston jako, jossa on k osaa ja joka minimoi mittafunktion d.
>
> Lisäksi käytössä oleva muisti on rajattu (tyypillisesti paljon pienemmäksi kuin aineiston koko). Tiedonsiirrantäkustannukset tulee minimoida.

BIRCH toteuttaa jälkimmäisen osan tiivistämällä aineiston muistiin mahtuvaksi piirrepuuksi. Puu muodostetaan piste kerrallaan, ja kun se on valmis, alkuperäistä aineistoa ei enää tarvita. Tietokanta käydään siis läpi vain kerran, joten ainakin vaatimus siirräntäkustannusten minimoinnista toteutuu.

2.1 Etäisyysmittoja

Tarkastellaan nyt klusteria, joka muodostuu metrisen d-ulotteisen avaruuden pisteistä. BIRCH perustuu siihen, että avaruus voidaan varustaa normilla; luvussa 5.2 nähdään, että tämä ei ole ehdoton vaatimus. Merkitään $\|\cdot\|$:lla tavallista euklidista normia: $\|a\|^2 = \sum_{i=1}^{d} a_i^2$.

Jos klusterissa x on n pistettä $\{x_i \mid i = 1, \ldots, n\}$, voidaan määritellä *keskiarvopiste* (centroid)

$$\bar{x} = \frac{1}{n} \sum_{1 \leqslant i \leqslant n} x_i. \tag{1}$$

Piste $\bar{x}$ ei välttämättä ole mikään aineiston pisteistä. On myös olemassa malleja, joissa keskiarvon asemesta tarkastellaan aineiston mediaanipistettä (medoid). Määritellään lisäksi klusterin *säde* R ja *halkaisija* D seuraavasti:

$$R = \left(\frac{1}{n} \sum_{1 \leqslant i \leqslant n} \|x_i - \bar{x}\|^2 \right)^{1/2}, \qquad D = \left(\frac{1}{n(n-1)} \sum_{1 \leqslant i,j \leqslant n} \|x_i - x_j\|^2 \right)^{1/2}. \tag{2}$$

Säde on siis pistejoukon keskihajonta ja halkaisija etäisyyksien neliökeskiarvo*.

Klusterin x etäisyys toiseen klusteriin y voidaan määritellä monella tavalla, ja BIRCH jättääkin tämän kysymyksen käyttäjän päätettäväksi. Mahdollisia etäisyysmittoja on lueteltu taulukossa 1. Mikään näistä mitoista ei liene muita "parempi", vaan erilaiset mitat sopivat erilaisille aineistoille ja klusterointitavoitteille. Alkuperäisessä artikkelissa [ZRL96] suositellaan mittoja D_2 ja D_4, koska ne soveltuvat parhaiten käytettäväksi BIRCHin osana olevan hierarkkisen klusterointialgoritmin kanssa.

*Lähteissä käytetään yleisesti termiä "pisteparien keskimääräinen etäisyys", vaikka D:n määritelmän mukaan aivan tästä ei olekaan kyse. Sama epätäsmällisyys esiintyy puhuttaessa etäisyysmitoista D_2 ja D_3.

ETÄISYYSMITTA	MÄÄRITELMÄ
Keskiarvopisteiden etäisyys	$D_0 = \|\bar{x} - \bar{y}\|$
Keskiarvopisteiden Manhattan-etäisyys ($x^{(i)}$ on vektorin x i:s komponentti)	$D_1 = \|\bar{x} - \bar{y}\|_1 = \sum_{1 \leqslant i \leqslant d} \lvert\bar{x}^{(i)} - \bar{y}^{(i)}\rvert$
Eri klustereiden pisteiden etäisyyksien neliökeskiarvo	$D_2 = \left(\frac{1}{\lvert x\rvert \lvert y\rvert} \sum_{\substack{1 \leqslant i \leqslant \lvert x\rvert \\ 1 \leqslant j \leqslant \lvert y\rvert}} \|x_i - y_j\|^2 \right)^{1/2}$
Yhdistetyn klusterin $z = (x + y)$ pisteiden etäisyyksien neliökeskiarvo	$D_3 = \left(\frac{1}{\lvert z\rvert \big(\lvert z\rvert - 1\big)} \sum_{1 \leqslant i,j \leqslant \lvert z\rvert} \|z_i - z_j\|^2 \right)^{1/2}$
Klustereiden yhdistämisestä aiheutuva varianssin muutos (huomaa neliöinti)	$D_4^2 = \sum_{1 \leqslant i \leqslant \lvert x+y\rvert} \big\|(x+y)_i - \overline{(x+y)}\big\|^2 - \sum_{1 \leqslant i \leqslant \lvert x\rvert} \|x_i - \bar{x}\|^2 - \sum_{1 \leqslant i \leqslant \lvert y\rvert} \|y_i - \bar{y}\|^2$

Taulukko 1: Erilaisia tapoja määritellä kahden klusterin x ja y välinen etäisyys. Tässä $|x|$ on x:n sisältämien pisteiden lukumäärä ja $(x + y)$ yhdistetty klusteri.

3 Piirrepuu

Tietojoukko oletetaan klusterointiongelman mukaisesti niin suureksi, että yksittäisillä pisteillä operointi ei voisi tuottaa tehokasta klusterointia. Aineistoa täytyy tiivistää jollain tapaa. Tästä tiivistyksestä toivotaan tietysti jäävän jäljelle jotain, mikä mahdollistaa klusteroinnin tekemisen myöhemmin. Alkuperäistä aineistoa ei tarvita enää tiivistyksen jälkeen, joten päästään tavoitteeseen lukea aineisto tietokannasta vain kerran.

3.1 Klusterointipiirteet

BIRCH muodostaa datapisteistä *aliklustereita*, jotka ovat eräänlaisia klustereiden prototyyppejä. Niitä on tyypillisesti paljon enemmän kuin lopulta saatavia tulosklustereita. Jokaisesta aliklusterista tallennetaan muistiin *klusterointipiirre* (cluster feature): klusterin x, joka sisältää n pistettä $\{x_i \mid i = 1, \ldots, n\}$, klusterointipiirre on kolmikko

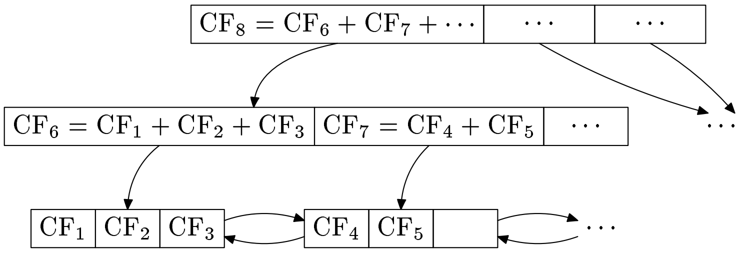
$$\mathbf{CF}(x) = \big(n, \mathrm{LS}_x, \mathrm{SS}_x\big),$$

jossa LS_x on pisteiden vektorisumma $\sum_{i=1}^{n} x_i$ ja SS_x on pisteiden neliösumma $\sum_{i=1}^{n} \|x_i\|^2$. Yksittäisten pisteiden koordinaatteja ei tallenneta.

Klusterointipiirre tiivistää n pistettä yhdeksi vektoriksi ja kahdeksi luvuksi. Näillä tiedoilla pärjätään itse asiassa varsin pitkään. Ensinnäkin huomataan heti, että yhdistettäessä kaksi aliklusteria saadaan yhdisteen klusterointipiirre laskemalla aliklustereiden piirteet yhteen:

$$\mathbf{CF}_1 + \mathbf{CF}_2 = \big(n_1 + n_2, \mathrm{LS}_1 + \mathrm{LS}_2, \mathrm{SS}_1 + \mathrm{SS}_2\big). \tag{3}$$

Tämä mahdollistaa sen, että aliklustereita voidaan koota askeltaen, piste kerrallaan. Toiseksi nähdään, että kahden aliklusterin välinen etäisyys käyttäen mitä tahansa edellä mainittua



Kuva 1: Piirrepuu. Solmujen klusterointipiirteet saadaan lapsisolmuista yhdistämällä.

etäisyysmittaa D_m voidaan laskea piirteiden avulla. Esimerkiksi

$$
\begin{aligned}
D_2(x,y)^2 &= \frac{1}{n_x n_y} \sum_{i,j} \|x_i - y_j\|^2 \\
&= \frac{1}{n_x n_y} \sum_{i,j} \Big(\|x_i\|^2 + \|y_j\|^2 - 2x_i \cdot y_j \Big) \\
&= \frac{1}{n_x n_y} \Big(n_y \mathrm{SS}_x + n_x \mathrm{SS}_y - 2\mathrm{LS}_x \cdot \mathrm{LS}_y \Big) \\
&= \frac{\mathrm{SS}_x}{n_x} + \frac{\mathrm{SS}_y}{n_y} - \frac{2}{n_x n_y} \mathrm{LS}_x \cdot \mathrm{LS}_y .
\end{aligned}
$$

Etäisyyksien laskemisen kannalta aliklusterista ei tarvitakaan tämän enempää tietoa, joten klusterointipiirrettä voidaan pitää riittävänä tilastollisena yhteenvetona aliklusterin pisteistä.

3.2 Piirrepuu

Piirrepuu (CF tree) on tasapainotettu hakupuu, joka muistuttaa B^+-puuta (kuva 1). Jokaiseen puun solmuun talletetaan joukko klusterointipiirteitä ja osoittimia lapsisolmuihin. Kukin piirre kuvaa aliklusteria, joka liittyy kyseiseen lapseen. Siirryttäessä puussa ylöspäin saadaan yhdisteklustereiden piirteet laskettua helposti solmun sisältämistä piirteistä kaavan (3) mukaan. Puun lehtisolmuissa on vain joukko klusterointipiirteitä.

Ainoastaan puun lehtiä käytetään klusteroinnissa, ja muita solmuja tarvitaankin vain sopivan lehtisolmun löytämiseen. Lehtisolmut on yhdistetty toisiinsa, joten niitä voidaan käsitellä helposti yhdessä.

Piirrepuuhun liittyy kaksi käyttäjän valittavissa olevaa parametria. Sivukoko P määrää puun solmun koon tavuina. Se ja aineiston ulottuvuuksien lukumäärä d kertovat, kuinka monta alkiota puun solmuihin sopii. Parametrin P merkitys ei ehkä ole niin suuri kuin B^+-puiden tapauksessa, koska pyrkimyksenä on pitää koko piirrepuu muistissa (eikä levyllä). Toinen parametri on lehtien kynnysarvo T: jokaisen lehtisolmun piirteen halkaisijan D (tai vaihtoehtoisesti säteen R) täytyy toteuttaa ehto $D < T$. Mitä suurempi T on, sitä enemmän pisteitä yhteen aliklusteriin mahtuu, ja vastaavasti piirrepuun koko on pienempi.

3.3 Piirrepuun muodostaminen

Datapisteen x lisääminen piirrepuuhun tapahtuu lähes kuten avaimen lisäys B^+-puuhun:

1. Lähdetään etenemään puun juuresta lehtiin päin. Kussakin solmussa valitaan se lapsi, jota vastaava aliklusteri on lähinnä pistettä x. Etäisyysmittana käytetään jotain metriikkaa D_m.

2. Kun tullaan lehteen, etsitään taas lähin solmun alkio L_i. Jos x:n voi sisällyttää aliklusteriin L_i rikkomatta kynnysarvoehtoa, niin tehdään se. Muussa tapauksessa lisätään lehteen uusi alkio, joka sisältää uuden pisteen x. Jos tämä alkio ei enää sovi lehteen, lehtisolmu jaetaan kahteen osaan valitsemalla ne kaksi alkiota, jotka ovat kauimpina toisistaan. Nämä alkiot laitetaan eri solmuihin ja loput alkiot jaetaan näihin kahteen solmuun etäisyyden perusteella.

3. Kun piste on lisätty johonkin lehteen, palataan puussa ylöspäin päivittäen aliklustereita. Jos lehtisolmu jaettiin, seuraavalle tasolle pitää lisätä uusi alkio, mikä taas näkyy yhtä korkeammalle tasolle. Pahimmassa tapauksessa jakoja tehdään puun juureen asti.

Puun muodostuksessa on muutamia erikoistapauksia, jotka täytyy ottaa huomioon [ZRL97]. Sivukoko voi aiheuttaa ongelmia: aliklusterit, jotka voisi liittää yhteen, joutuvat eri solmuihin (jolloin ne tietysti pysyvät erillisinä); myös päinvastainen tapaus on ongelmallinen. Voi myös käydä niin, että jos sama piste esiintyy aineistossa eri kohdissa, se voi joutua eri kerroilla eri aliklustereihin.

BIRCH yrittää myös pitää koko piirrepuun muistissa. Vaikka puu tiivistää aineistoa huomattavasti, voi silti käydä niin, että puu tarvitsee enemmän muistia kuin mitä on käytettävissä. Koska puun koko riippuu kynnysparametrista T, niin on selvää, että T pitäisi osata valita "sopivasti" tai ainakin jotain pitää tehdä puun kasvettua yli sille varatun muistin. Ongelman ratkaisu on aloittaa hyvin pienellä T:n arvolla (oletusarvo on 0). Jos puu kasvaa liian suureksi, on mahdollista kasvattaa T:tä ja rakentaa puu uudestaan käyttämättä liikaa muistia. Uuden T:n arvon valintaan ei kuitenkaan ole olemassa algoritmia, vaan BIRCH käyttää heuristiikkaa, jonka pitäisi uudelleenrakennettaessa puolittaa muistissa olevan puun koko.

Kuinka paljon puun rakentamiseen kuluu aikaa? Hieman yllättäen aikavaativuus on pisteiden määrän N suhteen lineaarinen. Tämä johtuu siitä, että puuhun ei talleteta kaikkia aineiston pisteitä. Puun sisältämien solmujen lukumäärä ja siten sen korkeus on käytettävissä olevan muistin (eikä aineiston koon) funktio. Aikavaativuus ei siis olekaan muotoa $N \log N$.

Oletetaan, että käytössä on M tavua muistia. Tällöin piirrepuussa on enintään M/P solmua. Jos yhteen solmuun sopii b alkiota, on puun korkeus noin $h = 1 + \log_b M/P$. Lisättäessä pistettä x siirrytään puun juuresta kohti lehteä, ja reitin varrella haetaan jokaisesta solmusta x:ää lähin alkio. Jokainen testaus operoi piirteillä eli vie ajan $O(d)$. Yhden pisteen lisääminen sujuu siis ajassa $O(hdb)$, joka on N:n suhteen vakio. Kaikkien pisteiden lisääminen vie siten ajan $O(N)$. Tämän lisäksi puun uudelleen rakentamisista tulee lisätermi, joka on kuitenkin N:n suhteen logaritminen, eikä siis muuta kokonaisaikavaativuutta.

4 BIRCH-algoritmi

Kuten on nähty, suurikin aineisto voidaan lineaarisessa ajassa tiivistää muistiin mahtuvaksi piirrepuuksi. On mielenkiintoista huomata, että mitään päätöksiä tai toimenpiteitä itse klusteroinnin suhteen ei ole vielä tehty: aineiston pisteet on vain alustavasti luokiteltu piirteiksi (eli aliklustereiksi). Se miten BIRCH tästä jatkaa on vain yksi mahdollinen tapa; muita vaihtoehtoja käsitellään seuraavassa luvussa.

BIRCH-algoritmi koostuu neljästä vaiheesta (taulukko 2). Näistä toinen ja neljäs vaihe ovat valinnaisia. Oleelliset toimenpiteet ovat siis piirrepuun muodostaminen ja piirteiden klusterointi: käsitellään näitä tarkemmin.

Kuten aiemmin todettiin, BIRCH aloittaa piirrepuun rakentamisen hyvin pienellä kynnysarvolla T. Jos puu kasvaa liian suureksi, tarkkuutta voidaan heikentää keräämällä enemmän pisteitä aliklustereihin. Tämä tehdään kasvattamalla T:tä ja siirtämällä vanhan piirrepuun

VAIHE	KUVAUS	TULOS
1	Luetaan aineisto ja rakennetaan piirrepuu	Piirrepuu
2	Tiivistetään piirrepuu haluttuun kokoon	Pienempi piirrepuu
3	Muodostetaan piirteistä klustereita	Klusterointi
4	Parannetaan klusterointia iteroimalla	Parempi klusterointi

Taulukko 2: BIRCH-algoritmin vaiheet

lehdet uuteen puuhun. Tässä vaiheessa voidaan puusta myös siivota poikkeavia havaintoja (outlier) eli pisteitä, jotka ovat kaukana varsinaisista klustereista. Poikkeukset tunnistaa siitä, että vastaava klusterointipiirre sisältää "paljon vähemmän" (lähteen [ZRL97] käyttämä heuristiikka) pisteitä kuin piirteet keskimäärin. Mahdolliset poikkeavat tapaukset kerätään listaan pois piirrepuusta. Kun koko aineisto on käsitelty, saatu lista yritetään integroida uudestaan piirrepuuhun. Tässä vaiheessa pisteet, jotka ovat vieläkin kaukana muista aliklustereista, ovat varmasti poikkeavia ja voidaan siis poistaa.

Poikkeavien tapausten suodattaminen pois aineistosta voi parantaa klusterointivaiheen tarkkuutta. Piirrepuun käytöllä on tämän ja tiivistyksen lisäksi se vaikutus, että se kerää lähellä toisiaan olevat pisteet läheisiin piirteisiin (tai saman piirteen alle). Tämä auttaa, jos klusterointialgoritmi on herkkä aineiston pisteiden järjestyksen suhteen.

4.1 Klusterointi

Ennen varsinaista klusterointia voidaan suorittaa vaihe 2, jossa piirrepuuta edelleen tiivistetään sellaiseksi, että klusterointialgoritmi selviytyy siitä ongelmitta. Vaiheessa 3 suoritetaan klusterointi piirrepuun lehdille. Periaatteessa piirteitä voitaisiin pitää tässä yksittäisinä pisteinä, mutta parempaan tulokseen päästään huomioimalla piirteessä oleva pisteiden lukumäärä n eräänlaisena painona. Voi myös olla mahdollista klusteroida itse piirteitä, jolloin etäisyydet lasketaan suoraan piirteen vektori- ja neliösummista.

Klusterointivaihe ei ole kovin mielenkiintoinen. BIRCHin kiinnostavuus liittyy oikeastaan vain siihen, miten suuresta aineistosta kerätään edustava yhteenveto klusterointialgoritmille. Valittu algoritmi on koostava (agglomeratiivinen) hierarkkinen klusterointi [Ols95]. Valinta riippuu ilmeisestikin käyttötarkoituksesta (kuten muutkin parametrit), ja joissain tapauksissa esimerkiksi jokin etäisyyteen perustuva menetelmä voisi olla parempi.

Hierarkkisessa klusteroinnissa [Leh02] yhdistetään lähekkäisiä pisteitä klustereiksi ja toistetaan tätä rekursiivisesti kunnes klustereita on jäljellä vain yksi. Lopulliset klusterit saadaan katkaisemalla syntyvä "klusteripuu" (dendrogrammi) sopivalta korkeudelta. Olsonin [Ols95] mukaan etäisyysmetriikat D_2 ja D_4 mahdollistavat hierarkkisen klusteroinnin ajassa $O(n^2)$. Tässä n on aliklustereiden lukumäärä, joka on huomattavasti pienempi kuin pisteiden lukumäärä N, joten BIRCHin voi edelleen väittää olevan lineaarinen N:n suhteen. Nähdään kuitenkin, että vaiheessa 2 tehtävä puun koon pienennys voi olla tarpeellinen, jotta klusterointialgoritmi pystyy suoriutumaan tehtävästään kohtuullisessa ajassa.

Klusterointivaihe näyttäisi itse asiassa muistuttavan piirrepuun rakennusta: kummassakin vaiheessa muodostetaan aineistosta hierarkkinen malli. Tämä antaa aiheen kysyä, mihin klusterointia BIRCHissä tarvitaan? Riittäisikö pelkkä piirrepuun muodostus? Vastaus on kielteinen. Piirrepuun solmuun mahtuu tietty kiinteä määrä alkioita, joten yksittäinen solmu ei aina vastaa hyvin aineiston klustereita. Kuten aliluvussa 3.3 mainittiin, yhdistettävät piirteet voivat joutua eri solmuihin tai eroavat piirteet voivat päätyä samaan solmuun. Ehkä olisi myös vaikeampaa valita hyvä kynnysparametrin T arvo kuin klustereiden lukumäärä.

Neljännessä vaiheessa yritetään parantaa vaiheen 3 suorittamaa klusterointia. Aloitetaan laskemalla jo saatujen klustereiden keskiarvopisteet. Jaetaan sitten pisteet uudestaan liittämällä kukin piste sitä lähinnä olevaan keskiarvopisteeseen. Tämä on tietysti tuttu k:n keskiarvon menetelmä, mutta etäisyysmitta voi olla jotain muuta kuin euklidinen etäisyys klusterin keskipisteeseen. Vaihe 4 on tarpeellinen myös silloin kun halutaan tietää, mihin klusteriin kukin piste kuuluu; hierarkkinen klusterointi ei tätä tietoa tarjoa.

5 BIRCHin jälkeen

BIRCH oli ilmeisesti ensimmäinen klusterointialgoritmi, joka huomioi suurten tietomäärien mukanaan tuomat vaatimukset. Alkuperäinen artikkeli [ZRL96] julkaistiin vuonna 1996. Tämän jälkeen ilmestyneiden artikkelien ja viittausten lukumäärän perustella voidaan päätellä BIRCHin olevan hyvin tunnettu.

Menetelmää on myös kritisoitu ja muutamia puutteita on löydetty:

- Kynnysarvon T alkuarvon valinta ja T:n kasvattaminen puun uudelleenkonstruoinnin yhteydessä sekä poikkeavien havaintojen seulonta tapahtuvat heuristisesti ilman kovin vakuuttavia perusteluja.
- Hierarkkinen klusterointi toimii neliöllisessä ajassa piirteiden lukumäärän suhteen, mikä ei täysin vastaa lupausta lineaarisesta suoritusajasta. BIRCH lukee alkuperäisen aineiston vain kerran, joten tässä suhteessa toki päästään tavoitteeseen pitää *siirräntäkustannukset* lineaarisina.
- Klusteroinnin hyvyydestä on vaikea sanoa mitään, koska juuri se osa systeemistä on mitä suurimmassa määrin käyttäjän määriteltävissä. Voi tietysti ajatella, että ”hyvä” klusterointi on samankaltainen kuin tavallisessa hierarkkisessa klusteroinnissa, mutta tiivistyksen vuoksi tilanne ei ole aivan sama. Myös etäisyysmitta jätetään käyttäjän valittavaksi.
- Vaiheessa 4 käytetyllä k:n keskiarvon menetelmällä saadaan konvekseja samankokoisia klustereita. Mitä tapahtuu, jos yhden ison klusterin lähellä on pienempi? Osa ison klusterin pisteistä on lähempänä pienen klusterin keskipistettä, joten ne päätyvät ”väärään” klusteriin. Myös BIRCHin käyttämä hierarkkinen klusterointi suosii konvekseja klustereita.

Heuristiikoista ja lukuisista valinnoista johtuen (tai niiden ansiosta) BIRCH mukautuu hyvinkin erilaisiin tarkoituksiin. Tästä syystä algoritmia ja sen ominaisuuksia on vaikea analysoida matemaattisesti, eikä sitä juuri olekaan tehty.

5.1 Muita klusterointitapoja

On selkeästi nähtävissä, että BIRCHin tärkein kontribuutio on esiklusterointi, eli aineiston tiivistys piirrepuuksi. Varsinaista klusterointia (vaiheet 3 ja 4) on pyritty myöhemmissä artikkeleissa parantamaan tai se on korvattu kokonaan jollain muulla menetelmällä.

Esimerkiksi käy klusterointialgoritmi BRIDGE [DLX01], jossa on yhdistetty etäisyyteen perustuva ja tiheysklusterointi vuorottelemalla k-MEANS- ja DBSCAN-algoritmeja [Ber02]. Tiheysklusteroinnissa yksi tehtävistä on löytää joukko tietyn pisteen ε-säteisen ympäristön pisteitä. Tähän tarkoitukseen voi käyttää R*-puuta, joka on eräs B^+-puuta muistuttava spatiaalinen tietorakenne. Osoittautuu, että BIRCHin piirrepuulla voi korvata R*-puun ja tuloksena on tehokkaampi ja laajoja aineistoja käsittelemään pystyvä systeemi.

Breunig, Kriegel ja Sander [BKS00] tarkastelevat lähemmin BIRCHin klusterointiprosessia. Heidän argumentaationsa lähtee siitä, että etäisyyteen perustuva klusterointi on helppo tehdä piirrepuun sisältämästä informaatiosta. Näin asia tietysti on: piirteen $\mathbf{CF} = (n, \mathrm{LS}, \mathrm{SS})$ keskiarvopiste on LS/n, jota voidaan pitää käsitellyn aineiston pisteenä painolla n. Tällainen strategia ei sovellu hierarkkiseen klusterointiin, jossa klustereille ei lasketa keskipisteitä. BIRCH kuitenkin käyttää hierarkkista klusterointia, joten näyttää, että etäisyyksiä on mahdollista laskea suoraan piirteistä ilman klusteroinnin laadun merkittävää heikkenemistä.

Breunigin ja kumppaneiden tulos osoittaa lähinnä sen, ettei piirteitä voi pitää vain n-kertaisina pisteinä. He tarkastelevat artikkelissaan algoritmia nimeltä OPTICS, joka on yhdistelmä tiheyteen perustuvaa ja hierarkkista klusterointia. Artikkelissa yhdistetään tähän BIRCH-tyyppinen aineiston tiivistys. Nähdään, että piirrepuu kelpaa käyttöön sellaisenaan. Klusterointivaiheen ongelmat vältetään määrittelemällä piirteiden välinen etäisyys huolellisesti. Voidaan määritellä *datakupla*, joka vastaa jotakuinkin piirrettä. Kupla määritellään kolmikolla (n, M, D), jossa n on kuplan pisteiden lukumäärä, M keskiarvopiste ja D kaavan (2) mukainen halkaisija. (Nämä suureet saadaan helposti laskettua piirteistä.) Kuplien etäisyys on nyt mahdollista määritellä sellaiseksi, että tiheysklusterointi onnistuu niillä hyvin.

Tulokset ovat kiinnostavia. Verrattuna OPTICS-algoritmiin ilman tiivistystä, päästään suoritusajoissa yli tuhatkertaiseen nopeuteen. Aineistoa voi tiivistää helposti kymmenesosaan ilman nähtävää klustereiden laadun heikkenemistä. Tiivistyssuhteella 95% on nähtävissä pientä eroa, ja laatu heikkenee selvästi vasta 99.5% jälkeen. (Aineisto oli tässä kokeessa isohko, n. 100 000 pistettä.)

5.2 Yleistyksiä

Eräs suunta, johon tiivistyksen ideaa on laajennettu, on *yleinen* metrinen avaruus. BIRCH olettaa, että pisteitä kuvataan vektoreilla, joiden välisiä etäisyyksiä voidaan laskea vektorin komponenteista. Entä jos tarkasteluavaruudessa on määritelty vain etäisyyden käsite? Etäisyys kahden pisteen välillä voi myös olla kallista laskea. Tällainen on esimerkiksi avaruus, jonka alkiot ovat merkkijonoja ja etäisyytenä käytetään muokkausetäisyyttä $d(s_1, s_2)$, joka kertoo, kuinka monta muokkausoperaatiota tarvitaan, että s_1:stä saadaan s_2.

On mahdollista määritellä klusterin *edustaja* (eräänlainen mediaanipiste) melko luontevasti myös tällaisissa tapauksissa [GRG+99]. Kuten arvata saattaa, vaikeuksia tulee vastaan klusteripiirteiden päivityksessä. Tämä johtuu siitä, että klusterin edustajapiste pitää valita uudestaan jokaisen pisteen lisäyksen jälkeen; sitä ei voi päivittää helposti askeltaen kuten vektoriavaruuden tapauksessa. Valinnan tekemiseen tarvitaan kuitenkin kaikkia jo nähtyjä pisteitä, joten koko aineiston täytyisi mahtua muistiin. Koska tätä ei voida vaatia, joudutaan päivityksessä turvautumaan heuristiikkoihin.

Toinen, yleisempi tapa tarkastella tiivistystä on *data squashing* [DVJ+99], joka perustuu siihen, että aineisto korvataan sellaisella uudella ja pienemmällä aineistolla, että probabilististen mallien parametrit (aineiston momentit) pysyvät likimäärin samoina. BIRCH toimii myös tällä periaatteella, mutta se käyttää vain kahta momenttia (pisteiden vektori- ja neliösumma), kun taas viitatussa artikkelissa lasketaan kaikki momentit jopa viidenteen kertalukuun asti.

Lopuksi on hyvä vielä mainita, ettei tiedon tiivistäminen BIRCHin tapaan ole ainoa tapa käsitellä suuria aineistoja. CURE [Ber02, s. 9] ottaa aineistosta satunnaisotoksen ja soveltaa sitten hierarkkista klusterointia.

6 Yhteenveto

BIRCH on algoritmi, joka soveltuu hyvin suurten tietojoukkojen klusterointiin. Se perustuu tiedon tiivistykseen, jossa aineistosta muodostetaan lineaarisessa ajassa yhteenveto. BIRCHin tapauksessa tämä yhteenveto on tasapainotettu hakupuu, joka sisältää klusterointipiirteitä. Varsinainen klusterointi suoritetaan näille piirteille.

Tiivistys ja piirrepuun käyttö on itse asiassa koko menetelmän keskeisin idea, jolla on ollut merkittävä vaikutus myöhempään suurten tietomäärien klusteroinnin tutkimukseen. BIRCHin klusterointivaihe on jäänyt vähäiselle huomiolle, eikä algoritmia olekaan juuri analysoitu kokonaisuutena. Klusteroinnin hyvyyden matemaattinen malli, jos sellaista onkaan, on nähty vähemmän tärkeäksi kuin tehokkaan menetelmän kehittäminen.

Viitteet

[Ber02] P. Berkhin. Survey of clustering data mining techniques. Research paper, Accrue Software, 2002. `http://www.accrue.com/products/rp_cluster_review.pdf`.

[BKS00] M. M. Breunig, H. Kriegel, and J. Sander. Fast hierarchical clustering based on compressed data and OPTICS. In *Proceedings of Principles of Data Mining and Knowledge Discovery (PKDD 2000)*, pages 232–242, Lyon, France, September 13–16 2000. Springer.

[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 (DASFAA 2001)*, pages 32–39, Hong Kong, April 18–20 2001.

[DVJ+99] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing flat files flatter. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 6–15, San Diego, California, USA, 1999. ACM Press.

[GRG+99] V. Ganti, R. Ramakrishnan, J. Gehrke, A. L. Powell, and J. C. French. Clustering large datasets in arbitrary metric spaces. In *15th International Conference on Data Engineering (ICDE)*, pages 502–511, Sydney, Australia, March 23–26 1999.

[Leh02] K. Lehmussaari. Hierarkkinen klusterointi. Teoksessa *Klusterointimenetelmätseminaari*, sivut 76–85. Raportti C-2002-54, Helsingin yliopisto, tietojenkäsittelytieteen laitos, marraskuu 2002.

[Ols95] C. F. Olson. Parallel algorithms for hierarchical clustering. *Parallel Computing*, 21:1313–1325, 1995.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, June 1996.

[ZRL97] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Journal of Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.