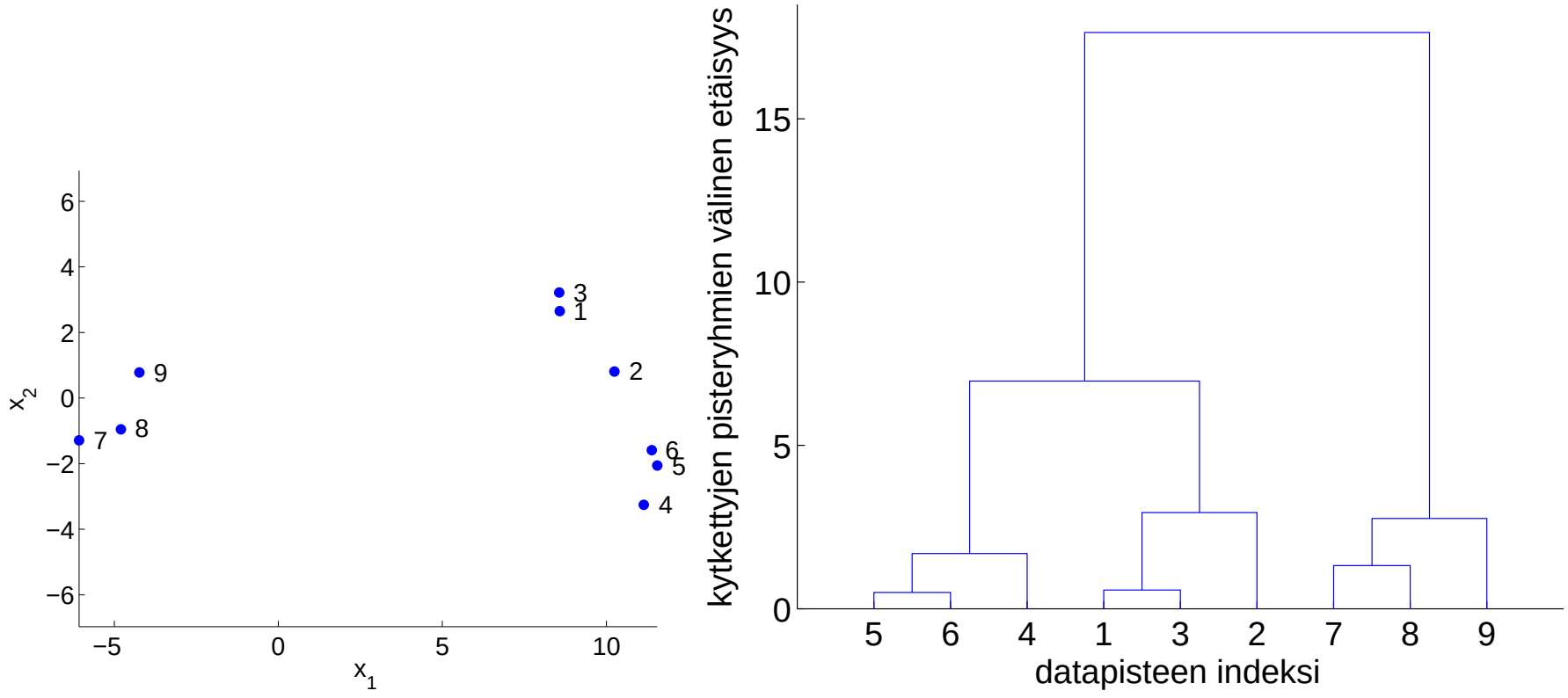
Luentorunko keskiviikolle 3.12.2008

- Hierarkkinen ryvästäminen

Ryvästyshierarkia & dendrogrammi

- Hierarkkinen ryvästäminen tuottaa yhden ryvästyksen sijasta sarjan ryvästyksiä
- Tulos voidaan visualisoida puurakenteena, *dendrogrammina*
- Dendrogrammi voi auttaa hahmottamaan aineiston rakennetta
- Dendrogrammista voidaan eristää erilaisia ryvästyksiä
- Dendrogrammi tarjoaa informaatiota sopivan klusterien määrän valitsemiseksi

Esimerkki: dendrogrammi



- Puun lehtinä kaikki datapisteet eli yhden pisteen pisteryhmät
- Alhaalta ylöspäin edetessä yhdistetään lähimmät pisteryhmät toisiinsa; tässä pisteryhmien välinen etäisyys ryhmien kauimmaisten pisteiden etäisyys
- Vaakaviivat sijaitsevat etäisyydellä, jolla kytketyt pisteryhmät yhdistettiin

Miten valita ryppäiden määrä K?

Hierarkkinen ryvästys ei vapauta meitä täysin ryppäiden määrän valinnan ongelmasta, mutta dendrogrammi auttaa tekemään informoidun valinnan.

- Leikataan etukäteen kiinnitetyllä etäisyystasolla d^*
- Leikataan kohdasta, jossa kahden peräkkäisen yhdistämisen etäisyyksien ero

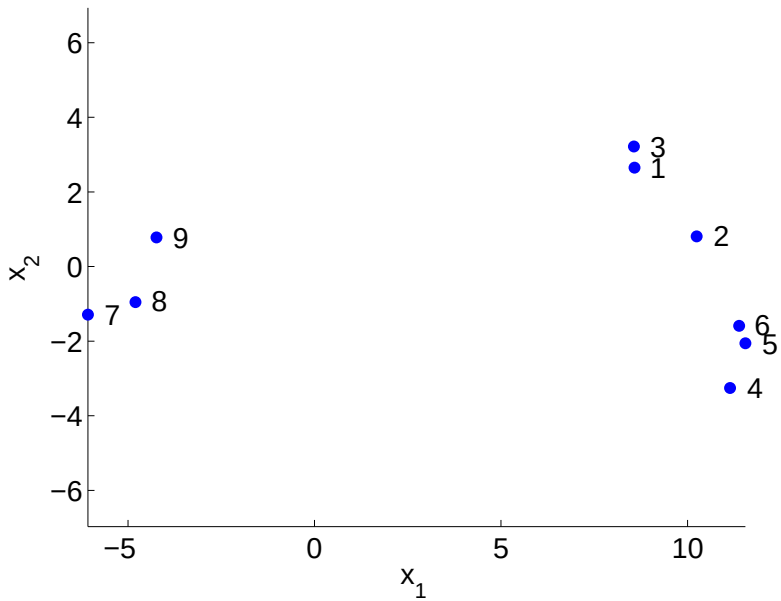
  $$d_{gap}(t) = d_{t+1} - d_t$$

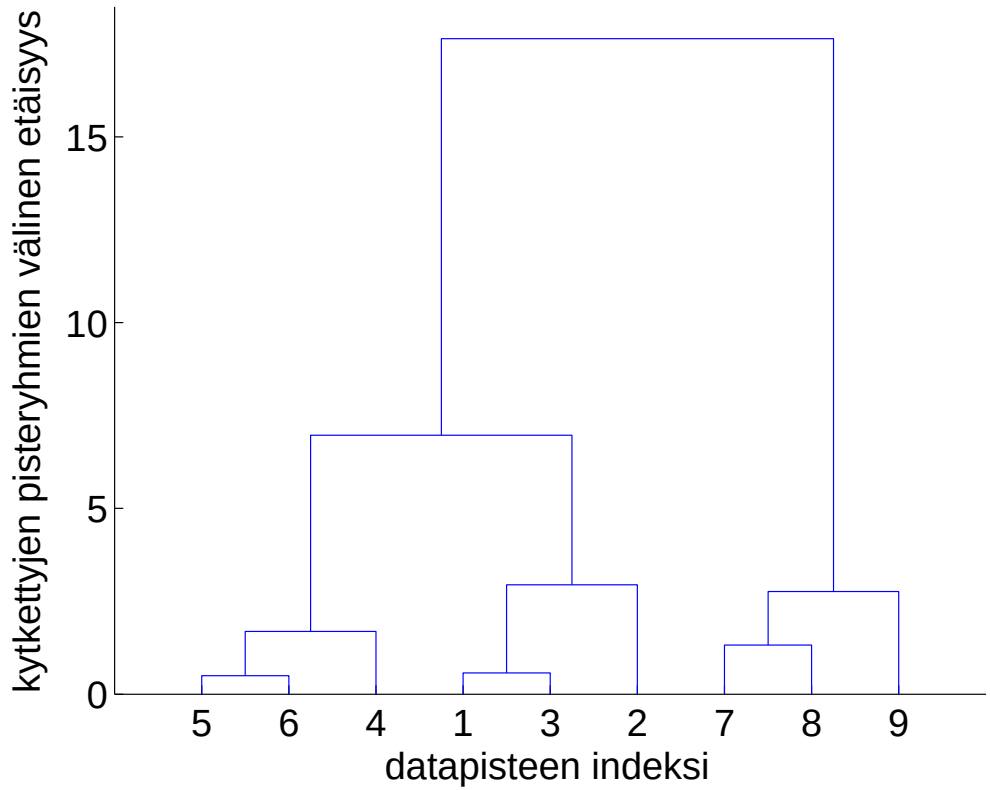
  on suurin.
- Voidaan luonnollisesti käyttää myös K:n keskiarvon ryvästyksen yhteydessä mainittuja kriteerejä (kyynärpääkriteeri, MDL, ...)

Esimerkki

Valitaan suurin yhdistämisetäisyys

- $d = 20 \implies K = 1$
- $d = 10 \implies K = 2$
- $d = 5 \implies K = 3$





Hierarkkiset ryvästämismenetelmät

Hierarkkinen ryvästysalgoritmi voi olla joko

- Koostava (agglomerative), jolloin algoritmi muodostaa vähittäin yhä suurempia ryppäitä liittämällä toisiinsa lähekkäin olevia ryppäitä
- Osittava (divisive), jolloin algoritmi muodostaa vähittäin yhä pienempiä osittamalla löyhiä ryppäitä kahteen osaan esim. K-means -algoritmilla ($K = 2$)

Koostava hierarkkinen ryvästäminen

- Koostavan hierarkkisen ryvästämisen alkutilanne on se, jossa kukin datapiste on oma ryppäänsä
- Ryppäiden yhdistäminen perustuu ryppäiden välisen etäisyyteen $d(C_i, C_j)$: joka vaiheessa toisiaan lähinnä olevat ryppäät yhdistetään
- Tavallisesti monotonisuusominaisuus on toivottava: $d_1 \leq d_2 \leq \ldots d_k$, missä d_t on askeleessa t yhdistettyjen ryppäiden etäisyys.

Algoritmi: koostava hierarkkinen ryvästäminen

% alusta kukin esimerkki omaan ryppääseensä
$C_i = \{x_i\}, i = 1, \ldots, n;$
% laske ryppäiden väliset etäisyydet taulukkoon D
$D_{ij} = d(C_i, C_j),\ i, j = 1, \ldots, n$
while vähintään kaksi rypästä jäljellä **do**
  % etsi kaksi toisiaan lähinnä olevaa rypästä
  $\{i', j'\} = argmin_{\{i,j|C_i,C_j \in \mathcal{C}\}} D_{ij};$
  % liitä rypäs $C_{j'}$ ryppääseen $C_{i'}$
  $C_{i'} = C_{i'} \cup C_{j'};\ \mathcal{C} = \mathcal{C} \setminus \{C_{j'}\};$
  % päivitä etäisyydet
  $D_{i',k} = d(C_{i'}, C_k), \forall C_k \in \mathcal{C}, C_k \neq C_{i'};$
end while

Ryppäiden etäisyysfunktiot (1/3)

Etäisyysfunktiota vaihtamalla saadaan seuraavat variantit:

- Lähinaapuriryvästys (nearest neighbor clustering, single-link clustering):

$$d(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} d(x_i, x_j)$$

- Kauimmaisen naapurin ryvästys (complete-link clustering):

$$d(C_i, C_j) = \max_{x_i \in C_i, x_j \in C_j} d(x_i, x_j)$$

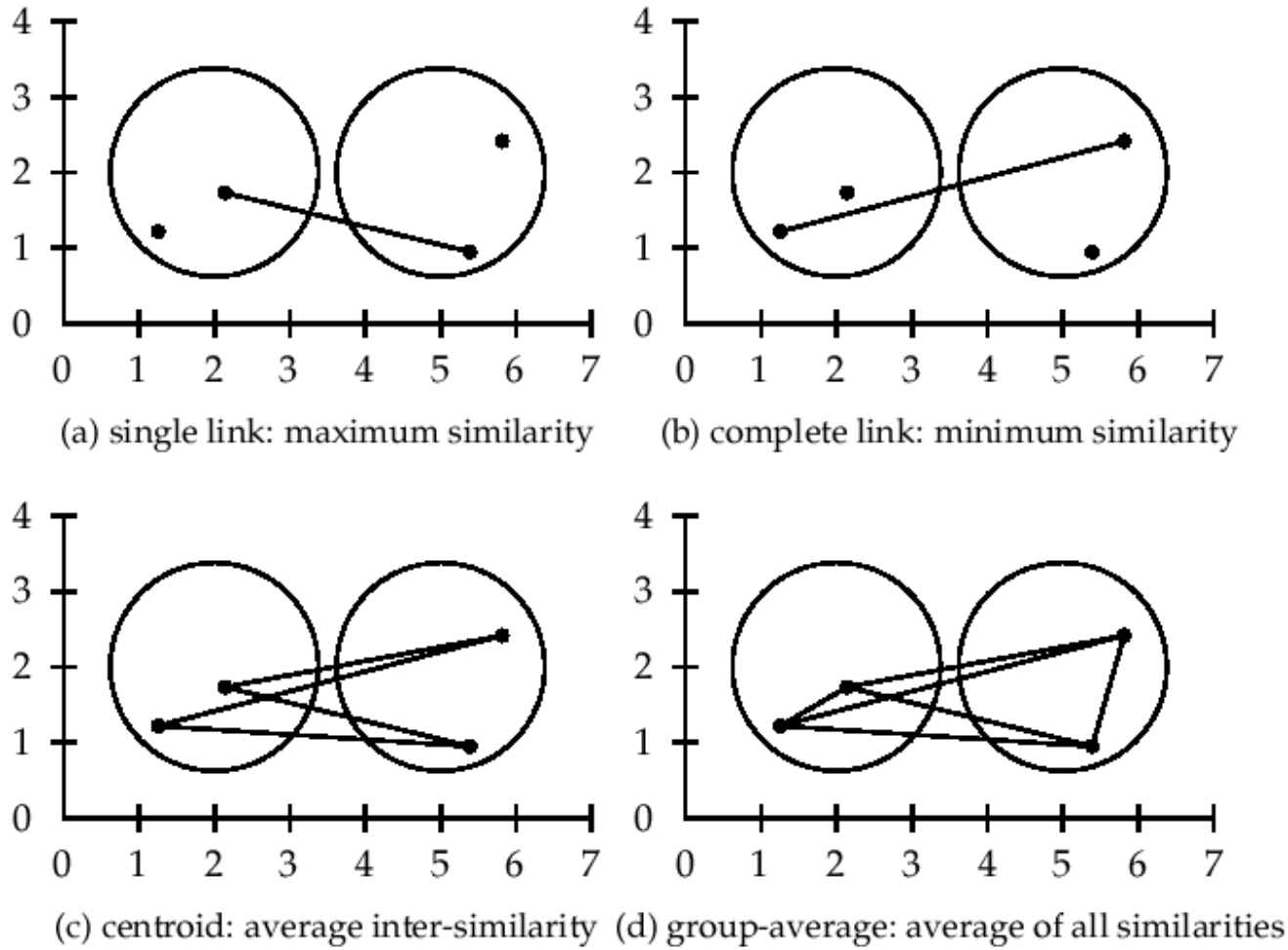
- Keskimääräisen etäisyyden ryvästys (average-link clustering):

$$d(C_i, C_j) = \frac{1}{(n_i + n_j)(n_i + n_j - 1)} \sum_{x_i, x_j \in C_i \cup C_j, x_i \neq x_j} d(x_i, x_j)$$

- Keskipisteryvästys (centroid clustering): $d(C_i, C_j) = d(r_i, r_j)$, missä r_i ryppään C_i edustajavektori

Ryppäiden etäisyysfunktiot (2/3)

Visualisoituna lähinaapuriryvästys (a), kauimmaisen naapurin ryvästys (b), keskipisteryvästys (c) ja keskimääräisen etäisyyden ryvästys (d)



(a) single link: maximum similarity (b) complete link: minimum similarity

(c) centroid: average inter-similarity (d) group-average: average of all similarities

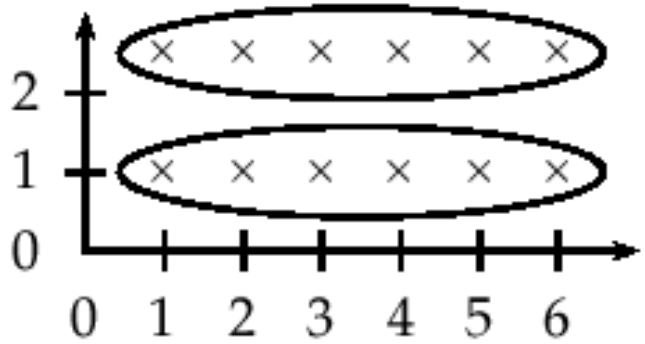
Ryppäiden etäisyysfunktiot (3/3)

Ryppäiden etäisyysfunktion valinta vaikuttaa ainakin seuraaviin ominaisuuksiin:

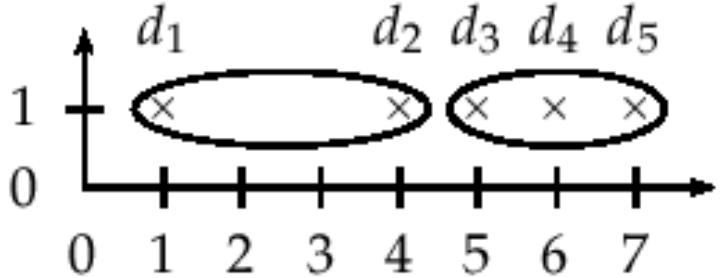
- Syntyvien ryppäiden muoto
- Implementaatio/aikavaativuus
- Monotonisuus yhdistämisetäisyyden suhteen
- Optimaalisuus

Ryppäiden muoto

- Lähinaapuriryvästyksellä on taipumus tuottaa ketjumaisia ryppäitä



- Kauimmaisen naapurin ryvästys tuottaa kompakteja ryppäitä, mutta on sensitiivinen yksittäisten kaukana olevien pisteiden (outlier) suhteen



- Keskimääräisen etäisyyden ryvästys ja keskipisteryvästys ovat näiden kahden ääripään välissä

Algoritmin aikavaativuus

1. Lähinnä toisiaan olevien ryppäiden liittäminen toisiinsa
   - while-silmukka suoritetaan tasan $n - 1$ kertaa, missä n on datapisteiden määrä
2. Toisiaan lähinnä olevien ryppäiden löytäminen:
   - triviaali ratkaisu on käydä ryppäiden pareittaisten etäisyyksien taulukko kokonaan läpi joka askeleessa, jolloin vertailujen määrä iteraatiota kohti $O(n^2)$
   - siis $O(n^3)$ etäisyysvertailua koko algoritmin suorituksen aikana
   - tämä voidaan pudottaa aikaan $O(n^2 \log n)$ tai $O(n^2)$ ryppäiden etäisyysfunktiosta riippuen
3. Etäisyyksien päivittäminen
   - Kun taulukoidaan ryppäiden pareittaiset etäisyydet, vain uuden ryppään etäisyydet vanhoihin täytyy päivittää ryppäiden liitosoperaation jälkeen
   - yhden iteraation aikana $O(n)$, koko suorituksen aikana $O(n^2)$ etäisyyslaskuoperaatiota.
   - operaatiot voidaan toteuttaa $O(1)$ ajassa

Yhdistettävän parin nopea etsintä

- Talletetaan kunkin ryppään etäisyydet muihin ryppäisiin prioriteettijonoon (esim. kekorakenne)
- $O(1)$ ajassa löydetään (keon päältä) tiettyä rypästä lähinnä oleva rypäs ja vastaava etäisyys
- $O(\log n)$ ajassa kunkin keon päivitys, yhteensä $O(n \log n)$ aika per iteraatio
- Koko algoritmin aikana kuluu aikaa $O(n^2 \log n)$

Parietäisyyksien nopea päivitys

Kun klusterit C_i ja C_j yhdistetään, kuinka klustereiden väliset etäisyydet päivitetään?

- yhdistäminen: $C_i = C_i \cup C_j$
- päivitetään etäisyydet $D_{ik}, k \neq j$
- jos etäisyydet D_{ik} ja D_{jk} on taulukoitu, päivitys voidaan tehdä $O(1)$ ajassa
  - Lähinaapuri: $D_{ik} = \min(D_{ik}, D_{jk})$
  - Kauimmainen naapuri: $D_{ik} = \max(D_{ik}, D_{jk})$
  - Keskimääräinen etäisyys:
    $D_{ik} = \frac{1}{p_{ijk}}(p_{ik} D_{ik} + p_{jk} D_{jk} + p_{ij} D_{ij} - p_i D_{ii} - p_j D_{jj} - p_k D_{kk})$,
    missä kertoimet p_k, p_{ik} ja p_{ijk} ovat ryppäiden C_k, $C_i \cup C_k$ ja $C_i \cup C_j \cup C_k$ sisältämien esimerkkiparien määrät.

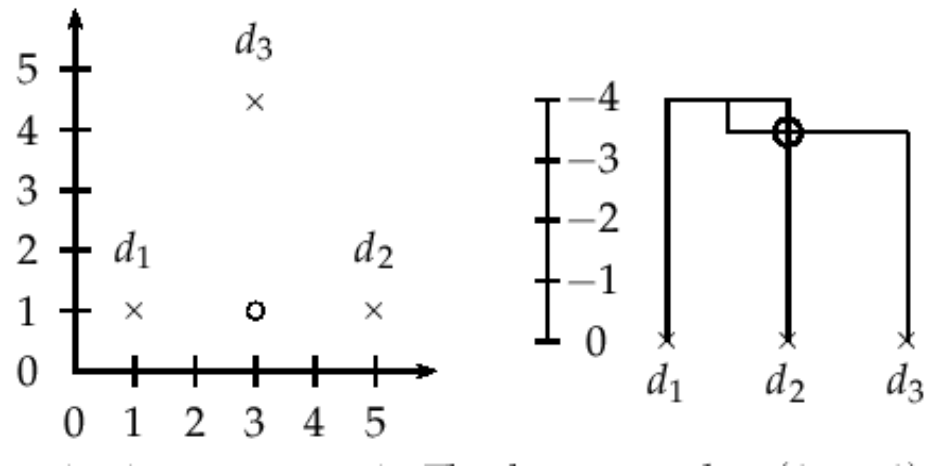
Entä keskipisteryvästys? Luennoija ei heti keksinyt $O(1)$ päivitystä, mutta sellainen voi silti olla olemassa...

Yhdistämisetäisyyden monotonisuus

- Hierakkinen ryvästysmenetelmä on monotoninen, jos algoritmin suorituksen aikana yhdistämisetäisyyksien jono on ei-vähenevä

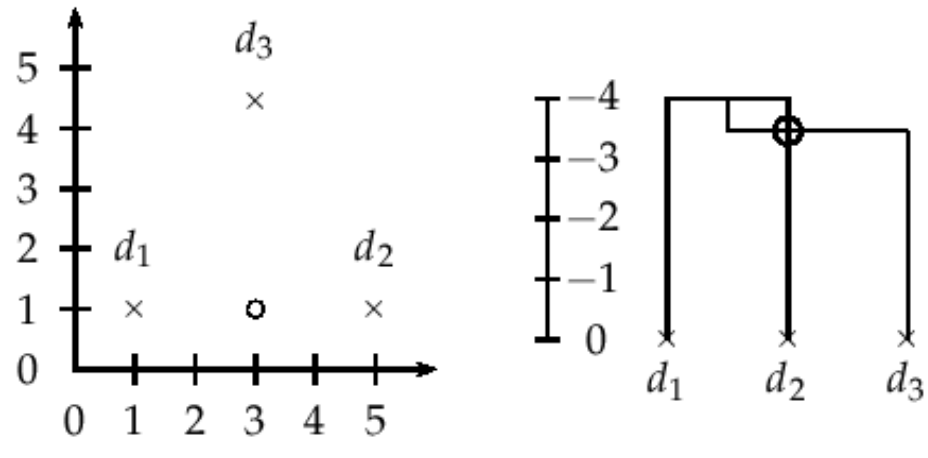
$$d_1 \leq d_2 \leq \cdots \leq d_{N-1}$$

- Ei-monotonisen menetelmän tuottama dendrogrammi voi sisältää inversioita, jotka ilmenevät ristikkäin menevinä kaarina



Keskipisteryvästäminen ei ole monotoninen

- Kuvassa pisteet $d_1 = (1 + \epsilon, 1)$, $d_2 = (5, 1)$, on yhdistetty etäisyydellä $4 - \epsilon$, mutta piste $d_3 = (3, 1 + 2\sqrt{3})$ ja ensimmäisen ryppään keskipiste $o = (3 + \epsilon/2, 1)$ yhdistetään etäisyydellä $2\sqrt{3} + \epsilon/2 \approx 3.4641 + \epsilon/2$
- Keskipisteryvästäminen (centroid clustering) ei siis ole monotoninen menetelmä
- Sen sijaan voidaan osoittaa, että lähi- ja kauimmaisen naapurin ryvästäminen sekä keskimääräisen etäisyyden ryvästäminen ovat monotonisia



Ryvästyksen optimaalisuus

- Ryppään C yhdistämisetäisyys $d(C)$ on pienin etäisyys, jolla C voidaan muodostaa kahdesta osasta C', C''
- Tämä etäisyys riippuu ainostaan käytetystä etäisyysmitasta, ei menetelmästä
- Ryvästyksen yhdistämisetäisyys määritellään ryppäiden yhdistämisetäisyyden maksimina $d(\mathcal{C}) = \max_{k=1}^{K} d(C_k)$
- Hierarkkisesti muodostettu ryvästys $\mathcal{C} = \{C_1, \ldots, C_K\}$ on optimaalinen, jos $d(\mathcal{C}) \leq d(\mathcal{C}')$ kaikilla $\mathcal{C}', |\mathcal{C}'| \leq K$ ts. ei ole olemassa ryvästystä, jossa on korkeintaan saman verran ryppäitä ja pienempi yhdistämisetäisyys

Monotonisuus vs. optimaalisuus

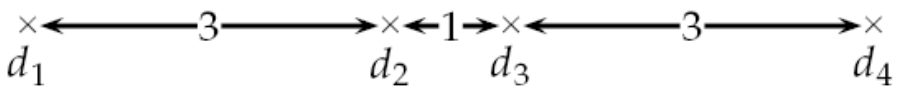
- Monotonisuus ja optimaalisuus mittaavat ryvästysmenetemästä hieman eri ominaisuutta
- Monotonisuus tarkoittaa, että algoritmin edetessä yhdistämisetäisyys ei koskaan vähene (dendrogrammissa ei ole ristiin meneviä kaaria)
- Optimaalisuus tarkoittaa, että ryvästyksen (suurin) yhdistämisetäisyys on mahdollisimman pieni (dendrogrammi on mahdollisimman matala)

Ryvästysmenetelmien optimaalisuus

- Voidaan osoittaa, että lähinaapuriryvästys on optimaalinen (todistus sivuutetaan)
- Kauimmaisen naapurin, keskimääräisen etäisyyden ja keskiarvoryvästysmenetelmät eivät sen sijaan ole optimaalisia

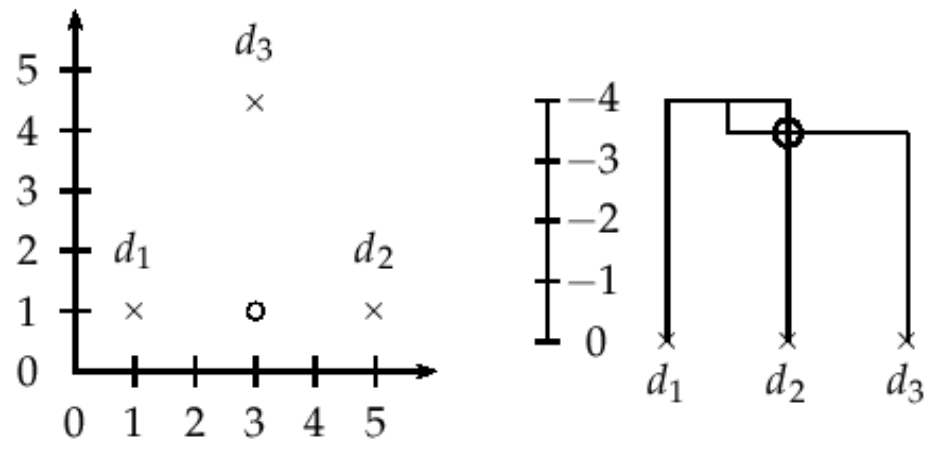
Kauimmainen naapuri ja keskimääräinen etäisyys eivät tuota optimaalista ryvästystä

- Kauimmaisen naapurin ja keskimääräisen etäisyyden ryvästysmenetelmät eivät ole optimaalisia, mikä nähdään alla olevasta esimerkistä
- Molemmat algoritmit liittävät ensin toisiinsa pisteet etäisyydellä 1 (d_2 ja d_3) ja päätyvät kahden ryppään ryvästykseen $\{\{d_1, d_2, d_3\}, \{d_4\}\}$ tai $\{\{d_1\}, \{d_2, d_3, d_4\}\}$.
- Optimaalinen ryvästys molemmissa tapauksissa $\{\{d_1, d_2\}, \{d_3, d_4\}\}$,
- Yhdistämisetäisyys kauimmaisen naapurin menetelmällä 4 (optimi 3) keskimääräisen etäisyyden menetelmällä $(1 + 3 + 4)/3 = 2.5$ (optimi 1.5)



Keskipisteryvästys ei ole optimaalinen

- Keskipisteryvästämisen epäoptimaalisuus nähdään kuvan esimerkistä
- Kuvassa pisteet $d_1 = (1 + \epsilon, 1)$, $d_2 = (5, 1)$, on yhdistetty etäisyydellä $4 - \epsilon$, joka on myös ryvästyksen $\{\{d_1, d_2\}, \{d_3\}\}$ $(K = 2)$ yhdistämisetäisyys
- Ryvästyksen $\{\{d_1, d_2, d_3\}\}$ (K=1) yhdistämisetäisyys on $d(\{\{d_1, d_2, d_3\}\}) =$ $3.46 < 4 - \epsilon$ eli pienempi määrä ryppäitä saadaan pienemmällä yhdistämisetäisyydellä



Hierarkkiset kokoavat ryvästysmenetelmät: yhteenveto

menetelmä	aikavaativuus	opti-maalinen?	kommentti
lähinaapuri	$O(n^2)$	on	ketjumaiset ryppäät
kauimmainen pari	$O(n^2) \log n$	ei	'outlier'-herkkä
keskimääräinen etäisyys	$O(n^2) \log n$	ei	useimmiten hyvä
keskipiste	$O(n^2) \log n$	ei	ei-monotoninen

Hierarkkinen vs . K-keskiarvon ryvästys

- K-means on nopea ($O(nK)$), epädeterministinen ja vaatii kiinteän ryppäiden määrän
- Hierarkkiset menetelemät ovat raskaampia ($O(n^2) - O(n^2 \log n)$), deterministisiä ja antavat paremmat työkalut ryppäiden määrän valintaan