Luentorunko perjantaille 28.11.2008

- Eräitä ryvästyksen keskeisiä käsitteitä
  - kustannusfunktio
  - sisäinen vaihtelu
  - edustajavektori
  - etäisyysmitta/funktio
- Osittamiseen perustuva ryvästys (yleisesti)
- K:n keskiarvon ryvästys (esimerkki osittamiseen perustuvasta ryvästyksestä)
- Hieman lisää etäisyysfunktiosta

Ryvästyksen kustannusfunktioista

- Ryvästyksen onnistumista mitataan tyypillisesti kustannusfunktioilla
- Kustannusfunktiot määritellään lähes poikkeuksetta datapisteiden välisten etäisyyksien perusteella
- Ryppäiden yhteenlasketun sisäisen vaihtelun (within cluster variation) minimoiminen on yksi mahdollisuus

$$F(\mathcal{C}) = \sum_{k=1}^{K} wc(C_k)$$

- Toimii mielekkäästi vain, jos ryppäiden määrä on kiinnitetty edeltäkäsin. Miksi?

Ryppään edustajavektorin määrittely

Määrittelemme seuraavassa kustannusfunktion, joka perustuu ryppäiden $C_1, \ldots, C_K$ *edustajavektoreihin* (keskipisteisiin tai keskimmäisiin edustajiin) $r_1, \ldots, r_K$

- Kukin r_k voi olla syöte, joka on jollakin tapaa keskeinen ryppään edustaja, tällöin $r_k \in C_k$
- Jos syötteet ovat jatkuva-arvoisia, keskiarvon ottaminen voi olla mielekästä:

$$r_k = \frac{1}{n_k} \sum_{x \in C_k} x$$

Ryppään sisäinen vaihtelu

- Yksinkertainen tapa mitata ryppäiden sisäistä vaihtelua on laskea yhteen kunkin ryppään syötteiden etäisyydet ryppään edustajavektoriin

$$wc(\mathcal{C}) = \sum_{k=1}^{K} wc(C_k) = \sum_{k=1}^{K} \sum_{x \in C_k} d(x, r_k)$$

- $wc(\mathcal{C})$ pyritään minimoimaan, eli ryvästys on siis sitä parempi, mitä lähempänä syötteet ovat ryppään edustajavektoria
- ratkaisu kuulostanee luontevalta
- mitä ongelmia voi seurata?

Osittamiseen perustuva ryvästys

- Annettuna syötejoukko $S = \{x_1, \ldots, x_n\}$ ja kokonaisluku K
- Tehtävä: etsi syötejoukon ositus K osajoukkoon $\mathcal{C} = \{C_1, \ldots, C_K\}$ siten, että
  1. osajoukot ovat mahdollisimman homogeenisia
  2. kukin syöte x_i kuuluu täsmälleen yhteen osajoukkoon C_j
- Homogeenisuutta mitataan kustannusfunktiolla $F(\mathcal{C})$; paras ryvästys on se, joka minimoi kustannusfunktion

$$\mathcal{C}^* = \text{argmin}_{\mathcal{C}} F(\mathcal{C})$$

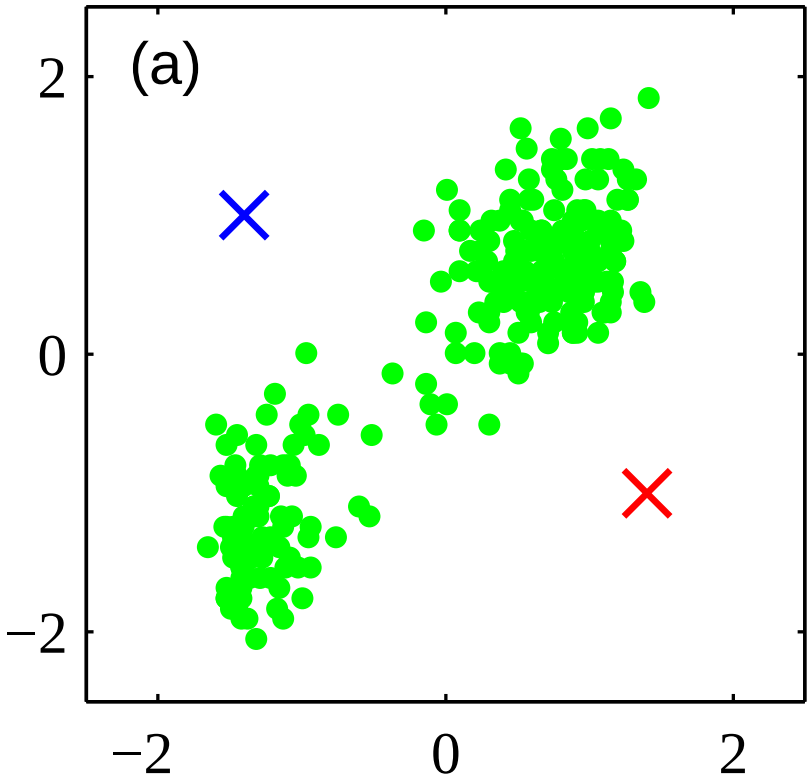
Osittamiseen perustuva ryvästys

- Käytännössä parhaan ryvästyksen löytäminen on laskennallisesti erittäin raskasta
  - kun K=2, erilaisia klusterointeja on $2^{|S|-1}$ kappaletta!)
  - mielekkäille arvotusfunktioille NP-kova ongelma, joten ei toivoa tehokkaasta täsmällisestä ratkaisualgoritmista
- Ongelma ratkaistaan käytännössä heuristisesti

K:n keskiarvon ryvästys (K-means clustering)

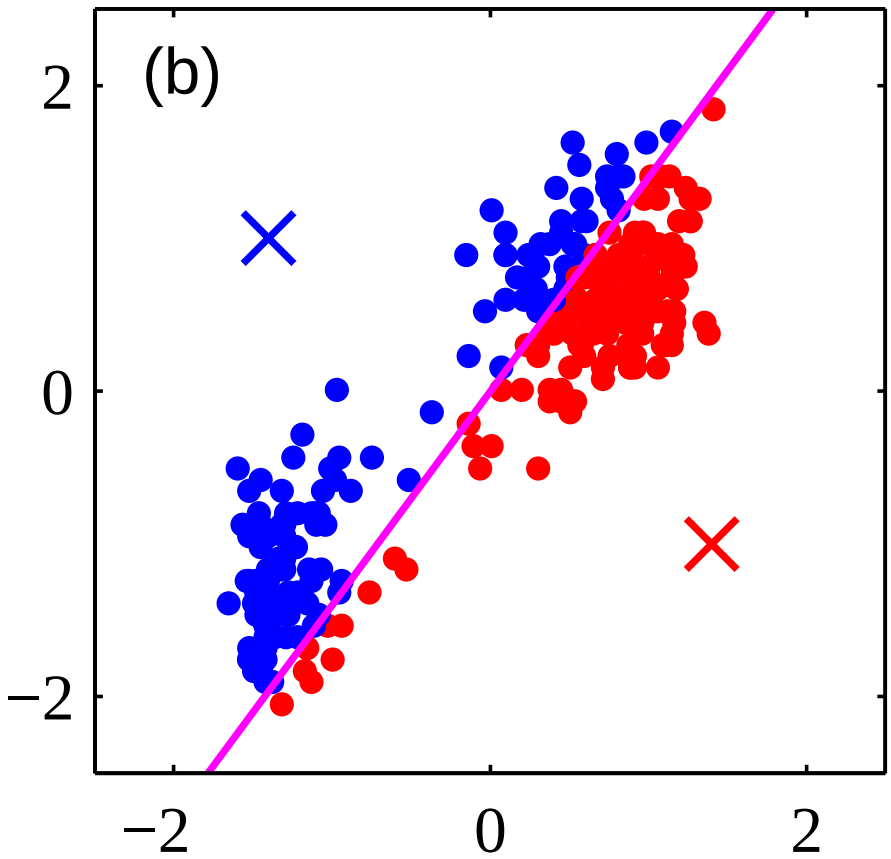
- K:n keskiarvon ryvästys on iteratiiviseen parantamiseen perustuva ryvästysmenetelmä.
- Aloitetaan satunnaisesta ryvästyksestä, ja muutetaan ryvästystä vaihettain siten, että kustannusfunktion arvo pienenee joka askeleessa
- Jatketaan kunnes ryvästys ei enää muutu

K:n keskiarvon ryvästys: esimerkki



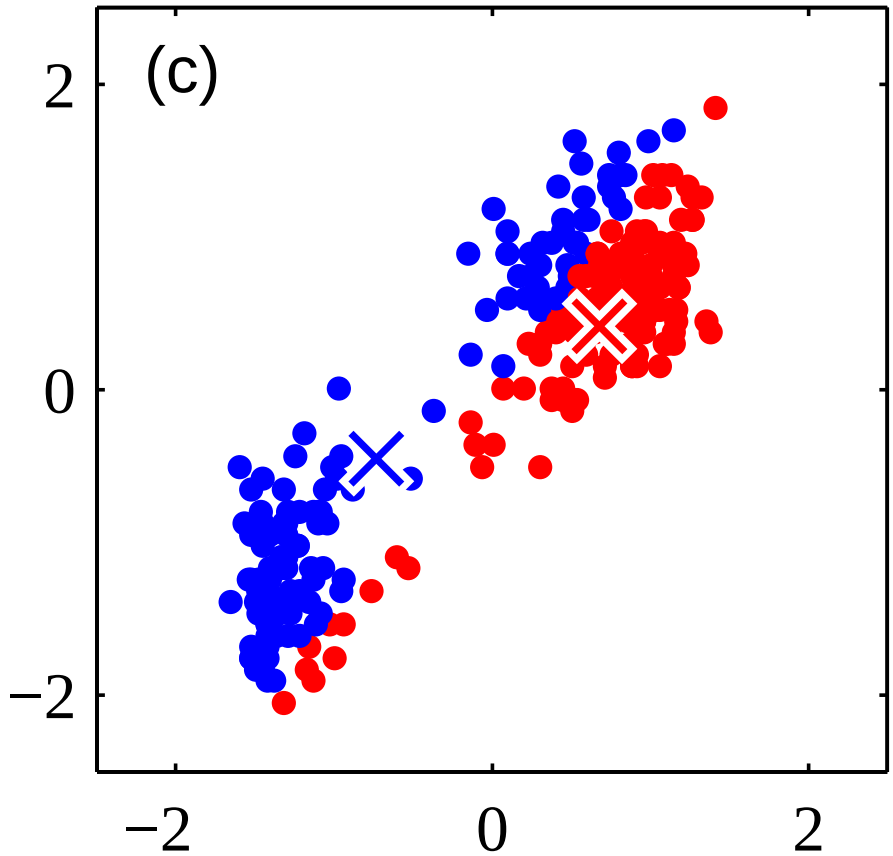
- data Old Faithful -nimisestä kuumasta lähteestä; vaaka-akseli purkauksen kesto, pystyakseli aika seuraavaan purkaukseen (nollakeskiarvoistettuina ja skaalattuina)
- K-means: sijoitetaan ensin k (tässä $k = 2$) klusterikeskipiste-ehdokasta satunnaisesti,

K:n keskiarvon ryvästys: esimerkki (2)



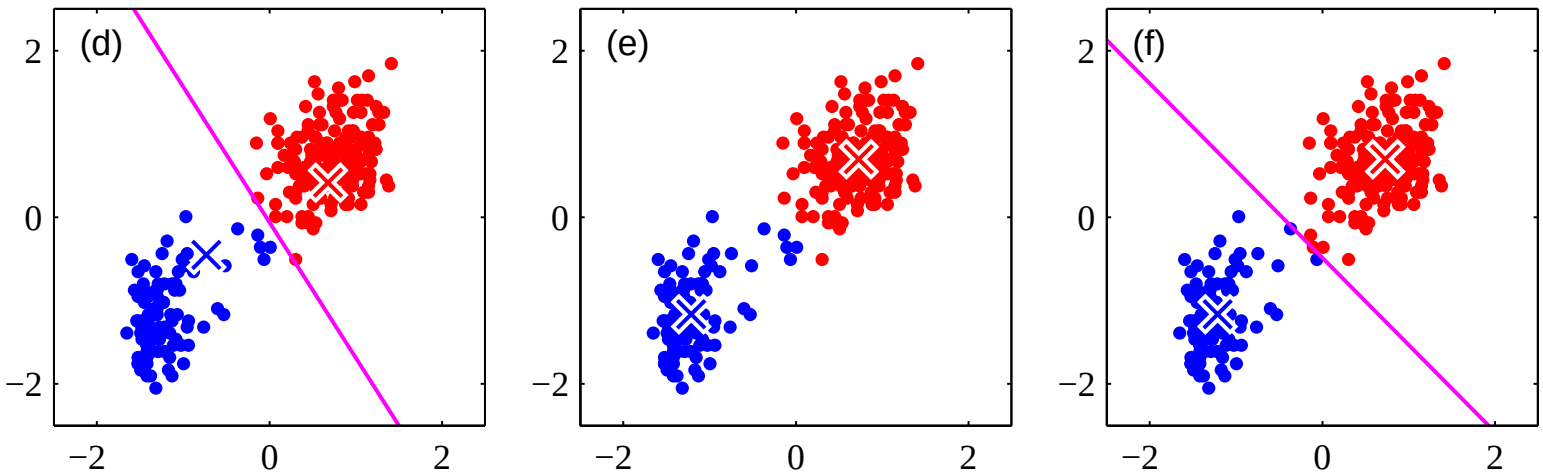
- assosioidaan kukin datapiste sitä lähinnä olevaan keskipisteeseen

K:n keskiarvon ryvästys: esimerkki (3)



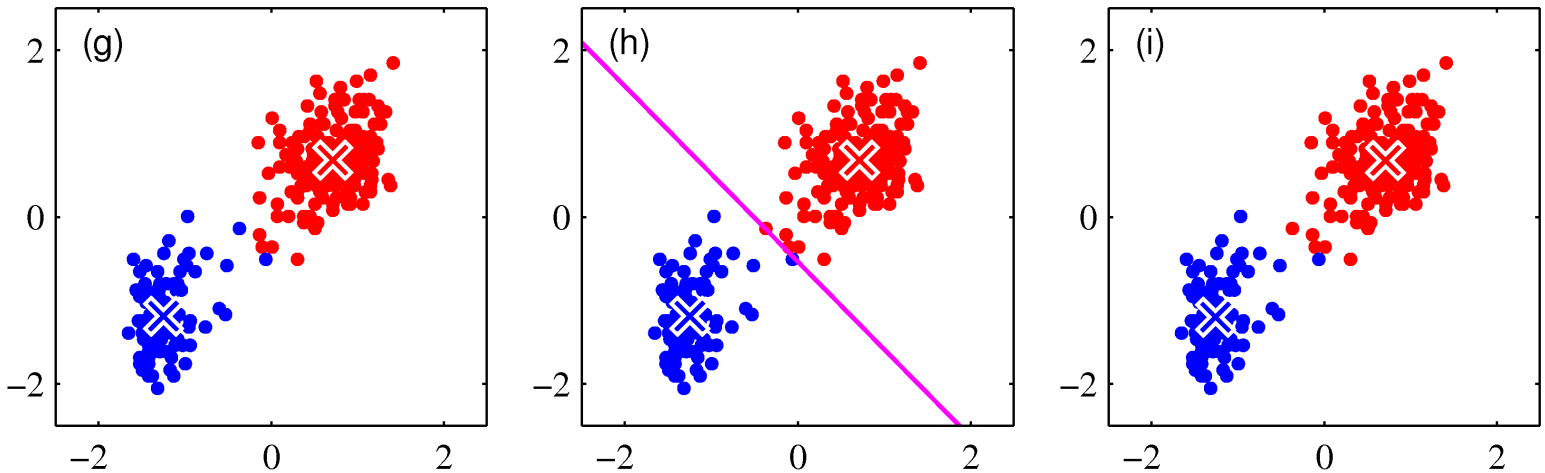
- lasketaan klusteriin kuuluvien alkioiden perusteella uudet keskipisteet (edustajavektorit)

K:n keskiarvon ryvästys: esimerkki (4)



- iteroidaan näitä vaiheita . . .

K:n keskiarvon ryvästys: esimerkki (5)



- . . . kunnes klusterit eivät enää muutu

Algoritmi (K:n keskiarvon ryvästys)

Algorithm K-means(S,K)

% alusta keskipisteet

r_k = satunnainen syöte joukosta S, kaikilla $k = 1, \ldots, K$;

% muodosta ryppäät

$k(x) = \mathbf{argmin}_{k=1}^{K} d(x, r_k)$, kaikilla $x \in S$

$C_k = \{x | k(x) = k\}$, kaikilla $k = 1, \ldots, K$

while muutoksia ryvästyksessä **do**

% laske uudet keskipisteet:

$r_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$, kaikilla $k = 1, \ldots, K$

% muodosta ryppäät:

$k(x) = \mathbf{argmin}_{k=1}^{K} d(x, r_k)$ kaikilla $x \in S$

$C_k = \{x | k(x) = k\}$ kaikilla $k = 1, \ldots, K$

end while

K-means -algoritmin pysähtyminen

- Askeltavien algoritmien pysähtyminen ei ole itsestään selvää, periaatteessa voisi olla mahdollista että algoritmi jää oskilloimaan eri ryvästysten välillä.
- Voidaan kuitenkin osoittaaa, että K-means -algoritmi pysähtyy lokaaliin minimiin.

Aikavaativuus

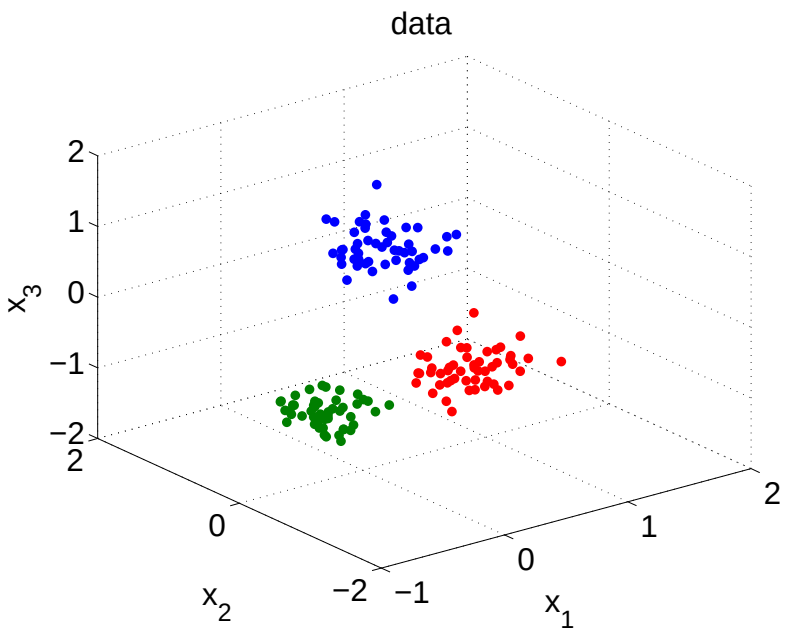
- Algoritmi toimii ajassa $O(KnI)$, missä I on iteraatioiden määrä (while-silmukka)
- Yhden iteraation vaativuus $O(Kn)$ seuraa argmin-operaation suorittamisesta: kutakin n syötettä verrataan kuhunkin K keskipisteeseen
- Iteraatioiden määrä I riippuu syötejoukosta ja siitä, miten aloituspisteet alustetaan.
- Käytännössä on havaittu, että tarvittavien iteraatioiden määrä on “pieni”
- Yllä olevassa jätetään etäisyyksien $d(x, y)$ laskennan kustannus huomiotta. Tyypillisesti kustannus on luokkaa $O(d)$, missä d on syötteen piirteiden määrä.

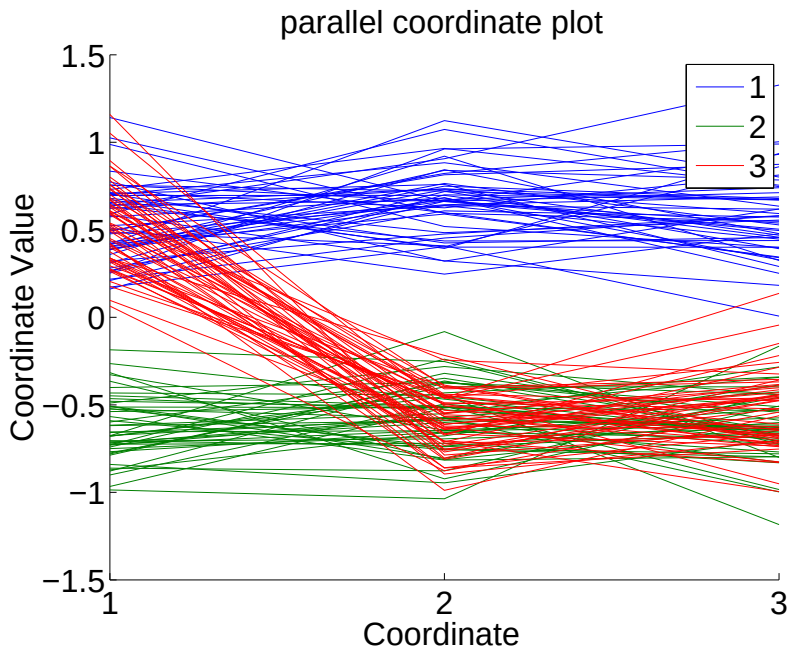
Algoritmin ongelmia: lokaali minimi

- Vaikka algoritmi pysähtyykin, saatu ratkaisu on herkkä ryppäiden keskipisteiden alustuksen suhteen: algoritmi päätyy herkästi lokaaliin minimiin, jos keskipisteet alustetaan "epäonnekkaasti"
- Tavallisin ratkaisu ongelmaan on suorittaa algoritmi useaan kertaan eri alkuasetuksilla ryppäiden keskipisteille ja valita tuloksista paras
- Edistyneempi menetelmä on nk. simuloitu jäähdytys (simulated annealing), jossa algoritmi voi välillä huonontaakin arvotusfunktion arvoa päästäkseen ulos lokaalista minimistä. Haittapuoli on suurempi suoritusaika.

Tulosten visualisointi: rinnakkaiset koordinaatit

- ongelma: miten klusteroinnin tuloksia visualisoidaan kun data on moniulotteista?
- yksi ratkaisu: rinnakkaiset koordinaatit (parallel coordinates)
  - tulostuksessa yksi käyrä vastaa yhtä datapistettä (tai vaikkapa kunkin klusterin keskipistettä)
  - eri klusteriin kuuluminen voidaan koodata viivan värillä



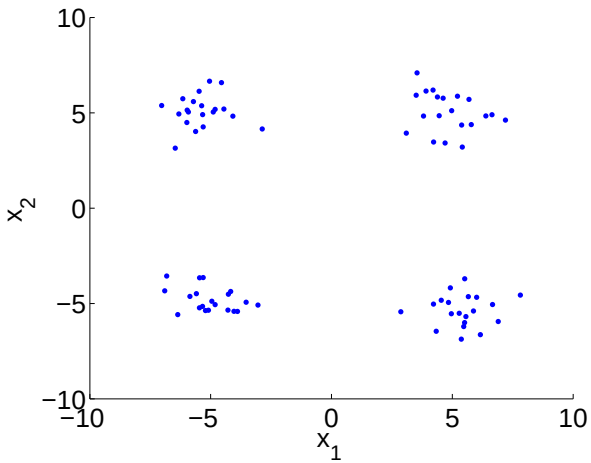


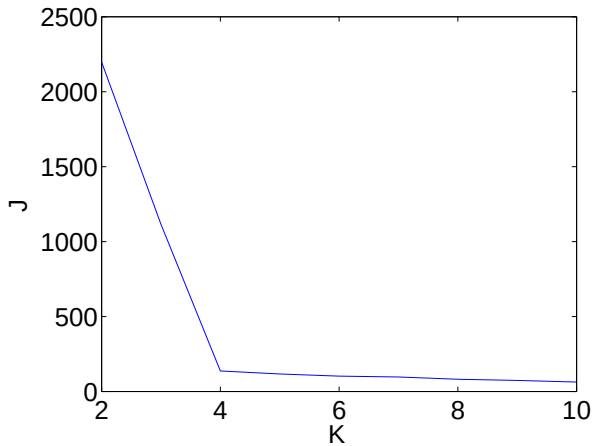
Miten valita ryppäiden määrä K?

- Mallinvalintaongelma (vrt. ohjattu oppiminen)
- Teoria on valitettavasti vielä suhteellisen kypsymätöntä
- Ratkaisuvaihtoehtoja
  - Ryppäiden määrän kasvattamisesta sakottaminen, erilaisia kriteerejä esim. Minimum Description Length, MDL:
    - Valitaan ryvästys, joka "pakkaa"datan parhaiten, kun datan kuvaus on ryppäiden keskipisteet + erotusvektori kullekin datapisteelle lähimpään keskipisteeseen
    - erotusvektorit koodataan siten, että lyhyet vektorit saava lyhyen koodin
  - Ryvästyksen stabiilisuus: jos ryppäiden määrä on "oikea", ryvästyksen ei pitäisi muuttua keskimäärin "paljoa", kun opetusaineistoa muutetaan “vähän“

Miten valita ryppäiden määrä K?

- "Yritys-ja-erehdys -menetelmä": kokeillaan eri arvoja $K = 1, 2, \ldots, n$ kunnes ryppäiden sisäinen varianssi on riittävän pieni
- Jos datassa on “optimaalinen K”, sisäinen varianssi pienenee nopeasti ryppäiden määrillä $< K$ ja sen jälkeen hitaasti
- “Kyynärpääkriteeri”





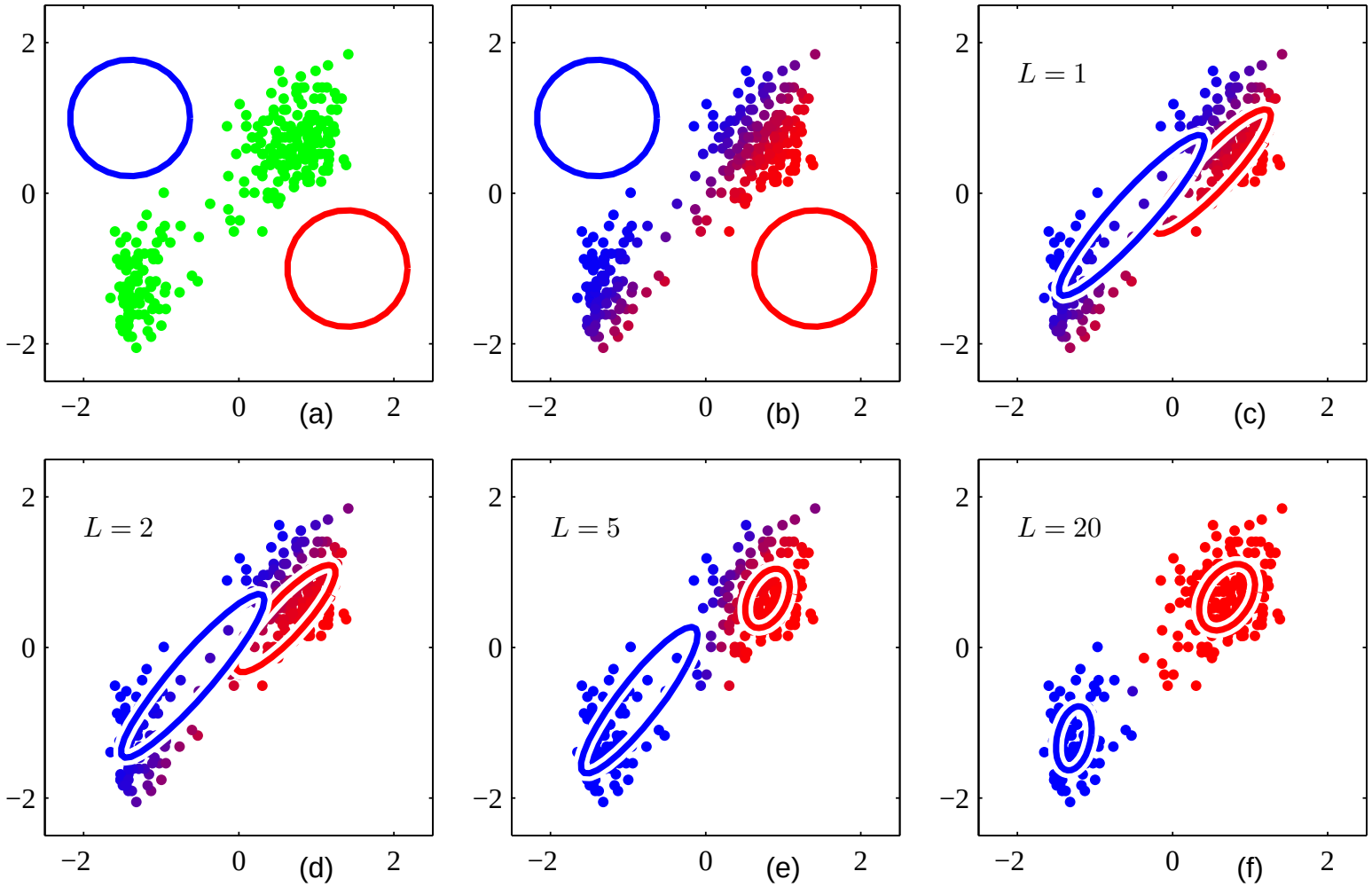
Ekskursio: laajennus sekoitemalleihin

- K:n keskiarvon klusterointi “pakottaa” jokaisen syötteen yhteen klusteriin
- kaikki klusteriin kuuluvat syötteet ovat yhtä merkittäviä, kun edustajavektori (keskiarvo) määritetään
- voisi ajatella, että kahden klusterin “rajaa” lähellä olevien syötteiden “pitäisi” vaikuttaa molempiin klustereihin

Ekskursio: laajennus sekoitemalleihin

- sekoitetodennäköisyysmalleihin perustuvassa klusteroinnissa syötteiden ajatellaan olevan peräisin K:sta lähteestä (esim. K:sta normaalijakaumasta eri parametrein), mutta lähteet ovat latentteja: ei tiedetä mikä syöte on mistäkin lähteestä
- optimaalinen klusterointi määritellään tyypillisesti sekoitemallin parametrien suurimman uskottavuuden (maximum likelihood) estimaatin avulla
- käytännössä joudutaan usein tyytymään lokaaliin maksimiin (Expectation Maximization, EM-algoritmi)
- sekoitemalleista kiinnostuneet voivat tutustua esim. teokseen Bishop: Pattern recognition and machine learning, luku 9.

Ekskursio: EM-algoritmi



Etäisyysfunktio

- klusteroinnissa käsitellään syötteiden välisiä suhteita, euklidinen etäisyys on erikoistapaus tästä
- etäisyysfunktio on yksi tapa kuvata pisteiden väliset suhteet
- etäisyysfunktio kuvaa kohteiden $\{o_1, ..., o_N\}$ väliset etäisyydet ei-negatiivisina reaalilukuina $d(o_m, o_n)$
- joskus etäisyysfunktion arvot lasketaan
- joskus etäisyysfunktio on suoraan lähtödatana
  - esimerkiksi psykologiassa koehenkilöiltä voidaan kysyä suoraan kohteiden välisiä etäisyyksiä;

Syötteiden väliset etäisyydet (1)

Joitakin tavallisimpia etäisyysmittoja syötteiden välisille etäisyyksille

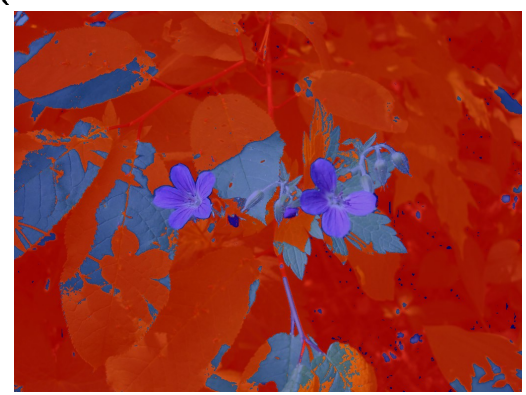
- Euklidinen etäisyys $d(x, z) = \sqrt{\sum_{j=1}^{d}(x_j - z_j)^2}$,
- Manhattan-etäisyys $d(x, z) = \sum_{j=1}^{d} |x_j - z_j|$,
- Hamming-etäisyys $d(x, z) = \sum_{j=1}^{d} 1_{x_j \neq z_j}$
- Kosini-etäisyys $d(x, z) = \cos(\theta) = \frac{x \cdot z}{\|x\|\|z\|}$

Syötteiden väliset etäisyydet (2)

- etäisyysfunktion määritteleminen on kuitenkin sovelluskohtaista ja heuristista
- alla pikselikohtaisen K-means ($K = 2$) klusteroinnin tuloksia kuvalle, joka on esitetty HSV-avaruudessa, eli kutakin pikseliä kohti kolme lukua: H (värisävy), S (värisaturaatio) sekä V (kirkkaus)

alkuperäinen kuva

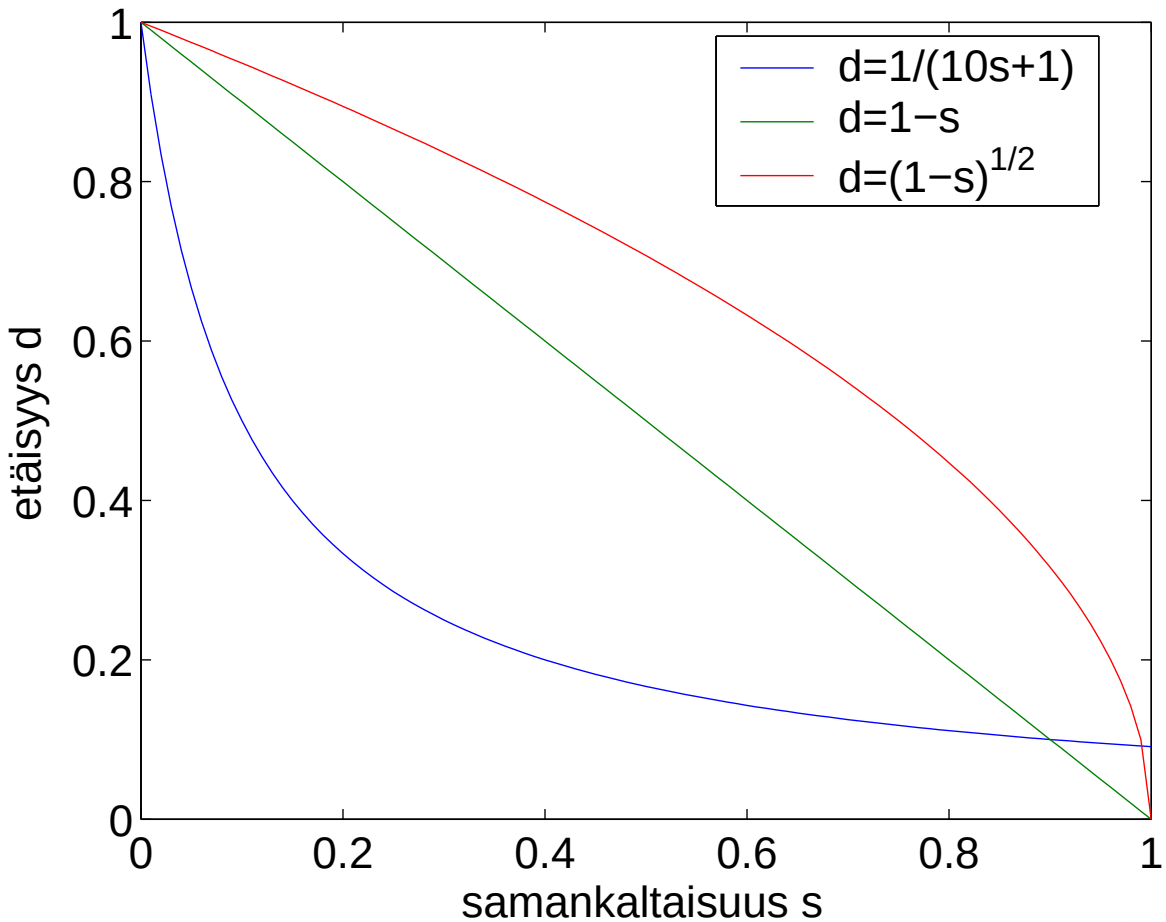
varianssit normalisoitu (mukava valoa etsivälle)



värisävymuuttuja skaalattu kertoimella 10 (mukava mehiläiselle)

Etäisyys vs. samankaltaisuus

- joskus lähtötietoina ei ole etäisyyksiä vaan samankaltaisuuksia $s(o_m, o_n)$
- samankaltaisuudet voidaan muuttaa etäisyyksiksi monotonisesti laskevalla funktiolla



Etäisyysfunktion räätälöinti

- Esikäsittelyn sijaan vaihtoehtoinen tapa käsitellä heterogeenistä dataa on räätälöidä etäisyysfunktion laskenta sopivaksi kullekin datatyypille.
- Tällöin dataa ei muunneta numeeriseen muotoon, vaan datapisteiden etäisyyttä laskettaessa käytetään datatyyppikohtaista etäisyysmittaa.
- Tehdään kullekin piirretyypille oma etäisyysfunktionsa. esim. seuraavasti
  - Numeeriset piirteet: $d(x_j, z_j) = (x_j - z_j)/\sigma_j$
  - Järjestysasteikon piirteet $d(x_j, z_j) = |rank(x_j) - rank(z_j)|/|rank(maximum) - rank(minimum)|$
  - Nominaaliasteikon piirteet: $d(x_j, z_j) = 1_{\{x_j \neq z_j\}}$

Rakenteiset piirteet

Muuttujat voivat olla muutakin kuin yksittäisiä arvoja, niillä voi olla sisäinen rakenne
Miten mitata etäisyyttä seuraavissa tapauksissa?

- Osajoukko: $x_j \subset$ { kehärata, kehä II, keskustatunneli, länsimetro}
- Järjestetty joukko: $x_j =$ (1. länsimetro, 2. kehärata, 3. kehä II, 4. keskustatunneli)