1. OHJAAMATON OPPIMINEN JA KLUSTEROINTI

1.1 Funktion optimointiin perustuvat klusterointialgoritmit

Klusteroinnin onnistumista mittaavan funktion J optimointiin perustuvissa klusterointialgoritmeissä ratkaisu esitetään usein parametrivektorin $\mathbf{\Theta} = (\mathbf{\Theta}_1^T, \ldots, \mathbf{\Theta}_m^T)^T$, missä $\mathbf{\Theta}_i$ on klusteriin C_i liittyvät parametrit, avulla

Parametrivektoreiden avulla voidaan määrittää löydettävien klustereiden rakenne (esim. hypertaso, hyperpallon kuori, toruksen kuori jne). Tämä on hyödyllinen ominaisuus esim. tietokonenäön sovellutuksista (kohteen ja taustan erottaminen toisistaan)

Tehdään siis *a priori* oletus klustereiden rakenteesta ja esitetään se parametrien avulla

Klustereiden lkm m oletetaan yleensä tunnetuksi

'C-means' tai 'k-means' -algoritmi

Muita nimiä ovat 'LBG' (Linde-Buzo-Gray), 'Isodata', 'Lloydin' -algoritmi

- Valitse luokkien lukumäärä c
- Aseta alkuarvot luokkien mallivektoreille $\mathbf{m}_j$
- Toista kunnes mallivektorit $\mathbf{m}_j$ eivät enää muutu
  - Luokittele näytteet $\mathbf{x}_k$ hakemalla kutakin näytettä lähin mallivektori
  - Päivitä luokkien mallivektorit laskemalla keskiarvo luokkaan osuneista näytteistä $\mathbf{m}_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$, missä N_i on klusteriin C_i kuuluvien havaintojen lkm

Eniten käytetty klusterointialgoritmi
Haittana: algoritmi antaa eri tuloksen eri ajokerroilla, johtuen mallivektorien satunnais alustuksesta

'C-means'-algoritmin variaatio

Klusterit $C_1, \ldots, C_c$ esitetään prototyyppivektoreiden $\mathbf{\Theta} = (\mathbf{m}_1^T, \ldots, \mathbf{m}_c^T)^T$ avulla ja klusteroinnin onnistumisen mittaamiseen käytetään seuraavanlaista kustannusfunktiota:

$$J_{SSE} = \sum_{i=1}^{c} \sum_{\mathbf{x}_k \in C_i} \|\mathbf{x}_k - \mathbf{m}_i\|^2 \qquad (1)$$

J_{SSE}:n minimi voidaan löytää seuraavalla algoritmilla:

- Valitaan prototyyppivektoreille $\mathbf{m}_1, \ldots, \mathbf{m}_c$ satunnaiset alkuarvot
- Toista kunnes klusterointi eli prototyyppivektorit eivät enää muutu
  - Jaetaan havainnot klustereihin s.e. $\mathbf{x} \in C_j$, jos

$$\frac{N_j}{N_j + 1} \|\mathbf{x} - \mathbf{m}_j\|^2 \leq \frac{N_i}{N_i - 1} \|\mathbf{x} - \mathbf{m}_i\|^2 \; \forall i = 1, \ldots, c, \qquad (2)$$

missä N_i on klusteriin C_i kuuluvien havaintojen lkm

– Päivitetään prototyyppivektorit seuraavasti: $\mathbf{m}_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$

Todistus:

- Oletetaan, että havainto $\mathbf{x}_j$ siirretään klusterista C_i klusteriin C_j
- Uuden klusterin C'_j prototyyppivektori on

$$\mathbf{m}'_j = \frac{1}{N_j + 1} \sum_{\mathbf{x}_k \in C'_j} \mathbf{x}_k = \mathbf{m}_j + \frac{\mathbf{x}_j - \mathbf{m}_j}{N_j + 1} \quad (3)$$

- Uuteen klusteriin C'_j liittyvä kustannus on silloin

$$J_{SSE}(C'_j) = \sum_{\mathbf{x}_k \in C'_j} \|\mathbf{x}_k - \mathbf{m}'_j\|^2 = J_{SSE}(C_j) + \frac{N_j}{N_j + 1} \|\mathbf{x}_j - \mathbf{m}_j\|^2 \quad (4)$$

- Vastaavasti, uuteen klusteriin C'_i liittyvä kustannus on

$$J_{SSE}(C'_i) = J_{SSE}(C_i) - \frac{N_i}{N_i - 1} \|\mathbf{x}_j - \mathbf{m}_i\|^2 \quad (5)$$

- J_{SSE}:n kokonaismuutos on

$$\Delta J_{SSE} = \frac{N_j}{N_j + 1} \|\mathbf{x}_j - \mathbf{m}_k\|^2 - \frac{N_i}{N_i - 1} \|\mathbf{x} - \mathbf{m}_i\|^2 \quad (6)$$

('C-means'-algoritmin perusversio ei ota kantaa kuinka havainnot jaetaan klustereihin ja kuinka klustereiden prototyyppivektoreita päivitetään)

1.2 Kilpailuun perustuvat klusterointialgoritmit

Kilpailuun perustuvissa klusterointialgoritmeissä klusterit $C_1, \ldots, C_m$ esitetään yleensä vektoreiden $\mathbf{w}_1, \ldots, \mathbf{w}_m$ avulla

Vektoreita (klustereita) $\mathbf{w}_1, \ldots, \mathbf{w}_m$ päivitetään havaintojen perusteella s.e. ne siirtyvät niihin kohtiin avaruutta, joissa havaintoja on tiheässä

Vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ kilpailevat keskenään. Havaintoa parhaiten vastaavaa voittajavektoria päivitetään s.e. se vastaa entistä paremmin havaintoa. Hävinneitä vektoreita ei päivitetä lainkaan tai päivitetään pienemmällä opetuskertoimella kuin voittajavektoria

Kilpailuun perustuvan klusterointialgoritmin yleistetty muoto ('General Competitive Learning Scheme' GCLS):

- Alustus: $t = 0$, $m = m_{\text{init}}$ (alkuarvaus klustereiden lkm:lle),
  (A) kaikkien tarvittavien parametrien alustus mm vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$
- Toista kunnes vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ eivät enää muutu merkittävästi tai

$t = t_{max}$ (yläraja iteraatioaskelien lkm:lle)

– $t = t + 1$
– Poimi satunnainen havainto $\mathbf{x} \in X$
– (B) Etsi havaintoa $\mathbf{x}$ parhaiten vastaava vektori $\mathbf{w}_j$
– (C) Jos ($\mathbf{x}$ ja $\mathbf{w}_j$ eivät ole riittävän samankaltaiset) TAI (jokin muu ehto) JA ($m < m_{\text{max}}$) (yläraja klustereiden lkm:lle), päivitä $m = m + 1$ ja määritä $\mathbf{w}_m = \mathbf{x}$
– (D) Muuten, päivitä vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ seuraavasti:

$$\mathbf{w}_j(t) = \begin{cases} \mathbf{w}_j(t-1) + \eta h(\mathbf{x}, \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on voittaja} \\ \mathbf{w}_j(t-1) + \eta' h(\mathbf{x}, \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ ei ole voittaja} \end{cases} \quad (7)$$

(η ja η' ovat opetuskertoimia ja $h(\cdot, \cdot)$ on jokin ongelmaan sopiva funktio)

Useimmat kilpailuun perustuvat klusterointialgoritmit voidaan esittää GCLS-tyyppisinä, kunhan kohdissa (A)-(D) tehdään sopivat valinnat

Oppiva vektorikvantisaatio

Oppiva vektorikvantisaatio ('Learning Vector Quantization', LVQ) on esimerkki GCLS-tyyppisestä klusterointialgoritmistä

LVQ-algoritmissä oletetaan klustereiden lkm $m = m_{\text{init}} = m_{\text{max}}$ tunnetuksi

LVQ-algoritmi saadaan kilpailuun perustuvien klusterointialgoritmien yleistetystä muodosta, kun

- (A) Alustetaan vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ satunnaisesti (ei muita parametrejä!)
- (B) $\mathbf{w}_j$ on havaintoa $\mathbf{x}$ parhaiten vastaava voittajavektori, jos $d(\mathbf{x}, \mathbf{w}_j) = \min_{k=1,\ldots,m} d(\mathbf{x}, \mathbf{w}_k)$. Funktio $d(\cdot, \cdot)$ on vektoreiden välinen etäisyys
- (C) Ehto ei toteudu koskaan, koska klustereiden lkm on kiinnitetty

- (D) Vektoreita päivitetään seuraavasti:

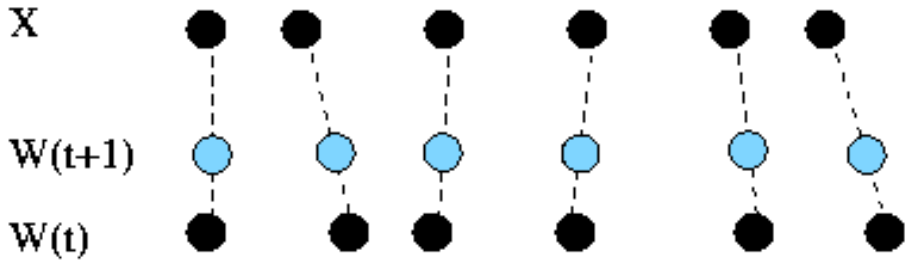
$$\mathbf{w}_j(t) = \begin{cases} \mathbf{w}_j(t-1) + \eta(\mathbf{x} - \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on voittaja} \\ \mathbf{w}_j(t-1), & \text{muuten} \end{cases} \quad (8)$$

Jos havaintojen luokat tunnetaan, voidaan vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ varata tietyille luokille ja tehdä päivitys seuraavasti:

$$\mathbf{w}_j(t) = \begin{cases} \mathbf{w}_j(t-1) + \eta(\mathbf{x} - \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on oikea voittaja} \\ \mathbf{w}_j(t-1) - \eta(\mathbf{x} - \mathbf{w}_j(t-1)), & \text{jos } \mathbf{w}_j \text{ on väärä voittaja} \\ \mathbf{w}_j(t-1), & \text{muuten} \end{cases} \quad (9)$$

(η on opetuskerroin)

LVQ-esimerkki: voittajavektori $\mathbf{w}(t)$ päivitetään vektoriksi $\mathbf{w}(t+1)$ havainnon $\mathbf{x}$ perusteella:



Itseorganisoituva kartta

('Self-Organizing Map' SOM)

SOM-algoritmissä havainnot yritetään esittää hilaan (kartta) järjestettyjen vektoreiden (karttayksiköiden) $\mathbf{w}_1, \ldots, \mathbf{w}_m$ avulla

SOM-algoritmin tavoitteena on asettaa vektoreiden arvot siten, että ne edustavat mahdollisimman hyvin havaintoja ja että hilassa toisiaan lähellä olevien vektoreiden arvot ovat samankaltaisempia kuin hilassa kaukana toisistaan olevien vektoreiden arvot

SOM-algoritmi saadaan kilpailuun perustuvien klusterointialgoritmien yleistetystä muodosta, kun

- (A) Alustetaan vektorit $\mathbf{w}_1, \ldots, \mathbf{w}_m$ (ei muita parametrejä)
- (B) $\mathbf{w}_{\text{BMU}}$ on havaintoa $\mathbf{x}$ parhaiten vastaava voittajavektori, jos $d(\mathbf{x}, \mathbf{w}_j) = \min_{k=1,\ldots,m} d(\mathbf{x}, \mathbf{w}_k)$. Funktio $d(\cdot, \cdot)$ on vektoreiden välinen etäisyys

- (C) Ehto ei toteudu koskaan, koska vektoreiden lkm on kiinnitetty
- (D) Päivitä vektoreita seuraavasti:

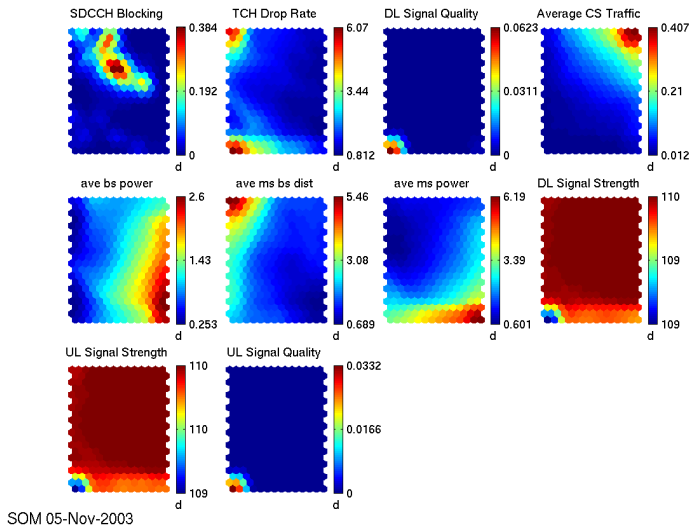
$$\mathbf{w}_j(t) = \mathbf{w}_j(t-1) + \eta h(j, \text{BMU})(\mathbf{x} - \mathbf{w}_j(t-1)), \qquad (10)$$

missä η on opetuskerroin ja $h(\cdot, \cdot)$ on naapurustofunktio, joka määrää mitkä ovat voittajavektorin päivitettäviä naapureita

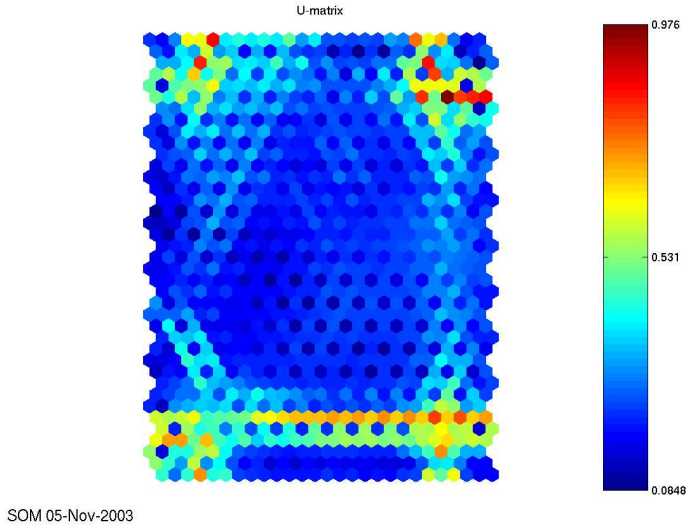
SOM-algoritmin antamaa tulosta voidaan analysoida ns. U-matriisin (kertoo kuinka samankaltaisia naapurivektorit ovat) ja komponenttitasojen (kertovat millaisia arvoja piirteet saavat erikohdissa karttaa) avulla

U-matriisista voidaan edelleen hakea klustereita esim. K-means algoritmilla käyttäen apuna jotain validointimenetelmää

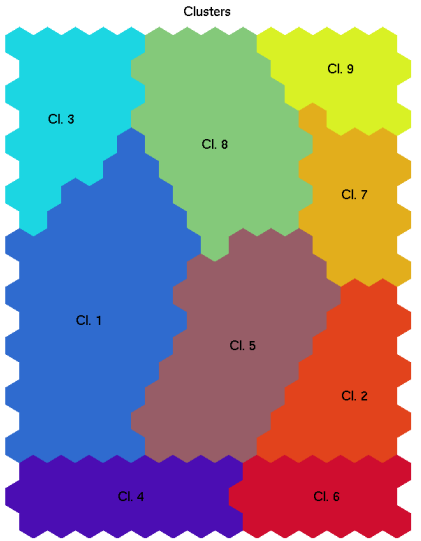
SOM:n komponenttitasot



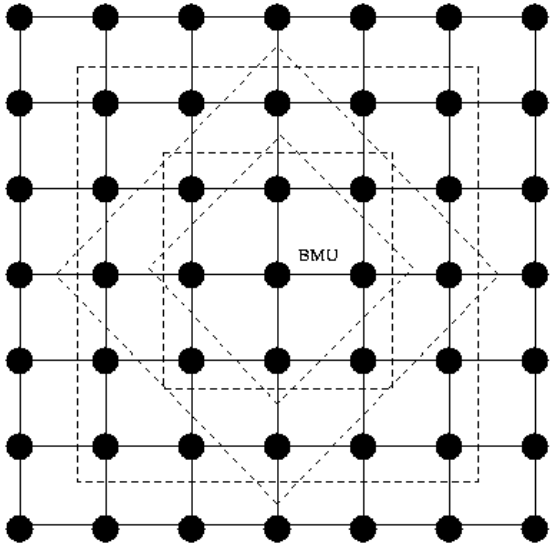
U-matriisi (Unified distance matrix)

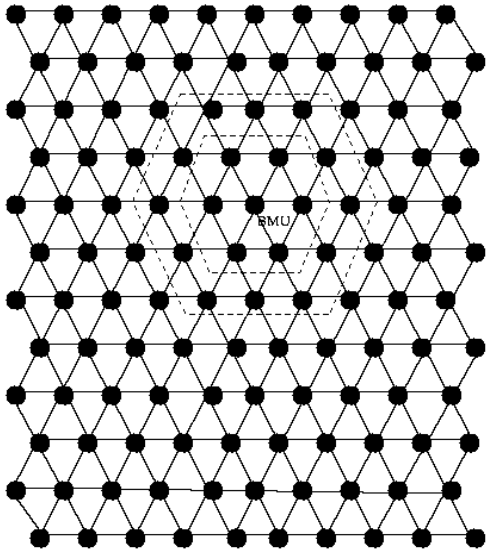


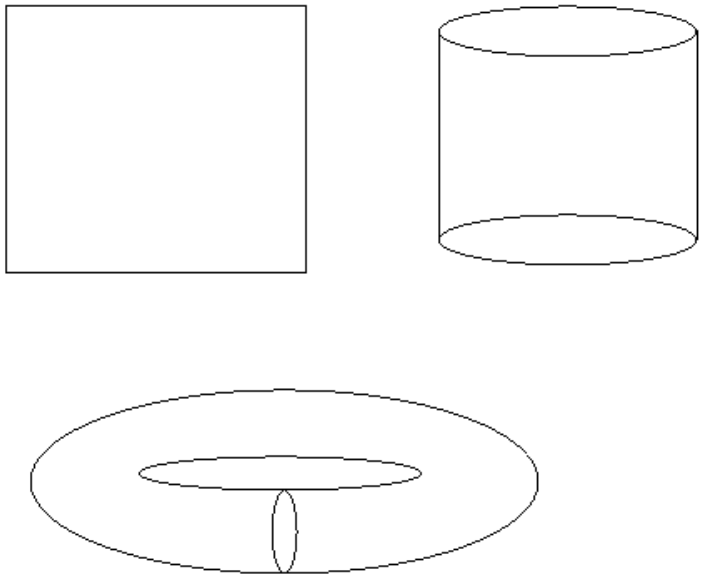
Klusteroitu U-matriisi



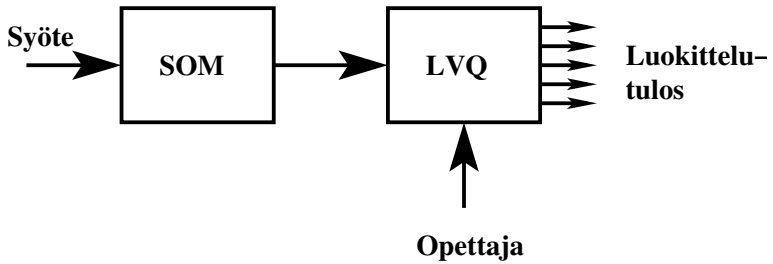
Erilaisia 2-ulotteisia hiloja ja naapurustoja:







Adaptiivinen hahmontunnistus

Toteutus SOM:n ja LVQ:n avulla:

- SOM:illa opetetaan alustavasti luokkien mallivektorit
- LVQ:lla hienosäädetään malleja käyttäen apuna dataa, jonka luokat tiedetään
- mallivektorit jäädytetään ja niitä voidaan käyttää uuden datan luokittelussa.