Luku 7. Itseorganisoiva kartta

T-61.2010 Datasta tietoon, syksy 2011

professori Erkki Oja

Tietojenkäsittelytieteen laitos, Aalto-yliopisto

24.11.2011

Tämän luennon sisältö

Itseorganisoiva kartta (SOM)

- *Itseorganisoiva kartta* (Self-Organizing Map, SOM) on informaatiotekniikan laboratoriossa professori Teuvo Kohosen 1980-luvun alussa kehittämä neuroverkkomalli
- Siitä on tullut 25 vuoden aikana eräs eniten käytettyjä neuroverkkoalgoritmeja maailmassa (viitteitä SOM:ään löytyy yli 7000)
- Katsotaan tässä luvussa mikä SOM on, mikä on sen yhteys neuroverkkoihin, miten se toimii ja esimerkkien avulla joitakin sovellusmahdollisuuksia.

▸ SOM-foneemikirjoitin ▸ SOM-köyhyyskartta

SOM-algoritmin perusidea

Kertaus: c-means

- Hyvä lähtökohta SOM-algoritmin ymmärtämiselle on ryhmittelyanalyysin *c-means-algoritmi* (Luku 6).
- Sehän meni seuraavasti:
  1. Valitse satunnaisesti c vektoria opetusjoukosta $\mathbf{x}_1, \ldots, \mathbf{x}_n$ pisteiksi $\mathbf{m}_1, \ldots, \mathbf{m}_c$;
  2. Toista kunnes pisteet $\mathbf{m}_1, \ldots, \mathbf{m}_c$ eivät muutu
     - Sijoita kukin vektori $\mathbf{x}_1, \ldots, \mathbf{x}_n$ siihen ryhmään C_i jonka keskipiste $\mathbf{m}_i$ on lähinnä;
     - Laske uudet keskipisteet kaavalla

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

missä n_i on C_i:n vektoreiden lukumäärä.

SOM-algoritmin perusidea (2)

Kertaus: c-means

- Tässä on oleellisesti kaksi vaihetta:
  1. Voittajan valinta: Voidaan ajatella että ryhmät, joita edustavat niiden keskipistevektorit $\mathbf{m}_1, \ldots, \mathbf{m}_c$, *kilpailevat* kustakin opetusvektorista $\mathbf{x}_j$ ja se, joka on lähinnä, voittaa kilpailun. Matemaattisesti: voittaja vektorille $\mathbf{x}_j$ on indeksi b jolle

$$b = b(\mathbf{x}_j) = argmin_{i=1,\ldots,c} \|\mathbf{x}_j - \mathbf{m}_i\|$$

  2. Voittajan mukautuminen/oppiminen: kun voittaja on kaapannut uuden vektorin, sen täytyy päivittää keskipistevektorinsa $\mathbf{m}_b$. c-means-algoritmissa tämä tehdään vasta kun kaikki $\mathbf{x}_j$:t on uudelleensijoitettu ryhmiin, mutta sen voi tehdä myös “lennossa” eli aina heti kun yksikin vektori on uudelleensijoitettu (ns. ISODATA-algoritmi).

SOM-algoritmin perusidea (3)

Kertaus: c-means

- SOM-algoritmi on helpompi ymmärtää jos kirjoitetaan keskipisteen päivitys uudella tavalla: olkoon $\mathbf{m}_b^{uusi}$ keskipiste sen jälkeen, kun uusi vektori $\mathbf{x}_j$ on lisätty ryhmään C_b
- Silloin myös ryhmän koko n_b kasvaa yhdellä
- Pätee:

$$\mathbf{m}_b^{uusi} = \frac{1}{n_b+1}[\sum_{\mathbf{x}\in C_b}\mathbf{x} + \mathbf{x}_j] \quad (1)$$

$$= \frac{1}{n_b+1}[\mathbf{x}_j + n_b\mathbf{m}_b] \quad (2)$$

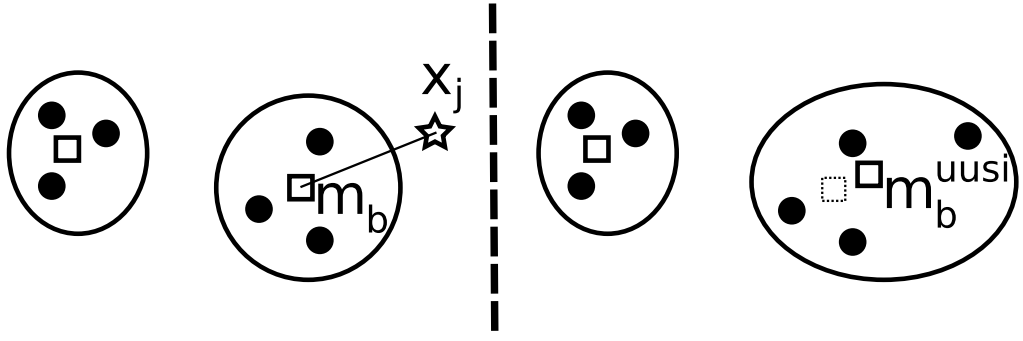
$$= \frac{1}{n_b+1}[\mathbf{x}_j + (n_b+1)\mathbf{m}_b - \mathbf{m}_b] \quad (3)$$

$$= \mathbf{m}_b + \frac{1}{n_b+1}[\mathbf{x}_j - \mathbf{m}_b] \quad (4)$$

SOM-algoritmin perusidea (4)

Kertaus: c-means

- Geometrisesti tämä merkitsee että uusi keskipiste $\mathbf{m}_b^{uusi}$ on liikkunut vanhan keskipisteen $\mathbf{m}_b$ ja uuden opetusvektorin $\mathbf{x}_j$ välisellä janalla pienen matkan kohti $\mathbf{x}_j$:tä, ikään kuin $\mathbf{x}_j$ vetäisi sitä puoleensa. Liike on sitä pienempi mitä enemmän ryhmässä on pisteitä



Kuva: c-means-algoritmin päivitysaskel

- Siinä siis oli c-means-algoritmi hiukan uudelleenmuokattuna.

SOM-algoritmin perusidea

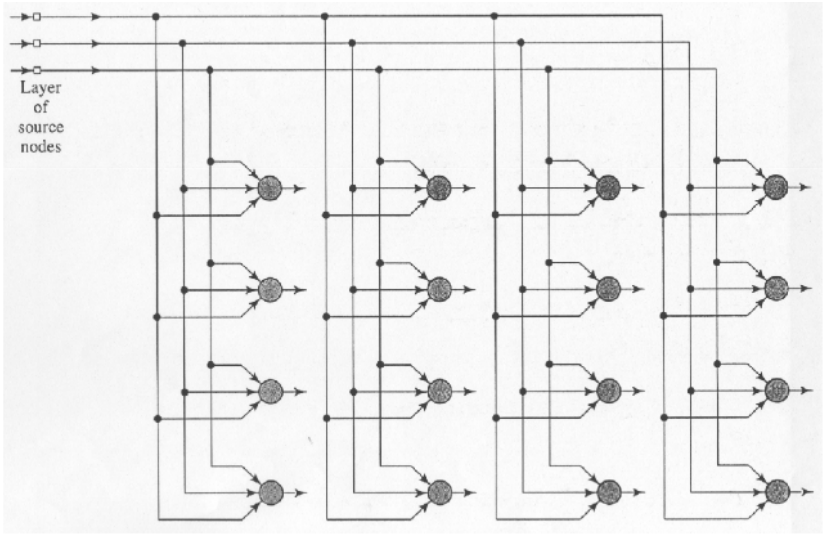
Voittajan yhteistyö naapurien kanssa

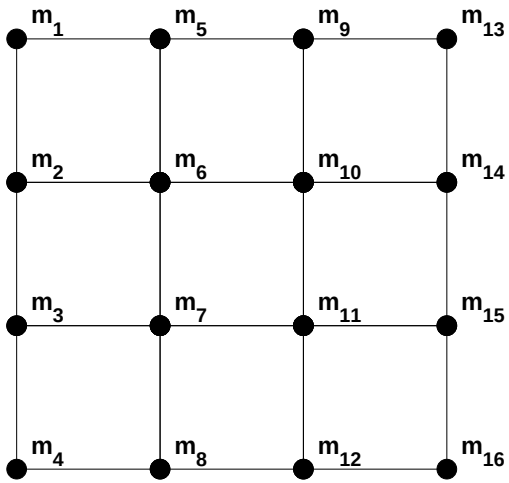
- SOM sisältää vielä yhden *aivan uuden periaatteen, joka tekee siitä eri menetelmän kuin c-means*:
  3. Voittajan yhteistyö naapureidensa kanssa: Kun edellä vain voittajan keskipistevektori $\mathbf{m}_b$ liikkui, niin SOM-menetelmässä myös voittajan *naapureiden keskipistevektorit liikkuvat*.
- Tämä sisältää kaksi kysymystä:
  1. Mikä on voittajan naapuri?
  2. Miten naapureiden keskipistevektorit liikkuvat?

SOM-algoritmin perusidea

Voittajan naapurusto

- Naapurusto määritellään aivan uudella tavalla (ei ole sama kuin esim. lähimmän naapurin luokitin!)
- Ajatellaan että itse asiassa keskipistevektorit ovat joidenkin *laskentayksiköiden* tai *keinotekoisten neuroneiden* painovektoreita, ja nämä yksiköt tai neuronit on järjestetty *säännölliseen hilaan*





SOM-algoritmin perusidea (2)

Voittajan naapurusto

Kuva: Kaksiulotteinen SOM-verkko: neuronin m_{11} naapureita ovat m_7, m_{10}, m_{12} ja m_{15}

- Tämä voisi vastata todellista fyysistä laskentapiiriä tai pientä aivokuoren osaa, vaikka SOM lähes aina toteutetaan ohjelmoimalla
- Hila on yleensä 2-ulotteinen kuten kuvassa, mutta voi olla myös 1- tai 3-ulotteinen (tai K-ulotteinen mutta silloin visualisointi vaikeaa).

SOM-algoritmin perusidea (3)

Voittajan naapurusto

- Nyt hila määrittelee naapurit: esim. 2-ulotteisessa hilassa lähimmät naapurit ovat viereiset laskentayksiköt oikealle, vasemmalle, ylös, alas (ns. 4-naapurusto, kuvassa $\mathbf{m}_{11}$:n naapurit ovat $\mathbf{m}_7$, $\mathbf{m}_{10}$, $\mathbf{m}_{12}$ ja $\mathbf{m}_{15}$), tai lisäksi diagonaalisesti (ns. 8-naapurusto) tai siihen voi kuulua myös seuraavat 16 ympäröivää yksikköä jne.
- 1-ulotteisessa hilassa yksiköt $b-1, b+1$, tai myös $b-2, b+2$ jne. (naapuruston koosta myöhemmin)

SOM-algoritmin perusidea

Naapureiden keskipistevektorien päivittäminen

- Entäs *toinen kysymys: Miten naapureiden keskipistevektorit liikkuvat?*
- Merkitään kaavassa (4) olevaa kerrointa $1/(n_b + 1)$ merkinnällä α, ja nyt saadaan SOM-päivityskaava painovektoreille:

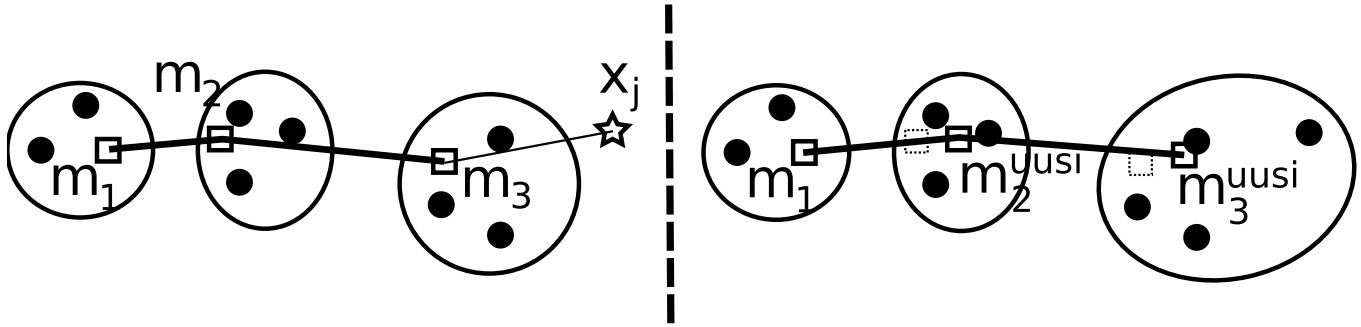
$$
\begin{aligned}
\mathbf{m}_b^{uusi} &= \mathbf{m}_b + \alpha[\mathbf{x}_j - \mathbf{m}_b] \text{ voittajayksiköllle } b \\
\mathbf{m}_k^{uusi} &= \mathbf{m}_k + \alpha[\mathbf{x}_j - \mathbf{m}_k] \text{ voittajan } b \text{ naapuriyksiköille } k \\
\mathbf{m}_l^{uusi} &= \mathbf{m}_l \text{ muille yksiköille.}
\end{aligned}
$$

- Osoittautuu että α:ksi voidaan valita sopivan pieni luku, jonka ei tarvitse olla yhtäsuuri kuin $1/(n_b + 1)$

SOM-algoritmin perusidea (2)

Naapureiden keskipistevektorien päivittäminen

- SOM poikkeaa c-means:stä siinä, että myös voittajan naapureita opetetaan. Alla olevassa kuvassa on 1-ulotteinen SOM, jossa painovektorit $\mathbf{m}_1$, $\mathbf{m}_2$ ja $\mathbf{m}_3$ vierekkäisen vektorin naapurustolla. Kun $\mathbf{m}_3$ voittaa (“best matching unit”, BMU), niin sitä ja myös sen naapuria $\mathbf{m}_2$ siirretään uuden datan suuntaan



Kuva: SOM-kartan (1D) päivittyminen: sekä voittaja että sen *naapuri* päivittyvät

SOM-algoritmin perusidea (3)

Naapureiden keskipistevektorien päivittäminen

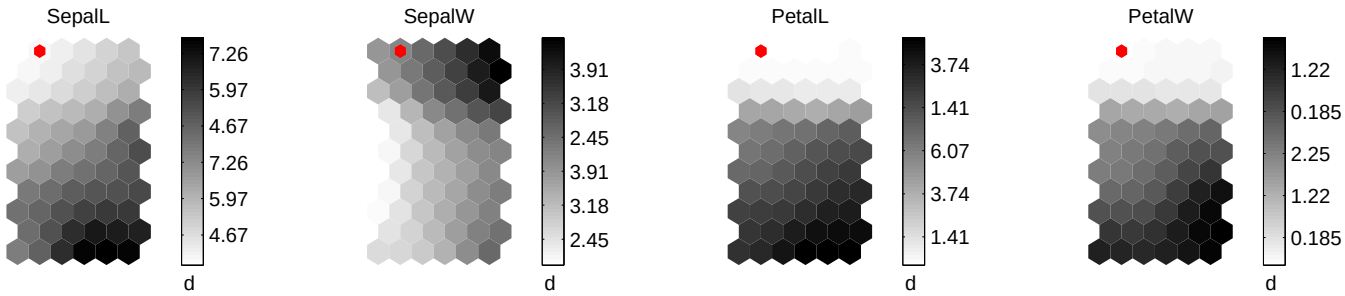
- Tässä kaavassa esitetään vain yksi askel SOM-algoritmia, yhdelle opetusvektorille
- Opetusta on jatkettava koko opetusjoukon yli, ja useaan kertaan (joitakin kymmeniä kertoja tyypillisesti)

  ▸ Koko esimerkki 2D-data ja 2D-kartta opetuksen aikana
- *Mitä tapahtuu?*
- Kaksi asiaa:
  1. Painovektorit levittäytyvät koko opetusjoukon alueelle (vrt. ryhmittely)
  2. Naapuriyksiköiden painovektorit pyrkivät olemaan lähekkäin (ns. *topologinen kartta*)

SOM-algoritmin perusidea

Kartan ja datan visualisointi

- Opetuksen tuottaman painovektorien levittäytyminen ja naapureiden läheisyys voidaan kuvata graafisesti kahdella tavalla
  - näyttämällä itse SOM-karttahilan ja kussakin neuronissa sen painovektorin



Kuva: 2-ulotteisen SOM-kartan neljä kompenttitasoa (4-ulotteiset painovektorit), kun syötteenä 4-ulotteista dataa. ▸ Karttahila ja painovektorit komponenteittain

SOM-algoritmin perusidea (2)

Kartan ja datan visualisointi

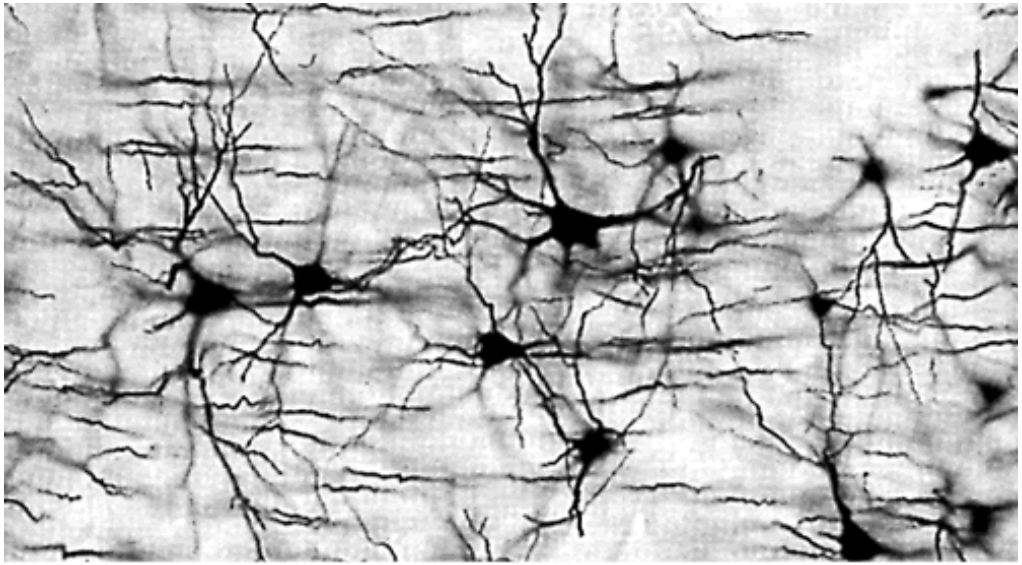
- näyttämällä syöteavaruuden (jonka silloin oltava 1-, 2- tai tietokoneavusteisesti 3-ulotteinen) ja siinä olevat opetusvektorit ja SOM-painovektorit. Naapuruus näytetään yhdistämällä naapurineuronien painovektorit viivalla (“elastinen verkko”) ▸ Kohosen kaktus

Yhteys biologiaan

- Isojen aivojen aivokuorella (apinoilla, kissoilla, hiirillä, ihmisillä, yms.) on *sensorisia alueita*, jotka käsittelevät esim. näköhavaintoja
- Erityisen paljon on tutkittu, kuinka yksinkertaiset visuaaliset havainnot, esimerkiksi tietynsuuntaiset viivat, kuvautuvat aivoihin
- Osoittautuu että näköaivokuoren tietyllä alueella on ns. suuntakarttoja, joissa kaikki eri suuntaiset viivat on koodattu samaan tapaan kuin SOM-pinnalla: kaikki suunnat ovat edustettuina, ja viereiset neuronit koodaavat lähes samansuuntaisia viivoja
- Suuntakartat eivät määräydy geneettisesti vaan oppimisen kautta

Yhteys biologiaan (2)

- SOM-algoritmi jäljittelee yksinkertaista oppimislakia, jolla todelliset hermosolut saattaisivat muodostaa topologisia suuntakarttoja.



Kuva: Hermosoluja

Suppenevuus 1-ulotteisessa tapauksessa

- Katsotaan nyt edellä esitettyä SOM-algoritmia:

$$\begin{aligned}
\mathbf{m}_b^{uusi} &= \mathbf{m}_b + \alpha[\mathbf{x}_j - \mathbf{m}_b] \text{ voittajayksikölle } b \\
\mathbf{m}_k^{uusi} &= \mathbf{m}_k + \alpha[\mathbf{x}_j - \mathbf{m}_k] \text{ voittajan } b \text{ naapuriyksiköille } k \\
\mathbf{m}_l^{uusi} &= \mathbf{m}_l \text{ muille yksiköille.}
\end{aligned}$$

- Oletetaan mahdollisimman yksinkertainen tapaus: 1-ulotteinen hila, 1- ulotteinen opetusdata: silloin "painovektorit" ovat skalaareja, suoran pisteitä $m_1, ..., m_c$.
- Naapurusto määritellään niin että neuronin j naapurit ovat $j-1, j+1$ paitsi päissä joissa neuronilla 1 on naapurina vain 2 ja neuronilla c vain $c-1$.
- Ajatellaan iso joukko "opetusvektoreita" (nyt skalaareja) x.

Suppenevuus 1-ulotteisessa tapauksessa (2)

- Tässä tapauksessa on mahdollista todistaa että painoluvut m_i järjestyvät varmasti opetuksessa joko suuruus- tai pienuusjärjestykseen, jos $0 < \alpha < 1$.
- Ensinnäkin, voidaan aina konstruoida sopiva joukko opetuslukuja x siten että ne *järjestävät* m_i:t

  ▸ 1D-kartan järjestyminen

- Toiseksi, jos m_i:t ovat järjestyksessä, niitä *on mahdoton saada enää epäjärjestykseen*. Oletetaan että $m_c > m_{c-1} > ... > m_2 > m_1$. Oletetaan että b:s yksikkö voittaa. Silloin vain m_{b-1}, m_b, m_{b+1} muuttavat paikkaa, muut pysyvät ennallaan.

Suppenevuus 1-ulotteisessa tapauksessa (3)

- Silloin

$$\begin{aligned} m_b^{uusi} - m_{b-1}^{uusi} &= m_b + \alpha(x - m_b) - m_{b-1} - \alpha(x - m_{b-1}) \\ &= (1 - \alpha)(m_b - m_{b-1}) > 0 \end{aligned}$$

joten niiden kohdalla järjestys säilyy. Aivan samoin osoitetaan että $m_{b+1}^{uusi} - m_b^{uusi} > 0$.

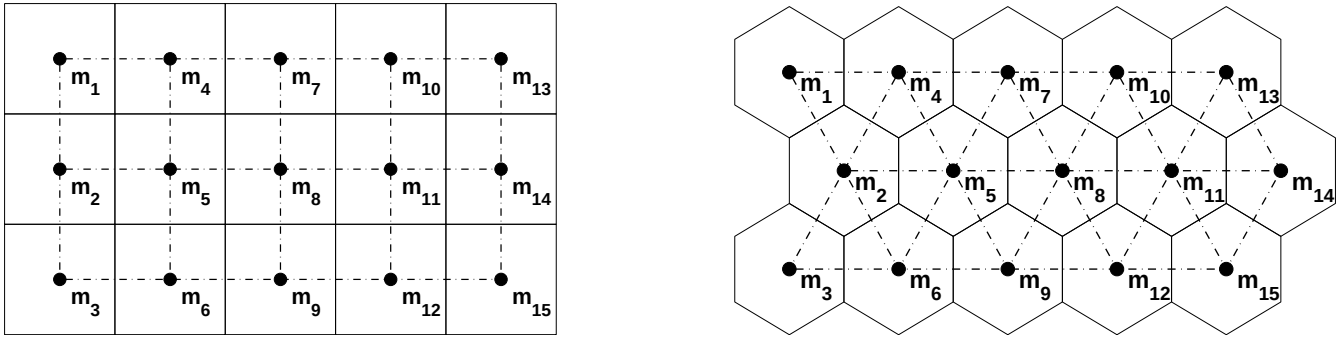
- Sitten

$$\begin{aligned} m_{b-1}^{uusi} - m_{b-2}^{uusi} &= m_{b-1} + \alpha(x - m_{b-1}) - m_{b-2} \\ &= m_{b-1} - m_{b-2} + \alpha(x - m_{b-1}) \end{aligned}$$

mutta tämä on positiivinen koska $m_{b-1} - m_{b-2} > 0$ ja toisaalta $x - m_{b-1} > 0$. Jälkimmäinen pätee siksi, että yksikkö b on voittaja ja siis x on lähempänä m_b:tä kuin m_{b-1}:tä. Aivan samoin osoitetaan että $m_{b+2}^{uusi} - m_{b+1}^{uusi} > 0$.

Käytännön valintoja

- Yleisimmät hilat ovat 1- tai 2-dimensioisia, koska kartta on silloin helppo visualisoida
- 2-dimensioisella hilalla on kaksi perusrakennetta: *suorakaidehila* ja *kuusikulmiohila*

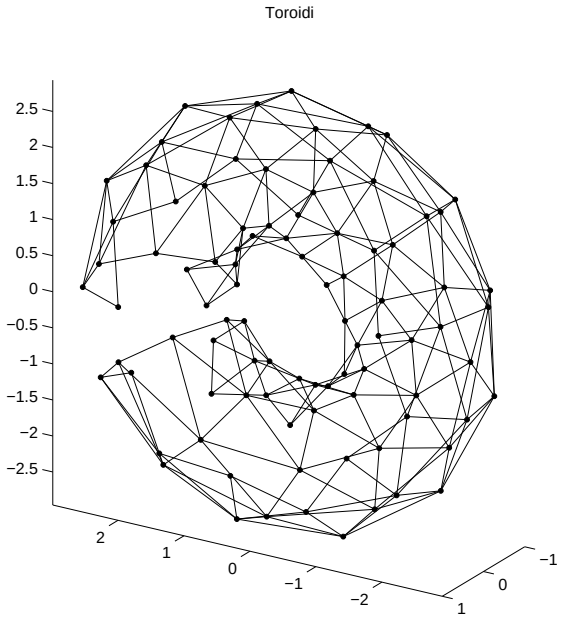


Kuva: SOM:n suorakaidehila ja kuusikulmiohila

- Kuusikulmio- eli heksagonaalihilan etu on se, että kullakin yksiköllä on 6 naapuria jotka kaikki ovat yhtä etäällä siitä

Käytännön valintoja (2)

- Reunoilla naapurusto on epätäydellinen, paitsi jos hila ajatellaan kierretyksi “munkkirinkeliksi” (toruspinnaksi), jolloin se on jatkuva eikä mitään reunoja ole



Kuva: SOM-hila “munkkirinkilinä”

Käytännön valintoja (3)

- Naapuruston koko: käytännössä osoittautuu, että SOM-algoritmissa on syytä pienentää naapurustoa vähitellen, samoin algoritmissa olevaa α-kerrointa. Näin saadaan parempia tuloksia.
- Nämä molemmat voidaan itse asiassa yhdistää kirjoittamalla SOM-algoritmi hieman toiseen muotoon: Sen sijaan että olisi

$$\begin{aligned}
\mathbf{m}_b^{uusi} &= \mathbf{m}_b + \alpha[\mathbf{x}_j - \mathbf{m}_b] \text{ voittajayksikölle } b \\
\mathbf{m}_k^{uusi} &= \mathbf{m}_k + \alpha[\mathbf{x}_j - \mathbf{m}_k] \ b\text{:n naapuriyksiköille } k \\
\mathbf{m}_l^{uusi} &= \mathbf{m}_l \text{ muille yksiköille.}
\end{aligned}$$

kirjoitetaankin tämä seuraavasti:

$$\mathbf{m}_k^{uusi} = \mathbf{m}_k + \alpha h(k, b)[\mathbf{x}_j - \mathbf{m}_k] \text{ kaikille yksiköille } k$$

Käytännön valintoja (4)

- Tämä on täysin sama kuin edellä oleva SOM-algoritmi, jos kaavassa oleva uusi termi, ns. *naapuruusfunktio* on

$$h(k,b) = \begin{cases} 1 & \text{jos } k \text{ kuuluu voittajan } b \text{ naapurustoon} \\ 0 & \text{muuten} \end{cases}$$

- Jotta kaavat olisivat aivan yhdenpitävät, on oltava myös $h(b,b) = 1$ eli pitää ajatella että b itsekin kuuluu omaan naapurustoonsa
- Nyt naapuruston valintaa voidaan “pehmentää” valitsemalla funktioksi $h(k,b)$ sileämpi funktio, esim. gaussinen $h(k,b) = e^{-\beta d^2(k,b)}$ missä $d(k,b)$ on yksikön k etäisyys voittajasta b *hilaa pitkin mitattuna* (huomaa että nytkin $h(b,b) = 1$)

Käytännön valintoja (5)

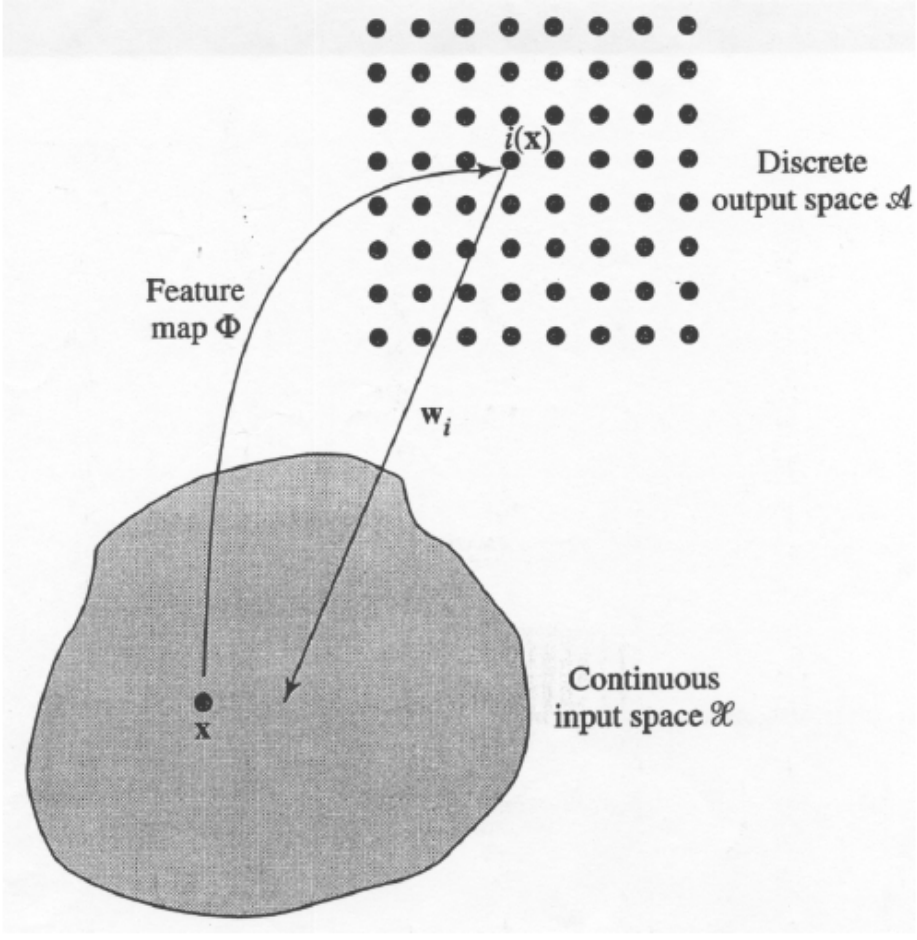
- Esim. yksiulotteisessa hilassa missä on yksiköt $1, 2, ..., c$, määritellään $d(k, b) = |k - b|$
- Paljon tämänkaltaisia virityksiä on valmiina ohjelmapaketissa SOM Toolbox

  ▸ http://www.cis.hut.fi/projects/somtoolbox/

Mihin SOM:ia käytetään?

- On taas erotettava toisistaan SOM:in *oppimisalgoritmi*, josta edellä puhuttiin, ja valmiin opetetun SOM-kartan *käyttö*
- Kartan käyttöä luonnehtii oheinen kuva: SOM muodostaa kuvauksen (funktion) syöteavaruuden $\mathbb{R}^d$ ja karttahilan välille

Mihin SOM:ia käytetään? (2)



Kuva: SOM: kuvaus syöteavaruuden ja karttahilan välillä

Mihin SOM:ia käytetään? (3)

- Kukin syöteavaruuden piste **x** (esim. kukin opetusjoukon vektori) kuvautuu hilaan siihen yksikköön joka on **x**:lle voittaja, eli yksikköön

$$b = b(\mathbf{x}) = argmin_{i=1,\ldots,c} \|\mathbf{x} - \mathbf{m}_i\|$$

- Siten SOM on *ryhmittelyalgoritmi*, jos kukin hilapiste ajatellaan omana ryhmänään (klusterinaan). Tiettyyn hilapisteeseen kuvautuu samankaltaisia vektoreita **x**
- Verrattuna tavalliseen ryhmittelyyn (esim. c-means) SOM:illa on ylimääräinen ominaisuus: vierekkäisiin hilapisteisiin kuvautuu yleensä myös samankaltaisia vektoreita
- Siten syöteavaruuden se alue, johon opetusvektorit kuuluvat, kuvautuu lähes jatkuvasti (topologisesti) hilaan

Mihin SOM:ia käytetään? (4)

- Hilaa voi nyt käyttää ikäänkuin *karttapintana tietokoneen ruudulla hakemaan erilaisia syöteavaruuden osia*, se on siis tehokas *visualisointikeino* hyvinkin korkeadimensioiselle datalle
- Kukin hilapiste k kuvautuu vastaavasti syöteavaruuteen $\mathbb{R}^d$ pisteeseen $\mathbf{m}_k$
- Siten SOM on myös *vektorikoodausalgoritmi*.
- Kuvaus syöteavaruuden ja SOM-hilan välillä ei kuitenkaan matemaattisessa mielessä voi olla puhtaasti topologinen (jatkuva molempiin suuuntiin), koska hilan dimensio on yleensä 2 ja syöteavaruuden $d > 2$. Kartassa on siellä täällä hyppäyksiä (vrt. tapaus 2-ulotteinen syöteavaruus, 1-ulotteinen kartta).

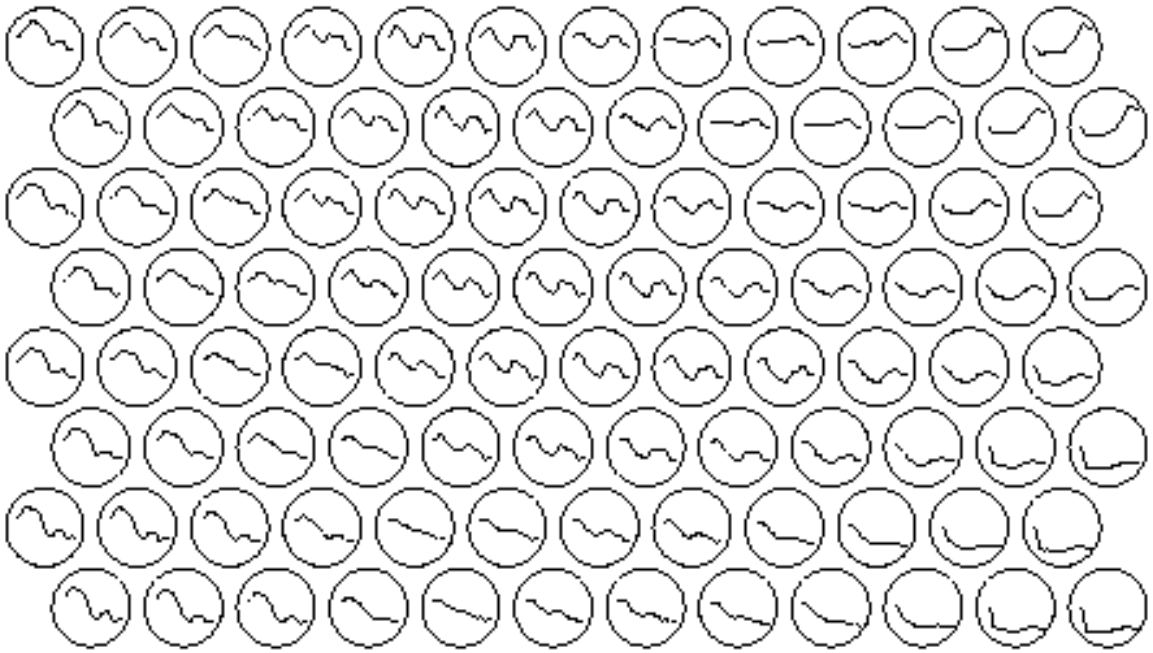
Mihin SOM:ia käytetään?
Esimerkkejä

- WEBSOM: menetelmä suurten tekstiaineistojen kuvaamiseksi kartalle
- PicSOM: kuvatietokantojen interaktiivinen hakumenetelmä
- Teollisuusssovelluksia.

Yhteenveto

Tässä luvussa esiteltiin itseorganisoiva kartta (SOM), jota voidaan käyttää korkeaulotteisen datan ryhmittelyyn ja visualisointiin.
SOM:n keksijä on akateemikko Teuvo Kohonen.

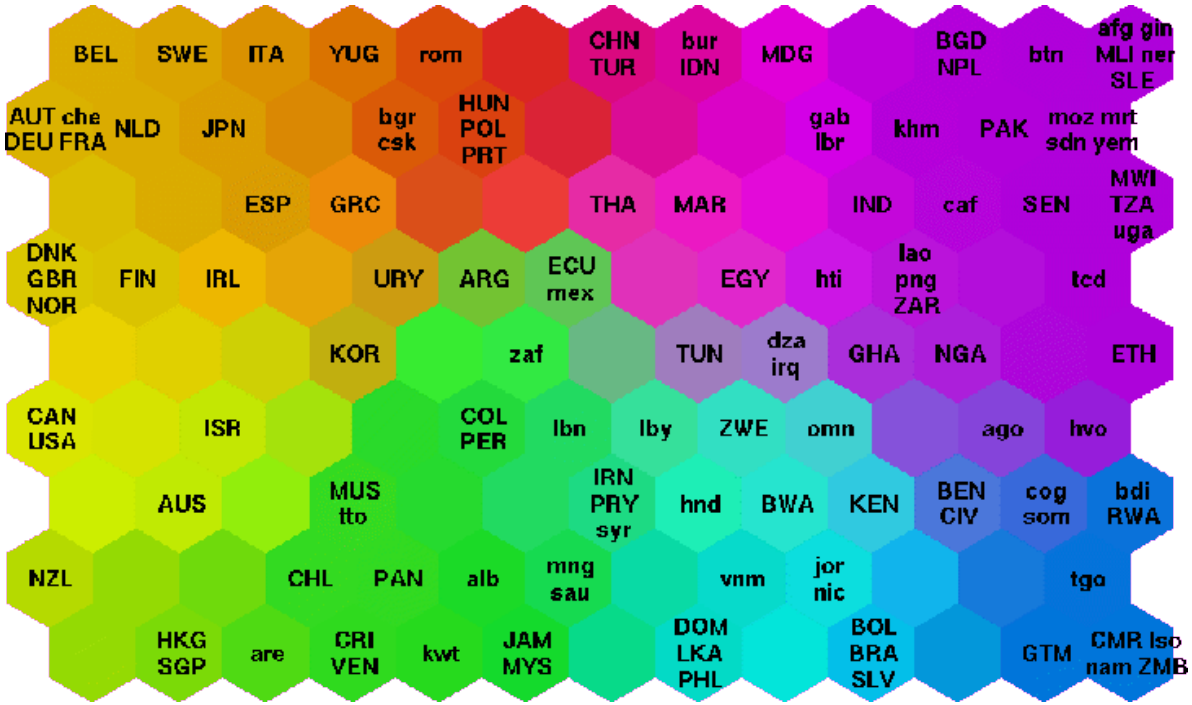
“Foneemikirjoitin”



Kuva: Erilaiset puheen spektrit kuvautuvat SOM-karttapinnalle. Neuronissa on piirrettynä äännettä vastaava spektrikuvio.

◂ Takaisin kalvoihin

“Köyhyyskartta”

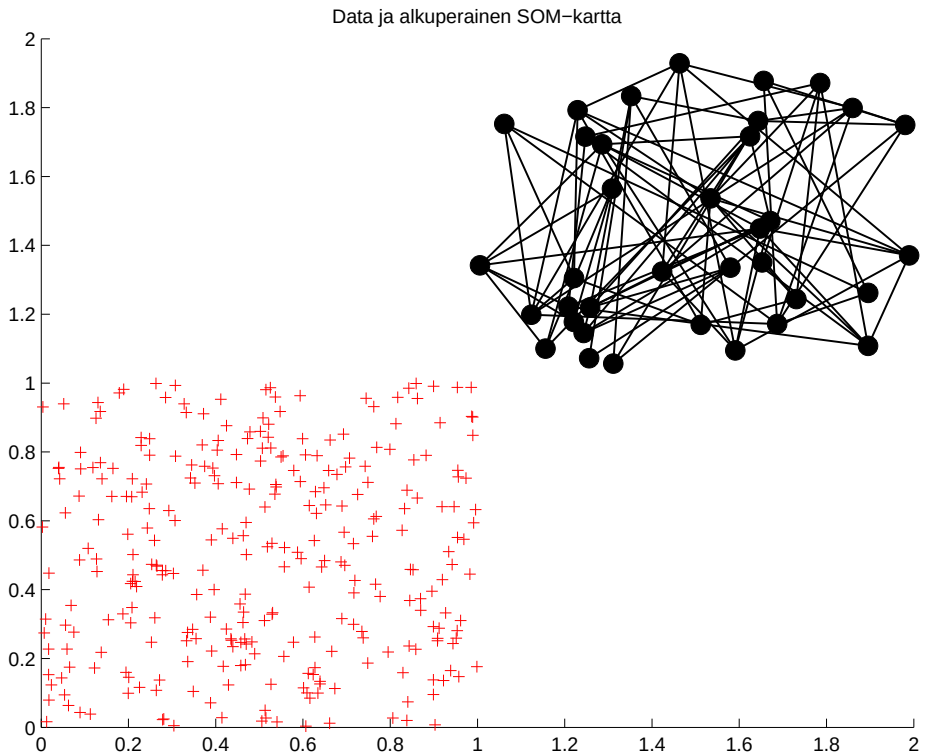


Kuva: Valtioiden taloudelliset indikaattorit SOM-karttapinnalla. Korkeaulotteinen data on saatu visualisoitua mielekkääksi 2-ulotteiseksi kuvaksi. Vierekkäisiin neuroneihin kuvautuu samankaltaisia datavektoreita (maita).

◂ Takaisin kalvoihin

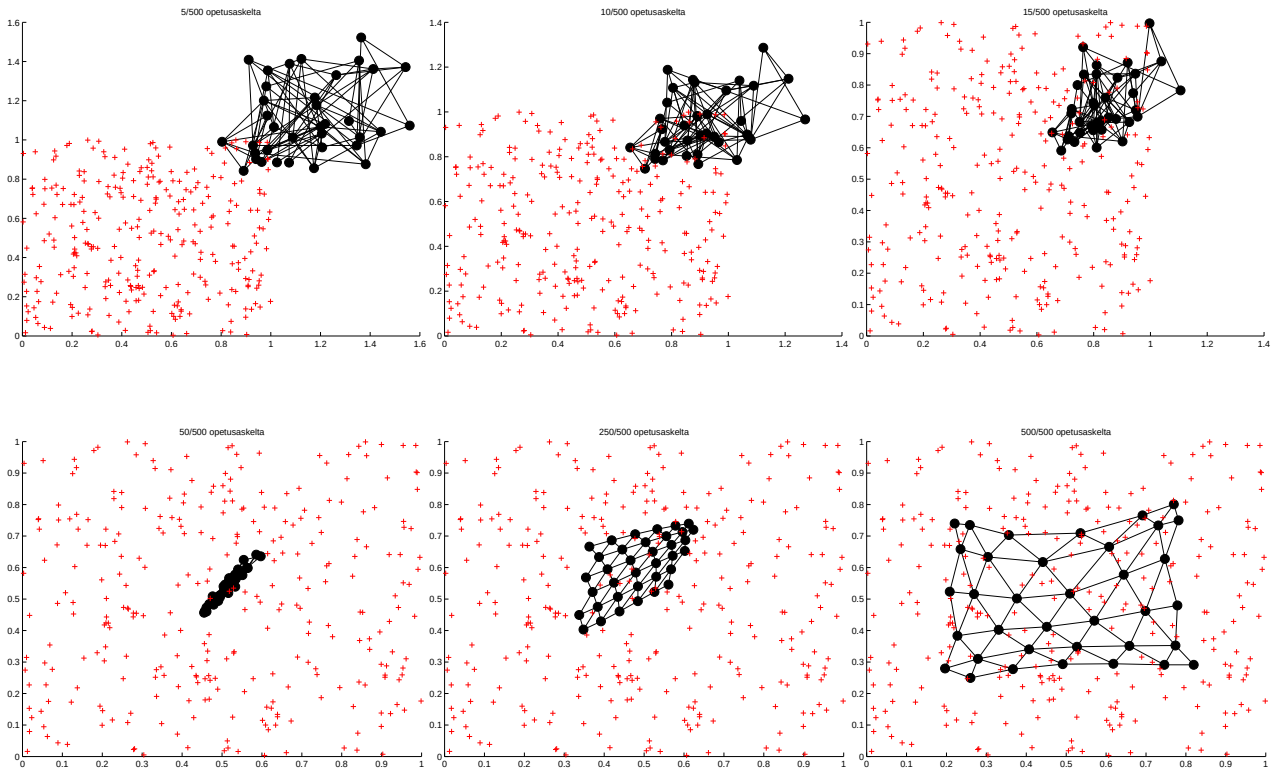
Esimerkki 2D-data ja 2D-kartta opetuksen aikana

Alkutilanne: datanäytteitä (+) yksikköneliössä ja 6×6 -kokoinen suorakulmaisen hilan SOM-kartta järjestäytymättömänä. Mustat pallot painovektoreita $\mathbf{m}_j$.



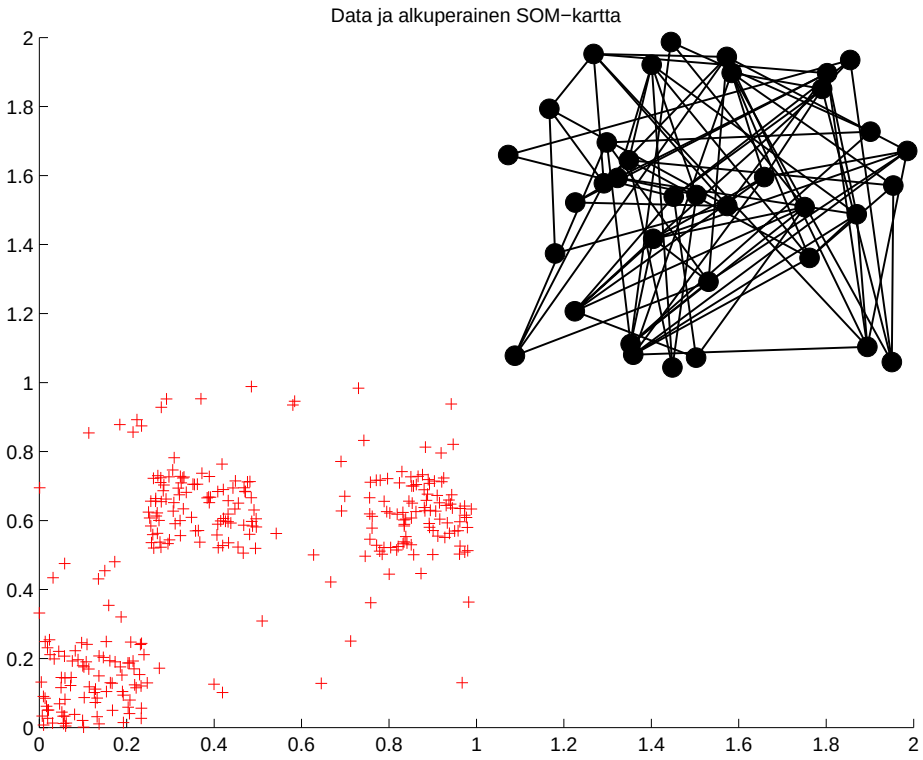
Esimerkki 2D-data ja 2D-kartta opetuksen aikana (2)

Opetuksen aikana kartta levittäytyy ja järjestyy opetusjoukon alueelle. Kuvat 5, 10, 15, 50, 250 ja 500 kierroksen jälkeen.



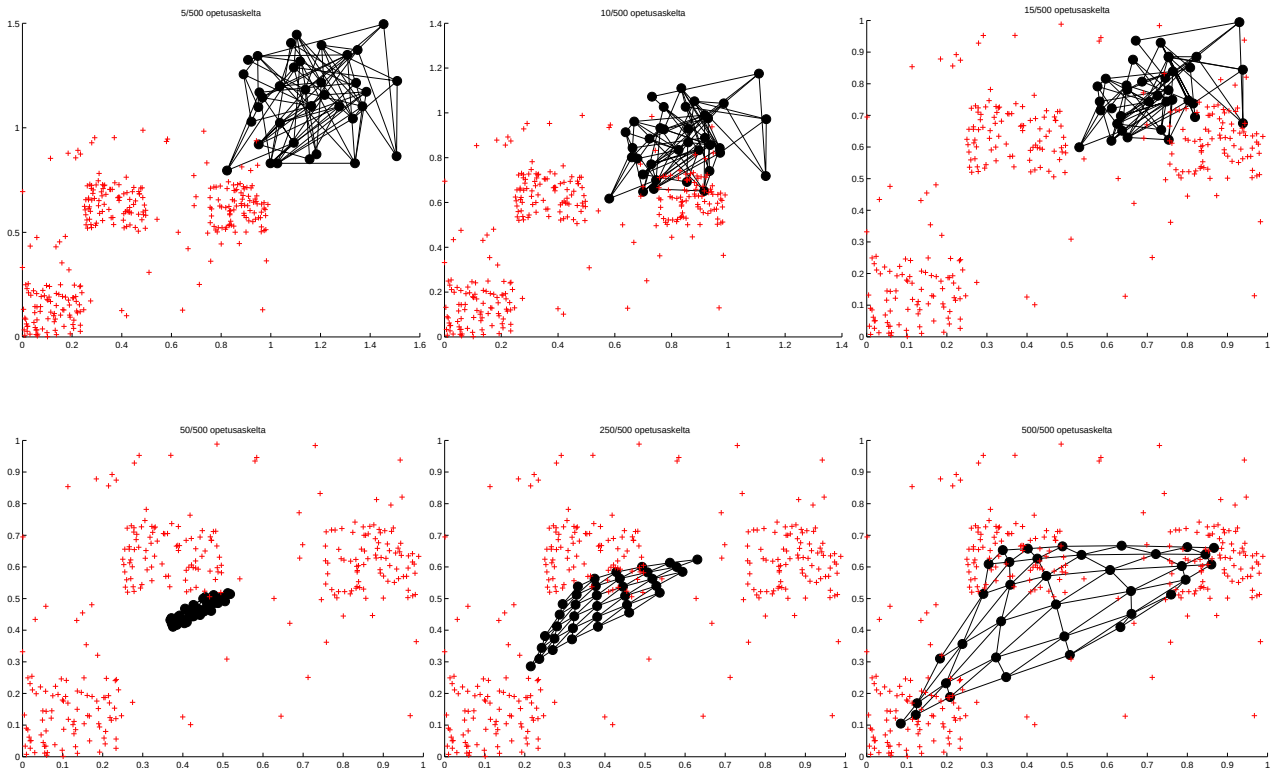
Esimerkki 2D-data ja 2D-kartta opetuksen aikana (3)

Alkutilanne #2: datanäytteitä (+) yksikköneliössä ja 6×6 -kokoinen suorakulmaisen hilan SOM-kartta järjestäytymättömänä. Mustat pallot painovektoreita $\mathbf{m}_j$.



Esimerkki 2D-data ja 2D-kartta opetuksen aikana (4)

Opetuksen aikana kartta levittäytyy ja järjestyy opetusjoukon alueelle. Kuvat 5, 10, 15, 50, 250 ja 500 kierroksen jälkeen.



Lähde: SOM Toolboxin `som_demo1`.

Takaisin kalvoihin

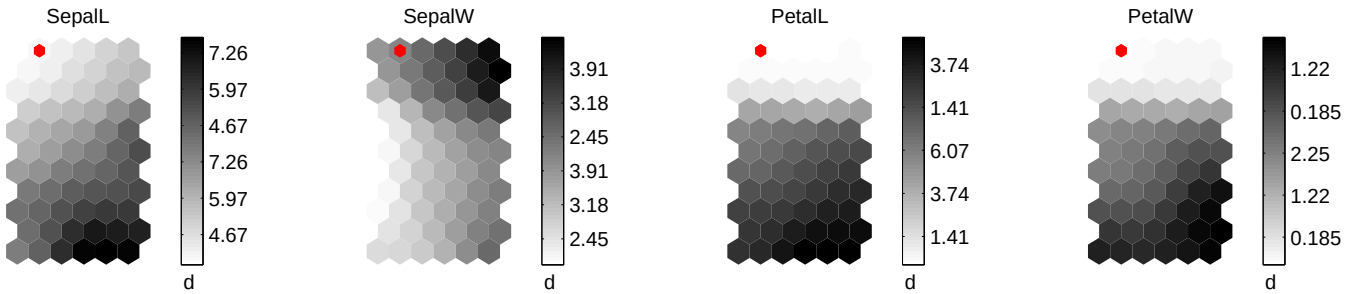
SOM:n komponenttitasoesitys

Datana tässä esimerkissä on käytetty klassista Fisherin liljadataa (Iris), jossa 150 neliulotteista havaintoa kolmesta liljalajista ('Setosa','Virginica', 'Versicolor'). Mitatut ominaisuudet aluslehden ('Sepal') pituus ('L') ja leveys ('W') sekä terälehden ('Petal') vastaavat.

$$\mathbf{X} = \begin{bmatrix} \text{SepalL} & 5.1000 & 4.9000 & \dots \\ \text{SepalW} & 3.5000 & 3.0000 & \dots \\ \text{PetalL} & 1.4000 & 1.4000 & \dots \\ \text{PetalW} & 0.2000 & 0.2000 & \dots \\ & \text{Setosa} & \text{Setosa} & \dots \end{bmatrix}$$

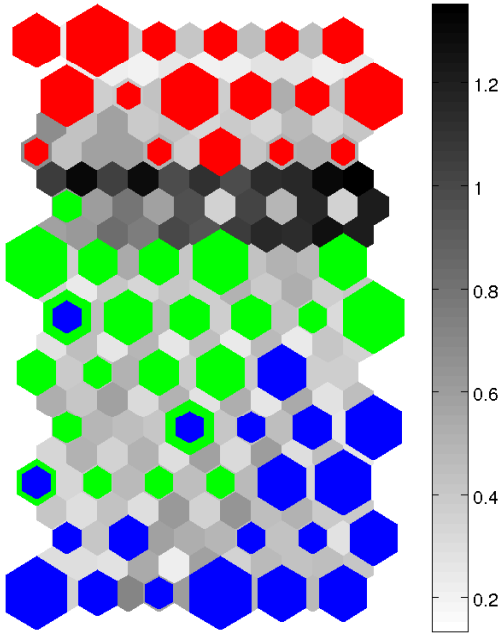
Koska data $\mathbf{x} \in \mathbb{R}^4$, niin myös painovektorit $\mathbf{m} \in \mathbb{R}^4$.

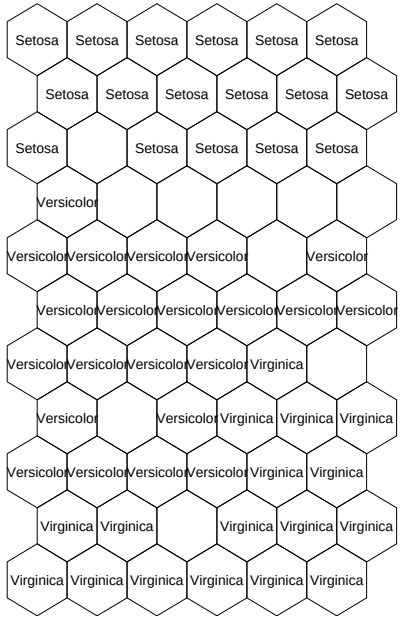
SOM:n komponenttitasoesitys (2)



Painovektorikartat komponenteittain eli ns. komponenttitasot. Yhden painovektorin $\mathbf{m}_{12} \in \mathbb{R}^4$ kohta on osoitettu punaisella pisteellä. Kyseisen kohdan arvot ovat pieniä muuttujille 'SepalL', 'PetalL' ja 'PetalW', mutta verrattain suuri muuttujalle 'SepalW'.

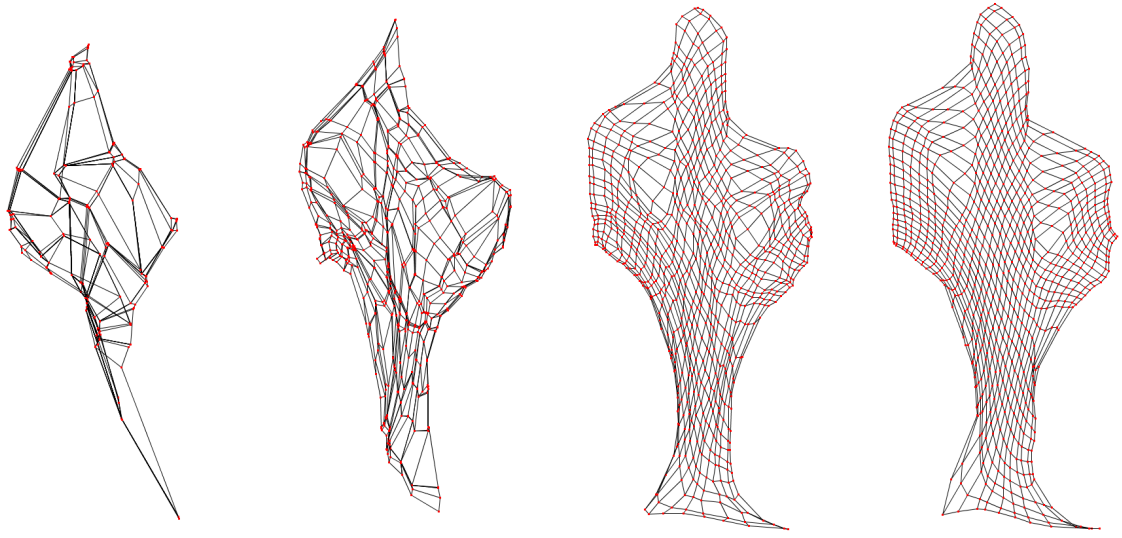
SOM:n komponenttitasoesitys (3)





Vasemmalla ns. U-matriisi, joka näyttää harmaasävyllä vierekkäisten $\mathbf{m}_i$:n etäisyydet, joista voi päätellä klusterirajoja. Värillinen täplä osoittaa kuinka monen havainnon lähin $\mathbf{m}_i$ on. Setosa-laji (punainen) on selkeästi erillään kahdesta muusta lajista, koska 'PetalL' ja 'PetalW' ovat pieniä (kts. komponenttitasot). Oikealla kunkin neuronin pääasiallisin laji. ◂ Takaisin kalvoihin

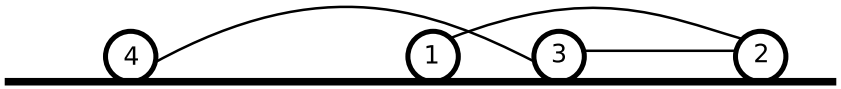
Elastinen verkko: “Kohosen kaktus”



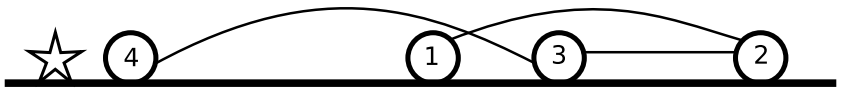
Kuva: “Kohosen kaktus”: kolmiulotteinen data ja kaksiulotteinen SOM. Kartta pyrkii täyttämään kolmiulotteisen muodon taipumalla datan mukaan.

◂ Takaisin kalvoihin

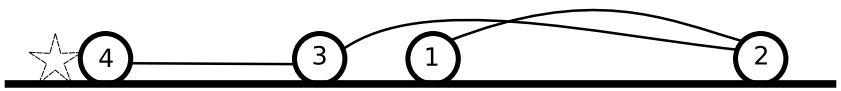
1D-kartan järjestyminen sopivalla 1D-datalla x



Yksiulotteiset painovektorit m_4, m_1, m_3 ja m_2 suoralla. Tavoitteena etsiä sellainen datasyöte x_1, x_2, ... että painovektorit tulevat järjestykseen.

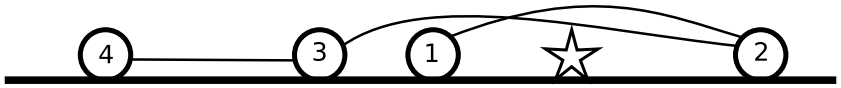


Opetusvektori (piste) x vasemmalla. Lähimpänä m_4.



Voittaja m_4 ja sen naapuri m_3 liikkuvat.

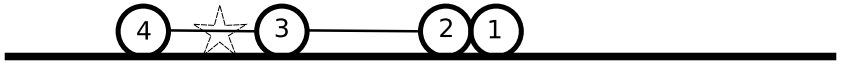
1D-kartan järjestyminen sopivalla 1D-datalla x (2)



Uusi opetuspiste x. Lähimpänä m_1.

Voittaja m_1 ja sen naapuri m_2 liikkuvat.

Uusi opetuspiste x_3. Lähimpänä m_3.



Liikutetaan voittajaa m_3 ja sen naapureita m_2 ja m_4. Nyt kartta on järjestynyt $m_4 < m_3 < m_2 < m_1$.

◂ Takaisin kalvoihin