Datan dimension vaikutus itseorganisoivaan karttaan

Kimmo Kiviluoto (35371P)
Informaatiotekniikan laboratorio / TKK
Tik-61.196 Informaatiotekniikan erikoistyö II

26.2.1996

1 Johdanto

Itseorganisoituva kartta- eli SOM-algoritmi muodostaa kuvauksen mahdollisesti korkeadimensioisesta syöteavaruudesta kartalle, joka tavallisesti on kaksidimensioinen. Kuvaus pyrkii säilyttämään topologian. Silloin, kun syöteavaruuden sisäinen dimensio on korkeampi kuin kartan dimensio, joudutaan kuitenkin tekemään kompromisseja topologian säilyttämisen suhteen. Tämän työn ensimmäisessä osassa (luku 2) tarkastellaan mahdollisuuksia estimoida syöteavaruuden dimensiota, jotta voitaisiin joko käyttää oikean dimensioista karttaa, tai ainakin havaita ristiriita kartan dimension ja syöteavaruuden sisäisen dimension välillä.

Toisessa osassa (luku 3.1) tarkastellaan sitä, minkälainen kuvaus syöteavaruudelta kartalle syntyy, ja miten syöteavaruuden dimensio vaikuttaa tähän kuvaukseen. Lisäksi tutkitaan, millaisia kvantifioitavia kriteerejä kuvauksen hyvyydelle voidaan asettaa. Aiemmin esiteltyjä kuvauksen hyvyyden mittojen esittelyn lisäksi kuvauksen jatkuvuuden kvantifioinnille ehdotetaan uutta mittaria, *topografista virhettä.*

Työn kolmannen osan (luku 4 aiheena on SOMin modifikaatio nimeltään AdSOM. Se pyrkii minimoimaan kuvausvirheiden määrän riippumatta siitä, onko syöteavaruuden sisäinen dimensio sama vai eri kuin kartan dimensio.

2 Syöteavaruuden dimension estimointi

Syötevektorit ovat havaintoja tarkastelun kohteena olevasta ilmiöstä. Kaikki mahdolliset syötevektorien arvot muodostavat ilmiön *tila-avaruuden* $\mathcal{S}$. Usein ilmiön dynamiikka on sellainen, että transienttien kadottua syötevektori voi kuulua vain tiettyyn tila-avaruuden osajoukkoon, jota kutsutaan *attraktoriksi* $\mathcal{A}$; $\mathcal{A} \subset \mathcal{S}$.

Todellisessa mittaustilanteessa on jatkuvasti läsnä tekijöitä, jotka aiheuttavat transientteja, joten mitatut syötevektorit eivät sijaitse tarkalleen attraktorilla. Jokaisen transientin jälkeen ilmiön tila kuitenkin lähestyy attraktoria eksponentiaalisen nopeasti. Siksi valtaosa havaintopisteistä sijaitsee lähellä attraktoria, ja syötevektorit muodostavat enemmän tai vähemmän kohinaisen kuvan attraktorista. Tarkasteltavan ilmiön tuntemisen kannalta olennaista on selvittää, millainen attraktori sillä on.

Merkittävin attraktorin ominaisuuksista on sen *dimensio* $d_{\mathcal{A}}$: se antaa käsityksen ilmiön monimutkaisuudesta. Tila-avaruuden dimensio $d_{\mathcal{S}}$ voi hyvin olla suurempi kuin $d_{\mathcal{A}}$. Tällöin nimellisesti kompleksi (ja sellaiselta näyttävä) ilmiö saattaa osoittautua pohjimmiltaan hyvin yksinkertaisten sääntöjen hallitsemaksi.

Mahdollisuus selittää monimutkaisia ilmiöitä yksinkertaisilla malleilla on ollut yksi suurimpia motivaation lähteitä kaaosteorian tutkimuksessa. Attraktorin toteaminen mataladimensioiseksi ei kuitenkaan vielä auta selittämään ilmiötä – se kertoo vasta, että selittäminen yksinkertaisilla malleilla voisi olla mahdollista. Näiden mallien muodostamisessa on pantu paljon toiveita neuroverkkoihin.

Neuroverkkojen yhteydessä puhutaan usein yksinkertaisesti syöteavaruudesta tekemättä eksplisiittisesti eroa, tarkoitetaanko koko tila-avaruutta $\mathcal{S}$ vai attraktoria $\mathcal{A}$ lähiympäristöineen – lähiympäristöineen siksi, että transienteista johtuen pisteet eivät sijaitse tarkalleen attraktorilla, ja lisäksi todelliset havainnot sisältävät aina jonkin verran mittauksista aiheutuvaa kohinaa. Tila-avaruuden ja attraktorin dimensio ei kuitenkaan yleensä ole sama; siksi puhumalla pelkästä "syöteavaruuden dimensiosta" ei selviä, kumpaa dimensiota ($d_{\mathcal{S}}$ vai $d_{\mathcal{A}}$) tarkoitetaan.

Täsmällisemmin asia voidaan ilmaista, kun todetaan, että ilmiöstä kokeiden avulla saatavat syötevektorit $\mathbf{x} \in \mathbb{R}^n$ ovat tyypillisesti sijoittuneet johonkin $\mathbb{R}^n$:n osajoukkoon $\mathcal{V}$, jossa niiden tiheys ei välttämättä ole vakio. Määritellään nyt, että *syöteavaruus* on joukko $\mathcal{V} \subset \mathbb{R}^n$, *syöteavaruuden geometrinen dimensio* on n, ja *syöteavaruuden luonnollinen dimensio* (tai sisäinen dimensio) on joukon $\mathcal{V}$ dimensio $d_{\mathcal{V}}$. Tällöin pätee $d_{\mathcal{S}} = n$ ja $d_{\mathcal{A}} \approx d_{\mathcal{V}}$.

Esimerkistä käyköön tilanne, jossa syötevektorit $\mathbf{x} \in \mathcal{V}$ sijaitsevat pinnalla $\mathbb{R}^3$:ssa. Tällöin geometrinen dimensio $n = 3$ ja luonnollinen dimensio $d_{\mathcal{V}} = 2$.

2.1 Erilaisia määritelmiä joukon dimensiolle

Intuitiivisesti määriteltynä dimensio kuvaa sitä, miten joukon mitta (esimerkiksi tilavuus) kasvaa sen koon (esimerkiksi säde) kasvaessa:

$$\text{mitta} = \text{koko}^{\text{dimensio}}$$
$$\Leftrightarrow \text{dimensio} = \lim_{\text{koko}\to 0} \frac{\log \text{mitta}}{\log \text{koko}},$$

missä raja-arvo on otettu siksi, että dimensio on luonteeltaan lokaali suure. Täsmällisemmin dimensio voidaan määritellä usealla vaihtoehtoisella tavalla; tässä luvussa esitellään niistä yleisimmin käytetyt. Luku seuraa varsin pitkälle Farmer et al:n käsittelyä [1].

2.1.1 Hausdorff-dimensio

Ensimmäisen (ja matemaattisesti täsmällisimmän) määritelmän joukon dimensiolle on esittänyt Hausdorff v. 1919. n-dimensioisessa euklidisessa avaruudessa sijaitsevan joukon *Hausdorff-dimensio* määritetään seuraavasti: Joukko peitetään n-dimensioisilla kuutioilla, joiden sivun pituudet ϵ_i voivat olla erisuuria. Määritellään

$$l_d(\epsilon) = \inf \sum_i \epsilon_i^d, \tag{1}$$

missä infimum on otettu yli kaikkien mahdollisten peitteiden, joissa $\epsilon_i \leq \epsilon \; \forall i$. Olkoon nyt

$$l_d = \lim_{\epsilon\to 0} l_d(\epsilon). \tag{2}$$

Tällöin d:lle on olemassa tietty kriittinen arvo, jonka yläpuolella $l_d = 0$ ja alapuolella $l_d = \infty$. Tämä kriittinen arvo $d = d_H$ on joukon Hausdorff-dimensio.

Hausdorff-dimension määritelmässä esiintyvän infimumin laskeminen ei käytännön datasta onnistu, koska se pitäisi etsiä yli kaikkien mahdollisten peitteitten. Siksi Hausdorff-dimensio on lähinnä teoreettinen työkalu; joukon dimension estimointia varten on tarkasteltava muilla tavoin määriteltyjä dimensioita.

2.1.2 Kapasiteetti

Joukon *kapasiteetin* eli *box-counting-dimension* d_B määritteli Kolmogorov seuraavasti:

$$d_B = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)}, \tag{3}$$

missä $N(\epsilon)$ on joukon ($\subset \mathbb{R}^n$) peittämiseen tarvittava ϵ-sivuisten n-dimensioisten kuutioiden lukumäärä.

Kapasiteetin määritelmässä kaikki kuutiot ovat saman kokoisia, joten sen laskeminen on varsin suoraviivaista. Käytännön tilanteissa näytevektoreita on aina äärellinen määrä, samoin resoluutio on äärellinen, joten rajalle $\epsilon \to 0$ ei päästä, vaan on tutkittava, miten d_B käyttäytyy pienillä ϵ:in arvoilla: $\mathbb{R}^n$ diskretoidaan ϵ-sivuisiksi kuutioiksi ja lasketaan, kuinka moneen kuutioon osuu vähintään yksi näytevektori. Toistamalla tämä usealla eri ϵ:in arvolla ja etsimällä mahdollisimman hyvin $(N(\epsilon), \epsilon)$-pareihin sopiva luku saadaan estimaatti d_B:lle.

Yleispätevä ratkaisu dimension määrittämiseen kapasiteetti ei ole. Sen suurin heikkous on, että sitä laskettaessa ei tehdä mitään eroa sen suhteen, kuinka monta näytevektoria kuutioon osuu. Syöteavaruuden $\mathcal{V} \in \mathbb{R}^n$ tiheys on tyypillisesti hyvin epätasainen $\mathbb{R}^n$:ssä: se on suurimmillaan attraktorin lähellä, ja pienenee etäisyyden kasvaessa attraktorista. Kuitenkin, jos kohinan osuus on suuri – mikä voi johtua joko siitä, että mitattattavassa ilmiössä on jatkuvasti transientteja, tai siitä, että itse mittauksissa todellisen signaalin lisäksi on merkittävä kohinakomponentti –, voi $\mathcal{V}$:n tiheys olla pieni mutta nollasta poikkeava laajalla alueella attraktorin ulkopuolellakin. Tällöin $d_B \to n$, kun syötevektorien määrä kasvaa.

2.1.3 Informaatiodimensio

Informaatiodimension esittelivät Balatoni ja Renyi. Tarkastellaan suuretta

$$I(\epsilon) = -\sum_i P_i \log P_i, \tag{4}$$

missä P_i on kuution i sisältämä todennäköisyys; kokeellisesta aineistosta se määritetään kuution i sisältämän syötevektorien määrän suhteena kaikkien vektorien määrään. Voidaan ajatella, että $I(\epsilon)$ kuvaa keskimääräistä informaatiota, joka saadaan mittauksella, jonka tarkkuus on ϵ.

Informaatiodimensio d_I saadaan kapasiteetin määritelmästä korvaamalla siinä esiintyvä ei-tyhjien kuutioiden määrän $N(\epsilon)$ logaritmi $I(\epsilon)$:illa:

$$d_I = \lim_{\epsilon \to 0} \frac{I(\epsilon)}{\log(1/\epsilon)}, \tag{5}$$

2.1.4 Korrelaatiodimensio

Viimeinen tässä käsiteltävä dimension määritelmä on korrelaatiodimensio, jonka esittelivät Grassberger ja Procaccia [2]. Se määritellään seuraavasti:

$$d_C = \lim_{\epsilon \to 0} \frac{\log C(\epsilon)}{\log(1/\epsilon)}, \tag{6}$$

missä $C(\epsilon)$ on tarkasteltavan joukon *korrelaatiointegraali*

$$C(\epsilon) = \frac{1}{N(N-1)} \sum_{i \neq j} u(r - \|\mathbf{x}_i - \mathbf{x}_j\|), \tag{7}$$

missä N on syötevektorien määrä ja u on Heavisiden askelfunktio: $u(a) = 0$, kun $a < 0$, $u(a) = 1$ muulloin. $C(\epsilon)$ siis kertoo niiden vektoriparien, joiden välinen etäisyys on pienempi kuin ϵ, osuuden kaikista vektoripareista.

2.1.5 Yhteyksiä

Voidaan osoittaa [1], että aina $d_H \leq d_B$. Lisäksi kapasiteetti, informaatio- ja korrelaatiodimensio ovat erikoistapauksia yleisemmästä dimension käsitteestä d_q, $q \in \mathbb{R}$, jolle pätee $q_1 < q_2 \Rightarrow d_{q_1} \geq d_{q_2}$, missä yhtäsuuruus pätee vain homogeenisille fraktaaleille eli objekteille, joilla sama rakenne toistuu läpi pituusasteikon. Koska edellä esitellyt dimensiot d_B, d_I ja d_C vastaavat q:n arvoja 0, 1 ja 2, tässä järjestyksessä, saadaan edellä esitellyille dimensioille seuraava keskinäisen järjestys: $d_B \geq d_I \geq d_C$ [3].

Sekä teoreettiset tarkastelut että kokeet idealisoiduissa tilanteissa ($N \to \infty$, ei kohinaa, ääretön resoluutio eli $\epsilon \to \infty$) osoittavat, että joitakin harvinaisia poikkeuksia lukuunottamatta eri tavoin määritellyt dimensiot ovat yleensä varsin lähellä toisiaan. Käytännössä merkittävimmät erot niiden välillä ovat erilaiset laskennalliset kustannukset (vrt. seuraava kappale) sekä erilainen sietokyky reaalisen datan epäideaalisuuksille (kts. esim. [2, 4]).

2.2 Algoritmeja dimension laskemiseen

Joukon dimension kokeelliseen määrittämiseen soveltuvat parhaiten informaatiodimensio d_I ja korrelaatiodimensio d_C – d_H:n määrittäminen on käytännössä mahdotonta, ja d_B toimii huonosti reaalisella, kohinaa sisältävällä datalla. Äärellinen resoluutio ja syötevektorien määrä N estävät suoraan etsimästä dimension raja-arvoa ϵ:in lähestyessä nollaa, joten dimensio täytyy määrittää kappaleessa 2.1.2 kuvatulla tavalla: lasketaan $I(\epsilon)$ tai $C(\epsilon)$ usealla eri ϵ:in arvolla, ja laaditaan tuloksista kuvaaja logaritmiselle asteikolle, jolloin kuvaajan kulmakertoimena saadaan estimaatti d_I:lle tai d_C:lle.

Teoreettiselle datalle edellä luonnosteltu lähestymistapa tuntuu sopivan hyvin. Reaaliselle, kohinaiselle datalle kulmakertoimen – ja siten myös dimension – määrittäminen kuvaajista on usein ongelmallista. Theiler [4] on käsitellyt perusteellisesti dimension määrittämisessä vastaantulevia ongelmia ja mahdollisia ratkaisuja niihin; tässä ei tila riitä kuin muutaman tärkeimmän ongelman mainitsemiseen.

2.2.1 Syötevektoreiden määrä

Tarvittava syötevektoreiden määrä N kasvaa nopeasti joukon dimension kasvaessa. Informaatiodimension kohdalla syynä tähän on se, että kun joukon dimensio on suuri, vektorit ovat hajallaan ympäri avaruutta ja ϵ:n pienetessä kuhunkin laatikkoon riittää vain hyvin pieni määrä vektoreita. Kun ϵ:a pienennetään tarpeeksi paljon, jokaiseen laatikkoon riittää lopulta vain yksi piste, jolloin $I(\epsilon)$ ei enää riipu lainkaan dimensiosta vaan ainoastaan N:stä.

Korrelaatiodimension tapauksessa Ruelle [5] on esittänyt seuraavan tuloksen tarvittavalle N:lle. Oletetaan, että $I(\epsilon)$ tai $C(\epsilon)$ vs. ϵ -kuvaajan kulmakerroin on määritetty ϵ-akselilla yhtä dekadia vastaavalta alueelta, mikä on lähellä lyhintä mahdollista, jotta kulmakertoimen saisi luotettavasti määritettyä. Tällöin joukon dimensiosta riippumatta on voimassa

$$\text{kulmakerroin} \leq 2\log_{10} N \tag{8}$$

Siksi dimensioestimaatti, joka ei ole selvästi pienempi kuin $2\log_{10} N$, ei välttämättä ole luotettava – se kertoo, että kulmakerroin on täytynyt määrittää epäilyttävän lyhyeltä matkalta.

2.2.2 Kohina

Merkitään kohinan amplitudia σ:lla. Tällöin $I(\epsilon)$ tai $C(\epsilon)$ vs. ϵ-kuvaajan alueella, jossa $\epsilon \leq \sigma$, kohinan osuus nousee merkittäväksi. Valkoisen kohinan dimensio on aina n, sama kuin tila-avaruuden dimensio. Siksi korrelaatiodimension kuvaajan kulmakerroin nopeasti kasvaa, kun ϵ pienenee arvosta $\epsilon = \sigma$. Joukon dimensio näyttää tällöin erilaiselta eri etäisyyksiltä katsottuna.

Informaatiodimension kuvaajasta kohina-alue on vaikeampi löytää kuin korrelaatiodimension kuvaajasta. Se näkyy lähinnä kuvaajan loivenemisena aikaisemmin kuin jos kohinaa ei olisi, sillä kohina saa pisteet "loppumaan" nopeammin ϵ:n pienetessä. Tämän avulla ei kuitenkaan juuri pysty arvioimaan kohinan amplitudia – on vaikea päätellä, johtuuko kuvaajan tasaantuminen jossakin tietyssä kohdassa kohinasta vai siitä, että pisteet alkavat muutenkin loppua.

2.2.3 Virhearviot

Theilerin [4] mukaan yleispäteviä menetelmiä d_I:n ja d_C:n virheen arvioimiseksi ei ole. Syitä tähän on useita. Jo kulmakertoimen teoreettisen virheen määrittäminen on vaikeaa, koska eri ϵ:n arvoilla saadut havaintopisteet eivät selvästikään ole riippumattomia. Tätäkin suurempi ongelma aiheutuu havaintojen epäideaalisuuksista, jotka riippuvat suuresti kulloisestakin tilanteesta.

Näyttäisi siltä, että estimoitaessa dimensiota edellä esitetyin menetelmin joudutaan tyytymään silmämääräiseen haarukointiin: etsitään suurin ja pienin arvo havaintoaineistosta piirrettyyn kuvaajaan sopivan jänteen kulmakertoimelle, ja koitetaan lisäksi löytää sille parhaiten sopiva arvo. Näin saadaan dimensioestimaatille karkeat rajat sekä todennäköisin arvo.

2.3 Kokeilut

2.3.1 Aineisto ja menetelmät

Edellisessä luvussa kuvattuja menetelmiä kokeiltiin käytännössä sekä keinotekoisella että todellisia ilmiöitä kuvaavalla datalla. Generoidut datajoukot olivat seuraavat:

- `d1-3` tasaisesta jakaumasta yksikköjanalla arvottiin 1 000 pistettä, ja niille tehtiin koordinaatistokierto siten, että ne sijaitsivat 30-ulotteisen yksikkökuution lävistäjällä
- `d1-4` kuin d1-3, mutta arvottiin 10 000 pistettä
- `d2-3` tasaisesta jakaumasta yksikköneliössä arvottiin 1 000 pistettä, ja ne kuvattiin 30-ulotteiseen yksikkökuutioon pinnaksi erilaisten alkeis- ja porrasfunktioiden avulla
- `d2-4` kuin d2-3, mutta arvottiin 10 000 pistettä
- `d2-5` kuin d2-3, mutta arvottiin 100 000 pistettä
- `d2-3n` saatiin d2-3:sta lisäämällä sen vektoreiden jokaiseen komponenttiin satunnaista, välille [0, 0.05] tasan jakautunutta kohinaa
- `d2-4n` saatiin d2-4:stä lisäämällä sen vektoreiden jokaiseen komponenttiin satunnaista, välille [0, 0.05] tasan jakautunutta kohinaa
- `twocl` arvottiin 2 000 pistettä kahdesta kaksiulotteisesta normaalijakaumasta, joiden keskipisteet olivat (-1,0):ssa ja (1,0):ssa, joiden kovarianssimatriisina oli identtimatriisi ja joiden *a priori*-todennäköisyydet olivat samat

Todellisia ilmiöitä kuvaavat datajoukot olivat seuraavat:

- `meesa` 6 616 kappaletta 36-ulotteisia paperitehtaan meesauunin tilaa kuvaavia mittausvektoreita
- `hehku` 5 159 kappaletta 27-ulotteisia terästehtaan kolmesta eri hehkuttamosta valmistuneita teräsrullia kuvaavia mittausvektoreita
- `welfare` 77 kappaletta maailman valtioita kuvaavia tunnuslukuvektoreita; komponentteja oli 39 kappaletta, mutta jotkin niistä puuttuivat useimmista vektoreista; siksi estimaatti laskettiin myös käyttäen vain 13 komponenttia, jotka olivat mukana kaikissa vektoreissa

Informaatiodimensio laskettiin siten, että syötevektorit normalisoitiin ensin yksikkökuutioon. Tämän jälkeen komponentit diskretoitiin diskretointiaskeleena ϵ niin, että ne kerrottiin $1/\epsilon$:lla ja pyöristettiin lähimpään kokonaislukuun alaspäin. Koska samassa kuutiossa sijaitsevat vektorit saadaan näin identtisiksi, niiden määrät voidaan nopeasti laskea konkatenoimalla vektoreiden komponentit täydennettynä nollilla saman pituisiksi, lajittelemalla (quicker sort tässä tapauksessa, mutta sopiva muunnos radix sortista toimisi nopeammin) näin saadut merkkijonot ja laskemalla perättäisten samojen merkkijonojen määrät. Näistä saadaan edelleen laskettua informaatiodimension laskukaavassa (5) esiintyvä $I(\epsilon)$ kaavan (4) avulla. Kuvatun algoritmin laskennallinen kompleksisuus on $O(N \log N)$.

Korrelaatiodimensio määritettiin laskemalla datavektoripareille niiden vektoreiden välinen etäisyys. Etäisyydet diskretoitiin, ja kullekin diskretointiaskelmalle osuvien vektoriparien määrä laskettiin, minkä tuloksena saatiin korrelaatiodimension laskukaavassa (6) esiintyvä korrelaatiointegraali $C(\epsilon)$ evaluoituna tietyillä diskreeteillä ϵ:n arvoilla. Laskennallinen kompleksisuus oli $O(N^2)$ eli selvästi informaatiodimension laskualgoritmia suurempi. Tästä syystä korrelaatiodimensiota ei laskettu laajimmille datajoukoille d1-4, d2-4, d2-4n ja d2-5.

Reaalidatassa esiintyi vektoreita, joista puuttui komponentteja. Informaatiodimension tapauksessa nämä karsittiin kokonaan datajoukosta; korrelaatiodimensiota laskettaessa puuttuvien komponenttien välinen etäisyys tulkittiin suurimmaksi mahdolliseksi.

Lopuksi kummassakin tapauksessa laadittiin logaritminen kuvaaja $I(\epsilon)$:sta tai $C(\epsilon)$:sta. Informaatiodimension tapauksessa kuvaajaan sovitettiin jänne suurimmasta käytetystä ϵ-arvosta ($\epsilon = 0.5$) siihen ϵ:n arvoon, jossa loiveneminen alkoi. Korrelaatiodimension tapauksessa kuvaajaan sovitettiin silmämääräisesti parhaiten sopivan näköinen jänne, jonka kulmakerroin antoi estimaatin dimensiolle (periaate näkyy tuloksia esittelevissä kuvissa 1 ja 2).

2.3.2 Tulokset

Datajoukoille laaditut dimensioestimaatit on koottu taulukkoon 1, ja korrelaatiointegraalikuvaajat on esitetty kuvissa 1 ja 2.

datajoukko	geom.dim.	N	d	$\hat{d}_I$	$\hat{d}_C$	$\hat{d}_C^{\min} \ldots \hat{d}_C^{\max}$
`d1-3`	30	1 000	1	1.0	1.0	0.9 ... 1.0
`d1-4`	30	10 000	1	1.0		
`d2-3`	30	1 000	2	2.3	1.7	1.7 ... 1.9
`d2-4`	30	10 000	2	2.4		
`d2-5`	30	100 000	2	2.5		
`d2-3n`	30	1 000	2	2.3	1.9	1.7 ... 12
`d2-4n`	30	10 000	2	2.8		
`twocl`	2	2 000	<2	1.5	1.8	1.4 ... 1.9
`meesa`	36	6 616	?	2.4	3.0	2.0 ... 4.0
`hehku`	27	5 159	?	4	4	3.2 ... 5.6
`welfare`	13 ... 39	77	?	1.4	5.5	5.0 ... 8.0

Taulukko 1: Käytetyt datajoukot ja niille lasketut dimensioestimaatit

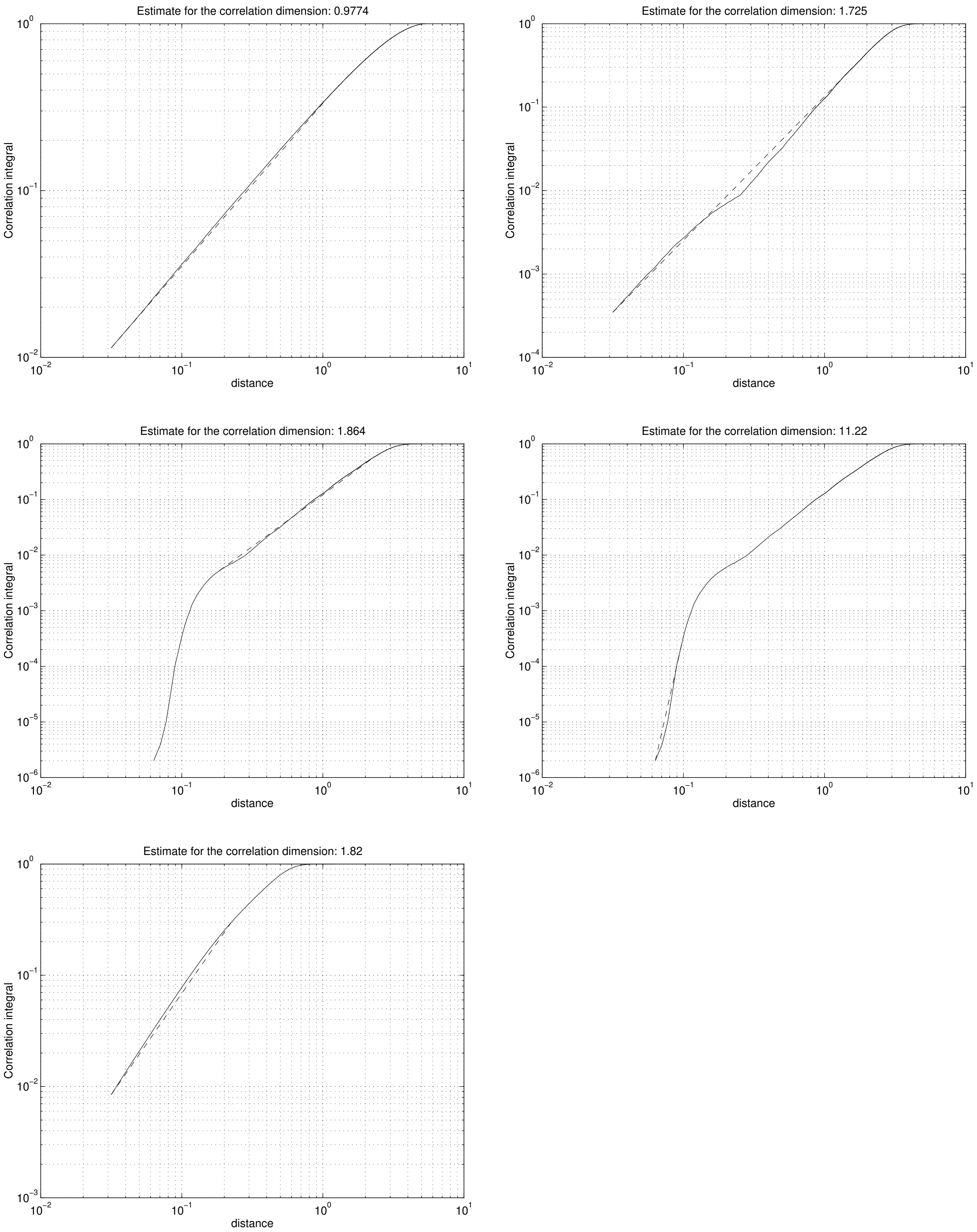
2.4 Estimointitulosten tarkastelua

Dimension estimoinnissa sekä korrelaatio- että informaatiodimension laskemisessa käytetyt algoritmit näyttävät toimivan hyvin, kun datapisteitä on riittävästi ja data on kohinatonta. Datapisteiden määrän pienetessä ja varsinkin kohinan lisääntyessä informaatiodimensioalgoritmin käyttökelpoisuus pienenee nopeasti.

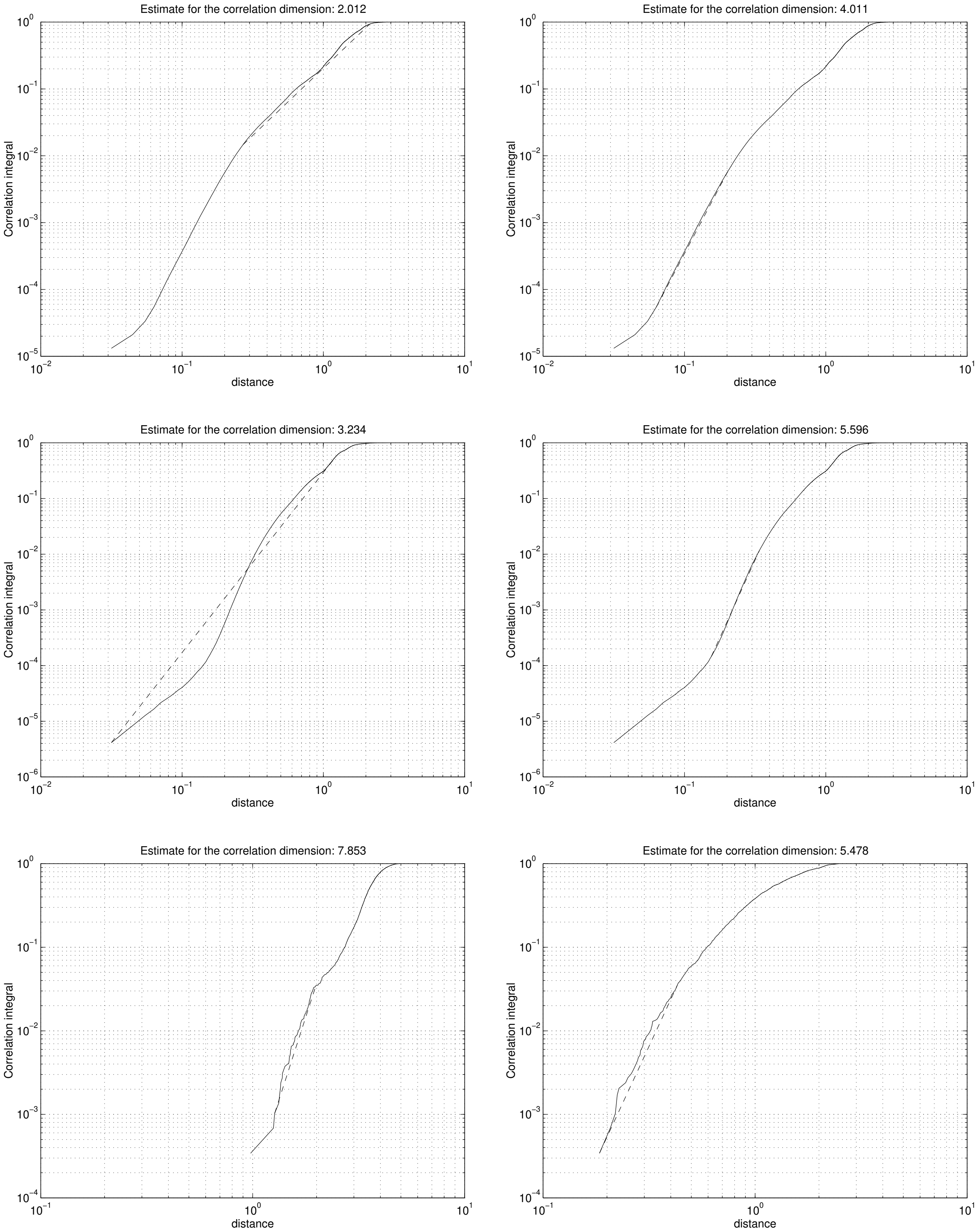
Korrelaatiodimensio ei ole aivan yhtä herkkä datapisteiden määrälle kuin informaatiodimensio. Luotettavien estimaattien saamiseksi korrelaatiointegraalin kulmakerroin on kuitenkin määritettävä riittävän pitkältä alueelta, mieluiten vähintään yhden dekadin matkalta pituusasteikolla, ja kuten kappaleessa 2.2.1 todettiin, tällöin dimensioestimaatille pätee aina rajoitus (8). Esimerkiksi datajoukossa `welfare` on selvästi liian vähän pisteitä, jotta dimensioestimaattia voitaisiin pitää edes suuntaa-antavana.

Mielenkiintoisia olivat kokeilut keinotekoisella datalla, jossa monidimensioisessa avaruudessa sijaitsevaan pintaan oli lisätty kohinaa. Korrelaatiointegraalin kohina-amplitudin ala- ja yläpuolella olevat pituusalueet erottuvat selvästi toisistaan, kuten kappaleen 2.2.2 mukaan pitäisikin tapahtua. Siten korrelaatiointegraalia voidaan käyttää hyväksi paitsi dimension estimoinnissa, myös ilmiön yleisemmässä tarkastelussa: sen avulla voidaan päätellä jotakin ilmiön erilaisista ominaisuuksista eri pituusalueilla. Esimerkiksi toisistaan klusterit, joiden sisäinen dimensio on toinen kuin klustereiden välinen, saattaisivat muodostaa korrelaatiointegraaliin erilliset alueet. Datajoukossa `twocl` ilmiö voidaan tosin havaita vain erittäin hyvällä tahdolla jos ollenkaan; syynä lienee se, että klusterit sijaitsevat hyvin lähellä toisiaan.

Datajoukkojen `meesa` ja `hehku` korrelaatiointegraalit loivenivat yllättävästi pituusasteikon alapäässä. Ilmiö aiheutuu datan keruussa tehtyjen mittausten äärellisestä tarkkuudesta: datavektoreiden komponenteista moni on ilmaistu vain yhdellä tai kahdella merkitsevällä numerolla.



Kuva 1: Generoitujen datajoukkojen korrelaatiointegraalikuvaajat. Ylinnä datajoukot `d1-3` ja `d2-3`, keskellä datajoukko `d2-3` jänne sovitettuna sekä kohinattomalle että kohinaiselle alueelle, ja alhaalla `twocl`.



Kuva 2: Reaalisten datajoukkojen korrelaatiointegraalikuvaajat. Ylinnä datajoukko `meesa`, keskellä `hehku`, ja alhaalla `welfare` sekä osittain vajailla 39- että täysillä 13-komponenttisilla datavektoreilla.

On syytä huomauttaa, että käytetyt reaaliset datajoukot olivat erittäin hankalia. Se, että dimension estimointi niille antoi tulokseksi vain karkeat ylä- ja alarajat, ei välttämättä anna vielä kovin hyvää kuvaa estimointialgoritmien toimivuudesta; vaadittaisiin lisää kokeiluja mieluiten sellaisella datalla, josta tiedetään, että se kuvaa jotakin tietyn dimensioista ilmiötä.

Tuloksista voidaan kuitenkin vetää se johtopäätös, että dimension estimointi on – ainakin jollakin tarkkuudella – mahdollista, jos vain datapisteitä on riittävästi. Korrelaatiodimensio näyttäisi olevan informaatiodimensiota käyttökelpoisempi ja tarkempi estimointialgoritmi, ja hiukan tehokkaammin koodaamalla se on myös nopeasti laskettavissa suurillekin datajoukoille (kts. esim. [6]). Korrelaatiodimensiota voisi siis käyttää yhtenä ilmiön analysoinnin työkaluna, ennenkaikkea antamaan karkea käsitys ilmiön monimutkaisuudesta.

Ongelmia dimension estimoinnissa saattaa kuitenkin tulla vastaan runsaasti, vrt. kappale 2.2. Lisäksi pelkkä ilmiön dimension tietäminen ei yleensä riitä vielä kovin pitkälle. Paljon arvokkaampaa olisi saada rakennetuksi ilmiötä kuvaava malli, jota voi käyttää hyväksi esimerkiksi riippuvuussuhteita analysoitaessa tai ennusteita laadittaessa. Mallista usein selviää myös ilmiön dimensio: jos esimerkiksi aikasarjaa ennustettaessa seuraava arvo voidaan luotettavasti ennustaa kolmen edellisen avulla, dimensio lienee lähellä kolmea.

Samalla tavoin voidaan kääntää kysymys siitä, miten selvittää ilmiön dimensio, jotta osataan valita oikea dimensio mallina käytettävälle itseorganisoivalle kartalle: parhaiten dataan sovittuvan SOMin dimensio on todennäköisesti lähellä ongelman dimensiota. Tarvitaan siis jonkinlaisia kriteerejä SOMin dataan sovittumisen arvioimiseksi; näitä tarkastellaan seuraavassa luvussa.

3 Hyvyyskriteerejä itseorganisoivan kartan muodostamalle kuvaukselle

3.1 Kartan muodostaman kuvauksen jatkuvuus ja erottelukyky

Itseorganisoivan kartan muodostama kuvaus voidaan esittää muodossa

$$\mathrm{f} : \mathcal{A} \to \mathcal{B}, \mathcal{A} \subset \mathbb{R}^m, \mathcal{B} \subset \mathbb{N}^d$$

m-dimensioisesta datamonistosta $\mathcal{A}$ d-dimensioiselle diskreetille kartalle $\mathcal{B}$. Ideaalitilanteessa kuvaus on mahdollisimman lähellä jatkuvaa bijektiivistä kuvausta, jonka käänteiskuvaus on samoin jatkuva (ts, homeomorfismia)

$$\mathrm{g} : \mathcal{A} \to \mathcal{C}, \mathcal{A} \subset \mathbb{R}^m, \mathcal{C} \subset \mathbb{R}^d_+;$$

Tällöin olisi sopivasti g:tä skaalaten voimassa $\mathrm{f} = \lceil \mathrm{g} \rceil$. Tällöin lähekkäin olevat datapisteet, ja vain ne, kuvautuvat kartalla lähekkäin oleviin pisteisiin (suora seuraus kuvauksen g homeomorfisuudesta).

Olkoon m datan geometrinen dimensio ja n datan luonnollinen dimensio (aina pätee $m \geq n$). Jos $n = d$, on mahdollista löytää edellä luonnehdittu g ja siten myös f. Tällöin kartan diskreettisyydestä aiheutuvaa virhettä lukuunottamatta topologia säilyy kuvauksessa.

Silloin kun $n > d$, kuvaus g ei voi olla bijektio, vaan se on väistämättä surjektio: joukon $\mathcal{C}$ pisteillä ei ole yksikäsitteistä alkukuvaa joukossa $\mathcal{A}$. Merkitään tällaisessa tapauksessa mielivaltaisen pisteen $\mathbf{y} \in \mathcal{C}$ alkukuvaa $\mathrm{g}^{-1}(\mathbf{y}) \subset \mathcal{A}$, ja määritellään kartan erottelukyky joukkojen

$g^{-1}(\mathbf{y})$ keskimääräisenä halkaisijana. Toisin sanoen, mitä pienempiä joukkojen $g^{-1}(\mathbf{y})$ halkaisijat keskimäärin ovat, sitä erottelukykyisempi kartta.

Kartan erottelukyky voi olla sovellusten kannalta liian heikko, jolloin kartta ei kvantisoi datamonistoa tarpeeksi hyvin. Erottelukykyä voi parantaa tinkimällä kuvauksen g jatkuvuudesta seuraavan esimerkin mukaisesti: Muodostetaan kuvaus pinnalta käyrälle siten, että pinnan jokainen piste kuvautuu sille käyrän pisteelle, joka on sitä lähinnä, ts.

$$g : \mathcal{S} \to \mathcal{C}, \mathbf{x} \mapsto \{\mathbf{y} \mid \|\mathbf{x} - \mathbf{y}\| < \|\mathbf{x} - \mathbf{y}'\| \quad \forall \mathbf{y}' \neq \mathbf{y}\}$$

Jos nyt $\mathcal{S} = I^2 = [0,1]^2$ ja $\mathcal{C} = \{(t,t)|t \in [0,1]\}$, jolloin $n = 2$ ja $d = 1$, kuvaus on selvästikin jatkuva, mutta sen erottelukyky on heikko – esimerkiksi pisteet $(1,0)$ ja $(0,1)$ kuvautuvat samalle pisteelle $(0.5, 0.5)$. Jos taas käyrä $\mathcal{C}$ muistuttaa muodoltaan avaruuden täyttävää käyrää (peano-käyrää), käyrä voi kulkea mielivaltaisen läheltä jokaista pinnan pistettä. Tällöin käyrän pisteiden alkukuvien $g^{-1}(\mathbf{y})$ halkaisijat ovat pieniä, ja kuvauksen erottelukyky on hyvä. Samalla kuvaukselle on kuitenkin ilmaantunut lukuisia epäjatkuvuuskohtia. (Matemaattisesti ottaen myöskin pisteitä, joissa kuvaus ei ole yksikäsitteinen; reaalimaailmassa tämä ei kuitenkaan haittaa, toisin kuin epäjatkuvuus).

Jatkuvuus ja erottelukyky ovat siis toistensa kanssa ristiriitaisia tavoitteita, jos $n > d$. SOMin käytännön sovelluksesta riippuu, kumpi tavoite asetetaan etusijalle. Data-avaruuden kvantisoinnissa tärkeintä on hyvä erottelukyky: mitkään toisistaan kaukana olevat pisteet eivät kuvaudu lähekkäin. Dataa analysoitaessa taas voi tärkeintä olla kuvauksen jatkuvuus: kaikki toisiaan lähellä olevat pisteet myös kuvautuvat lähekkäin. Erityisen tärkeää jatkuvuus on silloin, kun pyritään löytämään korkeadimensioisen datamoniston olennaisimmat piirteet kartan avulla. [1]

SOM pystyy erinomaisesti sopeutumaan näihin ristiriitaisiin vaatimuksiin. Se pyrkii automaattisesti approksimoimaan datamoniston dimensiota siten, että niillä alueilla, missä $n > d$, kartta vetäytyy rypyille (vrt. peano-käyrä) [7]. Toisaalta kartta pyrkii muodostamaan "mahdollisimman jatkuvan" kuvauksen – eräänlaisen regression, joka voi olla epälineaarinen – datamonistolta kartalle.

SOMin konvergenssiominaisuuksien analyysissään Ritter ja Schulten [8] osoittivat, että kartan rypyttyminen, kun $n > d$, on yhteydessä naapurustofunktion h leveyteen seuraavasti: Tarkastellaan kaksiulotteista karttaa, jonka syötteet ovat kolmiulotteisia, ja jolla on gaussinen naapurustofunktio $h_{ci} = \exp\left(-\|r_c - r_i\|^2/2\sigma^2\right)$. Syötevektoreiden $\mathbf{x} = (x_1, x_2, x_3)$ x_3-komponentti on tasan jakautunut välille $[-s, s]$, ja x_1- ja x_2-komponentit ovat tasan jakautuneita välille $[0, a]$, $a \gg s$. Oletetaan, että kartta on alunperin (x_1, x_2)-tason suuntainen. Kun s lähestyy arvoa $s^* = \sigma\sqrt{3e/2} \approx 2.02\sigma$ alhaalta, syötteet (x_1, x_2)-tason ulkopuolelta saavat kartan värähtelemään x_3-akselin suuntaisesti yhä intensiivisemmin, ja kun lopulta $s = s^*$, kartta siirtyy uuteen tasapainotilaan. [2]

Kartan sietokyky poikittaisille syötteille on siis suoraan verrannollinen naapurustofunktion leveyden määräävään parametriin σ. *Siten kartan välittämässä kuvauksessa voidaan korostaa joko kuvauksen jatkuvuutta tai sen erottelukykyä yksinkertaisesti naapurustofunktion leveyttä muuttamalla.*

[1] Edellä oli asetettu kuvaukselle tavoitteeksi hyvä erottelukyky ja jatkuvuus. Näiden lisäksi voi olla muitakin tavoitteita. Esimerkiksi kuvaukselta voisi toivoa, että se approksimoisi datajoukon tiheyttä – toisin sanoen, kuvajoukossa tietyn kokoisen pallon sisään kuvautuisi aina todennäköisyydeltään vakiosuuruinen osa datajoukosta.

[2] Monimutkaisemmille datamonistoille ei yleistä ratkaisua ainakaan vielä ole löydetty, mutta yllämainittu approksimaatio on käyttökelpoinen "paikallisesti" eli alueella, jolla datamoniston tiheys on likimäärin vakio. (huomautus alkuperäisestä lähteestä)

3.2 Kuvauksen hyvyyden mittaaminen

Kartan välittämän kuvauksen hyvyydelle voidaan asettaa useita keskenään osittain ristiriitaisia kriteereitä, joiden keskinäinen painotus riippuu kulloisestakin sovelluksesta. Kartan välittämän kuvauksen hyvyys on siksi syytä kvantifioida erikseen kunkin kriteerin suhteen.

Kuten kappaleessa 3.1, tässäkin rajoitutaan tarkastelemaan kuvauksen kahta tärkeintä ominaisuutta: sen erottelukykyä ja jatkuvuutta. Seuraavissa kappaleissa esitellään menetelmät näiden ominaisuuksien kvantifioimiseen.

3.2.1 Kvantisointivirhe

Kuvauksen erottelukyvyn mitaksi sopii kvantisointivirhe $\mathcal{E}_q$. Sillä tarkoitetaan näytevektoreiden keskimääräistä etäisyyttä lähimmästä neuronista (koodikirjavektorista):

$$\mathcal{E}_q = \frac{1}{N}\sum_{i=1}^{N} \min_k d(\mathbf{x}_i, \mathbf{w}_k),$$

missä $\mathbf{x}_i$:t ovat datavektoreita ja $\mathbf{w}_k$:t kartan painovektoreita.

Kvantisointivirhe mittaa sitä, kuinka hyvin kartta on sovittautunut datamonistoon ($\approx$ näytevektoreiden joukkoon) [3], toisin sanoen, kuinka hyvin kartta pystyy kvantisoimaan datamoniston. Se ei kuitenkaan kiinnitä lainkaan huomiota siihen, ovatko kartan neuronit jollakin tavoin järjestyneet vaiko ei.

Kartan pyrkimys järjestyä riippuu voimakkaasti naapurustofunktion leveydestä: mitä leveämpää naapurustoa käytetään, sitä suurempi kartan pyrkimys järjestykseen. Toisaalta neuronien on kaikkein "helpointa" sovittautua dataan silloin, kun niiltä ei vaadita minkäänlaista järjestyneisyyttä; kun naapurustofunktio on Kroneckerin δ. Tällöin SOM palautuu klassiseen vektorikvantisointi-algoritmiin [9], eikä sillä ole minkäänlaista pyrkimystä muodostaa jatkuvaa kuvausta datamonistosta kartalle.

3.2.2 Topografinen tulo

Ensimmäisenä esitelty menetelmä kvantifioida kartan muodostaman kuvauksen jatkuvuutta on Bauerin ja Pawelzikin [10] topografinen tulo (topographic product), joka perustuu siihen, että neuronien naapurustoja verrataan toisiinsa sekä kartalla että syöteavaruudessa. Kartalla neuronin naapurusto koostuu yksinkertaisesti ympäröivistä neuroneista, syöteavaruudessa naapurusto koostuu neuronin painovektoria lähimpänä olevia painovektoreita vastaavista neuroneista. Bauerin ja Pawelzikin mukaan se, että naapurustot ovat laajasti ottaen samat, osoittaa kartan kuvaavan naapurustosuhteet hyvin. Tällöin topologia säilyy kuvauksessa, eli kuvaus on jatkuva. Jos taas naapurustot eroavat toisistaan, se on merkkinä siitä, että kartta on laskostunut, ja kuvaus on siis epäjatkuva. –Sen kaavan nimi, jolla naapurustojen samuus tai erilaisuus syöteavaruudessa ja kartalla kvantifioidaan, on juuri "topografinen tulo".

Topografinen tulo on kuitenkin liian tiukka mitta kuvauksen jatkuvuudelle – riittävä, muttei välttämätön ehto. Kuten Villmann et al. [11] huomauttivat, esimerkiksi kourun muotoiseen

[3] Käytännön sovelluksissa kaikki tieto datamonistosta ja sen tiheysjakaumasta saadaan monistoa edustavien näytevektoreiden muodossa. Siksi ne voidaan – tietyllä varovaisuudella – samaistaa toisiinsa. Samoin kartan välittämän kuvauksen hyvyys lasketaan käytännössä jatkuvan tiheysfunktion sijasta diskreetin näytevektorien joukon avulla, joten kaavat eri virhemitoille on jäljempänä esitetty diskreetissä muodossa.

datamonistoon hyvin sovittautuva kartta vetäytyy myös kourulle. Topografinen tulo havaitsee tällöin, että kartta on mutkalla, mikä sen mukaan on merkki topografisesta virheestä, vaikka todellisuudessa kuvaus olisi topografisesesti virheetön.

Perusvirhe topografisessa tulossa on, että se ei ota lainkaan huomioon datamoniston muotoa. Etäisyydet lasketaan aina euklidista metriikkaa käyttäen, vaikka ne tulisi laskea ultrametrisesti eli datamonistoa seuraten. Virheen seurauksena topografinen tulo ei anna oikeita tuloksia, jos datamonisto on riittävän epälineaarinen.

3.2.3 Topografinen funktio

Villmann et aliin [11] esittelemä topografinen funktio $\Phi_L^M(s)$ approksimoi ultrametristä naapurustojen määrittelyä. Kahden neuronin n_i ja n_j painovektorit luokitellaan vierekkäisiksi silloin, kun jollekin näytevektorille $\mathbf{x}$ lähimmät painovektorit ovat $\mathbf{w}_i$ ja $\mathbf{w}_j$. Se, että neuronien painovektorit ovat lähekkäin, ei siis vielä riitä tekemään neuroneista naapureita; painovektorien välissä pitää olla myös ainakin yksi näytevektori.

Topografinen funktio määritellään seuraavasti:

$$\begin{aligned}\Phi_L^M(s) = \sum_{i \in L} \#\{n_j \mid j \in L,\ \|n_i - n_j\| > s, \\ \text{missä } i = \operatorname*{argmin}_k \|\mathbf{w}_k - \mathbf{x}\| \text{ ja } j = \operatorname*{argmin}_{k \neq i} \|\mathbf{w}_k - \mathbf{x}\| \text{ jollekin näytevektorille } \mathbf{x}\end{aligned} \tag{9}$$

Kaavassa 9 tarkoittaa $\#$ joukon kardinaliteettia, L on kartan neuronien indeksijoukko, ja neuronien etäisyys kartalla $\|n_i - n_j\|$ määritellään Turku- eli ℓ_1-metriikalla.

Topografinen funktio tuottaa naapurustosuhteita kuvaavan käyrän (kts. kuva 3, josta näkee, kuinka paljon ja kuinka laajoja poimuja kartassa on. On kuitenkin tilanteita, joissa käyrän sijasta yksi jatkuvuutta kuvaava luku olisi hyödyllisempi – tällaisia ovat esimerkiksi karttojen vertailu, tai saman kartan välittämän kuvauksen jatkuvuuden kehittymisen tarkastelu opetuskierrosten funktiona. Toinen topografisen funktion ongelma on, että se ei kiinnitä lainkaan huomiota siihen, kuinka laajalti neuronit ovat vierekkäisiä – onko niiden välillä vain yksi näytevektori, vai sijaitsevatko kaikki neuroneille kuvautuvat näytevektorit niiden välillä? Näistä syistä topografiselle funktiolle kehitettiin tässä työssä vaihtoehto, topografinen virhe.

Kuva 3: Yksidimensioisen kartan sovittautuminen yksikköneliöön ja kartasta laadittu topografinen funktio (kuva viitteestä [11])

3.2.4 Topografinen virhe

Topografinen virhe $\mathcal{E}_t$ määritellään seuraavasti:

$$\mathcal{E}_t = \frac{1}{N}\sum_{k=1}^{N} u(\mathbf{x}_k),$$

$$\text{missä } u(\mathbf{x}_k) = \begin{cases} 1, & \text{kun } ||n_i - n_j|| > 1,\ i = \text{argmin}_k\,||\mathbf{w}_k - \mathbf{x}|| \\ & \quad \text{ja } j = \text{argmin}_{k \neq i}\,||\mathbf{w}_k - \mathbf{x}|| \\ 0, & \text{muulloin} \end{cases} \tag{10}$$

Topografinen virhe siis kertoo, kuinka suurella osalla näytevektoreista on lokaali ympäristö, jossa on epäjatkuvuuskohta: $u(\mathbf{x}) = 1$ tasan silloin, kun jossakin kohdassa näytevektorin $\mathbf{x}$ ja toisiksi lähimmän painovektorin $\mathbf{w}_j$ välissä on alue, jossa mielivaltaisen pienessä ympäristössä olevat vektorit kuvautuvat ei-vierekkäisille painovektoreille.

Topografinen virhe luonnehtii kartan välittämän kuvauksen jatkuvuutta yhdellä luvulla. Lisäksi, jos oletetaan että syöteavaruus on riittävän säännöllinen ja että kartta on ehtinyt sovittautua syöteavaruuteen tarpeeksi hyvin, topografinen virhe kertoo paitsi epäjatkuvuuskohtien olemassaolosta, myös niiden yleisyydestä. Sen sijaan topografinen virhe ei anna käsitystä siitä, *millaisia* epäjatkuvuuskohdat ovat, toisin kuin topografinen funktio – se kertoo vain, kuinka paljon niitä on. Tämä tuntuu perustellulta: muutoin olisi mahdoton tietää, kuvaako suuri topografisen virheen arvo sitä, että kartassa on paljon lyhyitä epäjatkuvuuskohtia, vai sitä, että muutama epäjatkuvuuskohta kuvaa vierekkäiset syötevektorit kartan eri reunoihin.

3.3 Pohdintoja

Ideaalinen kartan välittämä kuvaus on – diskretoinnin vaikutus unohtaen – homeomorfismi syöteavaruudesta kartalle. Lisäksi syöteavaruuden tiheys kuvautuu ideaalisessa kuvauksessa johdonmukaisesti: painovektorien tiheyden tulee olla monotoninen syöteavaruuden tiheyden funktio.

Niin kauan kuin syöteavaruuden luonnollinen dimensio on sama kuin kartan dimensio, kuvaus voi olla lähellä ideaalista. Jos sen sijaan syöteavaruuden luonnollinen dimensio on suurempi, joudutaan tekemään kompromissi kuvauksen resoluution ja jatkuvuuden välillä. Kuvauksen ominaisuuksien painotus riippuu sovelluksesta, mutta tavallisesti kummastakin on hyötyä. Molemmat ominaisuudet tulisi siis pystyä kvantifioimaan, jotta eri parametrein ja alkuarvoin opetetuista kartoista voidaan valita paras.

Edellä esiteltiin algoritmit resoluution ja jatkuvuuden kvantifioinnille. Mikä kolmesta eri algoritmista jatkuvuuden kvantifioinnille valitaan, ei ehkä vaikuta lopputulokseen kovin paljoa. Topografinen tulo vaatisi datan olevan lineaarista, kaksi muuta algoritmia ei; koska topografisella tulolla ei ole mitään sellaisia hyviä ominaisuuksia joita muilla kahdella ei ole, sen käyttö jatkuvuuden kvantifiointiin ei ehkä ole perusteltua. Topografisen funktion hyväksi puoleksi jää sen antama käsitys kartan poimujen laajuudesta, mikä saattaa olla joissakin sovelluksissa arvokasta. Topografinen virhe taas mahdollistaa tulosten yksikäsitteisen vertailun ja epäjatkuvuuskohtien yleisyyden arvionnin.

4 Kuvausvirheiden minimointi paikallisesti adaptoituvan naapurustofunktion avulla

Kartan välittämän kuvauksen laatuun eniten vaikuttaa naapurustofunktion leveysparametri σ. Jos se on liian suuri, kartta kyllä oikenee ja kuvauksesta muodostuu jatkuva, mutta erottelukyky jää heikoksi. Jos se taas on liian pieni, ja jos syöteavaruuden luonnollinen dimensio on suurempi kuin kartan dimensio, kartta laskostuu edellisessä luvussa kuvatulla tavalla. Kartta voi myös jäädä kunnolla avautumatta, jos σ pienenee opetuksen alussa liian nopeasti.

Naapurustofunktion ei välttämättä tarvitse olla saman levyinen kaikkialla kartassa. Sen leveyden ei myöskään tarvitse olla aidosti vähenevä opetusaskelten määrän funktio; riittää, että naapurustofunktio on tarpeeksi leveä estämään kartan aukeamatta jäämisen – ja, jos halutaan, myös kartan laskostumisesta johtuvan epäjatkuvuuskohtien syntymisen.

Esitetyt havainnot ovat perustana SOMin modifikaatiolle nimeltään AdSOM.

4.1 AdSOM-algoritmi

AdSOM-algoritmi on hyvin pitkälti samanlainen kuin SOM-algoritmi. Erona on lähinnä kullekin neuronille i erikseen määritelty naapurustofunktion leveysparametri σ_i, sekä sen päivitysalgoritmi.

Opetuksen alkaessa σ_i:t alustetaan kartan säteen (suurimman diagonaalin puolikkaan) suuruisiksi. Tietyin väliajoin, aina joka N:s kierros, leveysparametreja σ_i pienenetään seuraavasti:

$$\sigma_i = \beta\sigma_i, \quad 0 < \beta < 1, \tag{11}$$

missä parametrit N ja β on valittava niin, että naapurustot eivät pienene liian nopeasti.

Toisin kuin tavallisessa SOMissa, leveysparametreja σ_i voidaan myös kasvattaa, jos havaitaan, että kartta ei muodosta jatkuvaa kuvausta. Epäjatkuvuuskohtia voi aiheuttaa kartan aukeamatta jääminen tai laskostuminen. Ne ilmenevät tilanteina, joissa syötevektorin kaksi lähintä painovektoria kuuluvat neuroneille, jotka eivät ole naapureita kartalla. Tällöin näiden neuronien ympäristölle lasketaan uudet leveysparametrit σ_i seuraavasti:

$$\sigma_i := \max\{\sigma_i, \sigma_i^*\},$$

$$\text{missä } \sigma_i^* = \begin{cases} \|n_j - n_k\| & \text{jos } \max\{\|n_i - n_j\|, \|n_i - n_k\|\} \leq \|n_j - n_k\| \\ \|n_j - n_k\| - s & \text{muutoin, jos } s := \min\{\|n_i - n_j\|, \|n_i - n_k\|\} < \|n_j - n_k\|, \\ 1 & \text{muulloin} \end{cases} \tag{12}$$

Tällöin järjestyksessä olevalle kartalle naapurustosäteet voivat pienetä lähelle nollaa, jolloin saadaan paras mahdollinen resoluutio. Jos taas jossakin kartan osassa laskostumisen estämiseksi tarvitaan leveämpiä naapurustoja, σ_i:t vakiintuvat siellä korkeammalle tasolle ilman, että koko kartta tarpeettomasti jäykistyy.

Erilevyisistä naapurustoista huolimatta jokaisella opetusvektorilla on oltava yhtä suuri painoarvo kartan muokkautumisessa, joten AdSOMissa myös naapurustofunktion muotoon jouduttiin tekemään muutos. Samalla otettiin huomioon Erwin et aliin [12] havainto, että kupera naapurustofunktio saa kartan järjestymään nopeammin: tämän vuoksi gaussisen funktion koverat reunat jätettiin pois, ja kääntöpisteissä ($r = \pm\sigma_i$) funktion liepeet vedettiin nollaan. Näin päädyttiin seuraavaan naapurustofunktion $h_i(r)$ määritelmään:

$$h_i(r) = \begin{cases} \dfrac{\exp\left[-\frac{1}{2}\left(\frac{r}{\sigma_i}\right)^2\right] - \exp\left(-\frac{1}{2}\right)}{\sigma_i\left[1 - \exp\left(-\frac{1}{2}\right)\right]} & \text{jos } r < \sigma_i \\ 0 & \text{muutoin} \end{cases} \tag{13}$$

4.2 Kokeilut

4.2.1 Menetelmät

AdSOM-algoritmin toimintaa verrattiin perus-SOMiin, jossa käytettiin kahta erilaista naapurustoa: tavallista kuplanaapurustoa, sekä kaavassa (13) määriteltyä gaussisen naapurustofunktion muunnelmaa. Kokeiluissa kartta oli yksidimensioinen, ja siinä oli 65 yksikköä. Datamoniston geometrinen dimensio oli kolme, ja sen luonnollinen dimensio vaihteli yhden ja kahden välillä.

AdSOMin naapurustofunktiota pienennettiin tehdyssä kokeessa hiukan toisin kuin edellä kaavassa (11) on kuvattu; sen sijasta käytettiin kaavaa

$$\sigma_i = \sigma_i^{\beta}, \quad 0 < \beta < 1, \tag{14}$$

Tällöin suuret σ_i:n arvot pienenivät nopeammin kuin kaavan (11) tapauksessa, mikä saattaa olla hyväkin. Toisaalta naapurustofunktion leveys oli aina vähintään yksi; kaavalla (11) järjestyksessä olevan kartan naapurustofunktio taas kapenee kohti Kroneckerin deltafunktiota, mikä mahdollistaa ideaalisen kvantisoinnin (klassinen vektorikvantisointi) ja minimoi reunaefektit.

Jokaista kolmesta kartasta kokeiltiin ensin alustaen painovektorit satunnaisesti ja opettaen karttaa yhteensä 30 000 kierrosta. Oppimisnopeusparametri α laski lineaarisesti 0,9:stä 0,05:een ensimmäisten 3 000 kierroksen aikana, ja seuraavien 27 000 kierroksen ajan se pysyi vakiona 0,05:ssä. Kuplanaapurusto-SOMissa naapurustosädeparametri σ pieneni ensimmäisten 3 000 kierroksen aikana 32:sta (kartan säde) yhteen ja gaussisessa SOMissa 32:sta kahteen, joihin arvoihin se jäi seuraaviksi 27 000 kierrokseksi. AdSOMin naapurustosäteet σ_i alustettiin samoin 32:een, jonka jälkeen ne saivat kehittyä AdSOM-algoritmin mukaisesti.

Toisessa kokeessa karttoja opetettiin niin, että painovektorit alustettiin edellisessä kokeessa pahiten laskostuneen kartan – gaussisen SOMin – arvoihin. Karttoja opetettiin 30 000 kierrosta, α pidettiin koko ajan arvossa 0,05, ja kupla- sekä gaussisen SOMin σ pidettiin yhdeksässä. Edellisen kokeen tavoin AdSOMin σ_i:t alustettiin arvoon 32, jonka jälkeen ne saivat kehittyä AdSOM-algoritmin mukaisesti.

Tulokset evaluoitiin laskemalla kvantisointi- ja topografinen virhe aina viidensadan opetusaskelen jälkeen. Lisäksi opetuksen kulkua seurattiin tarkkailemalla silmämääräisesti kartan sovittautumista dataan sekä naapurustojen leveysparametrien σ_i kehittymistä.

4.2.2 Tulokset

Saadut tulokset on esitetty kuvassa 4. Naapurustofunktion leveyttä muuttelemalla saadaan aikaiseksi hyvin erilaiset kuvaukset. Niillä syöteavaruuden alueilla, joilla kartalla ja datalla on sama dimensio, leveä naapurustofunktio aiheuttaa voimakkaan reunaefektin ja estää karttaa seuraamasta kunnolla syöteavaruuden mutkia. Sen sijaan alueilla, joilla syöteavaruuden dimensio on suurempi kuin kartalla, leveä naapurustofunktio saa kartan "keskiarvoistamaan", tavallaan seuraamaan syöteavaruuden "epälineaarisia pääkomponentteja"; kapean naapurustofunktion kartta puolestaan pyrkii approksimoimaan syöteavaruuden korkeampaa dimensiota laskostumalla.

AdSOMissa topografinen virhe pysyy jatkuvasti lähes nollassa – kun laskostuminen jollakin alueella alkaa, kartta jäykistyy sillä alueella niin, että laskos nopeasti häviää. Gaussinen kartta leveällä naapurustolla muodostaa täysin jatkuvan kuvauksen: kartta oikenee nopeasti, jonka jälkeen topografinen virhe pysyy nollassa.

Sen sijaan kuplanaapurusto ei muodosta jatkuvaa kuvausta edes leveää naapurustoa käytettäessä. Syynä tähän näyttäisi olevan se, että painovektorit muodostavat ryppäitä, joihin kuuluu noin naapuruston leveyden verran painovektoreita. Ryppäiden reunimmaiset painovektorit vetävät toisiaan puoleensa riittävästä pitääkseen ryppään kasassa, ja valtaosa ryppäälle osuvista näytevektoreista tulee juuri näille reunimmaisille vektoreille. Keskimmäiset vektorit eivät välttämättä sijaitse tasaisesti reunimmaisia vektoreita yhdistävällä janalla, vaan usein jonkin verran sen ylä- tai alapuolella, mikä ilmeisesti aiheuttaa kuvauksessa olevat epäjatkuvuudet.

Kvantisointivirhe on odotetusti pienin kartoilla, joilla on kapein naapurustofunktio; AdSOM häviää näille jonkin verran. Sen sijaan leveänaapurustoiset kartat kvantisoivat selvästi AdSOMia huonommin.

4.3 Pohdintoja

4.3.1 AdSOM

Simulointien perusteella AdSOM näyttäisi toimivan ainakin yksidimensioisessa tapauksessa hyvin: se onnistuu muodostamaan lähes jatkuvan kuvauksen uhraamatta erottelukykyä tarpeettoman paljoa. Samaan voitaisiin päästä myös tavallisella SOMilla kokeillen erilaisilla naapurustofunktion leveyden loppuarvoilla, jos syöteavaruuden luonnollinen dimensio olisi vakio. Simuloinneissa käytetyllä, vaihtuvadimensioisella datalla tämä ei kuitenkaan toimisi kovin hyvin.

AdSOMin parhaita puolia tavalliseen SOMiin verrattuna on, että se tekee mahdolliseksi hyvien kuvausten muodostamisen ilman työlästä sopivien parametrien haarukoimista – mikä sopii hyvin yleiseen itseorganisoitumisfilosofiaan. Lisäksi AdSOMia käyttäen kartan on erittäin vaikeaa jäädä kunnolla järjestymättä, esimerkiksi kierteelle tai kaksin kerroin; tavallisessa SOMissa näin voi käydä, jos opetuksen alkuvaiheessa naapurustosäde pienenee liian nopeasti.

AdSOMilla voisi myös olla mahdollista arvioida datan dimensiota. Jos naapurustosäteet σ_i pääsevät pienenemään opetuksen lopussa lähelle nollaa, kartan ja datan dimensiot ilmeisesti vastaavat toisiaan. Kartta on tällöin lähellä optimaalista: painovektorit sijaitsevat Voronoiympäristöjensä painopisteissä, mutta ne eivät pääse ajautumaan epäjärjestykseen – jos näin kävisi, naapurustosäteet kasvaisivat hetkeksi ja pienenisivät taas järjestyksen palattua.

Jatkuva kuvaus kartan dimensiota suurempidimensioisesta syöteavaruudesta voidaan muodostaa vain, jos naapurustofunktiot pysyvät riittävän leveinä. Hintana on reunaefekti: reunimmaiset painovektorit vetäytyvät kohti keskustaa. Jos kartan dimensiota ei voida kasvattaa, esimerkiksi visuaalisen tarkastelun tärkeyden vuoksi, reunaefektiä voidaan vähentää antamalla kartan reunimmaisille yksiköille suurempi oppimiskerroin α; kts esim [13].

Laskennallisesti AdSOM ei ole merkittävästi tavallista SOMia raskaampi. Toiseksi lähin painovektori löytyy tavallaan sivutuotteena lähintä vektoria etsittäessä; samoin naapurustosäteiden σ_i laskeminen on kevyt operaatio, ja opetuksen edistyessä sitä ei tarvitse tehdä kovin usein.

Kuva 4: Syötevektorit ja kartan painovektorit, kuvattuna kahdesta eri suunnasta, ja eri karttoja vastaavat topografiset ja kvantisointivirheet.

4.3.2 Naapurustofunktion muoto

Kuplanaapurusto on osoittautunut hyvin toimivaksi silloin, kun naapurustosäde pienennetään opetuksen lopussa yhteen. Kokeilut näyttäisivät kuitenkin vihjaavan, että naapuruston jäädessä leveämmäksi kuplanaapurusto ei jaa painovektoreita tasaisesti, kuten kaavan (13) gaussisen naapuruston muunnos tekee. Seurauksena on kuvan 4 mukaisesti epäjatkuva ja huonosti kvantisoiva kuvaus. Lisäksi visuaalisella tarkastelulla datasta saattaa saada harhaanjohtavan kuvan: esimerkiksi SOM_PAKin [14] umat-esitys näyttäisi datan jakautuneen erillisiin klustereihin, vaikka kysymyksessä on painovektoreiden virheellinen klusteroituminen tasaisesti jakautuneessa datassa.

Tavallinen gaussinen naapurusto on usein havaittu huonommin toimivaksi kuin kuplanaapurusto. Sen sijaan tehdyissä kokeissa mukaeltu gaussinen naapurusto toimi kuplanaapurustoa paremmin; tavallista gaussista naapurustoa ei tässä kokeiltu. Olisikin ehkä syytä tehdä perusteellisempi selvitys näiden kolmen eri naapurustofunktion käyttäytymisestä, kun datan dimensio ja naapurustofunktin loppuleveys vaihtelevat.

5 Yhteenveto

Syöteavaruuden dimensiolla on suuri vaikutus siihen, millaisen kuvauksen itseorganisoiva kartta muodostaa. Lisäksi se antaa käsityksen siitä, kuinka monimutkainen tarkasteltava ilmiö on. Siksi sitä olisi hyödyllistä pystyä estimoimaan.

Tässä erikoistyössä tarkasteltiin joukon dimension estimointiin esitettyjä algoritmeja ja kokeiltiin sekä keinotekoisella että todellisia ilmiöitä kuvaavalla datalla. Keinotekoisella datalla estimointialgoritmit toimivat hyvin; sen sijaan todellisella datalla saatiin vain suuruusluokka-arviot datan dimensioille. Algoritmit vaativat huomattavan paljon havaintovektoreita, ja niiden käytännön soveltamiseen liittyy joukko ongelmia, joihin on vaikea tai mahdoton löytää yleisiä ratkaisuja. Siitä huolimatta syöteavaruuden dimension karkea estimointi on täysin mahdollista. Lisäksi erityisesti korrelaatiodimensio-algoritmin avulla voidaan rajoitetussa määrin saada myös kvalitatiivista tietoa datasta, lähinnä sen kohinaisuudesta ja klusteroitumisesta.

Kun syöteavaruuden luonnollinen dimensio ei ole sama kuin käytetyn itseorganisoituvan kartan, ei ole mahdollista saada ideaalista, homeomorfismia muistuttavaa kuvausta syöteavaruuden ja kartan välille, vaan joko kuvauksen erottelukyvystä tai jatkuvuudesta on tingittävä. Kumpi ominaisuus asetetaan etusijalle, riippuu sovelluksesta. Vaihtokauppaa erottelukyvyn ja jatkuvuuden välillä voidaan säädellä kartan naapurustofunktion leveyden avulla.

Sekä kartan muodostaman kuvauksen erottelukyky että jatkuvuus on pystyttävä kvantifioimaan, jotta kartan muodostaman kuvauksen hyvyyttä voidaan arvioida. Erottelukykyä voidaan mitata kvantisointivirheen $\mathcal{E}_q$ avulla. Kuvauksen jatkuvuuden kvantifioimiseen on esitetty aiemmin kahta erilaista mittaa: topografista tuloa sekä topografista funktiota. Tässä työssä ehdotettiin näiden sijasta käytettäväksi topografista virhettä $\mathcal{E}_t$, joka antaa kahta muuta jatkuvuuden mittaa paremman käsityksen siitä, kuinka suuri osa datavektoreista kuvautuu jatkuvaan lokaaliin ympäristöön.

Työssä esiteltiin myös SOM-algoritmin muunnos AdSOM, joka pyrkii muodostamaan mahdollisimman jatkuvan kuvauksen syöteavaruudesta kartalle näiden dimensioista riippumatta. AdSOMilla on myös eräitä hyödyllisiä ominaisuuksia siinäkin tapauksessa, että kartan dimensio on sama kuin syöteavaruuden luonnollinen dimensio.

Paras mahdollinen kuvaus saadaan käyttämällä karttaa, jonka dimensio on sama kuin syöteavaruuden luonnollinen dimensio, kuten Speckmann et al. [15] on ehdottanut. Usein kartan avulla kuitenkin pyritään visualisoimaan dataa, ja tällöin kartan dimensioksi joudutaan valitsemaan joko kaksi tai mahdollisesti kolme. Ellei kartan muodostama kuvaus ole jatkuva, datasta muodostuu helposti väärä mielikuva – varsinkin, jos kuvauksen epäjatkuvuutta ei tiedosteta. Siksi kuvauksen jatkuvuus on aina selvitettävä, ja yleensä on myös syytä pyrkiä muodostamaan mahdollisimman jatkuva kuvaus, joko etsimällä sopiva naapurustofunktion leveys tai käyttämällä AdSOMia.

Viitteet

[1] J. Doyne Farmer, Edward Ott, and James A. Yorke, "The dimension of chaotic attractors", *Physica*, vol. 7D, pp. 153–180, 1983.

[2] Peter Grassberger and Itamar Procaccia, "Measuring the strangeness of strange attractors", *Physica*, vol. 9D, pp. 189–208, 1983.

[3] H.G.E. Hentschel and Itamar Procaccia, "The infinite number of generalized dimensions of fractals and strange attractors", *Physica*, vol. 8D, pp. 435–444, 1983.

[4] James Theiler, "Estimating fractal dimension", *Journal of the Optical Society of America*, vol. 7, no. 6, pp. 1055–1073, June 1990.

[5] D. Ruelle, "Deterministic chaos: the science and the fiction", *Proc. R. Soc. Lond*, vol. A 427, pp. 241–248, feb 1990.

[6] T.S. Parker and L.O. Chua, *Practical Numerical Algorithms for Chaotic Systems*, Springer-Verlag, New York, 1989.

[7] Teuvo Kohonen, *Self-Organizing Maps*, Springer Series in Information Sciences 30. Springer, Berlin Heidelberg New York, 1995.

[8] H. Ritter and K. Schulten, "Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection", *Biological Cybernetics*, vol. 60, no. 1, pp. 59–71, Nov. 1988.

[9] Teuvo Kohonen, "Self-organizing maps: optimization approaches", in *Artificial Neural Networks*, T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, Eds., Amsterdam New York Oxford Tokyo, 1991, vol. 2 of *International Conference on Artificial Neural Networks*, pp. 981–990, Elsevier Science Publishers B.V. (North-Holland).

[10] Hans-Ulrich Bauer and Klaus R. Pawelzik, "Quantifying the neighborhood preservation of self-organizing feature maps", *IEEE Transactions on Neural Networks*, vol. 3, no. 4, pp. 570–579, July 1992.

[11] Th. Villmann, R. Der, and Th. Martinetz, "A new quantitative measure of topology preservation in Kohonen's feature maps", in *Proc. ICNN'94, the IEEE Int. Conf. on Neural Networks*, Orlando, Florida, USA, June 1994, IEEE, pp. 645–648, IEEE.

[12] E. Erwin, K. Obermayer, and K. Schulten, “Self-organizing maps: stationary states, metastability and convergence rate”, *Biological Cybernetics*, vol. 67, no. 1, pp. 35–45, 1992.

[13] M. Herrmann, “Self-organizing feature maps with self-organizing neighborhood widths”, in *Proc. ICNN'95, the IEEE Int. Conf. on Neural Networks*, Perth, Western Australia, Nov. 1995, IEEE, vol. 6, pp. 2998–3003, IEEE.

[14] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen, *The Self-Organizing Map Program Package*, Laboratory of Computer and Information Science, Helsinki University of Technology, Espoo, version 3.1 edition, 1995, available via ftp://cochlea.hut.fi/.

[15] H. Speckmann, G. Raddatz, and W. Rosenstiel, “Considerations of geometrical and fractal dimension of SOM to get better learning results”, in *Proc. ICANN'94, Int. Conf. on Artificial Neural Networks*, Maria Marinaro and Pietro G. Morasso, Eds., London, UK, 1994, vol. I, pp. 342–345, Springer-Verlag.

Sisältö