T-61.2010 Datasta tietoon, syksy 2011

professori Erkki Oja professori Heikki Mannila

31.10.2011

Sisältö

1 Datasta tietoon -johdantoluento

Johdanto

Kurssin ensimmäinen luento esittelee kurssin tavoitteet ja sisällön sekä käytännön suorittamiseen liittyvät asiat.

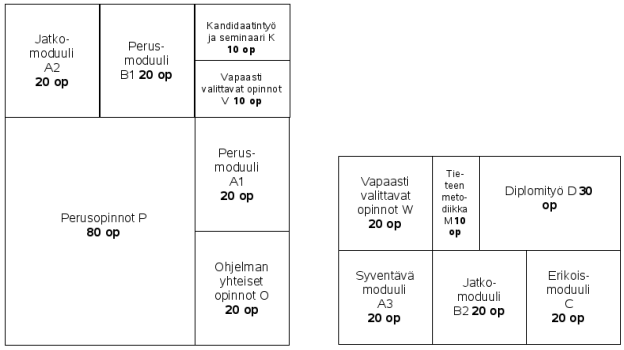
Kurssiesite ja kaikki materiaali on saatavilla Nopassa.

Kurssi suoritetaan tentillä ja pienellä harjoitustyöllä, joka on korvattavissa tietokonelaskareilla. Mukana viikottaisia pistetehtäviä. Tarkat ohjeet Nopassa.

1.1 Miksi tällainen kurssi?

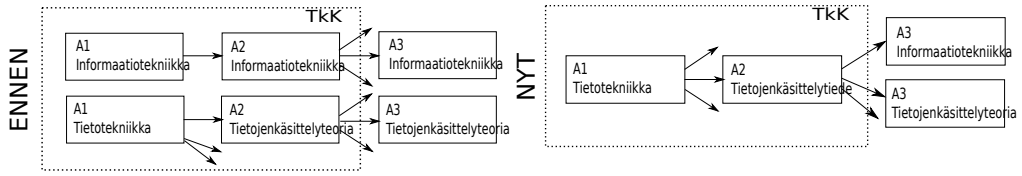
Miksi tällainen kurssi?

- On haluttu tarjota ohjelman yhteisissä opinnoissa (O-moduuli) kurssi, joka toimii johdatuksena *informaatiotekniikan* opinnoille: tietojenkäsittelytieteen jatkomoduulille A2 ja syventäville A3-moduuleille informaatiotekniikassa *kognitiivisessä neurotieteessä* ja *kieliteknologiassa*.



Kuva 1.1: TkT- (180 op) ja DI-tutkinto (120 op)

- HUOM! Informaatiotekniikan A1- ja A2-moduulit ovat poistuneet ja lukuvuodesta 2010-2011 lähtien tarjolla on *tietojenkäsittelytieteen jatkomoduuli A2*, jonka voi valita *tietotekniikan perusmoduulin A1* jatkoksi.
- Bioinformatiikan tutkinto-ohjelmassa kurssi kuuluu laskennallisen ja kognitiivisen biotieteen perusmoduuliin
- Tietojenkäsittelytieteen (TKT) laitos (v. 2008 asti entiset Informaatiotekniikan ja Tietojenkäsittelyteorian laboratorio) on hyvin *tutkimusintensiivinen*. Aalto-yliopiston Perustieteiden korkeakoulussa toimii viisi valtakunnallisista tutkimuksen huippuyksiköistä, joista kaksi TKT-laitoksella:



Kuva 1.2: (a) Moduulipolku lv 2009-2010 asti; (b) lv 2010-2011 lähtien.

1. “Adaptiivisen informatiikan tutkimusyksikkö” (Erkki Oja);
2. “Algoritmisen data-analyysin huippuyksikkö” (Heikki Mannila)

- Haluamme antaa jo perusopiskelijoille tilaisuuden tutustua meillä tehtävään tutkimukseen
- Ne opiskelijat jotka *eivät valitse* tietojenkäsittelytiedettä tai informaatiotekniikkaa pää- tai sivuaineekseen saavat kuitenkin jonkinlaisen käsityksen siitä mitä ala pitää sisällään
- Ne opiskelijat jotka *valitsevat* tietojenkäsittelytieteen (informaatiotekniikan) pää- tai sivuaineen saavat Datasta Tietoon -kurssissa katsauksen koko kenttään, joka sitten syvenee laitoksen muiden kurssien avulla ja esim. osallistumalla *kesäteekkarina* kesäteekkarina tutkimushankkeisiin (**haku tammikuussa!**)
- Muut aiheeseen liittyvät kurssit (those with *English names* will be given totally in English):

  T-61.2020 Datasta tietoon harjoitustyö (23.1.2012)

  T-61.3015 Digitaalinen signaalinkäsittely ja suodatus

  T-61.3025 Hahmontunnistuksen perusteet

  T-61.3040 *Statistical signal modeling*

  T-61.3050 *Machine learning: basic principles*

  T-61.5010 *Information visualization*

  T-61.5020 *Statistical natural language processing*

  T-61.5050 *High-throughput bioinformatics*

  T-61.5060 *Algorithmic methods of data mining*

  T-61.5070 *Computer vision*

  T-61.5080 *Signal processing in neuroinformatics*

  T-61.5090 *Image analysis in neuroinformatics*

  T-61.5100 *Digital image processing*

  T-61.5110 *Modeling biological networks*

  T-61.5120 *Computational genomics*

T-61.5130 *Machine learning and neural networks*

T-61.5140 *Machine learning: advanced probabilistic methods*

T-61.5150 *Speech recognition*

T-61.5900 Informaatiotekniikan erikoistyö

T-61.5910 *Research project in computer and information science*

T-61.60x0 *Special course in computer and information science I-VI*

T-61.60x0 *Special course in bioinformatics I-II*

T-61.6090 *Special course in language technology*

1.2 Kurssin suorittamiseen liittyviä asioita

Kotisivu, ilmoittautuminen ja suorittaminen

- Kurssin kotisivu `https://noppa.aalto.fi/noppa/kurssi/T-61.2010/`
- Ilmoittaudu WebOodin kautta `https://oodi.aalto.fi/a/`
- Kurssin suoritus: tentti (bonuspisteitä joulu-tammikuussa) ja pieni harjoitustyö (joka korvattavissa aktiivisella osallistumisella tietokonelaskareihin)

Käytäntöä Luennot

- Luennot maanantaisin ja torstaisin klo 14-16 salissa T1
- Alkupuolen (luennot 1-8, luvut 1-7) luennoi professori Erkki Oja `http://users.ics.tkk.fi/oja`
- Kaksi viimeistä luentoa (luennot 9-10, luvut 8-9) luennoi dosentti Kai Puolamäki `http://users.ics.tkk.fi/kaip/` (professori Heikki Mannilan osuus)
- Nopassa luentokalvot (kolme formaattia: luvuittain esitys, luvuittain tiivistelmä 4/A4; “koko kirja”)

Käytäntöä Laskuharjoitukset (“paperi”)

- Laskuharjoitukset (2 h / viikko) maanantaisin ja perjantaisin klo 12-14 salissa T4 (vaihtoehtoiset ryhmät). Alkavat pe 4.11.2011. Laskuharjoitusten assistenttina toimivat TkK Maria Osmala ja DI Janne Toivola
- Viisi kierrosta. Sisältävät sekä demotyyppisiä tehtäviä että yhden bonuspistetehtävän / kierros
- Harjoitustehtävät suomeksi ja englanniksi sekä ratkaisut englanniksi löytyvät kurssin kotisivulta

Käytäntöä Laskuharjoitusten bonuspistetehtävät

- Bonuspistetehtävän vastaukset käsin kirjoitettuna A4-kokoiselle paperille. Sivun oikeaan yläkulmaan ISOLLA OPISKELIJANUMERO ja kierros P1, P2, P3, P4 tai P5. Sivun yläosaan myös nimi
- Palautus metalliseen palautuslaatikkoon, T-talon pääaulassa, 3. kerroksessa heti rappusten jälkeen
- Ratkaisuun aikaa noin reilu viikko, DL merkattu tehtäväpapereihin
- Tarkistettujen tehtävien haku laskareista tai 3. kerroksen "Informaatiotekniikan" käytävän lötteröstä

Käytäntöä Pakollinen harjoitustyö

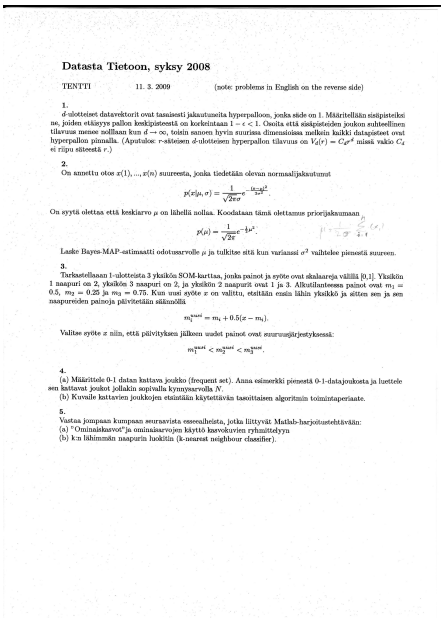
- Pieni Matlab-harjoitustyö (arvioitu työmäärä 8 h) on kiinteä osa kurssia ja sen voi tehdä itsenäisesti 15.1.2012 mennessä.
- Työohje on Nopassa. Aiheena ominaiskasvot (eigenfaces) liittyen pääkomponenttianalyysiin
- Harjoitustyön **voi korvata** aktiivisella osallistumisella ohjattuihin tietokoneharjoituksiin, joita on 2 h viikossa viidesti.

Käytäntöä Tietokoneharjoitukset

- Tietokoneharjoituksia samalla sisällöllä ma 16-18 Maari-A, ti 12-14 Maari-C, ti 14-16 Maari-A, ke 14-16 Maari-A
- Alkavat ma 7.11.2011. Ohjaajina TkK Elina Karp ja TkK Eric Malmi
- Viisi kierrosta. Demoja ja omaa laskentaa tietokoneella. Ei pisteitä, mutta korvaavat pakollisen harjoitustyön
- Tehtävät suomeksi kurssin Noppa-sivulta "viikkoharjoitukset"

Käytäntöä Tentti

- Ensimmäinen tentti la 17.12.2011 klo 10-13, toinen ke 11.1.2012 klo 13-16 (sali T1) ja viimeinen ennen kesää ke 7.3.2012 13-16 (sali T1).
- Tenttivaatimukset ja luentomateriaali kotisivuilla. Tentissä ei kaavakokoelmaa. Funktiolaskin sallittu.
- Kurssimateriaalina ovat luentokalvot ja harjoitustehtävät ratkaisuineen, jotka siis saatavana Nopasta
- Viisi tehtävää a 6 p, yhteensä 30 pistettä



Datasta Tietoon, syksy 2008

TENTTI 11. 3. 2009 (note: problems in English on the reverse side)

1.
d-ulotteiset datavektorit ovat tasaisesti jakautuneita hyperpalloon, jonka säde on 1. Määritellään sisäpisteiksi ne, joiden etäisyys pallon keskipisteestä on korkeintaan $1 - \epsilon < 1$. Osoita että sisäpisteiden joukon suhteellinen tilavuus menee nollaan kun $d \to \infty$, toisin sanoen hyvin suuriss dimensiossa melkein kaikki datapisteet ovat hyperpallon pinnalla. (Aputulos: r-säteisen d-ulotteisen hyperpallon tilavuus on $V_d(r) = C_d r^d$ missä vakio C_d ei riipu säteestä r.)

2.
On annettu otos $x(1), ..., x(n)$ suureesta, jonka tiedetään olevan normaalijakautunut

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

On syytä olettaa että keskiarvo μ on lähellä nollaa. Koodataan tämä olettamus priorijakaumaan

$$p(\mu) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\mu^2}.$$

Laske Bayes-MAP-estimaatti odotusarvolle μ ja tulkitse sitä kun varianssi σ^2 vaihtelee pienestä suureen.

3.
Tarkastellaan 1-ulotteista 3 yksikön SOM-karttaa, jonka painot ja syöte ovat skalaareja väliltä [0,1]. Yksikön 1 naapuri on 2, yksikön 3 naapuri on 2, ja yksikön 2 naapurit ovat 1 ja 3. Alkutilanteessa painot ovat $m_1 = 0.5$, $m_2 = 0.25$ ja $m_3 = 0.75$. Kun uusi syöte x on valittu, etsitään ensin lähin yksikkö ja sitten sen ja sen naapureiden painoja päivitetään sääntöllä

$$m_i^{\text{uusi}} = m_i + 0.5(x - m_i).$$

Valitse syöte x niin, että päivityksen jälkeen uudet painot ovat suurusjärjestyksessä:

$$m_1^{\text{uusi}} < m_2^{\text{uusi}} < m_3^{\text{uusi}}.$$

4.
(a) Määrittele 0-1 datan kattava joukko (frequent set). Anna esimerkki pienestä 0-1-datajoukosta ja luettele sen kattavat joukot jollakin sopivalla kynnysarvolla N.
(b) Kuvaile kattavien joukkojen etsintään käytettävän tasoittaisen algoritmin toimintaperiaate.

5.
Vastaa jompaan kumpaan seuraavista esseealheista, jotka liittyvät Matlab-harjoitustehtäviin:
(a) "Ominaisarvot" ja ominaisarvojen käyttö luovokuvien ryhmittelyyn
(b) k:n lähimmän naapurin luokitin (k-nearest neighbour classifier).

Kuva 1.3: Tyypillinen tenttipaperi. Lähde: tenttiarkisto.fi.

Käytäntöä Tentin pisteet ja bonuspisteet sekä arvosanarajat

- Paperilaskareissa 1 bonuspistetehtävä / kierros. Bonuspistetehtävät arvostellaan 0 / 0,5 / 1 bonuspiste.
- Lisäksi kurssipalautteen antamisesta saa 1 bonuspisteen.
- Maksimimäärä on 6 bonuspistettä, joka skaalataan tenttipisteiksi jakamalla kahdella (max 3 tenttipistettä). Tämä lisätään tenttipisteisiin sillä ehdolla, että tentistä on saanut vähintään 15 pistettä.
- Bonuspisteet ovat voimassa vain joulu- ja tammikuun tentissä!
- Arvosanarajat: 0 = 0–14,5p; 1 = 15,0–17,5p; 2 = 18,0–20,5p; 3 = 21,0–24,5p; 4 = 25,0–27,5p; 5 = 28,0–33,0p.

Käytäntöä Tiedotus

- Erityisistä muutoksista ilmoitetaan Nopassa
- Ongelmatapauksissa ota yhteyttä laskuharjoitusassistenttiin
- Kurssin opetushenkilökunnan yhteinen sposti `t612010@ics.tkk.fi`

Kurssin suorittamisesta

- Kurssin osaamistavoitteet: `https://noppa.aalto.fi/noppa/kurssi/T-61.2010/esite`
  - Kurssin jälkeen osaat selittää, kuinka luonnollinen data kuten kuvat, puhe, mittaussarjat esitetään digitaalisesti tietokannassa. Osaat soveltaa tilastomatemaattisia ja algoritmisia perusmenetelmiä (yksinkertaisimmassa muodossaan) tällaisen datan käsittelyyn. Kurssin lopussa osaat keskustella asiaankuuluvalla terminologialla, miten "datasta tietoon"käytännössä toteutuu.
- *Mitä "opetustekoja" tarjoamme "oppimistekojanne" varten? Miten tavoitteiden saavuttamista mitataan?*
- **Luennot** tarjoavat pohjan.
- Luentokalvot ovat saatavilla Nopassa. Tutustu aiheeseen etukäteen. Kertaa asioita luennon lopuksi. Kysy epäselviksi jääneistä asioista.
- Luentokalvot itsessään eivät ole täydellinen oppikirja!
- **Paperilaskareissa** käydään läpi matemaattista laskemista.
- Laskareiden ymmärtäminen vaatii omaa työtä: tehtävien läpikäymistä itse, välivaiheiden ratkaisua, luennoilla saatua laajempaa ymmärrystä.
- **Tietokonelaskareissa** sovelletaan ja kokeillaan.
- **Tentissä** on viisi kysymystä, joista yksi on suoraan paperilaskareista, ainakin yksi liittyy kurssin loppupuolen lukuihin (Mannila) ja yksi liittyy tietokonelaskareissa sovellettaviin asioihin.

Kurssipalautteesta

- Kurssin kurssipalaute kerätään joulu-tammikuussa
- Palautteen antamisesta saa **yhden bonuspisteen** joulu- ja tammikuun tenttiin
- Luemme palautteen sekä kurssin päättyessä että uuden alkaessa
- Palautteessa opiskelijoiden suurin yksittäinen haaste: kurssin matemaattisuus $\rightarrow$ tänä vuonna viikoittaiset bonuspistetehtävät
- Viime syksynä palautteessa kiitelty mm. laskutupaa, jossa saattoi harjoitella tenttiviikolla (tämä syksy?)

1.3 Kurssin sisältö luvuittain

Kurssin materiaalissa 9 lukua ja 10 luentoa. Nopassa näkyy aikataulu vs luennot.
Lukujen yhteys toisiinsa?

- Luku 1 + Luku 2 johdantoa. (2h)
- Luku 3 + Luku 4 numeerisen datamatriisin käsittelyä. (4h)
- Luku 5 estimointi (4 h)
- Luku 6 + Luku 7 hahmontunnistusta: luokittelu, ryhmittely (6 h)
- Luku 8 + Luku 9: diskreettejä hahmoja (4 h)

Kurssin sisältö luvuittain

1. Johdanto (1 h)
    - Miksi tällainen kurssi?
    - Käytäntöä
    - Kurssin sisältö
    - Kurssimateriaali
2. Datasta tietoon: mitä dataa, mitä tietoa? (1 h)
    - Data-analyysin ongelma
    - Mallit ja oppiminen
    - Esimerkkejä
    - Case study: WEBSOM
3. Data vektorina (2 h)
    - Vektorit, matriisit, etäisyysmitat
    - Datan piirreirrotus ja vektorointi
    - Dimensionaalisuuden kirous
    - Esimerkki piirrreirrotuksesta: PicSOM
4. Vektoridatan tiivistäminen ja dekorrelointi (2 h)
    - Pääkomponenttianalyysi
    - Esimerkkejä

- DSS-menetelmä: halutunlaisten aikakomponenttien etsiminen

5. Estimointiteorian perusteita (4 h)
    - Perusjakaumat 1-ulotteisina
    - Yleistys vektoridatalle, d:n muuttujan normaalijakauma
    - Suurimman uskottavuuden periaate
    - Bayes-estimointi
    - Regressiosovitus
    - Esimerkki regressiosta: neuroverkko
    - Esimerkkejä
6. Hahmontunnistuksen perusteita (4 h)
    - Johdanto
    - Hahmoalueet, erotinfunktio
    - Lähimmän naapurin luokitin (kNN)
    - Bayes-optimaalinen luokitin
    - Ryhmittelyanalyysi (*c-means*-algoritmi ja hierarkinen ryhmittely)
7. Itseorganisoiva kartta (2 h)
    - Perusidea
    - Yhteys biologiaan
    - Suppenevuus 1-ulotteisessa tapauksessa
    - Käytännön valintoja
    - Mihin SOM:ia käytetään?
    - Esimerkkejä
8. Hahmojen etsintä diskreetistä datasta (2 h)
    - Miten muodostetaan hyviä paikallisia kuvauksia datan osista
    - Algoritmi: Tasottainen algoritmi kattavien joukkojen etsintään
9. Web-etsintämenetelmien algoritmit (2 h)
    - Perusongelmat
    - Linkkirakenteen ottaminen huomioon relevanttien sivujen etsimisessä
    - Algoritmit: Keskukset ja auktoriteetit sekä PageRank (Google)

2 Datasta tietoon: mitä dataa? mitä tietoa?

2.1 Data-analyysin ongelma

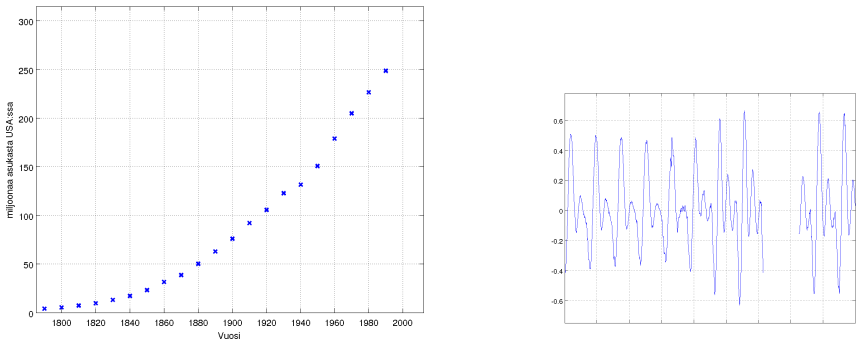
Data-analyysin ongelma

- Tulevien vuosien valtava haaste on digitaalisessa muodossa talletetun datan kasvava määrä
- Arvioita:
  - Yhdysvaltojen kongressin kirjasto Washingtonissa: 32 miljoonaa kirjaa ja lehteä, 3 miljoonaa äänitettä, 14.7 miljoonaa valokuvaa, 5.3 miljoonaa karttaa, 61 miljoonaa käsikirjoitusta. Kerätty *200 vuoden aikana*. Nyt sama datamäärä kertyy levyille *joka 15. minuutti* (noin 100 kertaa vuorokaudessa).
  - Tämä on 5 exatavua vuodessa. (Kertaus: Exatavu = 2^{60} tavua = 1,152,921,504,606,846,976 tavua $\approx 1.15 \times 10^{18}$ (triljoona) tavua).
  - Sama määrä tulisi, jos *kaikki ihmispuhe kaikkina aikoina* (n. 100.000 vuotta) koodattaisiin sanoiksi ja digitoitaisiin (R. Williams, CalTech).
- Aiemmin talletettu data oli lähinnä tekstiä ja numerodataa (taloudellishallinnollinen IT), mutta nyt yhä enemmän ns. reaalimaailman dataa (digitaaliset kuvat, videot, äänet, puhe, mittaustiedot, bioinformatiikan tietopankit jne.)
- Lopulta mikä tahansa tieto josta voi olla hyötyä saattaa tulla digitaalisesti haettavaksi, esim. Webin kautta
- Tämä asettaa suuria haasteita tallennus- ja tietokantatekniikoille
- Eräs keskeinen kysymys: *kuinka haluttu tieto (information, knowledge) löytyy?* Tarvitaan jonkinlaisia "älykkäitä" datan analyysi-, louhinta- ja hakumenetelmiä.

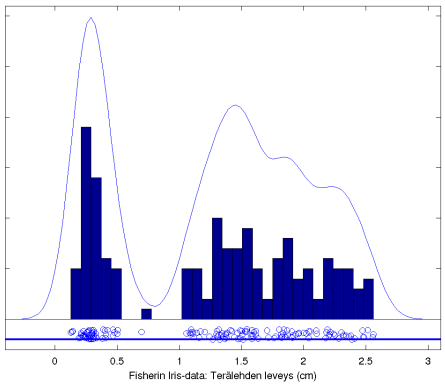
2.2 Mallit ja oppiminen

Mallit ja oppiminen

- Peruslähtökohta data-analyysille on datan *mallitus*
- Malli tarjoaa tiivistetyn *esitystavan (representaation)* datalle
- Mallin perusteella on paljon helpompi *tehdä päätelmiä* kuin raakadatasta
- Esimerkki: aikasarjan ennustaminen (kuva 2.1)
- Toinen esimerkki: datan todennäköisyysjakauma (kuva 2.2)

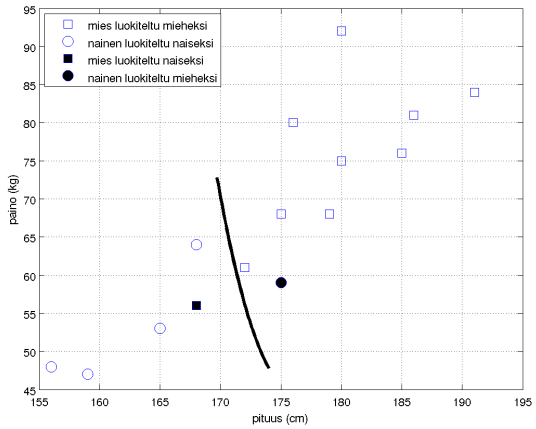


Kuva 2.1: (a) Yhdysvaltain asukasmäärä 1790-1990. Onko kasvu jatkunut? (b) Äänisignaalin aaltomuoto. Miten täytetään puuttuva kohta?

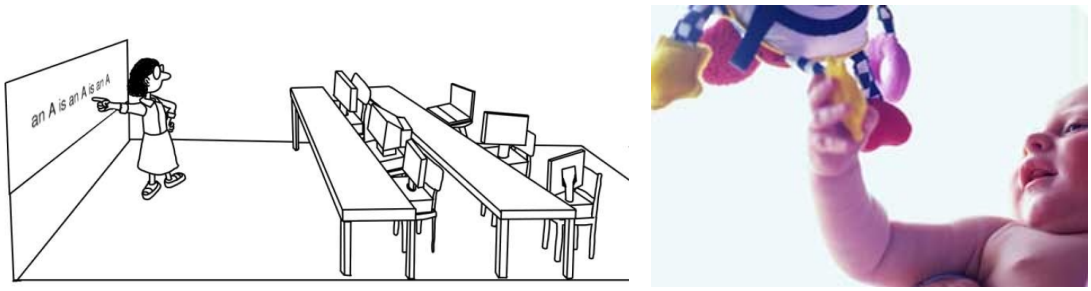


Kuva 2.2: Datapisteet, histogrammi ja estimoitu jakauma. Fisherin kuuluisa datajoukko kolmesta liljalajista. Terälehden leveys mitattu 50 yksilöstä kolmesta lajista. Yksi laji erottuu, kaksi menee “sekaisin”. HUOM! näytteet ripoteltu y-akselin suunnassa.

- Kolmas esimerkki: luokitus (kuva 2.3)
- Mistä malli sitten löytyy?
- Joskus voidaan käyttää olemassaolevaa tietoa (fysikaalisia luonnonlakeja, inhimillistä kokemusta, tms)
- Usein kuitenkin joudutaan käyttämään *tilastollisia malleja*, jotka muodostetaan suoraan datan perusteella (kuten edellisissä esimerkeissä)
- Kurssi “datasta tietoon” (ainakin alkuosa) käsittelee tilastollisia malleja ja niiden johtamista datajoukoista.
- Usein mallin automaattista muodostamista datajoukosta kutsutaan *koneoppimiseksi*



Kuva 2.3: Pituus- ja painohavainnot 15 ihmisestä. Laskettu luokitin, joka jakaa tason kahteen osaan (vain osa rajasta piirretty). Kun tunnetaan luokat, tiedetään, että luokitin tuottaa kahdessa tapauksessa virheellisen luokituksen.



Kuva 2.4: Koneoppimista vs ihmisen oppiminen.

- Sana tulee ihmisen oppimisesta, joka myös pohjimmiltaan on mallien oppimista
- Datasta oppimisen menetelmät jakaantuvat kahteen pääluokkaan: *ohjattu oppiminen* ja *ohjaamaton oppiminen*
- Ohjatussa (kone)oppimisessa annetaan joukko data-alkioita ja niitä vastaavia nimikkeitä joihin ne halutaan liittää: esimerkiksi äänipätkä ihmisen puhetta a-äänteen osalta ja kirjainsymboli “a”
- Tehtävä: muodosta malli joka liittää toisiinsa data-alkiot ja nimikkeet (automaattista puheentunnistusta varten)
- Vastaa ihmisellä opettajan johdolla tapahtuvaa oppimista
- Ohjaamattomassa (kone)oppimisessa annetaan vain joukko data-alkioita mutta ei mitään muuta; esimerkiksi iso määrä kaupan asiakkaistaan keräämiä tietoja

- Tehtävä: muodosta malli, joka yhdistää toisiinsa samoista tuotteista kiinnostuneet asiakkaat (täsmämainontaa varten)
- Vastaa ihmisellä itsekseen tapahtuvaa oppimista.

2.3 Esimerkkejä

Esimerkkejä

- Kieliteknologia
  - Konekääntäminen luonnollisten kielten välillä
  - Puheen tunnistaminen (audiotietokannan automaattinen muuuntaminen tekstiksi; TV-lähetysten on-line tekstitys)
- Käyttöliittymät
  - Puheentunnistus
  - Käsinkirjoitettujen merkkien tunnistus
  - Eleiden, ilmeiden, katseen suunnan tunnistus
  - Käyttäjän profilointi
- Web-haku
  - Googlen laajennukset; semanttinen verkko
  - Oppiva semantiikka; ontologioiden tms. tiedon rakenteiden automaattinen muodostus
- Teknistieteellinen data
  - Tietoliikenne (verkon kuormituksen ennustaminen; ympäristöään tarkkaileva kännykkä)
  - Neuroinformatiikka (biolääketieteen mittaukset kuten EEG, MEG, fMRI); ihmisen ja koneen väliset kehittyneet käyttöliittymät
  - Bioinformatiikka: geenisekvenssit, DNA-sirudata
  - Ympäristön tila, ilmasto
  - Sensoriverkot
- Taloudellinen data
  - Aikasarjojen (pörssikurssit, valuuttakurssit) ennustaminen
  - Yritysten tilinpäätöstietojen analyysi
  - Luottokorttien käytön seuranta
  - Asiakkaiden ryhmittely ja käyttäytymisen ennustaminen (ostoskorianalyysi)
- .... ja paljon paljon muuta!

2.4 Case study: WEBSOM

Case study: WEBSOM

- WEBSOM (Web Self Organizing Map) on informaatiotekniikan laboratoriossa prof. Teuvo Kohosen johdolla kehitetty dokumenttien selaus- ja hakujärjestelmä
- Se perustuu SOM (Self Organizing Map) -neuroverkkoon
- Suurimmassa sovelluksessa tehtiin kartta 7 miljoonalle dokumentille, jotka ovat elektronisessa muodossa olevia patenttien tiivistelmiä
- WEBSOMin visuaalisen käyttöliitymän avulla voi helposti selailla tietyn alan patentteja. Demo: `http://websom.hut.fi/websom/stt/doc/fin/`

3 Data vektoreina

Johdanto

Aiheeseen liittyviä laskari- ja demotehtäviä:

- H1/1, H1/2, H1/3: konvoluutio, Fourier-muunnos, signaalin suodatus
- H1/5, H1/6: korkeaulotteiset avaruudet
- T1: äänisignaalin suodattaminen Matlabilla

3.1 Datamatriisi

Datamatriisi $\mathbf{X}$ voidaan esittää matriisina, jossa n saraketta (havaintoa) ja d riviä (dimensio):

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(1) & \mathbf{x}(2) & \mathbf{x}(3) & \dots & \mathbf{x}(n) \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3n} \\ \vdots & & & \ddots & \vdots \\ x_{d1} & \dots & \dots & \dots & x_{dn} \end{bmatrix}$$

Tietokonelaskareissa esitellään datamatriisi $\mathbf{X}$, jonka dimensioina ovat ihmisen pituus (cm) ja paino (kg). Tällöin esimerkiksi

$$\mathbf{X} = \begin{bmatrix} 186 & 175 & 180 & 165 \\ 81 & 68 & 75 & 53 \end{bmatrix}$$

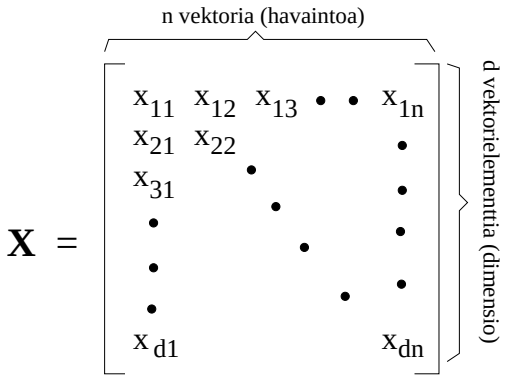
on havaintomatriisi neljästä kaksiulotteista havainnosta (neljän ihmisen pituudesta ja painosta):

$$\mathbf{x}(1) = \begin{bmatrix} 186 \\ 81 \end{bmatrix}, \quad \mathbf{x}(2) = \begin{bmatrix} 175 \\ 68 \end{bmatrix}, \quad \mathbf{x}(3) = \begin{bmatrix} 180 \\ 75 \end{bmatrix}, \quad \mathbf{x}(4) = \begin{bmatrix} 165 \\ 53 \end{bmatrix}.$$

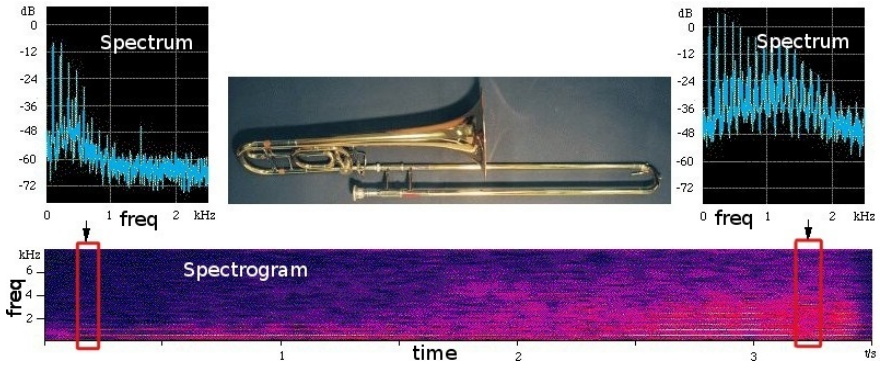
Arvot voidaan myös luetella yksitellen $x_{11} = 186$, $x_{12} = 175$, …, $x_{24} = 53$. Datapisteet $\mathbf{x}(i)$ voidaan ajatella geometrisina pisteinä. Kun datan dimensio on 2, niin pisteet voidaan visualisoida xy-koordinaatistossa. Jos datan dimensio on suurempi, visualisointi ja sen ymmärtäminen on hankalaa (mahdotonta?).

Datamatriisi

- Ajatellaan jotakin datajoukkoa joka on talletettu datamatriisiin $\mathbf{X}$:



- Matriisin sarakkeet ovat siis yksittäisiä datavektoreita ja kukin rivi vastaa tiettyä vektorialkiota (*mittaustulosta, piirrettä, attribuuttia, surretta*)
- Esimerkiksi datavektori voisi olla äänispektri soittimen äänestä, jolloin vektorialkiot vastaisivat taajuuksia. Koko datamatriisi on tällöin ns. *spektogrammi*



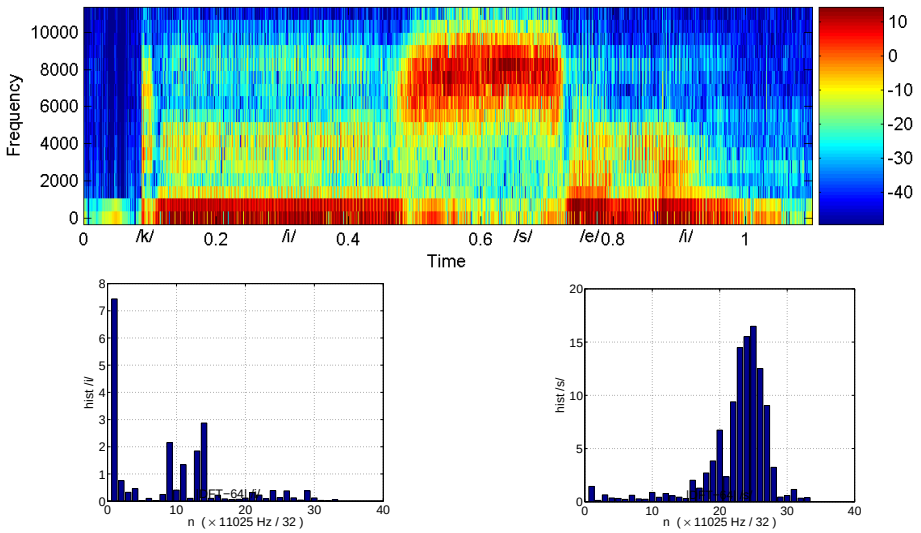
Kuva 3.1: Spetrogrammista selviää kunakin ajanhetkenä kunkin taajuuskomponentin voimakkuus.

- Spektrogrammissa siis matriisi $\mathbf{X} = \begin{bmatrix}\mathbf{x}(1) & \mathbf{x}(2) & \dots & \mathbf{x}(n)\end{bmatrix}$, jossa $\mathbf{x}(i)$:t ovat peräkkäisissä aikaikkunoissa (esim. 20 ms välein) laskettuja diskreettejä Fourier-muunnoksia

  (katso sivu 21)

- Toinen esimerkki spektrogrammista: ihmisen puhe

  (kuva 3.2)
- Kaikkea dataa ei suinkaan voi esittää vektorina mutta tämä kattaa hyvin paljon tapauksia. Vrt. myös relaatiotietokannat joissa tietueet on talletettu binäärivektoreina
- Merkitään seuraavassa d-ulotteista vektoria merkinnällä $\mathbf{x}$ ja sen alkioita $x_1, x_2, ..., x_d$.



Kuva 3.2: Ylhäällä spektrogrammi sanasta "kiisseli" Sanan Fourier-muunnoksen eli spektrin histogrammi kahdessa kohdassa sanaa: vasemmalla /i/, oikealla /s/. Vastaavan ajan mukana “tanssivan” patsasrivin näet “missä tahansa” audiosoittimessa.

- Oletan että vektorien yhteenlasku $\mathbf{x} + \mathbf{y}$, sisätulo $\mathbf{x}^T\mathbf{y}$ ja normi $\|\mathbf{x}\|$ ovat tunnettuja, samoin matriisilla kertominen $\mathbf{Ax}$, matriisien yhteenlasku $\mathbf{A} + \mathbf{B}$ ja tulo $\mathbf{AB}$ sekä matriisin ominaisarvot ja -vektorit $\mathbf{Ax} = \lambda\mathbf{x}$.

Esitietomatematiikkaa. Olkoon

$$
\begin{aligned}
\mathbf{x} &= \begin{bmatrix} 3 & 2 & 5 \end{bmatrix}^T \\
\mathbf{y} &= \begin{bmatrix} 1 & 0 & -3 \end{bmatrix}^T \\
\mathbf{A} &= \begin{bmatrix} 2 & -3 & 0 \\ 1 & 5 & 3 \\ 0 & 2 & 3 \end{bmatrix} \\
\mathbf{B} &= \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 2 \\ -2 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

Tällöin

$$
\begin{aligned}
\mathbf{x}+\mathbf{y} &= \begin{bmatrix}4 & 2 & 2\end{bmatrix}^T \\
\mathbf{x}^T\mathbf{y} &= \begin{bmatrix}3 & 2 & 5\end{bmatrix}\begin{bmatrix}1\\0\\-3\end{bmatrix} = 3\cdot 1 + 2\cdot 0 + 5\cdot(-3) = -12 \\
||\mathbf{x}|| &= \sqrt{3^2+2^2+5^2} = \sqrt{38} \approx 6.16 \\
\mathbf{A}\mathbf{x} &= \begin{bmatrix}2 & -3 & 0\\1 & 5 & 3\\0 & 2 & 3\end{bmatrix}\begin{bmatrix}3\\2\\5\end{bmatrix} = \begin{bmatrix}2\cdot 3 - 3\cdot 2 + 0\cdot 5\\1\cdot 3 + 5\cdot 2 + 3\cdot 5\\0\cdot 3 + 2\cdot 2 + 3\cdot 5\end{bmatrix} = \begin{bmatrix}0\\28\\19\end{bmatrix} \\
\mathbf{A}+\mathbf{B} &= \begin{bmatrix}3 & -3 & 1\\1 & 7 & 5\\-2 & 2 & 4\end{bmatrix}
\end{aligned}
$$

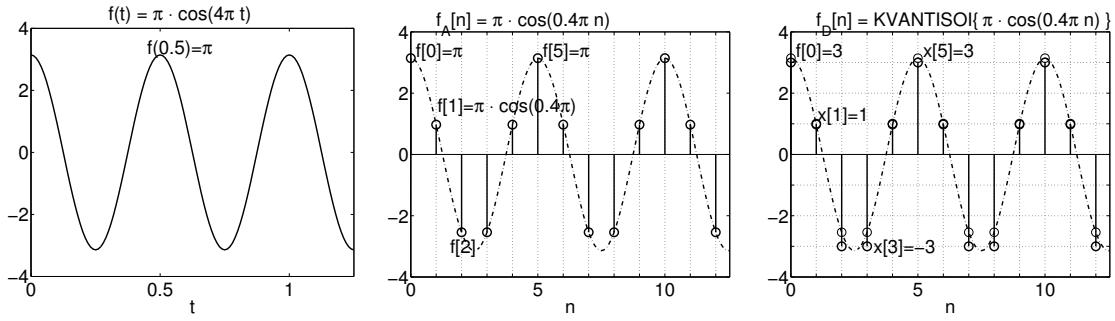
Sama Matlabilla tai Octavella

```
x = [3 2 5]'
y = [1 0 -3]'
A = [2 -3 0; 1 5 3; 0 2 3]
B = [1 0  1; 0 2 2; -2 0 1]
x+y
x'*y
sqrt(sum(x.^2))
norm(x)
A*x
A+B
```

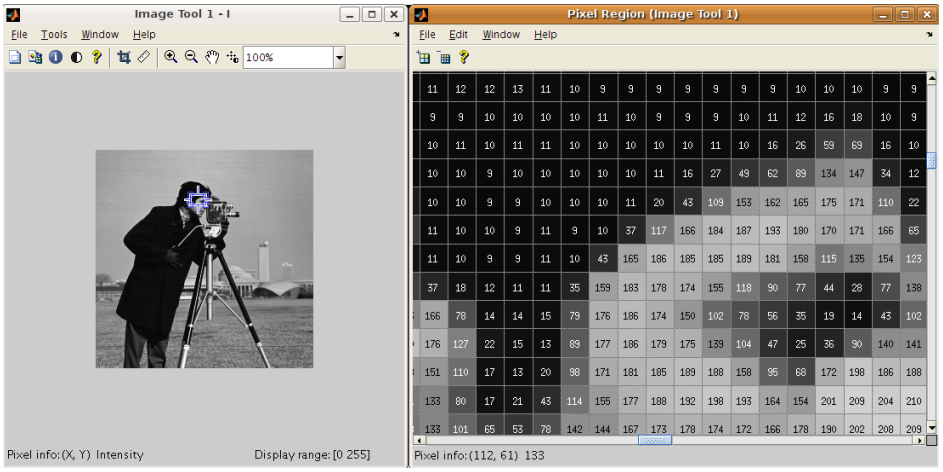
3.2 Piirreirrotus: ääni- ja kuvasignaalit

Datan piirreirrotus ja vektorointi Analogisen datan digitointi

- Usein data on jo vektorimuodossa: siis järjestetty joukko lukuja (jatkuva-arvoisia, binäärisiä ...).
- Jotta jatkuva (analoginen) data saataisiin tähän muotoon, se täytyy diskretoida (digitoida).
- Analoginen *signaali*, esim. ääni, kuvataan ajan mukana vaihtuvana jatkuvana funktiona $f(t)$
- Sekä aika t että amplitudi f täytyy diskretoida jotta ne voi esittää digitaalisessa muodossa
- Esim. digikamera, tietokoneen äänikorttiin kytketty mikrofoni
- Käytännössä mittalaite tekee sen analogia-digitaali (A/D) -muuntimensa avulla



Kuva 3.3: (a) analoginen signaali $f(t)$; (b) ajan suhteen digitoitu lukujono $f_A[n] \in R$, $n \in Z$; (c) ajan ja amplitudin suhteen digitoitu $f_D[n] \in Z$. Äänisignaalissa tyypillinen digitointi 8 bitillä antaa $2^8 = 256$ tasoa y-akselille välillä $-1 \ldots +1$.



Kuva 3.4: Matlabin "kameramies" ja yksityiskohta silmän alueelta. Resoluutio 256 × 256, tyyppi `uint8` eli $[0, 255] \in Z$, jossa 0=musta ja 255=valkea.

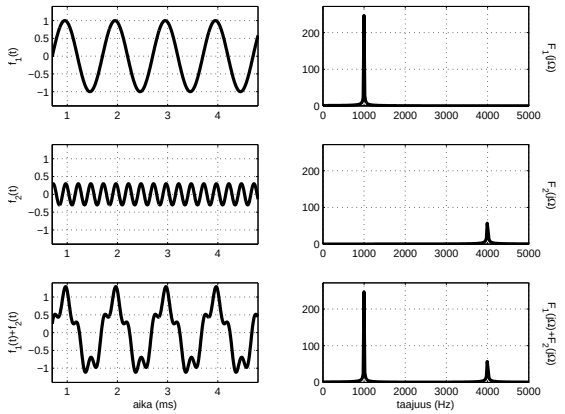
- Tähän liittyvä teoreettinen kysymys on *näytteistys (sampling)*: kuinka tiheään näytteitä tulee ottaa, ettei hävitetä tietoa?
- Yksiulotteinen signaali $f(t)$ muutetaan jonoksi $f_m = f(mT)$ missä T on näytteistysväli
- Jos näytteistystaajuus $1/T$ on riittävän suuri verrattuna $f(t)$:n suurimpaan taajuuteen, voidaan itse asiassa koko $f(t)$ palauttaa näytteistään (näytteistyslause, sampling theorem; ks. kurssi T-61.3015 Digitaalinen signaalinkäsittely ja suodatus)
- Demo näytteenotosta `http://www.jhu.edu/~signals/sampling/`

Datan piirreirrotus ja vektorointi Taajuusspektri

- Mitä tarkoittaa $f(t)$:n taajuus?
- Funktiolla $f(t)$ on *taajuusspektri* joka määritellään *Fourier-muunnoksen* avulla

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}\mathrm{d}t$$

  missä ω on (kulma)taajuus ja $i = \sqrt{-1}$
- Tämän kompleksifunktion itseisarvo voidaan piirtää käyränä, ja se on ns. amplitudispektri (magnitudispektri)
- Perusteluja: taajuus on jaksollisen funktion ominaisuus (montako kertaa sekunnissa jakso toistuu), ja sinikäyrä $f(t) = sin(\alpha t + \theta)$ on kaikkein tyypillisin jaksollinen funktio. α on sen (kulma)taajuus, θ on vaihe
- Sinikäyrän Fourier-muunnos on nolla paitsi kulmataajuuden arvoilla $\omega = \pm\alpha$ (miksi?), joten F-muunnos paljastaa sinin taajuuden.
- Jos $f(t)$ on summa monesta sinikäyrästä, niiden kaikkien taajuudet näkyvät piikkeinä *taajuusalueessa* eli kompleksitasossa.

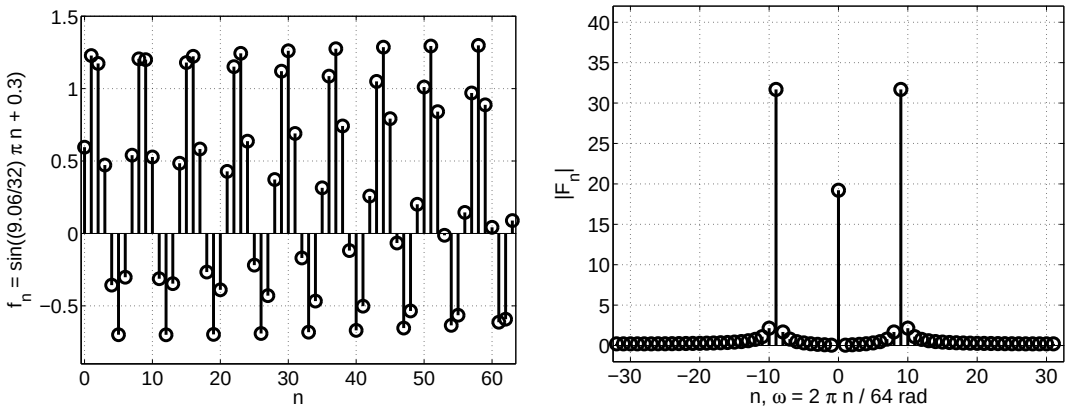


Kuva 3.5: Kaksi sinisignaalia ja niiden summa aika- ja taajuustasossa.

- *Diskreetti Fourier-muunnos* (DFT) määritellään digitoidulle signaalille $f_m, m = 0, ..., T-1$:

$$F_n = \sum_{m=0}^{T-1} f_m e^{-2\pi imn/T}$$

- Demo Fourier-sarjalla approksimoinnista `http://www.jhu.edu/~signals/fourier2/`



Kuva 3.6: 64-pituinen lukujono f_m, jossa keskiarvo ja yksi sini. Oikealla sen DFT-64 F_n, jossa korkeimmat piikit $n = 0$ ja $n = \pm 9$.

Esimerkki. Lasketaan tässä lukujonon $f_m = \begin{bmatrix} 2 & 0 & 1 & 1 \end{bmatrix}$ diskreetti Fourier-muunnos $F_n = \sum_m f_m e^{-2\pi i m n/T}$. Toisin sanoen $f_0 = 2$, $f_1 = 0$, $f_2 = 1$, $f_3 = 1$, ja $T = 4$. Tällöin

$$
\begin{aligned}
F_0 &= \sum_{m=0}^{3} f_m e^{-2\pi i m \cdot 0/4} = 2 + 0 + 1 + 1 = 4 \\
F_1 &= \sum_{m=0}^{3} f_m e^{-2\pi i m \cdot 1/4} = 2 + 0 \cdot (-i) + 1 \cdot (-1) + 1 \cdot i = 1 + i \\
F_2 &= \sum_{m=0}^{3} f_m e^{-2\pi i m \cdot 2/4} = 2 + 0 \cdot (-1) + 1 \cdot 1 + 1 \cdot (-1) = 2 \\
F_3 &= \sum_{m=0}^{3} f_m e^{-2\pi i m \cdot 3/4} = 2 + 0 \cdot i + 1 \cdot (-1) + 1 \cdot (-i) = 1 - i
\end{aligned}
$$

eli $F_n = \begin{bmatrix} 4 & 1+i & 2 & 1-i \end{bmatrix}$.

Huomaathan, että kompleksista eksponenttifunktiota voidaan hyvin tarkastella napakoordinaatistossa ja se voidaan hajottaa Eulerin kaavalla suorakulmaiseen koordinaatistoon:

$$e^{i\omega} = \cos(\omega) + i \cdot \sin(\omega)$$

Esimerkiksi $e^{-2\pi i \cdot 1 \cdot 1/4} = e^{i(-pi/2)} = \cos(-\pi/2) + i \sin(-\pi/2) = 0 + i(-1) = -i$.

Matlabissa tai Octavessa (`http://octave.sourceforge.net`) tämä voidaan laskea

```
f = [2 0 1 1]
F = fft(f)
```

Äänenkäsittelyssä lukujonon pituus T on joku kahden potenssi, tyypillisesti esim. 256, 512 tai 1024.

Datan piirreirrotus ja vektorointi Taajuussuodatus

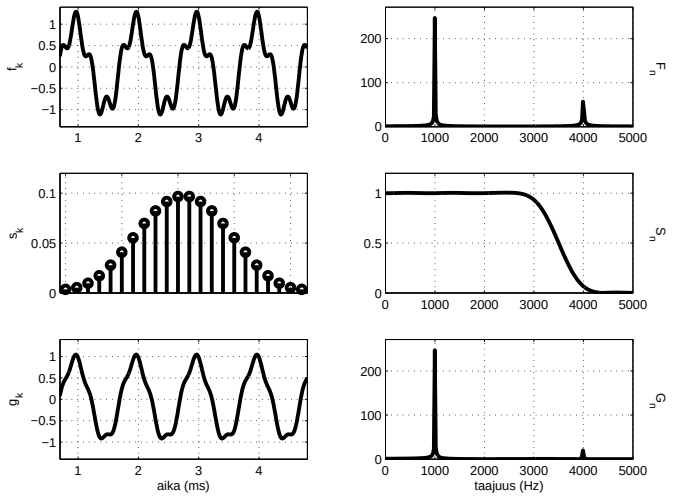
- Usein aikasignaalista halutaan poistaa tiettyjä taajuuksia, esim. korkeita taajuuksia jotka vastaavat kohinaa tai verkkovirran taajuus 50 Hz.
- Tässä käytetään hyväksi lineaarisia suotimia, jotka laskennallisesti toteutetaan ns. *konvoluutiona*:

$$g_k = \sum_{m=-\infty}^{\infty} f_m s_{k-m}$$

  missä f_m on suodatettava digitaalinen signaali ja s_j on vastaavasti suodin (äärellinen lukujono).
- On helppo todistaa että konvoluution Fourier-muunnokselle pätee

$$G_n = F_n \cdot S_n$$

  eli taajuustasossa spektrit kerrotaan keskenään
- Jos siis halutaan estää / korostaa tiettyjä taajuuksia, valitaan suodinjono s_k siten että sen F-muunnoksessa S_n nämä taajuudet ovat heikkoja / vahvoja.
- Esimerkki suodatuksesta aika- ja taajuustasossa. Tässä tehdään alipäästösuodatus (keskiarvoistaminen) aikatasossa $g_k = \sum_m f_m s_{k-m}$ vasemmassa sarakkeessa tai vastaavasti taajuustasossa $G_n = F_n \cdot S_n$. Vertaa ekvalisaattorin toimintaan kotistereoissa tai musiikkisoittimessa.



Kuva 3.7: Signaalin suodatus. Vasemmalla aikatason digitaaliset lukujonot (signaalit): sisääntulo f_k, suodinjono s_k ja suodatuksen lopputulos g_k. Oikealla vastaavat Fourier-muunnetut spektrit $f_k \leftrightarrow F_n$. Korkeataajuinen komponentti vaimenee suodatuksessa.

- Suodattimen suunnittelu käytännössä on oma taiteenlajinsa, ks. T-61.3015 Digitaalisen signaalinkäsittelyn ja suodatuksen kurssi.

- Joko aika-alueen digitaalinen signaali f_m tai taajuusalueen digitaalinen signaali $|F_n|$ (amplitudispektri, mahdollisesti täydennettynä vaiheella) voivat sellaisenaan olla vektoreita joita jatkossa analysoidaan. Esimerkkinä puheen spektrogrammi.

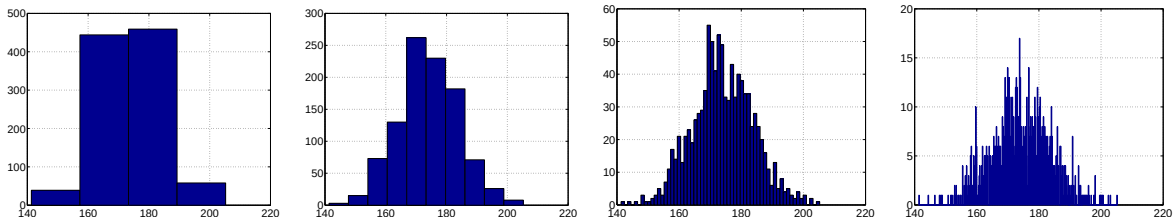
- Kuville on olemassa 2-ulotteinen DFT jota voi vastaavalla tavalla käyttää kuvien suodatukseen; vrt. digikameroiden kuvankäsittelyoptiot.

- Spektriä voi myös kokonaisuudessaan tai osina käyttää tehokkaana piirrevektorina kuville. Se paljastaa esim. kuvassa olevien viivojen yleisimmät suunnat. Enemmän kurssissa T-61.5100 Digitaalinen kuvankäsittely.

Kuvan 3.7 tulkintaan liittyen: vasemman sarakkeen f_k, s_k ja g_k ovat kaikki diskreettejä lukujonoja, joista syöte f_k ja vaste g_k on kuvaan piirretty jatkuvalla käyrällä, joka on saatu yhdistämällä diskreetit vierekkäiset pisteet yhteen suoralla viivalla. Syötteen ja vasteen osalta kuvassa on noin 4 millisekunnin jakso ja jaksonaika $T_0 = 1$ ms vastaten $f_0 = 1/T_0 = 1000$ Hz eli 1000 värähdystä sekunnissa. Näytteenottotaajuus esimerkissä on $f_T = 10000$ Hz eli 10000 näytettä sekunnissa. Kun suodinjono s_k on noin 21 merkkiä pitkä, niin se vastaisi ajallisesti noin 0.2 millisekuntia. Suodin on alipäästötyyppinen eli keskiarvoistaa, mikä nähdään siinä, että g_k on "pehmeämpi" kuin syöte.

Muita tyypillisiä piirteitä eri datatyypeille Histogrammi

- Hyvin yleinen ja tehokas piirretyyppi on *histogrammi*, jossa suureen arvot "lokeroidaan"



Kuva 3.8: Normaalijakautuneita satunnaislukuja $N = 1000$ kpl. Sama data, mutta pylväiden lukumäärä 4, 10, 60 ja 300. Histogrammissa pylvään korkeus kertoo, kuinka monta arvoa osui lokeroon.

- DFT on esimerkki tästä eli taajuushistogrammi: paljonko signaalissa tai kuvassa on tiettyä taajuutta

- Kuvan värihistogrammi kertoo, montako pikseliä kuvassa on kutakin väriä (dimensio 16.7 miljoonaa tiivistämättömänä, kun kolmevärinen 8-bittinen esitys)

- Tekstidokumentin sanahistogrammi kertoo montako kertaa kukin sana esiintyy dokumentissa (dimensio 50.000 - 100.000 tiivistämättömänä)

Muita tyypillisiä piirteitä eri datatyypeille Kuva ja ääni

- Hyvä piirteiden lähde ovat standardit, etenkin JPEG (joint photographic experts group) ja MPEG (moving pictures expert group); esim. MPEG-1 Audio Layer 3 eli MP3 äänelle ja musiikille

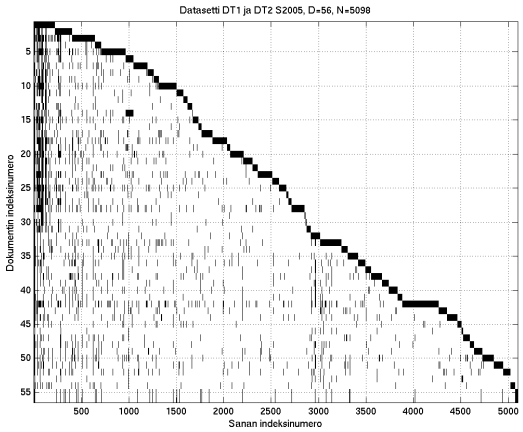
Kuva 3.9: Alkuperäinen kuva, josta kolme eri JPEG-tallennusta. Yksityiskohtakuvien tavukoot 42 kT, 12 kT, 6 kT.

- Ne on kehitetty äänen, kuvien ja videon pakkaamiseen, mutta pakattua muotoa voi käyttää myös piirteenä automaattisessa analyysissä

Muita tyypillisiä piirteitä eri datatyypeille Teksti vektorina

- Edellä mainittiin jo sanahistogrammit
- Jos yksittäisten dokumenttien sanahistogrammit ovat datamatriisin $\mathbf{X}$ sarakkeina, sitä kutsutaan termi-dokumentti-matriisiksi.
- Kuvan (kuva 31)

  esimerkissä termi-dokumenttimatriisi,jossa kukin rivi vastaa yhtä dokumenttia ja kukin sarake vastaa yhtä sanaa. Sanojen lukumäärät ≥ 1 on korvattu mustalla pisteellä. Sarakkeet on järjestetty sanojen esiintymisjärjestykseen. Ensimmäisessä dokumentissa yli 200 sanaa, toisessa myös noin 200 sanaa, joista noin 150 uusia verrattuna ensimmäiseen, jne. Lähde: 56 harjoitustyötä DT-kurssilla 2005. HUOM! Dataan on tahallisesti lisätty yksi “lähes” kopiodokumentti. Mikä dokumentti?



Kuva 3.10: Termi-dokumentti-matriisi, jossa kukin rivi ($D = 56$) vastaa yhtä dokumenttia ja kukin sarake ($N = 5098$) vastaa yhtä sanaa.

- Käytännössä matriisia useimmiten muokataan seuraavasti:

  1. Tavallisimmat sanat (ja, siis, ...) poistetaan; 2. sanoja (termejä) painotetaan niiden tärkeyden mukaan, usein käyttäen ns. *käänteistä dokumenttitaajuutta*

  $$w(sana) = \log(\frac{L}{df(sana)})$$

  missä $w(sana)$ on painokerroin, $df(sana)$ on sanan *dokumenttifrekvenssi* (kuinka monessa kokoelman dokumentissa tuo sana esiintyy), ja L on kokelman dokumenttien lukumäärä

- Nyt kunkin dokumentin sanahistogrammissa arvot kerrotaan painoilla w.

- Siten esim. sana “ja” joka luultavasti esiintyy jokaisessa dokumentissa (suomenkielisessä kokoelmassa) saa histogrammiarvon 0.

- Hankalampi tapaus ovat synonyymit ja sanojen taivutusmuodot. Näitä pohditaan kurssissa T-61.5020 Statistical Natural Language Processing (Luonnollisen kielen tilastollinen mallinnus)

- Samaa vektorointimuotoa voi käyttää mille tahansa *merkkijonoille*; esim. geneettisessä koodissa on 4 kirjainta A,C,G,T jotka ovat lyhenteitä tietyille nukleotideille. “Sana” voi olla esim. 3 merkin pituinen osajono: AAA, AAC, ... TTT joita on vain $4^3 = 64$ erilaista ja siten “termi-dokumenttimatriisin” sarakkeet 64-dimensioisia.

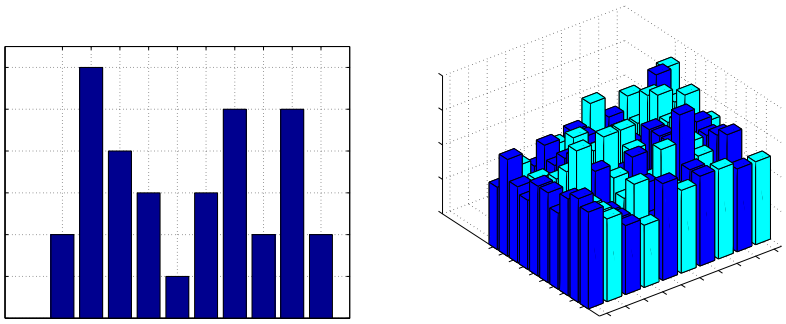
- Samantapainen idea yleistyy *puille* ja niitäkin yleisemmille tietorakenteille.

3.3 Dimensionaalisuuden kirous

Dimensionaalisuuden kirous Otoksen koko suhteessa avaruuden dimensioon

- Ajatellaan histogrammia moniulotteisessa avaruudessa. Katsotaan oheista (kuva **??**(a)

  1-ulotteista histogrammia, jossa 10 lokeroa. Jos halutaan tasossa (2-ulotteinen avaruus) 10 lokeroa / dimensio, tulee jo $10 \times 10 = 100$ lokeroa (pientä suorakaidetta). Riittääkö data täyttämään ne?



Kuva 3.11: Datahistogrammi (a) 1D; (b) 2D

- d-ulotteisessa avaruudessa 10^d; esim. jos $d = 79$, niin 10^{79}, joka on maailmankaikkeuden atomien arvioitu lukumäärä. Näin jo 79-ulotteisessa vektoriavaruudessa, jossa on 10 lokeroa kullakin koordinaattiakselilla, tulisi vain 1 atomi kuhunkin lokeroon
- *Mikä tahansa vektorijoukko korkeaulotteisessa avaruudessa on tavattoman harva*
- Tämä on *dimensionaalisuuden kirous*: tarvitaan valtavasti dataa minkäänlaisen mallin muodostamiseksi korkeaulotteiselle datalle

Dimensionaalisuuden kirous Korkeaulotteisten vektoriavaruuksien omituisuuksia

- Ajatellaan neliötä, kuutiota, hyperkuutiota jonka sivun pituus on 1 ja dimensio d. Ajatellaan datajoukkoa (datamatriisin $\mathbf{X}$ sarakkeet) joka on jakautunut tasaisesti hyperkuutioon. (Kussakin kahdessa osajoukossa joilla on sama tilavuus on keskimäärin sama määrä vektoreita). Kun d on suuri, törmätään joihinkin yllättäviin ilmiöihin.
- 1. Melkein koko hyperkuutio tarvitaan kattamaan pienikin osuus vektoreista.

  Mikä on sellaisen pienemmän hyperkuution sivun pituus, joka sisältää osuuden p kaikista vektoreista? (Esim. kymmenesosan, $p = 0.1$). Merkitään sivun pituutta $s(p)$

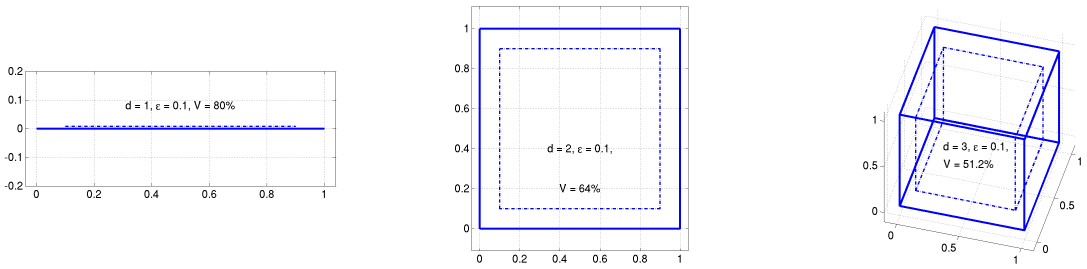
  Neliö: $s(0.1) = \sqrt{0}.1 \approx 0.32,\ s(p) = \sqrt{p}$

  Kuutio: $s(0.1) = (0.1)^{1/3} \approx 0.46\ s(p) = p^{1/3}$

10-ulotteinen: $s(0.1) = (0.1)^{1/10} \approx 0.80$

d-ulotteinen hyperkuutio: $s(p) = p^{1/d}$

Mille tahansa osuudelle p pätee $s(p) \to 1$ kun $d \to \infty$. Lähes koko hyperkuutio tarvitaan kattamaan pienikin osuus datasta.



Kuva 3.12: Sisäpisteiden suhteellinen määrä pienenee dimension kasvaessa.

- 2. Pisteet ovat kaukana toisistaan: melkein jokainen vektori (datapiste) on lähempänä kuution reunaa kuin toista pistettä.

  Hyperkuution keskipisteen etäisyys reunasta on 0.5. Se on maksimietäisyys; yleensä pisteiden etäisyys reunasta on pienempi.

  Olkoon datapisteitä n kpl ja avaruuden dimensio d. Voidaan osoittaa että pisteiden keskimääräinen etäisyys on

  $$D(d, n) = \frac{1}{2}(\frac{1}{n})^{1/d}.$$

  Nähdään että olipa n mikä tahansa, $D(d, n) \to 0.5$ kun $d \to \infty$.

- 3. Jos tehdään tasavälinen histogrammi, niin melkein kaikki lokerot ovat hyperkuution pinnalla.

  Olkoon lokeroiden lukumäärä dimensiota kohti b, kaikkiaan silloin lokeroita b^d. Niistä osuus $(\frac{b-2}{b})^d$ on kuution sisäosassa. Mutta tämä menee nollaan kun $d \to \infty$.

- Kaikki yhdessä osoittavat että korkeaulotteinen hyperkuutio ei vastaa käsitystämme kuutiosta, vaan on pikemminkin “piikikäs”.

- Opetus: data-analyysissä olisi tärkeää saada dimensio jotenkin pudotettua mahdollisimman pieneksi, että vältetään yllä olevat ilmiöt.

Esimerkki piirreirrotuksesta: PicSOM

`http://www.cis.hut.fi/picsom/`

Yhteenveto

Yhteenveto

- Tässä luvussa esitettiin datamatriisi $\mathbf{X}$ yhtenä tapana esittää havaintoja.
- Datamatriisista voidaan irrottaa piirteitä, jotka kuvaavat tehokkaammin dataa. Piirteitä voidaan käyttää paremmin esimerkiksi luokittelussa tai ryhmittelyssä.
- Laskenta tulee monella tapaa ongelmalliseksi, jos mitattavien suureiden määrä eli datamatriisin dimensio kasvaa suureksi. Seuraavassa luvussa esitellään yksi tapa pudottaa tehokkaasti datan dimensiota.

4 Vektoridatan tiivistäminen ja dekorrelointi

Johdanto

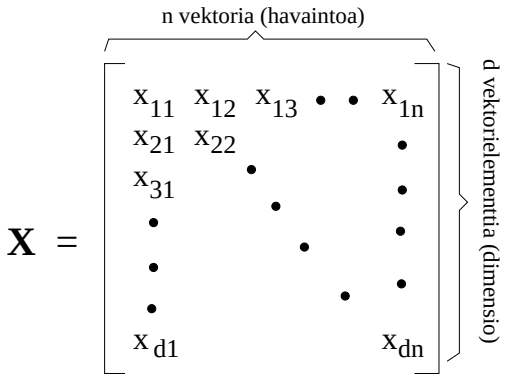
Aiheeseen liittyviä laskaritehtäviä:

- H2/1: pääkomponenttianalyysi (PCA)
- T2: ominaiskasvot

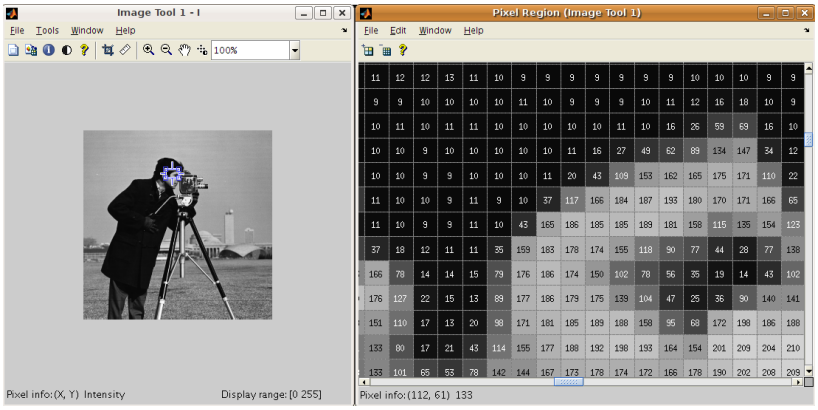
4.1 Datamatriisi

Datamatriisi

- Palautetaan mieleen *datamatriisi* $\mathbf{X}$:



- Usein matriisin sarakkeilla (vektoreilla $\mathbf{x}(t),\ t = 1, \ldots, n$) ei ole mitään määrättyä järjestystä vaan ne ovat vain otos jostakin perusjoukosta (esim. tietyn kokoisten digitaalikuvien joukko).
- Joskus vektoreilla $\mathbf{x}$ on kuitenkin selkeä ajallinen järjestys; ks. DSS-menetelmä luvun lopussa.
- Usein vaikeus on vektoreiden suuri dimensio (esim. tekstidokumenttien sanahistogrammit: dimensio voi olla 50.000; digitaalinen kuva rivi riviltä skannattuna: dimensio on rivit $\times$ sarakkeet)
- Kuitenkin vektorialkioiden välillä on voimakkaita riippuvuuksia (esim. kuvan vierekkäiset pisteet)
- Nämä riippuvuudet poistamalla päästään alentamaan dimensiota voimakkaasti.
- Menetelmiä: pääkomponenttianalyysi, joka poistaa korrelaatiot (2. kertaluvun tilastolliset riippuvuudet); riippumattomien komponenttien analyysi, joka pyrkii poistamaan kaikki riippuvuudet.



Kuva 4.1: Matlabin “kameramies” ja yksityiskohta. Huomaa luonnollisen kuvan rakenne: vierekkäisillä pikseleillä on yleensä lähes sama arvo (tasainen pinta) eli ne korreloivat.

4.2 Pääkomponenttianalyysi (PCA)

Pääkomponenttianalyysi (PCA)

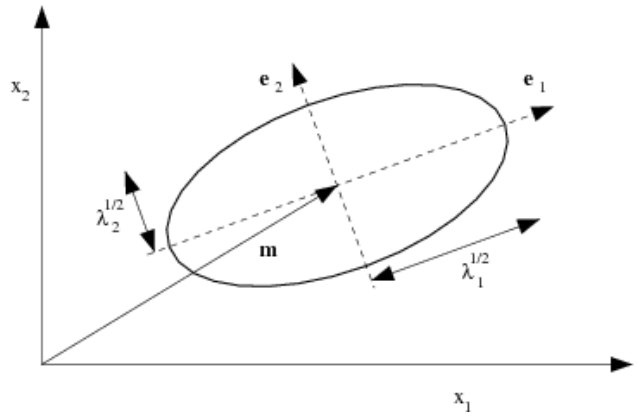
- Pääkomponenttianalyysi (Principal Component Analysis, PCA; eli Hotelling-muunnos eli Karhunen-Loève -muunnos) on klassinen tekniikka datan ja signaalien analyysisssä
- PCA *poistaa korrelaatiot* vektorialkioiden välillä ja samalla löytää kierron vektoriavaruudessa siten että uusilla (kierretyillä) koordinaateilla on maksimaalisen suuret varianssit (energiat).
- Tietyillä ehdoilla PCA löytää tilastollisesti riippumattomat komponentit (jos data normaalijakautunutta).
- Katsotaan seuraavassa matemaattisesti miten tämä tehdään.
- Lähtökohtana ovat datamatriisin sarakkeet: vektorit $\mathbf{x}$ joilla on d alkiota. Datamatriisi sisältää otoksen $\mathbf{x}(1), \ldots, \mathbf{x}(n)$ näitä vektoreita.
- Tyypillisesti $\mathbf{x}$:n alkiot voivat siis olla piirrearvoja, kuvapikseleiden harmaatasoja, histogrammiarvoja tms. kuten edellisissä luvuissa on esitetty
- Pääkomponenttimuunnoksessa vektorit $\mathbf{x}$ nollakeskiarvoistetaan ensin vähentämällä keskiarvo:

$$\mathbf{x} \leftarrow \mathbf{x} - \mathrm{E}\{\mathbf{x}\}$$

- Tässä merkintä $\mathrm{E}\{\mathbf{x}\}$ tarkoittaa keskiarvoa: käytännössä se lasketaan seuraavasti:

$$\mathrm{E}\{\mathbf{x}\} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}(i)$$

- Oletetaan seuraavassa että tämä on tehty ja vektorit $\mathbf{x}$ ovat siis nollakeskiarvoisia.
- Sitten $\mathbf{x}$ muunnetaan lineaarimuunnoksella toiseksi vektoriksi $\mathbf{y}$ jossa on m alkiota, $m \leq d$, niin että korrelaatioista johtuva redundanssi poistuu.
- Samalla kertaa $\mathbf{x}$:n uusille koordinaattiakseleille otettujen projektioiden varianssit maksimoituvat



Kuva 4.2: Vektorijakauman (kuvattu ellipsillä) pääakselit kahdessa dimensiossa: $\mathbf{e}_1$ ja $\mathbf{e}_2$. Keskiarvo on $\mathbf{m}$. Ominaisarvot λ_1 ja λ_2 antavat varianssit pääakselien suunnassa

- Ensimmäinen akseli $\mathbf{e}_1$ kuvassa 4.2 (pääkomponentin suunta) vastaa suurinta varianssia, toinen on suurin varianssi ensimmäistä vasten kohtisuorassa suunnassa jne.
- Viimeisillä pääkomponenteilla on jo niin pieni varianssi, että ne voidaan kokonaan jättää pois. Tähän perustuu vektoreiden tiivistäminen (dimension alentaminen).

  Esimerkki 3D-, 2D- ja 1D-datasta

Pääkomponenttianalyysi (PCA) Matemaattinen johtaminen maksimivarianssikriteerillä

- Matemaattisesti: ajatellaan lineaarikombinaatiota

$$y_1 = \sum_{k=1}^{d} w_{k1} x_k = \mathbf{w}_1^T \mathbf{x}$$

  vektorin $\mathbf{x}$ alkioista $x_1, \ldots, x_n$ (projektiota suunnalle $\mathbf{w}_1$)

- Summa y_1 on nimeltään $\mathbf{x}$:n *ensimmäinen pääkomponentti*, jos y_1:n varianssi $\mathrm{E}\{y_1^2\}$ on mahdollisimman suuri. Taas varianssi voidaan käytännössä laskea summana yli otoksen $\mathbf{x}(i)$. Vektori $\mathbf{w}_1$ on vastaava *pääkomponenttivektori*.

- Jotta $\mathrm{E}\{y_1^2\}$ ei kasvaisi äärettömän suureksi, täytyy vektorin $\mathbf{w}_1$ pituutta jotenkin rajoittaa. Kätevin rajoite on että sen normi on vakio, käytännössä 1

- Siten PCA-kriteeri on seuraava: maksimoi

$$J_1^{PCA}(\mathbf{w}_1) = \mathrm{E}\{y_1^2\} = \mathrm{E}\{(\mathbf{w}_1^T\mathbf{x})^2\} = \mathbf{w}_1^T\mathrm{E}\{\mathbf{x}\mathbf{x}^T\}\mathbf{w}_1 = \mathbf{w}_1^T\mathbf{C}_\mathbf{x}\mathbf{w}_1$$
$$\text{missä} \|\mathbf{w}_1\| = 1$$

- Matriisi $\mathbf{C}_\mathbf{x} = \mathrm{E}\{\mathbf{x}\mathbf{x}^T\}$ on $d \times d$ ja nimeltään $\mathbf{x}$:n *kovarianssimatriisi*

- Se lasketaan käytännössä nolla-keskiarvoistetusta datamatriisista $\mathbf{X}$ kaavalla $\mathbf{C}_x = \frac{1}{n}\mathbf{X}\mathbf{X}^T$

- Ratkaisu on

$$\mathbf{w}_1 = \mathbf{e}_1$$

  missä $\mathbf{e}_1$ on $\mathbf{C}_\mathbf{x}$:n ominaisvektori vastaten suurinta ominaisarvoa λ_1.

- Miksi?

- Katsotaan tätä harjoitustehtävänä, mutta eräs peruste on seuraava: $\mathbf{w}_1^T\mathbf{C}_\mathbf{x}\mathbf{w}_1$ on vektoreiden $\mathbf{w}_1$ ja $\mathbf{C}_\mathbf{x}\mathbf{w}_1$ sisätulo, joka on samalla niiden pituuksien (euklidisten normien) tulo kertaa niiden välisen kulman kosini. (Muista sisätulon määritelmä!)

- Sovittiin että $\|\mathbf{w}_1\| = 1$ joten jää kulman kosini kertaa $\|\mathbf{C}_\mathbf{x}\mathbf{w}_1\|$. Kosini maksimoituu kun kulma on nolla, mutta silloin pätee $\mathbf{C}_\mathbf{x}\mathbf{w}_1 = \lambda\mathbf{w}_1$ missä λ on jokin skalaarivakio.

- Tämä on matriisin $\mathbf{C}_\mathbf{x}$ ominaisarvoyhtälö jonka ratkaisuina ovat sen d ominaisvektoria. Mikä niistä pitää valita?

- Silloin myös pätee $\|\mathbf{C}_\mathbf{x}\mathbf{w}_1\| = \lambda$ joten λ:n pitäisi olla mahdollisimman suuri. Valitaan siis suurinta ominaisarvoa vastaava ominaisvektori pääkomponenttivektoriksi $\mathbf{w}_1$.

- Maksimivarianssikriteeri

$$max\ \mathrm{E}\{y_k^2\} = \mathrm{E}\{(\mathbf{w}_k^T\mathbf{x})^2\}$$

  voidaan yleistää m:lle pääkomponentille kun lisätään rajoitusehdot: joko

$$\mathrm{E}\{y_m y_k\} = 0,\ k < m \tag{4.1}$$

  tai

$$\mathbf{w}_i^T\mathbf{w}_j = \delta_{ij}$$

- Ratkaisuna on että k:s pääkomponentti on $y_k = \mathbf{e}_k^T\mathbf{x},\ k = 1, \ldots, n$

Pääkomponenttianalyysi (PCA) Johtaminen pienimmän neliösumman virheeseen (MSE) perustuen

- Toinen tapa määritellä PCA on vektoreiden $\mathbf{x}$ *pienimmän neliösumman virhe* (MSE) kun ne esitetään *PCA-kehitelmässä* jossa on mukana m termiä
- Merkitään joukkoa ortogonaalisia vektoreita $\mathbf{w}_1, \ldots, \mathbf{w}_m$
- MSE-kriteeri:

$$J_{MSE}^{PCA} = \mathrm{E}\{\|\mathbf{x} - \sum_{i=1}^{m} (\mathbf{w}_i^T \mathbf{x}) \mathbf{w}_i\|^2\} \tag{4.2}$$

- On helppo näyttää että voimme kirjoittaa

$$\begin{aligned} J_{MSE}^{PCA} &= \mathrm{E}\{\|\mathbf{x}\|^2\} - \mathrm{E}\{\sum_{i=1}^{m} (\mathbf{w}_i^T \mathbf{x})^2\} & (4.3) \\ &= \mathrm{trace}(\mathbf{C}_\mathbf{x}) - \sum_{i=1}^{m} \mathbf{w}_i^T \mathbf{C}_\mathbf{x} \mathbf{w}_i & (4.4) \end{aligned}$$

- Voidaan osoittaa että kriteerin (4.4) minimi ortogonaalisuusehdon vallitessa on m:n ensimmäisen ominaisvektorin $\mathbf{e}_1, \ldots, \mathbf{e}_m$ muodostama kanta
- PCA-kehitelmä tietylle vektorille $\mathbf{x}$ on silloin

$$\mathbf{x} = \sum_{i=1}^{m} (\mathbf{e}_i^T \mathbf{x}) \mathbf{e}_i \tag{4.5}$$

- Pienimmän MSE-virheen arvo on silloin

$$J_{MSE}^{PCA} = \sum_{i=m+1}^{n} \lambda_i \tag{4.6}$$

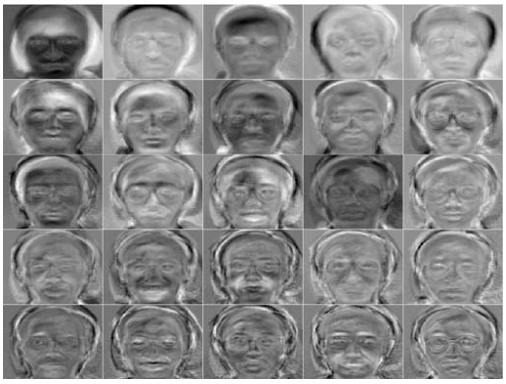
- Tämä on niiden pienimpien ominaisarvojen summa, jotka vastaavat poisjätettyjä ominaisvektoreita $\mathbf{e}_{m+1}, \ldots, \mathbf{e}_n$
- Tästä voi myös laskea kuinka monta pääkomponenttia täytyy ottaa mukaan kehitelmään (4.5) jotta päästään haluttuun kompressioon.

4.3 PCA-esimerkkejä: ominaiskasvot

Esimerkki: ominaiskasvot

- Eräs suosittu pääkomponenttianalyysin (PCA) sovellus on ns. *ominaiskasvot* (eigenfaces)

- Siinä kerätään joukko kasvokuvia, yleensä normeerattuja niin että kasvonpiirteet ovat suunnilleen samassa paikassa kuva-alueessa
- Kuvat skannataan vektoreiksi vaakariveittäin
- Nämä vektorit ovat nyt datamatriisin $\mathbf{X}$ sarakkeita.
- Lasketaan ominaisvektorit, jotka voidaan visualisoida ominaiskasvoina



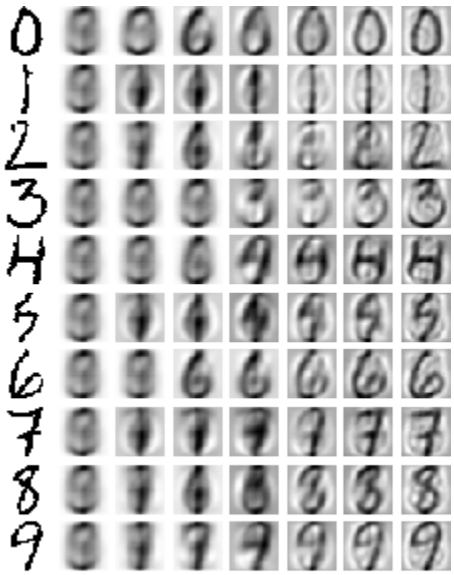
Kuva 4.3: Ominaiskasvoja

- Näitä voi käyttää kuvien tunnistamiseen projisoimalla kuvat mataladimensoiseen (2- tai 3-dim.) pääkomponenttiavaruuteen; saman henkilön eri kuvat osuvat siellä lähekkäin. Esimerkki projektiosta T2-harjoituksissa
- Tiivistettyä esitystä voi hyödyntää myös datan pakkaamisessa

  Tietokoneharjoitusten T2 esimerkki ominaiskasvoista

Esimerkki: käsinkirjoitetut merkit

- Seuraava esimerkki näyttää, kuinka datan rekonstruktio paranee kun yhä enemmän pääkomponentteja otetaan mukaan kehitelmään (4.5)
- Vasemmanpuoleisessa sarakkeessa on käsinkirjoitettuja numeroita digitoituna 32×32 kuvamatriisiin
- Seuraavassa sarakkeessa on niiden keskiarvovektori, joka siis vähennetään ennen PCA-analyysiä
- Vektorien $\mathbf{x}$ dimensio on siis 1024
- Kuva näyttää rekonstruktion PCA:lla kun 1, 2, 5, 16, 32, ja 64 pääkomponettia on mukana kehitelmässä.



Kuva 4.4: Käsinkirjoitettuja merkkejä. Vasemmalta alkuperäinen, keskiarvo, rekonstruktiot käyttäen 1, 2, 5, 16, 32 ja 64 pääkomponenttia.

Esimerkki: ilmastodata

- Otetaan toinen esimerkki: ilmastodatan ns. *kokeelliset ortogonaalifunktiot (Experimental Orthogonal Functions, EOF)*
- Ilmastodata (säädata) tarkoittaa säännöllisin välein (tunneittain, päivittäin jne.) mitattuja lämpötiloja, ilmanpaineita, sademääriä jne. usealla paikkakunnalla
- Amerikkalainen järjestö NCAR (National Center for Atmospheric Research, Kansallinen ilmakehätutkimuksen keskus) julkaisee Webissä tällaisia mittaustietoja pitkältä ajanjaksolta ja koko maapallon peittävältä mittaushilalta (jossa puuttuvat arvot esim. keskellä valtameriä on käytännössä laskettu tietyillä kaavoilla).
- Me käytimme heidän dataansa jossa ajanjakso on 1948–2004, mittaukset päivittäisiä, ja hila 2.5 asteen välein maapallon pinnalla
- Näistä voi tehdä datamatriisin $\mathbf{X}$ jossa vaakadimensio on aika (56 x 365 päivää = 20.440 päivää) ja pystydimensio on paikka (73 x 144 = 10.512 mittauspistettä). Matriisin koko siis 10.512 x 20.440.
- Säätieteilijät haluaisivat hajottaa (keskiarvoistetun) datamatriisin summaksi

$$\mathbf{X} = E\{\mathbf{X}\} + \sum_{i=1}^{m} \mathbf{w}_i \mathbf{y}_i^T$$

jossa sarakevektoreilla $\mathbf{w}_i$ on paikkadimensio ja rivivektoreilla $\mathbf{y}_i$ on aikadimensio

- Tämän voi kätevästi tehdä PCA:lla: huomaa että PCA-kehitelmä kaavasta (4.5) antaa koko datamatriisille
$$\mathbf{X} - E\{\mathbf{X}\} = \sum_{i=1}^{m} \mathbf{e}_i \mathbf{e}_i^T (\mathbf{X} - E\{\mathbf{X}\})$$
ja tässä sarakevektoreilla $\mathbf{e}_i$ on paikkadimensio, rivivektoreilla $\mathbf{e}_i^T(\mathbf{X} - E\{\mathbf{X}\})$ on aikadimensio.

- Ominaisvektoreita $\mathbf{e}_i$, kun ne kuvataan graafisesti karttapinnalla, sanotaan kokeellisiksi ortogonaalifunktioiksi (EOF).

- Oheinen kuva näyttää 3 ensimmäistä funktiota ja vastaavat aikakäyrät ilmanpainedatalle. (Ne on laskettu vain ajalle 1957–2000).

4.4 DSS-menetelmä

Esimerkki: ilmastodata DSS-menetelmä: halutunlaisten aikakomponenttien etsiminen

- (DSS = denoising source separation)

- Selostetaan seuraavassa lyhyesti kuinka yo. menetelmää voi parantaa jos tiedetään minkälaisia aikakomponentteja halutaan

- Useat sääilmiöt ovat jaksollisia, esim. vuoden, 2 vuoden jne. jaksoja

- Tämä voidaan ottaa huomioon yhdistämällä PCA ja aikasuodatus

- Tehdään ensin säämittausmatriisille $\mathbf{X}$ pääkomponenttianalyysi kuten edellä selostetaan

- Skaalataan sitten aikakäyrät $\mathbf{e}_i^T(\mathbf{X} - E\{\mathbf{X}\})$ niin, että niiden varianssi (keskimääräinen energia) on yksi; käytännössä skaalaustekijä on $1/\sqrt{n\lambda_i}$ missä λ_i on ominaisvektoria $\mathbf{e}_i$ vastaava ominaisarvo

- Nimittäin
$$E\{\mathbf{e}_i^T(\mathbf{X} - E\{\mathbf{X}\})(\mathbf{X} - E\{\mathbf{X}\})^T\mathbf{e}_i\} = n\mathbf{e}_i^T\mathbf{C}_x\mathbf{e}_i = n\lambda_i$$

- Suodatetaan nyt skaalatut aikakäyrät taajuussuotimella jolla on haluttu taajuusominaisuus ja tehdään vielä toisen kerran pääkomponenttianalyysi

- Ensimmäinen pääkomponentti vastaa käyrää jolla suodatettuna on suurin varianssi (energia), ja jossa siis on eniten haluttuja taajuuksia.

- Näin löytyy hyvin El Niño -ilmiö; ks. kuvat.

- Ilmastodatasta saadaan näin paljastettua uutta tietoa

- Tämä saattaa olla uusi aluevaltaus ilmastomittausten analyysissä

- Ensimmäinen julkaisu (Ilin, Valpola, Oja) kesällä 2005
  (Ilin, Valpola, Oja 2005)

Yhteenveto

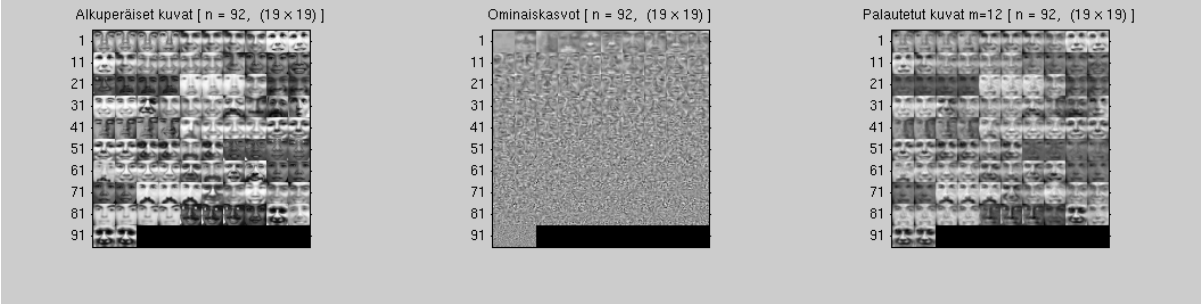
Yhteenveto

Tässä luvussa

- esiteltiin pääkomponenttianalyysimenetelmä (PCA) datan dekorrelointiin ja koordinaatiston kiertoon maksimivarianssin suuntaan
- johdettiin ensimmäisen pääkomponenttivektorin suunta
- käytettiin PCA:ta lukuisissa tapauksissa mm. ominaiskasvojen (eigenfaces) tapauksessa

Liite: Kuvia esitykseen, luku 4

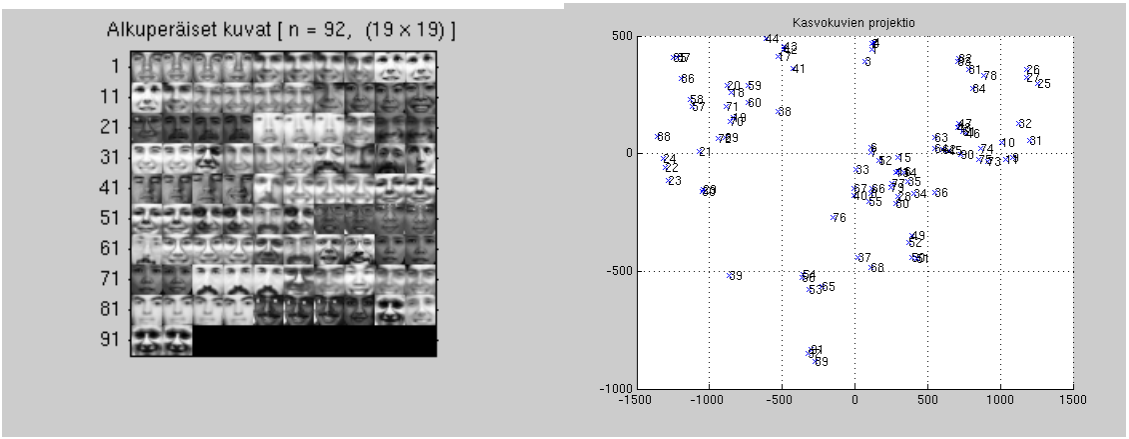
Ominaiskasvoja



Kuva 4.5: T2-harjoitusten datasetti: 92 kasvokuvaa resoluutiolla $19 \times 19 = 361$. Vasemmalla alkuperäiset kuvat. Keskellä ominaisvektorit eli ominaiskasvot. Oikealla palautetut kasvokuvat 12 ominaiskasvon perusteella. Alkuperäisessä datassa siis 92×361 lukuarvoa, kun taas pakkausta ja purkua varten tarvitaan 12 ominaiskasvoa ja 12D-projektiopisteet $12 \times (361 + 92)$.

Takaisin kalvoihin

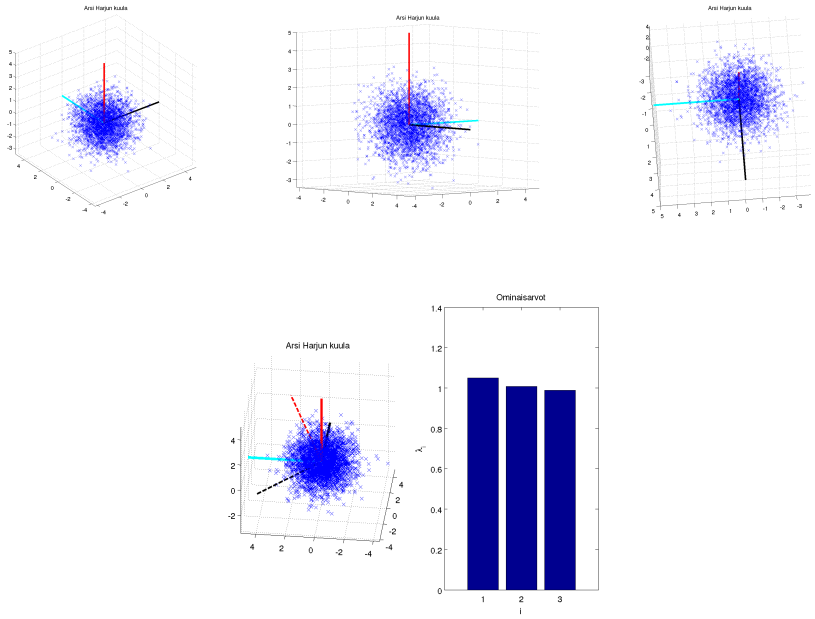
Kasvokuvien projektio 2D-tasolle



Kuva 4.6: T2-harjoituksen esimerkki: (a) 92 alkuperäistä kasvokuvaa ja (b) niiden 2D-projektiot. Samasta henkilöstä otettujen valokuvien 2D-projektiopisteet ovat lähellä toisiaan.

Takaisin kalvoihin

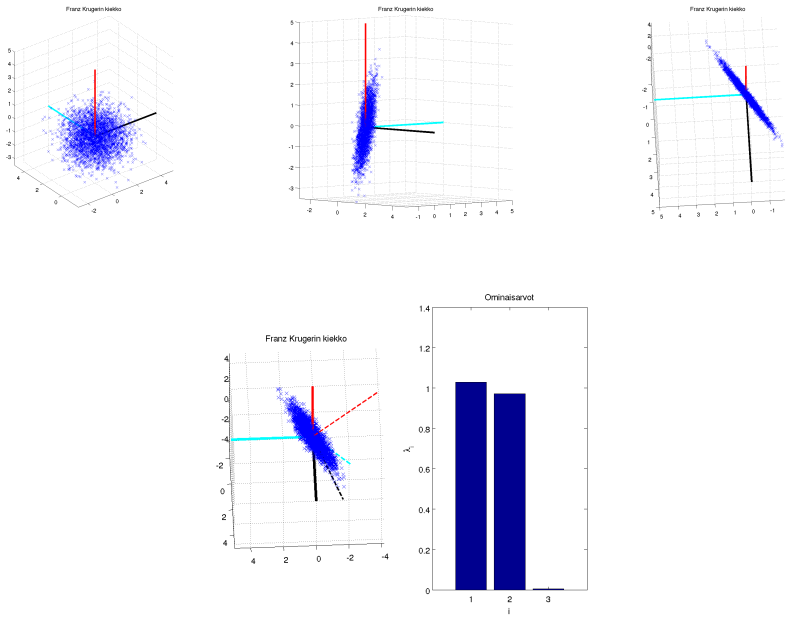
PCA - koordinaattiakselien kierto Arsi Harjun 3D-kuula ja geometrinen tulkinta



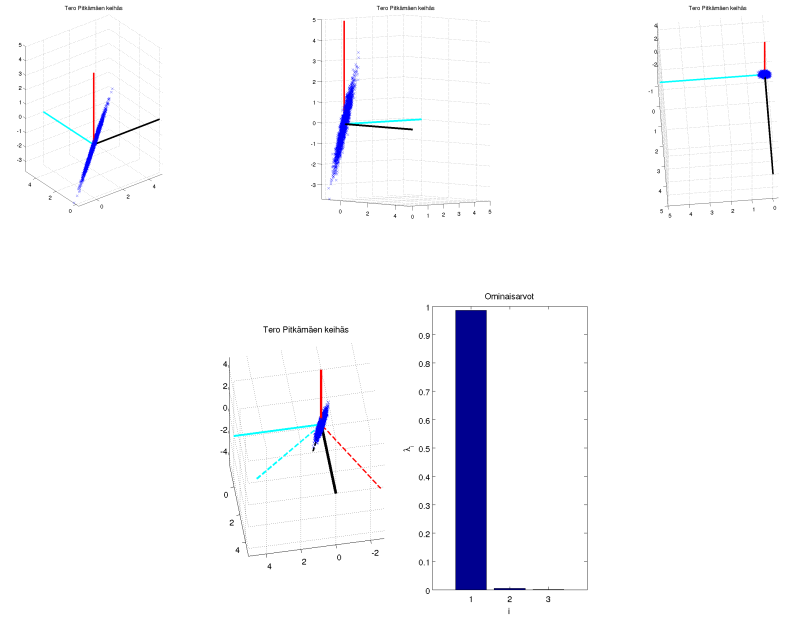
Kuva 4.7: Ylärivi: Arsi Harjun gaussinen kuula eri kulmista katsottuna. Alhaalla kuula PCA:n jälkeen. Ominaisarvot λ_1, λ_2, λ_3 melko samansuuruiset: yhtä satunnaista kaikissa suunnissa. 3D 2D 1D

PCA - koordinaattiakselien kierto Frantz Krugerin 2D-kiekko ja geometrinen tulkinta

PCA - koordinaattiakselien kierto Tero Pitkämäen 1D-keihäs ja geometrinen tulkinta



Kuva 4.8: Ylärivi: Frantz Krugerin gaussinen kiekko eri kulmista katsottuna. Alhaalla kiekko PCA:n jälkeen. PCA3:n suuntaan ominaisarvo λ_3 lähellä nollaa. 3D 2D 1D



Kuva 4.9: Ylärivi: Tero Pitkämäen gaussinen keihäs. Alhaalla keihäs PCA:n jälkeen PCA1:n suuntaisesti, jolloin vain ominaisarvo λ_1 on merkittävä. Dimension pudotus 3D $\rightarrow$ 1D. 3D 2D 1D Takaisin

5 Estimointiteorian perusteita

Johdanto

Aiheeseen liittyviä laskaritehtäviä:

- H2/3, H2/4: ML- ja Bayes-estimointi

5.1 Perusjakaumat 1-ulotteisina

Perusjakaumat 1-ulotteisina

- Kun datasta halutaan muodostaa malleja, ne ovat yleensä tilastollisia (esim. regressio, luokittelu, ryhmittely, ...) esimerkki regressiosta esimerkki luokittelusta esimerkki ryhmittelystä
- Siksi tarvitaan todennäköisyyslaskentaa
- Peruskäsite siellä on todennäköisyysjakauma (*kertymäfunktio*) skalaariarvoiselle satunnaismuuttujalle x:
$$F(x_0) = P(x \leq x_0) \tag{5.1}$$
antaa todennnäköisyyden sille että $x \leq x_0$.
- Funktio $F(x_0)$ on ei-vähenevä, jatkuva ja $0 \leq F(x_0) \leq 1$.
- *Tiheysfunktio* x:lle pisteessä $x = x_0$
$$p(x_0) = \left.\frac{dF(x)}{dx}\right|_{x=x_0} \tag{5.2}$$
on funktion $F(x)$ derivaatta pisteessä $x = x_0$
- Vastaavasti tietenkin
$$F(x_0) = \int_{-\infty}^{x_0} p(x)dx$$
- Tiheysfunktiolla on seuraavat ominaisuudet:
$$\begin{aligned} \int_{-\infty}^{\infty} p(x)dx &= 1 \\ \int_{-\infty}^{\infty} xp(x)dx &= \mu = \mathrm{E}\{x\} \\ \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx &= \sigma^2 = \mathrm{Var}\{x\}. \end{aligned}$$
- Tässä μ ja σ ovat x:n odotusarvo ja keskihajonta.

Perusjakaumat 1-ulotteisina Esimerkkejä

- Oletan tunnetuiksi jotkut yleisimmät tiheysfunktiot, kuten *normaalijakauma* (eli Gaussin jakauma)

$$p(x) = \frac{1}{\sqrt{2\pi}\,\sigma}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

eksponentiaalinen jakauma

$$p(x) = \lambda e^{-\lambda x}$$

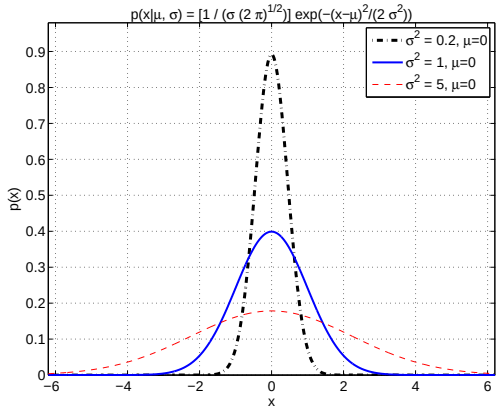
ja *tasainen jakauma* (välillä [a,b])

$$p(x) = 1/(b-a),\ \text{kun } x \in [a,b],\ 0 \text{ muuten}$$

Perusjakaumat 1-ulotteisina Normaalijakauman tiheysfunktio p(x)

Parametrit μ (keskiarvo) ja σ (keskihajonta, σ^2 varianssi)

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

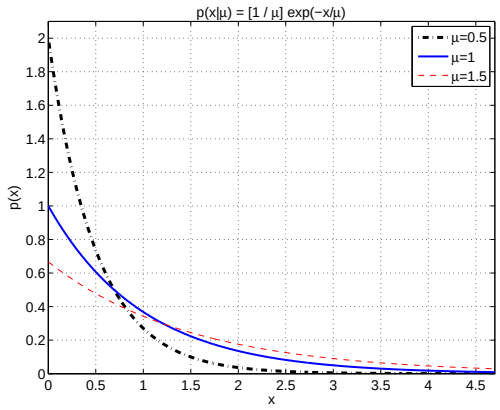


Kuva 5.1: Normaalijakauma kolmella eri σ:n arvolla

Perusjakaumat 1-ulotteisina Eksponentiaalijakauman tiheysfunktio p(x)

Parametri λ ($1/\mu$, jossa μ keskiarvo)

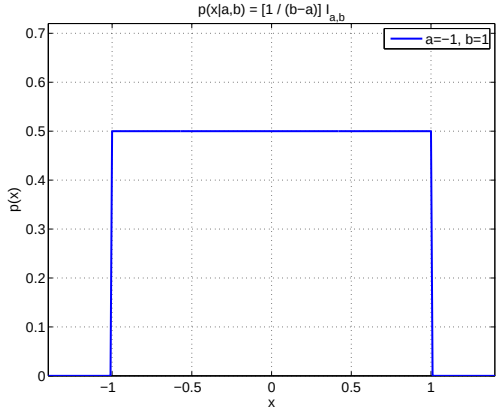
$$p(x) = \lambda e^{-\lambda x}$$



Kuva 5.2: Eksponenttijakauma kolmella eri λ:n arvolla

Perusjakaumat 1-ulotteisina Tasainen jakauman tiheysfunktio p(x)
Tasainen välillä $[a, b]$

$$p(x) = \frac{1}{b-a}, \quad \text{kun } x \in [a, b],\ 0 \text{ muuten}$$



Kuva 5.3: Tasajakauma

5.2 Yleistys vektoridatalle, d:n muuttujan normaalijakauma

Yleistys vektoridatalle

- Kun puhutaan vektoridatasta, on pakko yleistää yo. jakaumat moniulotteisiksi: olkoon taas datavektori $\mathbf{x} = (x_1, x_2, \ldots, x_d)^T$

- Sen todennäköisyysjakauma (kertymäfunktio) on

$$F(\mathbf{x}_0) = P(\mathbf{x} \leq \mathbf{x}_0) \tag{5.3}$$

missä relaatio "$\leq$" ymmärretään alkioittain

- Vastaava moniulotteinen tiheysfunktio $p(\mathbf{x})$ on tämän osittaisderivaatta kaikkien vektorialkioiden suhteen:

$$p(\mathbf{x}_0) = \frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}\cdots\frac{\partial}{\partial x_d}F(\mathbf{x})\bigg|_{\mathbf{x}=\mathbf{x}_0} \tag{5.4}$$

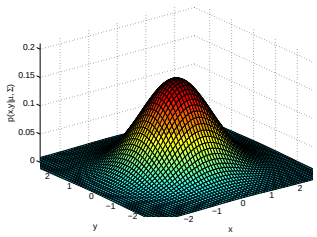
- Käytännössä tiheysfunktio on tärkeämpi.

Yleistys vektoridatalle Kahden muuttujan normaalijakauma, symmetrinen, $d = 2$

- "Symmetrinen" 2-ulotteinen ($d = 2$) normaalijakauma on

$$p(\mathbf{x}) = \frac{1}{2\pi\sigma^2}e^{-\frac{(x_1-\mu_1)^2+(x_2-\mu_2)^2}{2\sigma^2}}$$

missä jakauman odotusarvo (keskipiste) on piste $\mathbf{m} = [\mu_1,\ \mu_2]$ ja keskihajonta joka suuntaan on sama σ.



Kuva 5.4: Symmetrinen 2D-normaalijakauma

Yleistys vektoridatalle Kahden muuttujan normaalijakauma, $d = 2$

- Yllä oleva *2-ulotteinen normaalijakauma* voidaan yleisemmin kirjoittaa muotoon

$$p(\mathbf{x}) = K \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})\right)$$

jossa skaalaustermi K ($d = 2$)

$$K = \frac{1}{(2\pi)^{d/2}\det(\mathbf{C})^{1/2}} = \frac{1}{2\pi\det(\mathbf{C})^{1/2}}$$

ja $\mathbf{C}$ (toisinaan Σ) on (2×2)-kokoinen kovarianssimatriisi ja $\mathbf{m} = [\mu_1\ \ \mu_2]^T$ keskiarvovektori (huippukohta)

- Keskihajonta / varianssi on 2-ulotteisessa tapauksessa monimutkaisempi kuin 1-ulotteisessa tapauksessa: varianssin $\sigma^2 = \mathrm{E}\{(x-\mu)^2\}$ korvaa kovarianssimatriisi

$$\mathbf{C} = \begin{bmatrix} \mathrm{E}\{(x_1-\mu_1)^2\} & \mathrm{E}\{(x_1-\mu_1)(x_2-\mu_2)\} \\ \mathrm{E}\{(x_1-\mu_1)(x_2-\mu_2)\} & \mathrm{E}\{(x_2-\mu_2)^2\} \end{bmatrix}$$

- Esimerkkejä 2-ulotteisen normaalijakauman tiheysfunktioista

  $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen diagonaalinen $\mathbf{C}$, jossa lisäksi $\sigma_1 = \sigma_2$

  $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen diagonaalinen $\mathbf{C}$, jossa $\sigma_1 \neq \sigma_2$

  $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen ei-diagonaalinen $\mathbf{C}$

Yleistys vektoridatalle d:n muuttujan normaalijakauma

- Yleistys d:hen dimensioon suoraviivainen: matriisialkiot ovat $\mathbf{C}_{ij} = \mathrm{E}\{(x_i-\mu_i)(x_j-\mu_j)\} = \mathrm{Cov}(x_i, x_j)$
- Kovarianssimatriisi $\mathbf{C}$ on siis symmetrinen neliömatriisi kokoa $(d \times d)$
- Silloin *d-ulotteisen normaalijakauman* tiheysfunktio voidaan kirjoittaa vektori-matriisimuotoon:

$$p(\mathbf{x}) = K \cdot \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x}-\mathbf{m})\right) \tag{5.5}$$

- missä $\mathbf{m} = [\mu_1, \ldots, \mu_d]^T$ on keskiarvovektori (jakauman keskipiste, huippukohta) ja K on normalisoiva termi

$$K = \frac{1}{(2\pi)^{d/2}\det(\mathbf{C})^{1/2}}$$

- Termi K tarvitaan vain, jotta $p(\mathbf{x})$:n integraali d-ulotteisen avaruuden yli olisi 1
- Voidaan integroimalla johtaa että

$$\begin{aligned} \mathrm{E}\{\mathbf{x}\} &= \mathbf{m} = \int_{\mathcal{R}^\lceil} \mathbf{x}p(\mathbf{x})d\mathbf{x} \\ \mathrm{E}\{(\mathbf{x}-\mathbf{m})(\mathbf{x}-\mathbf{m})^T\} &= \mathbf{C} \end{aligned}$$

- Huomaathan, että vaikka $d > 1$, niin $p(\mathbf{x}) \in R^1$ eli tiheysfunktion arvo on skalaari, sillä matriisikertolaskut eksponenttifunktiossa $(1 \times \underline{d})(\underline{d} \times \underline{\underline{d}})(\underline{\underline{d}} \times 1) = (1 \times 1)$

Yleistys vektoridatalle Korreloimattomuus ja riippumattomuus

- Vektorin $\mathbf{x}$ alkiot ovat *korreloimattomat* jos niiden kovarianssit ovat nollia eli $\mathrm{E}\{(x_i - \mu_i)(x_j - \mu_j)\} = 0$. Toisin sanoen, kovarianssimatriisi $\mathbf{C}$ on diagonaalinen

$$\mathbf{C} = \begin{bmatrix} \sigma_1^2 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2^2 & 0 & \dots & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & & \dots & \sigma_d^2 \end{bmatrix}$$

- Vektorin $\mathbf{x}$ alkiot ovat *riippumattomat* jos niiden *yhteistiheysjakauma* voidaan lausua *marginaalijakaumien* tulona:

$$p(\mathbf{x}) = p_1(x_1)p_2(x_2)....p_d(x_d)$$

- Riippumattomuus tuottaa aina korreloimattomuuden, mutta ei välttämättä toisinpäin

- Osoitetaan että jos *normaalijakautuneen vektorin alkiot* ovat korreloimattomat, niin ne ovat myös riippumattomat: katsotaan termiä $(\mathbf{x} - \mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m})$ tapauksessa missä $\mathrm{E}\{(x_i - \mu_i)(x_j - \mu_j)\} = 0$ (korreloimattomuus).

- Mutta silloin $\mathbf{C}$ on lävistäjämatriisi ja

$$(\mathbf{x} - \mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m}) = \sum_{i=1}^{d} (\mathbf{C}_{ii})^{-1}(x_i - \mu_i)^2$$

- Kun otetaan tästä eksponenttifunktio kuten kaavassa (6.1) saadaan

$$\begin{aligned} p(\mathbf{x}) &= K \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m})^T\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m})\right) \\ &= K \cdot \exp\left(-\frac{1}{2}\sum_{i=1}^{d} (\mathbf{C}_{ii})^{-1}(x_i - \mu_i)^2\right) \\ &= K \cdot \exp(-\frac{1}{2}\mathbf{C}_{11}^{-1}(x_1 - \mu_1)^2)... \exp(-\frac{1}{2}\mathbf{C}_{dd}^{-1}(x_d - \mu_d)^2) \end{aligned}$$

  joten $p(\mathbf{x})$ hajaantuu marginaalijakaumien tuloksi, eli alkiot x_i ovat riippumattomia.

5.3 Suurimman uskottavuuden periaate

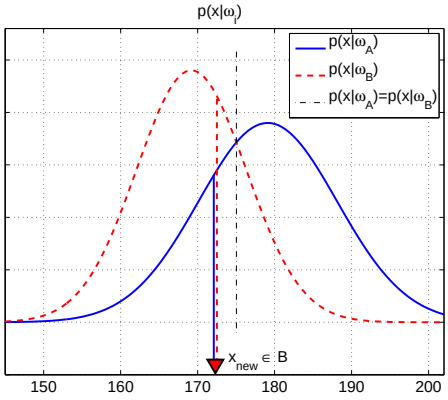
Suurimman uskottavuuden periaate Parametrisen tiheysfunktion estimointi

- Edellä esiteltiin teoriassa vektorimuuttujan jakauma, etenkin normaalijakauma.

- Jos on kuitenkin annettuna vain vektorijoukko datamatriisin $\mathbf{X}$ muodossa, mistä tiedämme sen jakauman? Data $\mathbf{X}$ ja eräs mahdollinen $p(x)$

- Jakauma olisi erittäin hyödyllinen, koska sen avulla voidaan esim. vastata seuraavaan kysymykseen: kun on annettuna uusi vektori $\mathbf{x}$, millä todennäköisyydellä se kuuluu samaan joukkoon kuin datajoukko $\mathbf{X}$ (tai jokin sen osajoukko).

- Etenkin luokittelu perustuu usein jakaumien estimointiin (esimerkki kuvassa 64)



Kuva 5.5: Esimerkki luokittelusta. Olkoon data $\mathbf{X}_A$ naisten ja $\mathbf{X}_B$ miesten pituuksia (cm), ja niistä estimoidut normaalijakaumat $p_A(x) \equiv p(x|\omega_A)$ ja $p_B(x) \equiv p(x|\omega_B)$. Jakaumia voidaan käyttää luokittelemaan uusi datapiste $p(x_{new}|\omega_B) > p(x_{new}|\omega_A)$. Havainto x_{new} luokiteltaisiin naiseksi.

- Eräs mahdollisuus on d-ulotteinen histogrammi: mutta dimensionaalisuuden kirous tekee tästä hyvin hankalan jos d vähänkin suuri

- Parempi menetelmä on yleensä *parametrinen tiheysestimointi*, joka tarkoittaa, että oletetaan tiheysfunktiolle $p(\mathbf{x})$ jokin muoto (esim. normaalijakauma) ja pyritään estimoimaan datajoukosta vain sen *parametrit* – normaalijakaumalle siis keskiarvovektori $\mathbf{m} = [\mu_1 \ \ldots \ \mu_d]^T$ ja kovarianssimatriisi $\mathbf{C} = (\mathbf{C}_{ij})$

- Tämä tekee yhteensä vain $d+d(d+1)/2$ parametria, koska $\mathbf{C}$ on symmetrinen matriisi kokoa $(d \times d)$

- Vertaa histogrammiin, jossa lokeroiden määrä kasvaa eksponentiaalisesti l^d.

- Merkitään tuntemattomien parametrien muodostamaa järjestettyä joukkoa vektorilla $\boldsymbol{\theta}$ ja tiheysjakaumaa $p(\mathbf{x}|\boldsymbol{\theta})$

- Tarkoittaa: funktion varsinainen argumentti ovat vektorialkiot $x_1, \ldots, x_d$ mutta funktio riippuu myös parametreista ($\boldsymbol{\theta}$:n alkioista) aivan niinkuin vaikkapa normaalijakauman muoto muuttuu kun kovarianssimatriisi muuttuu

- Funktion yleinen muoto oletetaan siis tunnetuksi (parametreja vaille) ja parametrivektorin $\boldsymbol{\theta}$ alkiot ovat vakioita mutta tuntemattomia.
- Miten estimaatit $\hat{\boldsymbol{\theta}}$ sitten ratkaistaan?

Suurimman uskottavuuden periaate

- *Suurimman uskottavuuden* menetelmässä (maximum likelihood) Milton: ML method valitaan se parametrivektori $\boldsymbol{\theta}$ joka maksimoi datavektorijoukon yhteisen tiheysfunktion, ns. *uskottavuusfunktion* $L(\boldsymbol{\theta})$

$$L(\boldsymbol{\theta}) = p(\mathbf{X} \mid \boldsymbol{\theta}) = p(\mathbf{x}(1), \mathbf{x}(2), \ldots, \mathbf{x}(n) \mid \boldsymbol{\theta}) \tag{5.6}$$

- Tässä $\mathbf{X}$ on siis koko datamatriisi ja $\mathbf{x}(1), ..., \mathbf{x}(n)$ ovat sen sarakkeet
- *Ajatus: valitaan sellaiset arvot parametreille, että havaittu datajoukko on todennäköisin mahdollinen* Esimerkki
- Huomaa että kun numeerinen datamatriisi $\mathbf{X}$ sijoitetaan tähän funktioon, se ei olekaan enää $\mathbf{x}$:n funktio vaan ainoastaan $\boldsymbol{\theta}$:n.
- Siten parametrit saadaan maksimoimalla (“derivaatan nollakohta”)

$$\frac{\partial}{\partial \boldsymbol{\theta}} p(\mathbf{X} \mid \boldsymbol{\theta}) \bigg|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}_{ML}} = \mathbf{0} \tag{5.7}$$

- Yleensä aina ajatellaan että datavektorit ($\mathbf{X}$:n sarakkeet) ovat riippumattomia, jolloin uskottavuusfunktio hajoaa tuloksi:

$$L(\boldsymbol{\theta}) = p(\mathbf{X} \mid \boldsymbol{\theta}) = \prod_{j=1}^{n} p(\mathbf{x}(j) \mid \boldsymbol{\theta}) \tag{5.8}$$

- Useimmiten uskottavuusfunktiosta $L(\boldsymbol{\theta})$ otetaan vielä logaritmi $\ln L(\boldsymbol{\theta})$, koska silloin (a) laskenta muuttuu yleensä helpommaksi (useimpien tiheysfunktioiden tulo muuttuu summaksi) ja (b) sekä $L(\boldsymbol{\theta})$:n että $\ln L(\boldsymbol{\theta})$:n ääriarvo (maksimi) on samassa pisteessä.

Suurimman uskottavuuden periaate Esimerkki 1-ulotteiselle normaalijakaumalle

- Esimerkki: otetaan 1-ulotteinen (skalaari)data, eli datamatriisissa on vain skalaarilukuja $x(1), \ldots, x(n)$. Oletetaan että näytteet ovat riippumattomia toisistaan.
- Oletetaan että ne ovat normaalijakautuneita, mutta emme tiedä niiden odotusarvoa μ emmekä varianssia σ^2. Lasketaan nämä suurimman uskottavuuden menetelmällä.

- Uskottavuusfunktio “ensimmäiselle datapisteelle”:

$$p(x(1) \mid \mu, \sigma^2) = (2\pi\sigma^2)^{-1/2} \exp\left[-\frac{1}{2\sigma^2}[x(1) - \mu]^2\right]$$

- Uskottavuusfunktio $L(\boldsymbol{\theta})$ koko datasetille:

$$p(\mathbf{X} \mid \mu, \sigma^2) = (2\pi\sigma^2)^{-n/2} \exp\left[-\frac{1}{2\sigma^2}\sum_{j=1}^{n}[x(j) - \mu]^2\right] \tag{5.9}$$

- Otetaan logaritmi $\ln L(\boldsymbol{\theta})$:

$$\ln p(\mathbf{X} \mid \mu, \sigma^2) = -\frac{n}{2}\ln(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{j=1}^{n}[x(j) - \mu]^2 \tag{5.10}$$

- Etsitään ääriarvo asettamalla derivaatta nollaksi ja saadaan yhtälö odotusarvolle μ

$$\frac{\partial}{\partial\mu}\ln p(\mathbf{X} \mid \mu, \sigma^2) = \frac{1}{\sigma^2}\sum_{j=1}^{n}[x(j) - \mu] = 0 \tag{5.11}$$

- Ratkaistaan tämä μ:n suhteen, saadaan otoksen keskiarvo

$$\mu = \hat{\mu}_{ML} = \frac{1}{n}\sum_{j=1}^{n} x(j) \tag{5.12}$$

- Vastaava yhtälö varianssille σ^2 on

$$\frac{\partial}{\partial\sigma^2}\ln p(\mathbf{X} \mid \mu, \sigma^2) = 0 \tag{5.13}$$

  josta tulee ML-estimaattina otoksen varianssi

$$\hat{\sigma}^2_{ML} = \frac{1}{n}\sum_{j=1}^{n}[x(j) - \hat{\mu}_{ML}]^2. \tag{5.14}$$

5.4 Bayes-estimointi

Bayes-estimointi

- Bayes-estimointi on periaatteeltaan erilainen kuin ML-estimointi
- Oletamme että tuntematon parametrijoukko (vektori) $\boldsymbol{\theta}$ ei olekaan kiinteä vaan silläkin on jokin oma jakaumansa $p(\boldsymbol{\theta})$

- Kun saadaan mittauksia / havaintoja, niiden avulla $\boldsymbol{\theta}$:n jakauma *tarkentuu* (esim. sen varianssi pienenee).
- Käytännössä Bayes-estimointi ei välttämättä ole paljonkaan ML-menetelmää raskaampi mutta antaa parempia tuloksia
- Muistetaan todennäköisyyslaskennasta *ehdollisen todennäköisyyden* kaava: $P(A|B) = P(A,B)/P(B)$ missä A ja B ovat joitakin tapahtumia (ajattele vaikka nopanheittoa)
- Tätä voi hyödyntää ajattelemalla datajoukkoa ja sitä mallia (todennäköisyysjakauman parametreja) josta datajoukko on tullut:

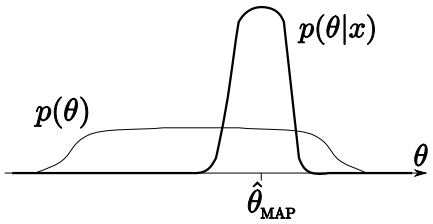
$$P(malli, data) = P(malli|data)P(data)$$

$$= P(data, malli) = P(data|malli)P(malli)$$

  jossa vain on kirjoitettu datan ja mallin yhteisjakauma kahdella eri tavalla
- Mutta tästä seuraa ns. *Bayesin kaava*

$$P(malli|data) = \frac{P(data|malli)P(malli)}{P(data)} \tag{5.15}$$

- Ajatus on että meillä on jokin käsitys mallin todennäköisyydestä ennen kuin mitään dataa on olemassa: $P(malli)$, ns. ***prioritodennäköisyys***
- Kun dataa sitten saadaan, tämä tarkentuu todennäköisyydeksi $P(malli|data)$, ns. *posterioritodennäköisyys*



Kuva 5.6: Bayesiläinen laskenta: Priori $p(\theta)$ ja posteriori $p(\theta|\mathbf{X})$, kun havaittu dataa $\mathbf{X}$. Posteriorista laskettu piste-estimaatti θ_{MAP}.

- Bayesin kaava kertoo kuinka tämä tarkentuminen lasketaan.
- Jos meillä on taas datamatriisi $\mathbf{X}$ ja tuntematon parametrivektori $\boldsymbol{\theta}$, niin Bayes antaa

$$p(\boldsymbol{\theta}|\mathbf{X}) = \frac{p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{X})}$$

- (HUOM matemaattinen merkintätapa: kaikkia todennäköisyysjakaumia (tiheysfunktioita) merkitään vain p:llä, ja argumentti kertoo mistä p:stä on kysymys. Tämä on tietysti matemaattisesti väärin mutta yleisesti käytetty tapa.)
- Yo. kaavassa nähdään että priorijakauma kerrotaan *uskottavuusfunktiolla* ja jaetaan *datan jakaumalla*, jotta saataisiin posteriorijakauma.
- Bayes-analyysissä "keksitään" priorijakauma ja pyritään muodostamaan posteriorijakauma
- Usein kuitenkin tyydytään etsimään posteriorin maksimikohta $\boldsymbol{\theta}$:n suhteen: Maksimi-posteriori eli MAP-estimointi
- Tällä on selkeä yhteys ML-estimointiin: nimittäin Bayesin kaava antaa

  $$\ln p(\boldsymbol{\theta}|\mathbf{X}) = \ln p(\mathbf{X}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta}) - \ln p(\mathbf{X})$$

  ja etsittäessä maksimikohtaa $\boldsymbol{\theta}$:n suhteen tulee gradienttiyhtälö

  $$\frac{\partial}{\partial \boldsymbol{\theta}} \ln p(\mathbf{X}|\boldsymbol{\theta}) + \frac{\partial}{\partial \boldsymbol{\theta}} \ln p(\boldsymbol{\theta}) = \mathbf{0} \tag{5.16}$$

- Huomataan että ML-menetelmään verrattuna tulee ylimääräinen termi $\partial(\ln p(\boldsymbol{\theta}))/\partial\boldsymbol{\theta}$.
- Siitä on joskus suurta hyötyä, kuten seuraavassa kohdassa nähdään.

5.5 Regressiosovitus

Regressiosovitus

- Regressio eroaa tähänastisista ohjaamattoman oppimisen menetelmistä (PCA, DSS, todennäköisyysjakaumien estimointi) siinä, että kuhunkin datavektoriin $\mathbf{x}(i)$ liittyy jokin lähtösuure $y(i)$, jonka arvellaan noudattavan mallia

  $$y(i) = f(\mathbf{x}(i), \boldsymbol{\theta}) + \epsilon(i)$$

- Tässä f on jokin funktio, ns. regressiofunktio, ja $\epsilon(i)$ on virhe (y-akselin suunnassa)
- Funktiolla f on tunnettu (tai oletettu) muoto mutta tuntematon parametrivektori $\boldsymbol{\theta}$

  Yleinen esimerkki polynomisesta sovituksesta

Regressiosovitus ML-estimointi lineaarisessa regressiossa

- Esimerkkinä lineaarinen regressio:

$$y(i) = \theta_0 + \sum_{j=1}^{d} \theta_j x_j(i) + \epsilon(i) = \theta_0 + \boldsymbol{\theta}^T \mathbf{x}(i) + \epsilon(i)$$

- Tunnettu tapa ratkaista parametrit $\hat{\boldsymbol{\theta}}$ on *pienimmän neliösumman* keino: minimoi $\boldsymbol{\theta}$:n suhteen

$$\frac{1}{n} \sum_{i=1}^{n} [y(i) - f(\mathbf{x}(i), \boldsymbol{\theta})]^2$$

- Itse asiassa tämä on *ML-estimaatti tietyllä ehdolla*: jos regressiovirhe $\epsilon(i)$ (kaikille arvoille i) on riippumaton $\mathbf{x}(i)$:n arvosta ja normaalijakautunut odotusarvolla 0 ja hajonnalla σ. (Hyvin luonnollinen oletus!)

- Silloin ML-estimaatti parametrille $\boldsymbol{\theta}$ saadaan uskottavuusfunktiosta ($Y = (y(i))_i$)

$$p(\mathbf{X}, Y|\boldsymbol{\theta}) = p(\mathbf{X}) p(Y|\mathbf{X}, \boldsymbol{\theta}) = p(\mathbf{X}) \prod_{i=1}^{n} p(y(i)|\mathbf{X}, \boldsymbol{\theta})$$

  jossa on vain käytetty ehdollisen todennäköisyyden kaavaa, huomattu että $\mathbf{X}$ ei riipu regressiomallin parametreistä ja oletettu eri mittapisteissä olevat $y(i)$:t riippumattomiksi (kuten ML-estimoinnissa yleensä oletetaan)

- Otetaan logaritmi

$$\ln p(\mathbf{X}, Y|\boldsymbol{\theta}) = \ln p(\mathbf{X}) + \sum_{i=1}^{n} \ln p(y(i)|\mathbf{X}, \boldsymbol{\theta})$$

- Mutta jos $\mathbf{X}$ eli sen sarakkeet $\mathbf{x}(i)$ ovat kiinteitä niinkuin ehdollisessa todennäköisyydessä ajatellaan, on $y(i)$:n jakauma sama kuin $\epsilon(i)$:n mutta keskiarvo on siirtynyt kohtaan $f(\mathbf{x}(i), \boldsymbol{\theta})$ - siis normaalijakauma:

$$p(y(i)|\mathbf{X}, \boldsymbol{\theta}) = vakio \times \exp(-\frac{1}{2\sigma^2}[y(i) - f(\mathbf{x}(i), \boldsymbol{\theta})]^2)$$

- Vihdoin saadaan uskottavuusfunktion logaritmiksi:

$$\ln p(\mathbf{X}, Y|\boldsymbol{\theta}) = \ln p(\mathbf{X}) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} [y(i) - f(\mathbf{x}(i), \boldsymbol{\theta})]^2 + n \ln(vakio)$$

- Maksimoinnissa (derivointi $\boldsymbol{\theta}$:n suhteen) $\boldsymbol{\theta}$:sta riippumattomat termit katoavat

- Täten maksimointi on täsmälleen sama kuin pienimmän neliösumman minimointi! (Koska $p(\mathbf{X})$ on datamatriisin jakauma eikä riipu mistään mallista)

Regressiosovitus Bayesin priorijakauman hyväksikäyttö lineaarisessa regressiossa

- Mitä Bayes voi antaa tähän lisää?
- ML:ssä maksimoitiin $p(\mathbf{X}, Y|\boldsymbol{\theta})$. Bayes-estimoinnissa maksimoidaan $p(\boldsymbol{\theta}|\mathbf{X}, Y) \propto p(\mathbf{X}, Y|\boldsymbol{\theta}) \cdot p(\boldsymbol{\theta})$, joka logaritmin jälkeen on $\ln p(\mathbf{X}, Y|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})$
- Eli maksimoitavassa funktiossa on priorijakaumasta tuleva termi $\ln p(\boldsymbol{\theta})$ ja maksimoitava funktio on
$$-\frac{1}{2\sigma^2}\sum_{i=1}^{n}[y(i) - f(\mathbf{x}(i), \boldsymbol{\theta})]^2 + \ln p(\boldsymbol{\theta})$$
- Tästä näkee, että jos kohinan $\epsilon(i)$ varianssi σ^2 on hyvin suuri, niin 1. termi on pieni ja $\boldsymbol{\theta}$:n priorijakauma määrää pitkälle sen arvon.
- Päinvastoin jos σ^2 on pieni, 1. termi dominoi ja priorijakaumalla on hyvin vähän vaikutusta lopputulokseen
- Tämä tuntuu hyvin luonnolliselta ja hyödylliseltä!
- Esimerkiksi jos on syytä olettaa että kaikki parametrit ovat (0,1)-normaalijakautuneita, on
$$p(\boldsymbol{\theta}) = vakio \times \exp(-1/2\sum_{k=1}^{K}\theta_k^2),$$
$$\ln p(\boldsymbol{\theta}) = \ln(vakio) - 1/2\sum_{k=1}^{K}\theta_k^2,$$
ja prioritermin maksimointi johtaa pieniin arvoihin parametreille.
- Tällöin regressiomalli yleensä käyttäytyy paremmin kuin jos parametreilla on hyvin suuria arvoja.

Esimerkki lineaarisesta regressiosta

- Katsotaan esimerkkinä lineaarista regressiota
- Nyt oletetaan että datasta tiedetään seuraavaa: yhteys on joko 1. asteen (lineaarinen) tai ehkä toisen asteen polynomi, joka kulkee hyvin luultavasti origon kautta. Mallin virhe (mitattujen $y(i)$- arvojen poikkeama mallin antamista arvoista) on normaalijakautunut hajonnalla σ.
- Malli on silloin
$$y(i) = a + bx(i) + cx(i)^2 + \epsilon(i)$$
- Käytetään Bayes-estimointia. Priorit valitaan normaalijakautuneiksi siten että a:n keskiarvo on 0 hajonta on 0.1, b:n keskiarvo on 1 ja hajonta 0.5 ja c:n keskiarvo on 0 ja hajonta 1.

- Näiden tiheydet ovat siis $vakio \times e^{-50a^2}$, $vakio \times e^{-2(b-1)^2}$, $vakio \times e^{-0.5c^2}$ ja maksimoitava funktio on (kun vakiot tiputetaan pois)

$$-\frac{1}{2\sigma^2}\sum[y(i) - a - bx(i) - cx(i)^2]^2 - 50a^2 - 2(b-1)^2 - 0.5c^2$$

- Derivoimalla a:n suhteen ja ratkaisemalla saadaan

$$a = \frac{1}{n + 100\sigma^2}\sum[y(i) - bx(i) - cx(i)^2]$$

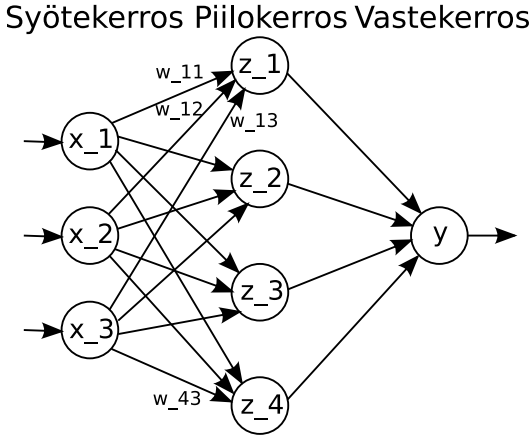
  ja vastaavat yhtälöt b:lle ja c:lle, joista ne kaikki voidaan ratkaista (lineaarinen yhtälöryhmä kolmelle muuuttujalle).

- Ilman Bayesia yo. kaavasta putoaa kokonaan pois termi $100\sigma^2$, joten Bayes pakottaa a:n aina pienempään arvoon. Ainoastaan kun mallin virhevarianssi σ^2 on hyvin pieni, Bayesin vaikutus menee vähäiseksi (silloin data on hyvin luotettavaa ja etukäteistiedon merkitys vähenee).

5.6 Esimerkki regressiosta: neuroverkko

Esimerkki regressiosta: neuroverkko

- *Neuroverkko* on oheisen kuvan esittämä laskentamalli, joka laskee kohtalaisen monimutkaisen funktion tulosuureistaan (inputeistaan)

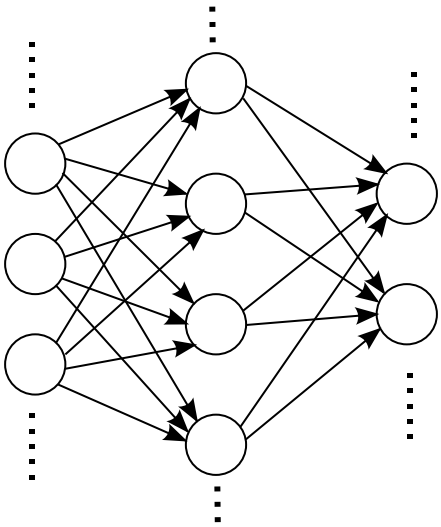


Kuva 5.7: Monikerrosperseptroniverkko (MLP) yhdellä ulostulolla.

- Malli koostuu “neuroneista” jotka laskevat funktioita tulosuureistaan

- Ensimmäisen kerroksen, ns. piilokerroksen neuronit laskevat kukin funktion $z_i = f(\mathbf{x}, \mathbf{w}_i)$ tulosuureista jotka ovat syötteenä olevan vektorin $\mathbf{x}$ alkiot

- Funktio on epälineaarinen ja $\mathbf{w}_i$ on i:nnen neuronin parametrivektori (jossa alkioita yhtä monta kuin $\mathbf{x}$:ssä)
- Toisen kerroksen neuroni laskee vain painotetun summan piilokerroksen lähtösuureista z_i: $y = \sum_i v_i z_i$
- Toisessa kerroksessa voi olla myös useampia rinnakkaisia neuroneita, jolloin verkko laskee regressiofunktion vektoreista $\mathbf{x}$ vektoreihin $\mathbf{y}$;



Kuva 5.8: Yleinen monikerrosperseptroniverkko (MLP).

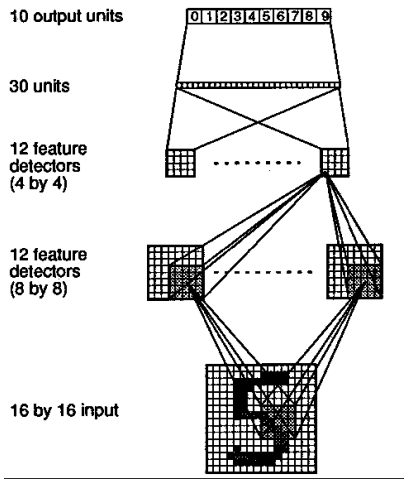
- MLP-verkon sovelluksena vaikkapa numerontunnistus
- Neuroverkoista on tullut tavattoman suosittuja hyvin monilla aloilla, koska ne laskevat hyvin isoja regressiomalleja, ovat osoittautuneet suhteellisen tarkoiksi ja tehokkaiksi, ja parametrien laskemiseksi isoissakin malleissa (satoja tai tuhansia neuroneita) on hyvin tehokkaita koneoppimisalgoritmeja ja helppokäyttöisiä ohjelmistopaketteja
- Raportoituja sovelluksia on satoja ellei tuhansia
- Enemmän kurssissa T-61.5130 Machine Learning and Neural Networks.

Yhteenveto

Yhteenveto

Tässä luvussa

- tutustuttiin tiheysfunktioihin

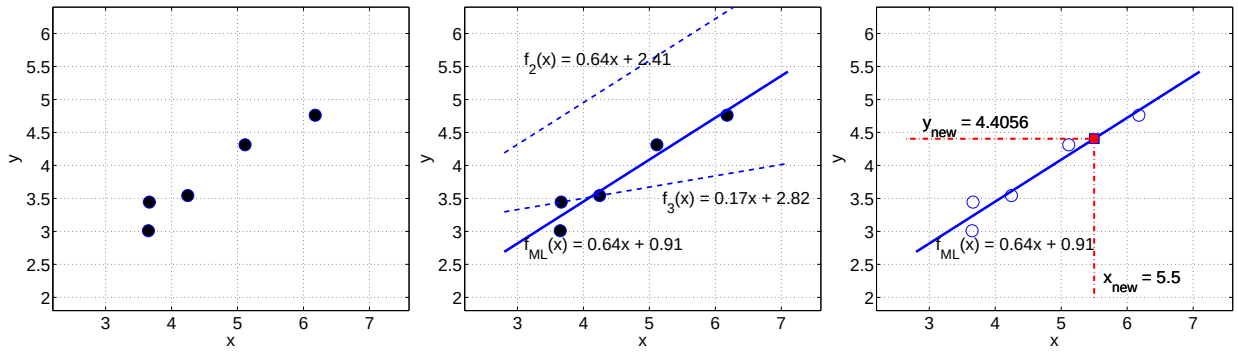


Kuva 5.9: Esimerkki neuroverkon käytöstä käsinkirjoitettujen merkkien tunnistamisessa. Neuroverkko oppii aineistosta kunkin numeron ja antaa suurimman vasteen kyseiselle ulostuloneuronille.

- esiteltiin suurimman uskottavuuden menetelmä parametrien estimointiin
- liitettiin etukäteistieto (a priori) parametrien estimointiin Bayesin teoreemaa käyttäen
- laskettiin regressiota eri menetelmillä (ML, Bayes, neuroverkko)

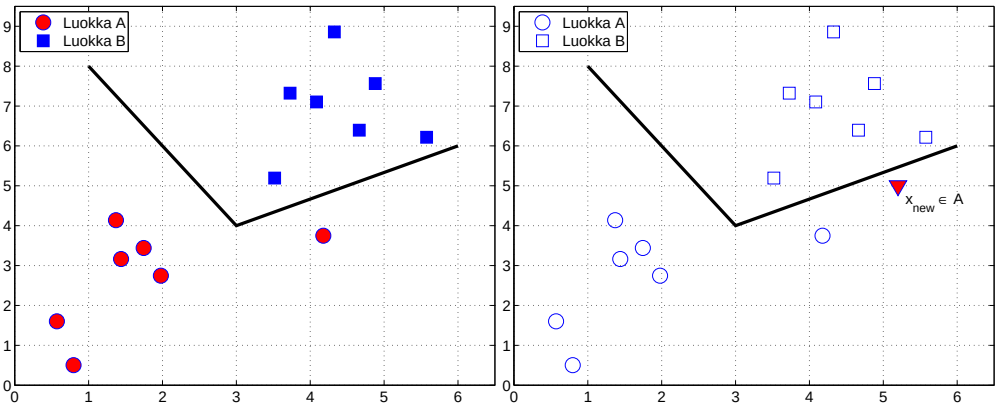
Liite: Kuvia esitykseen, luku 5

Esimerkki yksinkertaisesta lineaarisesta regressiosta



Kuva 5.10: Vasen kuva: datapisteitä (x_i, y_i). Tavoite selittää y x:n avulla käyttämällä funktiota $f(x, \theta) = kx + b$. Keskikuva: Kolme eri parametrisovitusta θ_{ML}, θ_2, θ_3. Parametrit estimoidaan pienimmän neliösumman kriteerin mukaisesti ja $f_{ML}(x) = 0.64x + 0.91$ antaa parhaimman sovituksen. Oikea kuva: opittua funktiota f voidaan käyttää arvioimaan uudelle x_{new}:lle "sopiva" y_{new}. Takaisin kalvoihin

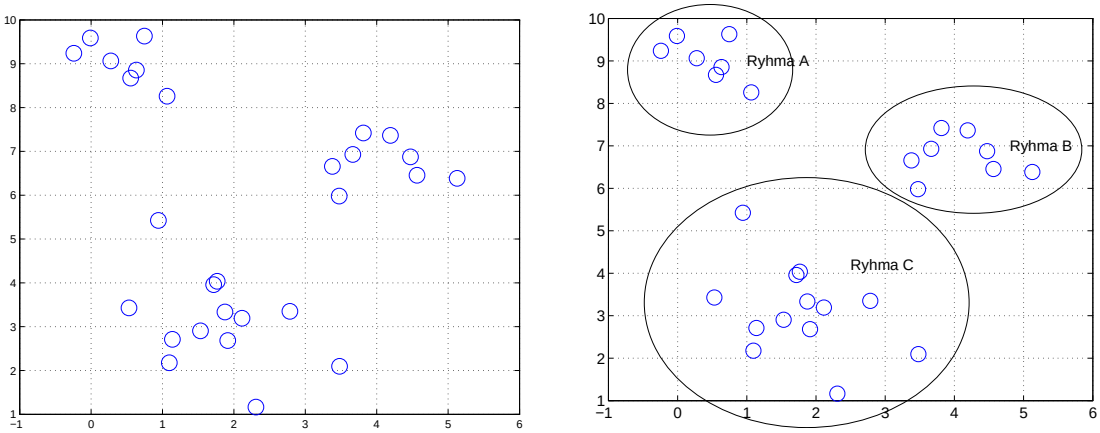
Esimerkki yksinkertaisesta luokittelusta



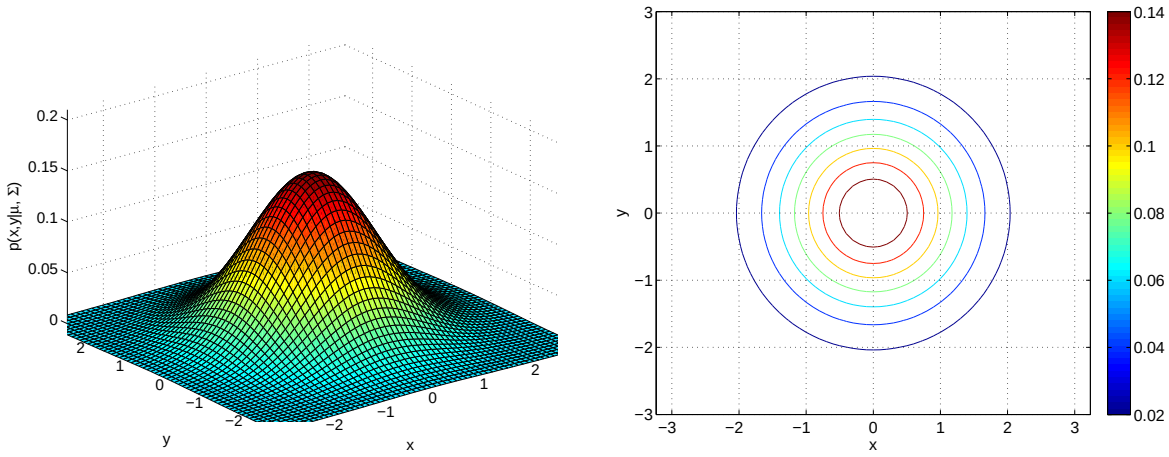
Kuva 5.11: Vasemmalla 2-ulotteinen data-aineisto, jossa lisänä luokkainformaatio jokaisesta datapisteestä joko A (pallo) tai B (neliö). Datasta on opittu luokittelija (luokkaraja). Oikealla luokittelijaa käytetään antamaan "sopiva" luokka-arvo uudelle datapisteelle x_{new}. Tässä esimerkissä luokittelija voi tuntua antavan "väärän" tuloksen. Takaisin kalvoihin

Esimerkki ryhmittelystä

Esimerkki 2-ulotteisesta normaalijakaumasta $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen diagonaalinen C, jossa lisäksi $\sigma_1 = \sigma_2$



Kuva 5.12: Vasemmalla 2-ulotteista dataa ilman luokkainformaatiota. Oikealla näytteet ryhmitelty kolmeen ryppääseen, joita aletaan kutsua ryhmiksi A, B, C. Takaisin kalvoihin

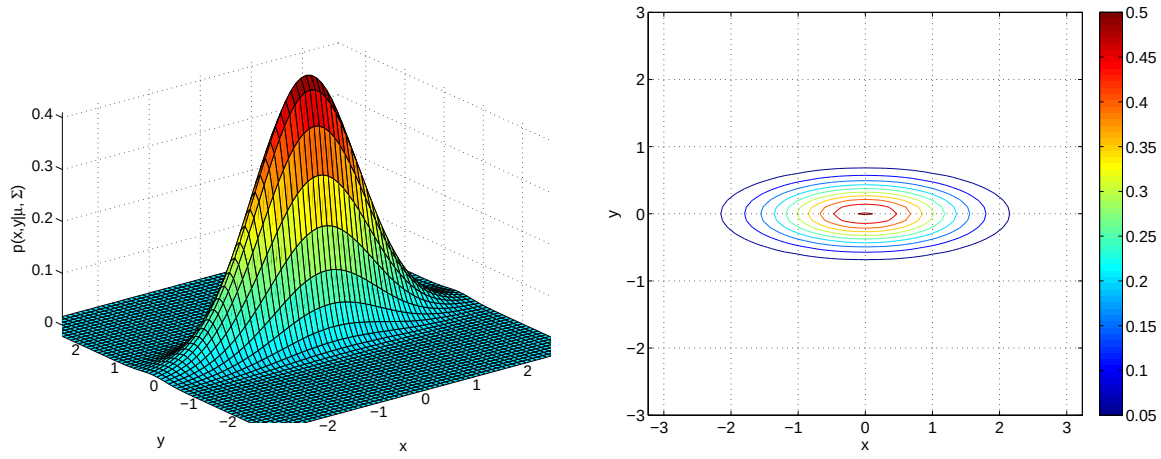


toinen esimerkki kolmas esimerkki Takaisin kalvoihin

Kuva 5.13: Esimerkki 2-ulotteisesta normaalijakaumasta: $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen diagonaalinen $\mathbf{C}$, jossa lisäksi $\sigma_1 = \sigma_2$: $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Tässä esim. $p(\begin{bmatrix} 0 & 0 \end{bmatrix}^T) \approx 0.159$ ja $p(\begin{bmatrix} 1 & -2 \end{bmatrix}^T) \approx 0.0131$.
toinen esimerkki kolmas esimerkki Takaisin kalvoihin

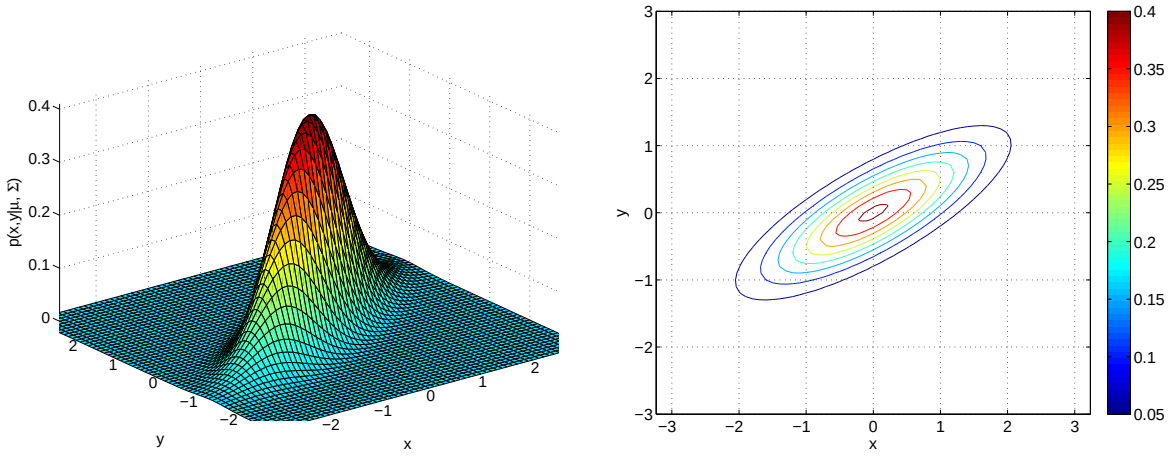
Esimerkki 2-ulotteisesta normaalijakaumasta $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen diagonaalinen $\mathbf{C}$, jossa $\sigma_1 \neq \sigma_2$

Esimerkki 2-ulotteisesta normaalijakaumasta $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen ei-diagonaalinen $\mathbf{C}$



ensimmäinen esimerkki kolmas esimerkki Takaisin kalvoihin

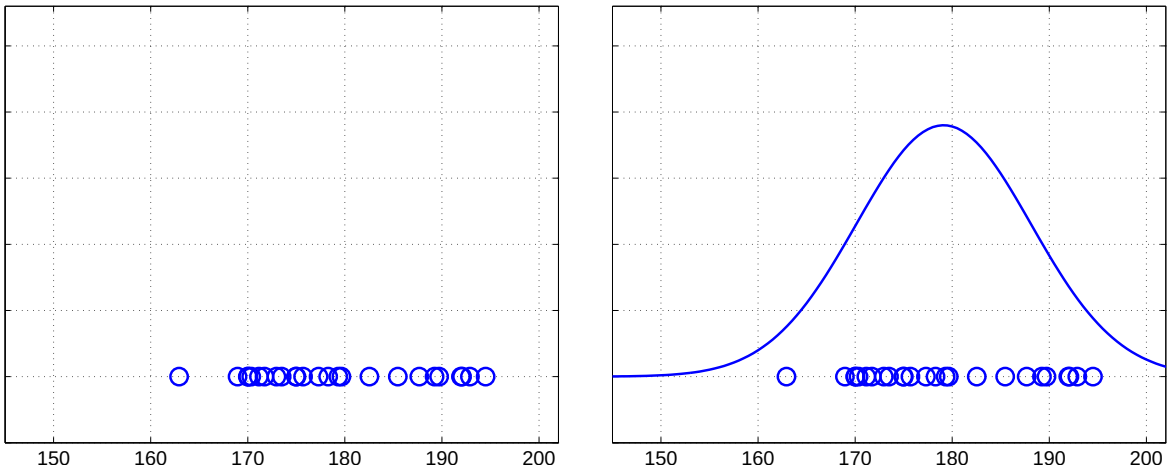
Kuva 5.14: Esimerkki 2-ulotteisesta normaalijakaumasta: $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen diagonaalinen $\mathbf{C}$, jossa $\sigma_1 \neq \sigma_2$: $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}$. Tässä esim. $p(\begin{bmatrix} 0 & 0 \end{bmatrix}^T) \approx 0.503$ ja $p(\begin{bmatrix} 1 & -0.3 \end{bmatrix}^T) \approx 0.195$.
ensimmäinen esimerkki kolmas esimerkki Takaisin kalvoihin



ensimmäinen esimerkki toinen esimerkki Takaisin kalvoihin

Kuva 5.15: Esimerkki 2-ulotteisesta normaalijakaumasta: $\mu_1 = 0$, $\mu_2 = 0$, symmetrinen eidiagonaalinen $\mathbf{C}$: $\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $\mathbf{C} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 0.4 \end{bmatrix}$. Tässä esim. $p(\begin{bmatrix} 0 & 0 \end{bmatrix}^T) \approx 0.411$ ja $p(\begin{bmatrix} 1 & 1 \end{bmatrix}^T) \approx 0.108$.
ensimmäinen esimerkki toinen esimerkki Takaisin kalvoihin

Data X ja siitä eräs tiheysfunktio $p(x)$



Kuva 5.16: Vasemmalla 1-ulotteinen datamatriisi $\mathbf{X}$, jossa 25 näytettä, esimerkiksi ihmisten pituuksia (cm). Oikealla siihen sovitettu tiheysfunktio $p(x)$, joka tällä kertaa normaalijakauma. Normaalijakauman parametrit $\hat{\mu}$ ja $\hat{\sigma}$ on *estimoitu* datasta. Silmämääräisesti $\hat{\mu} \approx 179$ ja $\hat{\sigma} \approx 9$. Takaisin kalvoihin

Suurimman uskottavuuden menetelmä Maximum likelihood method by Milton and Arnold

1. Obtain a random sample $x(1), x(2), \ldots, x(n)$ from the distribution of a random variable X with density p and associated parameter θ
2. Define a likelihood function $L(\theta)$ by

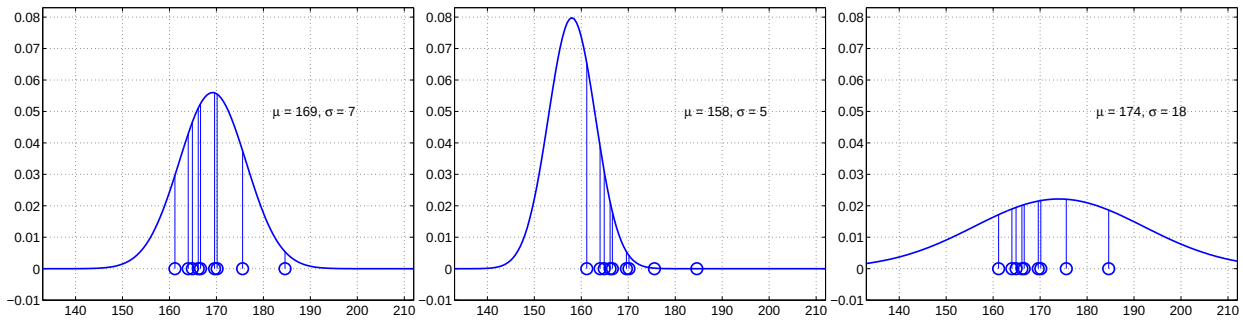
$$L(\theta) = \prod_{i=1}^{n} p(x(i))$$

3. Find the expression for θ that maximizes the likelihood function. This can be done directly or by maximizing $\ln L(\theta)$
4. Replace θ by $\hat{\theta}$ to obtain an expression for ML estimator for θ
5. Find the observed value of this estimator for a given sample

p. 233. Milton, Arnold: Introduction to probability and statistics. Third edition. McGraw-Hill, 1995. Takaisin kalvoihin

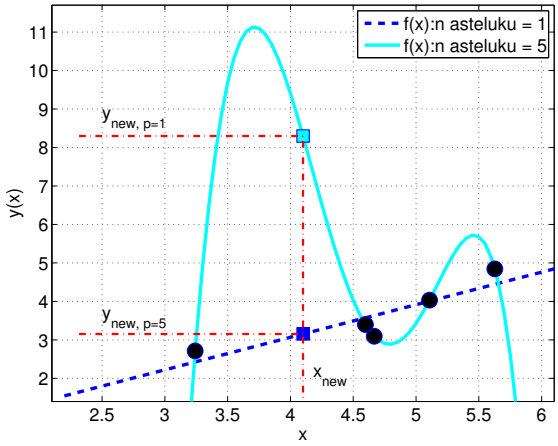
ML:n perusajatus "Valitaan todennäköisimmin datan generoinut jakauma"

Esimerkki lineaarisesta regressiosta ja ylioppimisesta Polynomisovitus asteilla p=1 ja p=5 pieneen datamäärään N=5



Takaisin kalvoihin

Kuva 5.17: Annettuna datajoukko $x(1), \ldots, x(9) \in R^1$ palloina ja kolme gaussista tiheysfunktiota eri parametriyhdistelmillä $\boldsymbol{\theta}_1 = \{\mu = 169,\ \sigma = 7\}$, $\boldsymbol{\theta}_2 = \{\mu = 158,\ \sigma = 5\}$ ja $\boldsymbol{\theta}_3 = \{\mu = 174,\ \sigma = 18\}$. Lasketaan uskottavuusfunktio $L(\boldsymbol{\theta}) = p(x(1)|\boldsymbol{\theta}) \cdot p(x(2)|\boldsymbol{\theta}) \cdot \ldots \cdot p(x(9)|\boldsymbol{\theta})$. Ensimmäinen tuottaa selvästi suurimman $L(\boldsymbol{\theta})$:n arvon annetulla datasetillä: $L(\boldsymbol{\theta}_1) = 0.029 \cdot 0.042 \cdot \ldots \cdot 0.005 > L(\boldsymbol{\theta}_3) > L(\boldsymbol{\theta}_2)$. Valitaan siten $\boldsymbol{\theta}_1 = \{\mu = 169,\ \sigma = 7\}$ (*näistä kolmesta vaihtoehdosta!*), koska $\boldsymbol{\theta}_1$ on todennäköisimmin generoinut datan $\mathbf{X}$.
Takaisin kalvoihin



Kuva 5.18: Regressiosovitus $y(i) = f(\mathbf{x}(i), \boldsymbol{\theta}) + \epsilon(i)$. Tunnetaan viisi havaintoa $\{x(i), y(i)\}_i$ (mustat pallot). Asteluvun $p = 1$ polynomifunktio $f_{p=1}(.)$ (sininen katkoviiva) sovittuu melko hyvin (pienehköt virheet $(y(i) - f(x(i), \boldsymbol{\theta}_1))^2$). Sen sijaan polynomisovitus $f_{p=5}(.)$ asteluvulla $p = 5$ vie neliövirheen nollaan, mutta on ylioppinut tunnettuun dataan ja siten huono uusille havainnoille (esim. x_{new}). Takaisin kalvoihin

6 Hahmontunnistuksen perusteita

Johdanto

Aiheeseen liittyviä laskari- ja demotehtäviä:

- H3/1, H3/2: MLE- ja Bayes-regressio
- H3/3: kNN-luokitin
- H3/4: optimaalinen Bayes-luokitin
- H4/2: hierarkkinen ryhmittely
- H4/3: c-means
- T3: regressiosovitus, kNN-luokitin
- T4: hierarkkinen ryhmittely, c-means

6.1 Luokittelu

Luokittelu

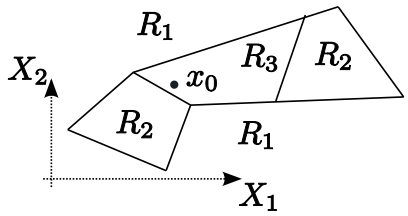
- *Hahmontunnistus* on tieteenala, jossa *luokitellaan* joitakin kohteita niistä tehtyjen havaintojen perusteella luokkiin
- Keskeiset ongelmat ovat havaintojen teko, niiden pelkistys ja esikäsittely, muuntaminen sopivimmaksi kuvaukseksi eli ns. *piirreirrotus*, sekä piirrevektorien perusteella tapahtuva luokittelu.
- Esimerkki: käsinkirjoitettujen numeroiden, vaikkapa kirjeiden osoitteissa olevien postinumeroiden (00076 Aalto) automaattinen tunnistus kirjeiden luokittelua varten. Postinumero pitää valokuvata ja kukin erillinen numero pitää digitaalisesti erotella ja normalisoida (esikäsittely), ja sen jälkeen etsiä sopivat numeeriset tunnusluvut (piirteet) joiden avulla luokittelu on mahdollisimman tehokasta
- Toinen esimerkki: tietyn objektin havaitseminen muiden objektien seasta. Katso kurssin "T-61.5070 Computer Vision" harjoitustyö
- Esimerkkejä hahmontunnistussovelluksista:
  - Satelliittikuvien tulkinta
  - Robottinäkö
  - Tekstin tunnistus

- Puheen tunnistus, “älykkäät” käyttöliittymät
- Henkilön tunnistus (sormenjälki, puhe, nimikirjoitus, kasvokuva, silmän iris, ...)
- Lääketieteellisten aineistojen seulonta, esim. irtosolunäytteet
- Laadunvalvonta, pullonpalautusautomaatit
- Dokumenttien automaattinen luku ja käsittely (shekit, lomakkeet, ...)
- Auton rekisterinumeron tunnistus
- .. ja paljon muuta. Kyseessä on selkeä kasvuala.

Hahmoalueet, erotinfunktio

Hahmoalueet, erotinfunktio

- *Hahmo* on tunnistettavan kohteen numeerinen esitystapa, joka voi olla vektori, matriisi, puu, graafi, merkijono, ...
- Tässä suppeassa esityksessä keskitytään taas vektoreihin: ajatellaan siis että kukin kohde on esitettävissä vektorina $\mathbf{x} = (x_1, \ldots, x_d)^T$ (pisteenä d-ulotteisessa vektoriavaruudessa).
- Se miten tällaiseen vektoriin päädytään on hyvin sovelluskohtaista ja itse asiassa ehkä vaikein osa koko hahmontunnistusjärjestelmän suunnittelua; emme puutu siihen tässä.
- Kaksi keskeistä kysymystä silloin on: 1. Mitkä hahmot ovat keskenään samanlaisia? 2. Minkälaisia ryhmiä hahmot muodostavat?
- Vastauksia näihin etsitään geometriasta, ajattelemalla siis hahmot pisteinä d-ulotteisessa avaruudessa ja katsomalla niiden välisiä *vektorietäisyyksiä* (esim. euklidinen etäisyys)
- Myös todennäköisyyslaskennalla on tärkeä rooli kuten nähdään
- Merkitään nyt luokkia merkinnällä $\omega_1, ..., \omega_M$ (omega), siis niitä on M kappaletta (esim. käsinkirjoitetut numerot: $M = 10$)
- Geometrisesti ajatellaan että koko d-ulotteinen vektoriavaruus jakaantuu M:ään alueeseen, ns. *hahmoalueeseen* $R_1, ..., R_M$ joiden keskinäiset leikkaukset ovat nollia ja joiden unioni on koko avaruus
- Jos hahmovektori $\mathbf{x}$ kuuluu alueeseen R_j, päätellään että $\mathbf{x}$:n luokka on ω_j.
- Esimerkki: Kuvassa on kolme hahmoaluetta, joista R_2 kahdessa osassa. Alueiden keskinäiset leikkaukset tyhjiä, yhdiste koko R^2. Kuvassa siis $\mathbf{x}_0$ kuuluu alueeseen R_3 joten sen luokka on ω_3
- Alueet ovat mielivaltaisen muotoisia, ja voi olla vaikeaa laskennallisesti päätellä mihin alueeseen $\mathbf{x}$ kuuluu



Kuva 6.1: Hahmoalueet

- Siksi usein käytetään *erotinfunktioita* $g_1(\mathbf{x}), ..., g_M(\mathbf{x})$ joiden avulla luokittelusääntö on seuraava:

  jos $g_j(\mathbf{x}) = max_i g_i(\mathbf{x})$, niin $\mathbf{x} \in \omega_j$

- Erotinfunktio voi olla vaikkapa datasta estimoitu tiheysfunktio $p_j(\mathbf{x}|\theta)$

- Yhteys hahmoalueisiin on silloin se, että

$$R_j = \{\mathbf{x}|g_j(\mathbf{x}) = max_i g_i(\mathbf{x})\}$$

- *Luokkaraja* luokkien i ja j välillä on $\{\mathbf{x}|g_i(\mathbf{x}) = g_j(\mathbf{x})\}$, siis niiden pisteiden joukko joissa erotinfunktiot saavat saman arvon. Se on pinta, jonka dimensio on $d - 1$

- Esimerkki: Ylläolevassa esimerkkikuvassa siis (esimerkiksi) $g_3(\mathbf{x}_0) > g_2(\mathbf{x}_0) > g_1(\mathbf{x}_0)$. Koska hahmoavaruus $d = 2$ (alueita), niin luokkarajat ovat käyriä

Hahmoalueet, erotinfunktio Luokitteluvirheen minimointi

- Oleellista on löytää sellaiset hahmoalueet tai ekvivalentisti erotinfunktiot, että kun yo. sääntöä käytetään, *luokitteluvirhe minimoituu*

- Luokittelu on ohjattua oppimista ("supervised learning"). On käytettävissä joukko hahmovektoreita $\mathbf{x}$, joiden *oikea luokka* ω *tiedetään*. Usein luokkatieto on "kallista" – esimerkiksi asiantuntijan "käsin" luokittelemaan aineistoa

- Tunnettu joukko jaetaan tarvittaessa erillisiksi opetusjoukoksi ja testijoukoksi

- Hahmoalueet tai erotinfunktiot muodostetaan *opetusjoukon* avulla

- *Testijoukkoa* voidaan käyttää luokitteluvirheen laskemiseen: annetaan testijoukko ilman luokkatietoa luokittimille, ja lasketaan kuinka usein luokitin päätyi väärään lopputulokseen Esimerkki

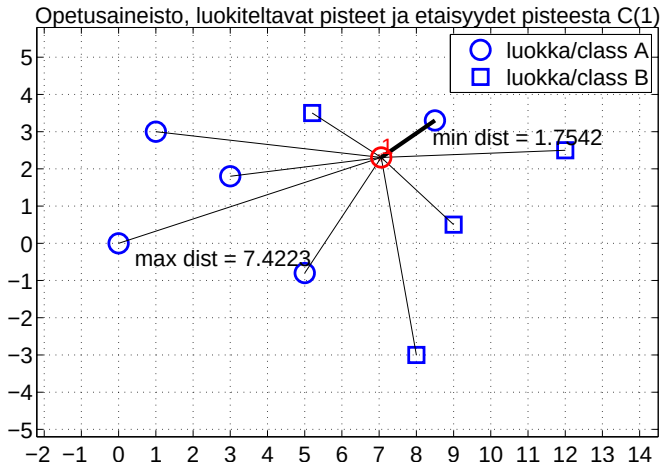
- Koska opetusjoukko on äärellinen, ei yleensä voida muodostaa virheetöntä luokitinta, eli kun uusia hahmovektoreita luokitellaan, menee luokka joskus väärin

6.2 Lähimmän naapurin luokitin (kNN)

Lähimmän naapurin luokitin (kNN)

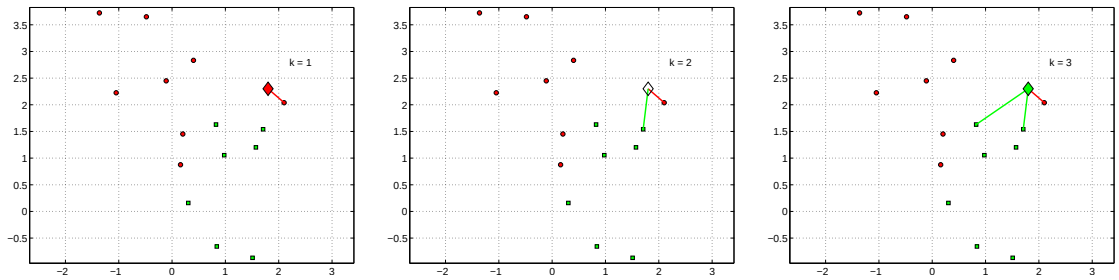
- Idea on hyvin yksinkertainen: olkoon uusi luokiteltava vektori $\mathbf{x} \in R^d$, jolle pitäisi siis löytää "oikea" luokka ω
- Käytettävissä opetusjoukko $(\mathbf{x}, \omega)_j$ eli tunnetaan valmiiksi luokiteltua (M luokkaa) dataa $\mathbf{x}_{\omega_i}(j) \in R^d$: $\{\mathbf{x}_{\omega_1}(1), \ldots, \mathbf{x}_{\omega_1}(n_1)\}$, ..., $\{\mathbf{x}_{\omega_M}(1), \ldots, \mathbf{x}_{\omega_M}(n_M)\}$.
- Lasketaan etäisyydet pisteen $\mathbf{x}$ ja opetusjoukon pisteiden välillä ($n = n_1 + \ldots + n_M$) ja etsitään ne k vektoria, jotka ovat *lähinnä* $\mathbf{x}$:ää (k lähintä naapuria)
- Usein etäisyysmittana käytetään euklidista etäisyyttä

$$D(\mathbf{x}, \mathbf{z}) = \|\mathbf{x} - \mathbf{z}\| = \sqrt{\sum_{i=1}^{d} (x_i - z_i)^2}$$



Kuva 6.2: Euklidinen etäisyys lähimpään naapuriin ≈ 1.75.

- Neliöjuurta ei käytännössä tarvitse ottaa, koska se ei vaikuta kun vertaillaan etäisyyksiä keskenään
- Luokitellaan $\mathbf{x}$ siihen luokkaan, mihin *enemmistö naapureista kuuluu*. Tasapelin sattuessa heitetään lanttia.
- Usein luokittelussa kaksi luokkaa $M = 2$. Tällöin k kannattaa valita parittomaksi ($k = 1, 3, 5, \ldots$)



Kuva 6.3: Luokiteltava piste vinoneliö. Opetusaineistossa $M = 2$ luokkaa (punaiset pallot $\{\mathbf{x}_{\text{pallo}}(1), \ldots, \mathbf{x}_{\text{pallo}}(8)\}$ ja vihreät neliöt $\{\mathbf{x}_{\text{nelio}}(1), \ldots, \mathbf{x}_{\text{nelio}}(7)\}$). Vas. $k = 1$, vinoneliö luokitellaan punaiseksi palloksi. Kesk. $k = 2$, tasapeli, arvotaan. Oik. $k = 3$, kaksi neliötä, yksi pallo, vinoneliö luokitellaan vihreäksi neliöksi.

- Menetelmä on idealtaan yksinkertainen ja myös varsin tehokas
- Jos opetusjoukko on suuri ja dimensio d korkea, on lähimpien naapureiden haku raskas operaatio (sama ongelma kuin tietokannoissa); tähän on kehitetty monenlaisia menetelmiä, esim. hakupuita.
- Huomaa siis, että tietokone "ei näe", mitkä ovat lähimmän naapurit vaan se joutuu laskemaan kaikki etäisyydet (ainakin ensimmäisellä kerralla)
- Luokitinta voi kokeilla Web-demoilla, esim. `www.cs.cmu.edu/~zhuxj/courseproject/knndemo/KNN.html`

6.3 Bayes-optimaalinen luokitin

Bayes-optimaalinen luokitin

- Tämä perustuu taas Bayesin kaavaan ja filosofiaan
- Koska kyseessä on luokittelu, on tärkein muuttuja *luokan ω_i todennäköisyys*, sekä ennen havaintojen saamista että sen jälkeen.
- Merkitään luokkien prioritodennäköisyyksiä $P(\omega_1), ..., P(\omega_M)$ ja posterioritodennäköisyyksiä sitten kun on tehty havainto, siis hahmovektori $\mathbf{x}$ on käytettävissä, $P(\omega_1|\mathbf{x}), ..., P(\omega_M|\mathbf{x})$
- Esim. postinumeroiden luokittelussa priori vastaa kysymykseen "kuinka todennäköinen postinumeroissa on numero kolmonen"; tämän voi laskea postin tilastoista
- Posteriori vastaa kysymykseen "kuinka todennäköisesti juuri tämä annettu numero on kolmonen"

- Bayesin kaavan mukaan näiden välinen yhteys on seuraava:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})}$$

  Siellä olevien kahden uuden jakauman tulkinta: $p(\mathbf{x})$ on kaikkien hahmovektoreiden yhteinen tiheysfunktio, $p(\mathbf{x}|\omega_i)$ on luokkaan ω_i kuuluvien hahmovektoreiden tiheysfunktio, ns. luokkatiheysfunktio luokalle ω_i

- Käytännössä luokkatiheysfunktiot voidaan estimoida opetusnäytteestä esim. olettamalla ne gaussisiksi ja laskemalla (suurimman uskottavuuden estimoinnilla) kullekin luokalle keskiarvovektori $\mathbf{m}_i$ (i viittaa luokkaan) ja kovarianssimatriisi $\mathbf{C}_i$ juuri siitä luokasta peräisin olevien opetusnäytteen vektorien perusteella

- Prioritodennäköisyydet estimoidaan tapauskohtaisesti (vrt. postinumerot)

- Järkevimmältä tuntuu MAP (maksimi-posteriori)-luokitus: erotinfunktiona käytetään posterioritodennäköisyyttä, eli luokittelusääntö: jos $P(\omega_j|\mathbf{x}) = max_i P(\omega_i|\mathbf{x})$, niin $\mathbf{x} \in \omega_j$

- Valitaan siis *todennäköisin luokka*.

- Bayesin kaavasta:

  jos $p(\mathbf{x}|\omega_j)P(\omega_j) = max_i p(\mathbf{x}|\omega_i)P(\omega_i)$, niin $\mathbf{x} \in \omega_j$

  koska termi $p(\mathbf{x})$ on kaikille sama ja voidaan tiputtaa pois vertailussa

- Jos prioritodennäköisyydet ovat samat, tulee vain

  jos $p(\mathbf{x}|\omega_j) = max_i p(\mathbf{x}|\omega_i)$, niin $\mathbf{x} \in \omega_j$

- Erotinfunktiona voidaan myös käyttää edellisten logaritmia, joka on usein laskennallisesti helpompi.

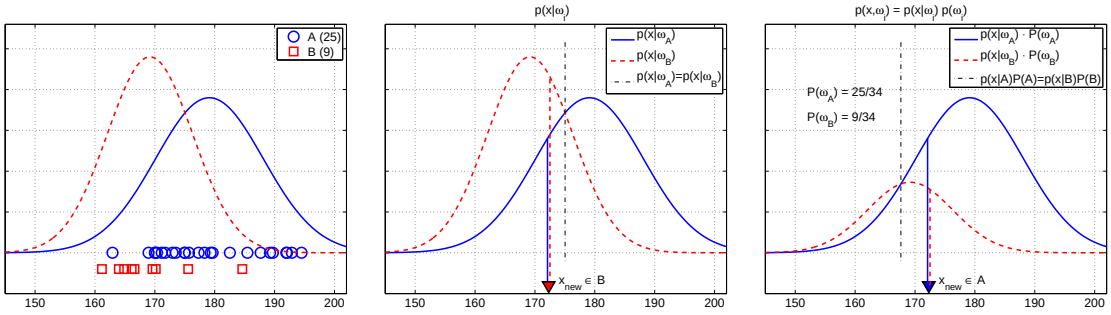
- Ero pelkkään suurimman uskottavuuden menetelmän (ML) antamaan luokkatiheysfunktioon $p(\mathbf{x}|\omega_j)$ on siis luokan prioritiedon hyväksikäyttö

- Kuvallinen esimerkki luokan priorin vaikutuksesta, kun mitattu 25 miehen pituus (sininen pallo) ja 9 naisen pituus (punainen neliö), ja luokiteltavana uusi havainto $x_{new} = 172$.

Bayes-luokitin normaalijakautuneelle datalle Millaiseksi sääntö muodostuu jos $p(\mathbf{x}|\omega_j)$:t ovat normaalijakautuneita?

- Muistetaan moniulotteisen normaalijakauman tiheysfunktio (Luku 5):

$$p(\mathbf{x}|\omega_j) = K_j \cdot \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{m}_j)^T \mathbf{C_j}^{-1}(\mathbf{x} - \mathbf{m}_j)\right) \tag{6.1}$$



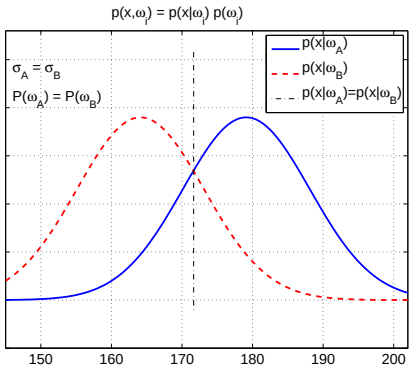
Kuva 6.4: (a) Opetusdata luokista A ja B sekä niistä estimoidut gaussiset todennäköisyysjakaumat $p(x|\omega_i)$. (b) ML-luokittelu käyttäen $p(x|\omega_i)$, jolloin $x_{\text{new}} = 172$ luokittuu B:ksi. (c) Bayes-optimaalinen luokitin $p(x|\omega_i) \cdot P(\omega_i)$, jossa luokkatodennäköisyydet $P(\omega_A) = 25/34$ ja $P(\omega_B) = 9/34$. Nyt $x_{\text{new}} = 172$ luokittuu A:ksi. Huomaa siis, että luokkaraja muuttuu.

missä $\mathbf{m}_j = (\mu_{1j}, \ldots, \mu_{dj})^T$ on keskiarvovektori (jakauman keskipiste, huippukohta) ja K_j on normalisoiva termi

$$K_j = \frac{1}{(2\pi)^{n/2} \det(\mathbf{C}_j)^{1/2}}$$

- Nähdään että jos kovarianssimatriisit ovat samat (tai oletetaan samoiksi), kaikki termit K_j ovat samat ja voidaan tiputtaa vertailussa pois.
- On sama verrataanko funktioiden $p(\mathbf{x}|\omega_j)$ vai niiden logaritmien arvoja (logaritmi on monotonisesti kasvava funktio)
- Luokittelu siis (olettaen myös luokkien priorit samoiksi): jos $[-(\mathbf{x} - \mathbf{m}_j)^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m}_j)] = max_i[-(\mathbf{x} - \mathbf{m}_i)^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{m}_i)]$, niin $\mathbf{x} \in \omega_j$
- Jos prioritodennäköisyydet eivät ole samat, tulevat niiden logaritmit myös mukaan
- Esimerkki: Katsotaan yksinkertaisuuden vuoksi 2 luokan tapausta:
- Kun kehitetään auki neliömuodot ja taas tiputetaan yhteiset osat pois, jäljelle jää yksinkertainen sääntö:

  jos $(\mathbf{m}_1 - \mathbf{m}_2)^T \mathbf{C}^{-1} \mathbf{x} > \mathbf{m}_2^T \mathbf{C}^{-1} \mathbf{m}_2 - \mathbf{m}_1^T \mathbf{C}^{-1} \mathbf{m}_1$ niin $\mathbf{x} \in \omega_1$, muuten $\mathbf{x} \in \omega_2$.
- Huomaa että vasen puoli on $\mathbf{x}$:n *lineaarinen* funktio, päätöspinta luokkien ω_1 ja ω_2 välillä on *hypertaso*.
- Esimerkki 1D:ssä:



Kuva 6.5: 1D-tapaus kahden luokan luokkatiheysfunktioista samalla keskihajonnalla $\sigma = \sigma_1 = \sigma_2$ ilman luokan prioria. Luokkarajaksi tulee symmetriasyistä hypertaso.

Bayes-luokitin binääridatalle Millainen sääntö binääridatalle?

- Ajatellaan seuraavaa tapausta: joukolle ihmisiä kohdistetaan mielipidekysely, joissa d kysymystä, ja kuhunkin pitää vastata "samaa mieltä" tai "eri mieltä".
- Koodataan yhden ihmisen vastaukset vektoriksi $\mathbf{x} = (x_1, ..., x_d)^T$ missä $x_i = 1$ jos samaa mieltä kysymyksen i kanssa, $x_i = 0$ jos eri mieltä.
- Kyselyn tekijä haluaa jakaa ihmiset kahteen luokkaan sillä perusteella, miten todennäköisesti on vastattu "samaa mieltä".
- Käytetään *Bernoulli-jakaumaa* $P(X = x_i) = p^{x_i}(1-p)^{1-x_i}$ parametrilla p
- Olkoot x_i:t riippumattomia toisistaan ja $P(x_i = 1|\omega_1) = p,\ P(x_i = 1|\omega_2) = q,\ p > q.$
- Oletetaan että etukäteisarviot luokkien suhteellisista suuruuksista (prioritodennäköisyydet) ovat $P(\omega_1),\ P(\omega_2)$.
- Mikä on Bayesin luokitin?
- Muodostetaan ensin luokkatiheysjakaumat $p(\mathbf{x}|\omega_1),\ p(\mathbf{x}|\omega_2)$.
- Mikä on esim. vektorin $\mathbf{x} = (11100101)$ todennäköisyys luokassa ω_1?
- Se on $ppp(1-p)(1-p)p(1-p)p$ (kysymysten riippumattomuus: $p(x_1, \ldots, x_d|\omega_1) = p(x_1|\omega_1) \cdot \ldots \cdot p(x_d|\omega_1)$)
- Huomaa että tämä voidaan kirjoittaa muotoon $\Pi_{i=1}^{8} p^{x_i}(1-p)^{1-x_i}$
- Tämä on siis $p(\mathbf{x}|\omega_1)$. Luokalle ω_2 vastaavasti mutta p:n tilalla q.

- Erotinfunktio $g_1(\mathbf{x}|\omega_1) = p(\mathbf{x}|\omega_1) \cdot P(\omega_1)$ luokalle ω_1, kun siitä otetaan logaritmi:

$$\begin{aligned}
\ln g_1(\mathbf{x}|\omega_1) &= lnp(\mathbf{x}|\omega_1) + lnP(\omega_1) \\
&= \sum_{i=1}^{d}[x_i lnp + (1-x_i)ln(1-p)] + lnP(\omega_1) \\
&= \sum_{i=1}^{d}[x_i ln\frac{p}{1-p} + ln(1-p)] + lnP(\omega_1) \\
&= ln\frac{p}{1-p}\sum_{i=1}^{d} x_i + dln(1-p) + lnP(\omega_1).
\end{aligned}$$

  Tästä on helppo muodostaa päätössääntö, kun prioritodennäköisyydet sekä p, q on annettu. Esim. jos priorit ovat samat, ne voi tiputtaa vertailussa pois; jos vaikkapa $p = 0.8, q = 0.5$ tulee vertailusäännöksi $g_1(\mathbf{x}|\omega_1) > g_2(\mathbf{x}|\omega_2)$

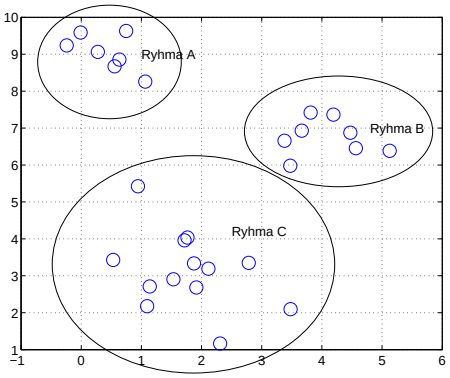
$$\sum_i x_i > 0.661d$$

  Tämän voi tulkita niin että "kuulut luokkaan yksi, jos vastasit 'samaa mieltä' vähintään 66.1 prosenttiin kysymyksistä".

6.4 Ryhmittelyanalyysi

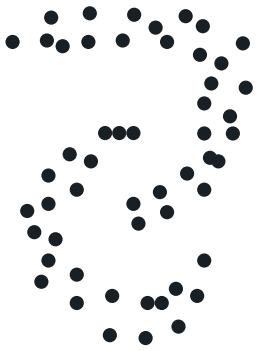
Ryhmittelyanalyysi

- Edellä on oletettu, että on olemassa luokiteltu opetusnäyte eli joukko luokaltaan tunnettuja vektoreita, joiden avulla luokitin (esim. lähimmän naapurin luokitin tai Bayes-luokitin) voidaan rakentaa.
- Monessa ongelmassa näin ei kuitenkaan ole: vektoreiden luokitteleminen "käsin" voi olla vaikeaa tai kallista.
- Esim. satelliittikuvien analyysissä on kuva-aineistoa käytettävissä valtavia määriä, mutta se ei ole luokiteltua.
- Silloin usein turvaudutaan *ryhmittelyanalyysiin* jolla pyritään löytämään samankaltaisten vektorien joukkoja tai ryppäitä aineistosta.
- Joukot vastaavat usein joitakin mielekkäitä luokkia, ja koska joukkoja on paljon vähemmän kuin vektoreita, niiden luokittelu on helpompaa.
- Esimerkki: 2D-datavektorit $\mathbf{x}(1), \ldots, \mathbf{x}(28)$ ryhmitellään kolmeen joukkoon. Soikiot on piirretty lopuksi visualisoimaan, mitkä datapisteet (indeksinumerot) kuuluvat mihinkin joukkoon



Kuva 6.6: Datan ryhmittely kolmeen joukkoon.

- $\Rightarrow$ Jokaisen joukon sisällä vektorit ovat samankaltaisia toistensa kanssa, kun ne taas poikkeavat toisten joukkojen vektoreista paljon
- Esimerkki "tulkinnallisesta tilanteesta": mikä olisi sopiva ryhmittelyn tulos?



Kuva 6.7: Miten ryhmittelisit tämän datan?

- Kurssilla esitetään kaksi tunnettua ryhmittelyalgoritmia: hierarkkinen ja (dynaaminen) c-means-ryhmittely
- Eri ryhmittelyalgoritmit voivat tuottaa hyvin erilaisen lopputuloksen samalle aineistolle

Ryhmittelyanalyysi Matemaattisia huomioita

- Meillä on taas n vektoria $\mathbf{x}(1), \ldots, \mathbf{x}(n)$ joiden dimensio on d (datamatriisin sarakkeet). Niillä *ei ole* nyt mitään luokkanimikkeitä
- Tehtävänä on löytää c ryhmää (joukkoa, rypästä, klusteria) $C_1, ..., C_c$ siten että niiden keskinäiset leikkaukset ovat tyhjät ja unioni on koko vektorijoukko

- Tyypillisesti siis tietty ryhmä C_i on joukko vektorien indeksejä
- Esim. jos $n = 25$ ja $c = 5$ (hyvin pieni ryhmittelyongelma), on erilaisten ryhmitysten lukumäärä 2.436.684.974.220.751. Ongelma on siis kombinatorisesti hyvin vaikea
- Ryhmittelyanalyysi on "ohjaamatonta luokittelua", koska ei ole "opettajaa" joka kertoisi oikeat luokat.

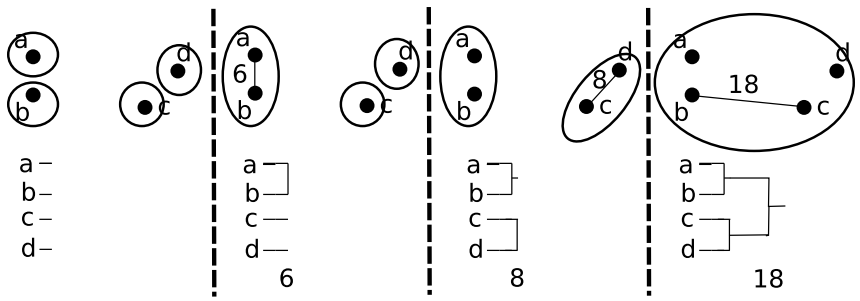
Ryhmittelyanalyysi Sovelluksia

- Sovelluksia:
  - Taksonomia biologiassa: mitä ovat eläin- ja kasvilajit
  - Lääketiede, biologia (potilasaineistot, geenipankit)
  - Yhteiskuntatieteet (kansanluokat, nelikentät)
  - Kuva-analyysi (satelliittikuvat, visuaalinen laadunvalvonta)
  - jne.

6.5 Hierarkkinen ryhmittely

Hierarkkinen ryhmittely Algoritmi

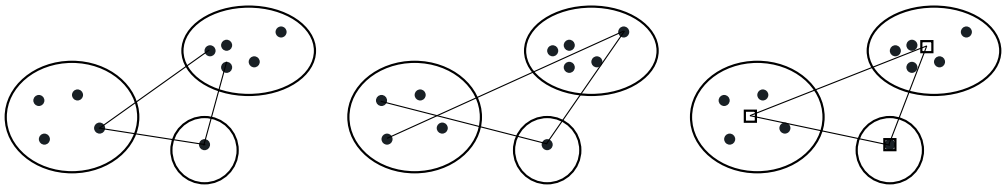
- Idea on hyvin yksinkertainen: aluksi kukin vektori muodostaa oman ryhmänsä. Niitä on siis n kpl
- Sitten yhdistetään ne ryhmät jotka ovat *lähinnä toisiaan*
- Tätä jatketaan kunnes kaikki vektorit ovat samassa ryhmässä.
- Ryhmittely voidaan esittää hierarkkisena ryhmittelypuuna
- Puun voi *katkaista* sopivalta tasolta jos esim. tiedetään montako ryhmää halutaan, tai jossakin vaiheessa lähimpien ryhmien etäisyys kasvaa isoksi (jolloin niiden yhdistäminen ei enää tunnu järkevältä).
- Edellisessä esimerkissä ryhmittelypuussa oli hyppy yhden ja kahden klusterin välissä: kaksi klusteria olisi luonnollinen valinta



Kuva 6.8: Hierarkkinen ryhmittely käyttäen lyhintä etäisyyttä. Alussa neljä datapistettä on neljässä ryhmässä. Yhdistetään kaksi lähintä ryhmää etäisyydellä/kustannuksella 6. Piirretään ryhmittelypuuta. Jatketaan kunnes kaikki pisteet yhdessä ryhmässä.

Hierarkkinen ryhmittely Ryhmien yhdistäminen

- Oleellinen kysymys silloin on: mikä on kahden ryhmän C_i, C_j välinen etäisyys?
- Kolme suosituinta mahdollisuutta
  1. *Lyhin* ("single") etäisyys vektorien $\mathbf{x} \in C_i$ ja $\mathbf{y} \in C_j$ välillä
  2. *Pisin* ("complete") etäisyys vektorien $\mathbf{x} \in C_i$ ja $\mathbf{y} \in C_j$ välillä.
  3. Ryhmien *keskipistevektorien* ("average") välinen etäisyys



Kuva 6.9: Ryhmien välisen etäisyyden määräytyminen: "single", "complete", "average"

- Jokaisella on tietty vaikutus ratkaisuna tulevien ryhmien muotoon. Lyhin etäisyys suosii "pitkulaisia" ryhmiä, pisin etäisyys taas "pallomaisia" ryhmiä, keskipisteiden etäisyys on näiden väliltä Esimerkki etäisyysmitan vaikutuksesta
- Huomaa että kahdessa jälkimmäisessä tavassa ryhmittely voidaan tehdä jopa tuntematta vektoreita $\mathbf{x}(i)$, kunhan tiedetään kaikki niiden väliset etäisyydet $d[\mathbf{x}(i), \mathbf{x}(j)]$ (vierekkäisyysmatriisi). Siten menetelmät sopivat myös muunlaiselle tiedon esitykselle kuin vektoreille (merkkijonot, puut, graafit, ...), kunhan etäisyydet voidaan laskea.
- Kuitenkin ryhmien keskipistevektoreita käyttävä menetelmä edellyttää että keskipiste on laskettavissa, ja tällöin vektorit $\mathbf{x}(i)$ on tunnettava.

Dynaaminen ryhmittely

Dynaaminen ryhmittely

- Tämä on eräs klassisimpia ohjaamattomia koneoppimismenetelmiä
- Nyt täytyy tietää (arvata, kokeilla) ryhmien lukumäärä c
- Kutakin ryhmää C_i edustaa laskennallinen keskipistevektori $\mathbf{m}_i$
- Ryhmä C_i määritellään joukkona

$$C_i = \{\mathbf{x} | \|\mathbf{x} - \mathbf{m}_i\| \leq \|\mathbf{x} - \mathbf{m}_j\|, j \neq i\}$$

  eli niiden vektoreiden joukko jotka ovat lähempänä C_i:n omaa keskipistettä kuin mitään muuta keskipistettä.

- Järkevä ryhmittelykriteeri on nyt:

$$minimoi\ J = \sum_{i=1}^{c} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = \sum_{i=1}^{c} J_i$$

- Osasumma J_i mittaa ryhmän C_i sisäistä varianssia eli kuinka lähellä keskipistettä vektorit $\mathbf{x}$ ovat.
- Ongelma: kuinka J minimoidaan?
- Huomaa että J on siis ryhmittelyn funktio: pisteet $\mathbf{x}$ eivät "liiku" minnekään, niitä vain allokoidaan eri ryhmiin niin että kriteeri minimoituu. Kun yksikin piste siirtyy ryhmästä toiseen, kriteerin J arvo muuttuu.
- Kombinatorinen ratkaisu (kokeillaan kaikki mahdolliset ryhmittelyt ja valitaan se jossa J on pienin) ei ole laskennallisesti mahdollinen
- Mitä siis voidaan tehdä?

6.6 c-means ryhmittelyalgoritmi (k-means, KM)

c-means ryhmittelyalgoritmi (k-means, KM)

- Tunnetuin dynaaminen ryhmittelymenetelmä on ns. *c-means* -algoritmi (eli c:n keskipisteen algoritmi; vaihtoehtoinen nimi k-means, KM):
  1. Valitse satunnaisesti c pistettä joukosta $\mathbf{x}(1), ..., \mathbf{x}(n)$ pisteiksi $\mathbf{m}_1, ..., \mathbf{m}_c$;
  2. Toista kunnes pisteet $\mathbf{m}_1, ..., \mathbf{m}_c$ eivät muutu

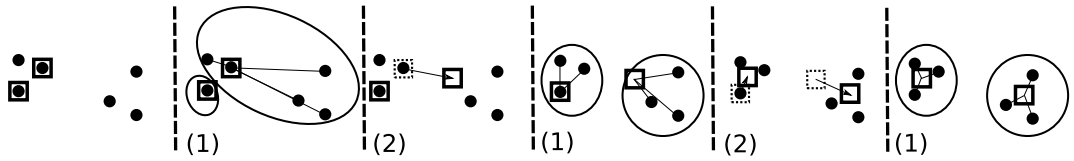
– Sijoita kukin piste $\mathbf{x}(1), ..., \mathbf{x}(n)$ siihen ryhmään C_i jonka keskipiste $\mathbf{m}_i$ on lähinnä;

– Laske uudet keskipisteet kaavalla

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

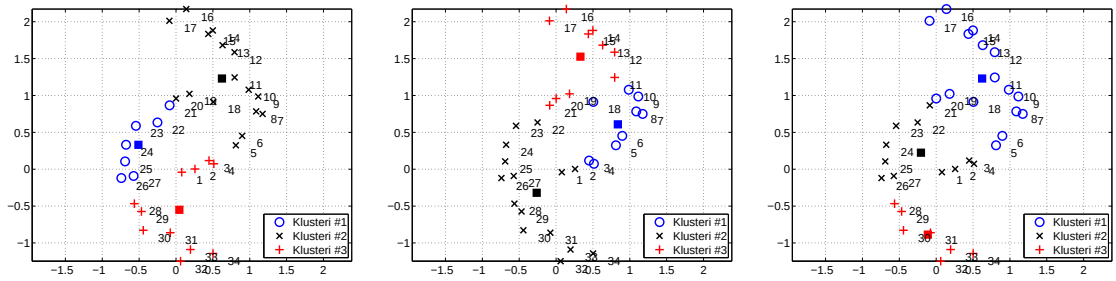
missä n_i on C_i:n pisteiden lukumäärä.

- Algoritmi kuvallisesti:



Kuva 6.10: Valittu $c = 2$, keskiarvopisteet $\mathbf{m}_i$ (neliö) valittu pisteiden joukosta satunnaisesti. Algoritmissa vuoroin (1) sijoitetaan pisteet (pallo) siihen ryhmään, jonka keskiarvovektori (neliö) on lähinnä, ja (2) lasketaan uusi keskiarvopiste $\mathbf{m}_i$ (katkoviivaneliö $\rightarrow$ neliö). Kun muutoksia ryhmiin ei tule, suoritus päättyy.

- Koska aloituspisteet arvotaan, voi eri ajokerroilla tulla erilaisia tuloksia



Kuva 6.11: Valittu $c = 3$, keskiarvopisteet $\mathbf{m}_i$ neliöinä. Kolme eri ajoa eri alkupisteillä.

- Algoritmiin löytyy demoja, esim. `http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html`

c-means ryhmittelyalgoritmi (k-means, KM) Mitä algoritmissa tapahtuu?

- Ajatellaan kriteeriä

$$J = \sum_{i=1}^{c} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = \sum_{i=1}^{c} J_i$$

ja katsotaan mitä tapahtuu jos siirrämme *yhden vektorin* $\mathbf{x}$ ryhmästä i ryhmään k

- Ainoastaan kaksi osasummista muuttuu: J_i ja J_k, koska kaikki muut ryhmät säilyvät ennallaan
- Niiden uudet arvot ovat

$$J_i^{uusi} = J_i - \|\mathbf{x} - \mathbf{m}_i\|^2 \quad (6.2)$$
$$J_k^{uusi} = J_k + \|\mathbf{x} - \mathbf{m}_k\|^2. \quad (6.3)$$

- Koko kriteeri J *pienenee* jos

$$J_i^{uusi} + J_k^{uusi} < J_i + J_k$$

  joka tapahtuu jos ja vain jos

$$\|\mathbf{x} - \mathbf{m}_k\| < \|\mathbf{x} - \mathbf{m}_i\|.$$

- Siis aina kun jokin vektori siirtyy ryhmään jonka keskipiste on *lähempänä* kuin vanhan ryhmän keskipiste, kriteeri J pienenee
- Mutta juuri näinhän c-means-algoritmissa tapahtuu.
- Siis kriteeri J pienenee aina kun ryhmittely muuttuu
- (Lisäksi pitää todistaa että myös algoritmin 2. vaihe eli keskipisteiden päivitys pienentää kriteeriä: harjoitustehtäväksi.)
- Koska J on positiivinen, se ei voi pienentyä ikuisesti vaan sen on supettava johonkin arvoon.
- Algoritmi ei takaa että löytyisi kaikkein paras ryhmittely, jossa J:llä on kaikkein pienin arvonsa, vaan algoritmi suppenee *paikalliseen minimiin*
- Silti se on käytännössä hyvä ja suosittu menetelmä.

c-means ryhmittelyalgoritmi (k-means, KM) Yhteys koodausteoriaan

- c-means-algoritmilla on yhteys koodausteoriaan: ryhmäkeskipisteet $\mathbf{m}_i$ voidaan ajatella *koodeiksi* eli kompressoiduksi esitykseksi kaikista ryhmän C_i vektoreista
- Jos sekä lähettäjällä että vastaanottajalla on tiedossa *koodikirja* eli vektoreiden $\mathbf{m}_i$ arvot, voi vektoreiden $\mathbf{x}$ koodauksen - dekoodauksen hoitaa seuraavasti:
  1. Lähettäjä etsii vektoria $\mathbf{x}$ lähimmän koodivektorin $\mathbf{m}_i$ ja lähettää vain sen indeksin i (koodaus)
  2. Vastaanottaja etsii indeksiä i vastaavan koodivektorin $\mathbf{m}_i$ ja käyttää sitä $\mathbf{x}$:n tilalla (dekoodaus)
- Jos vektorit $\mathbf{x}$ ovat isoja, on bittien säästö hyvin merkittävää
- Haittapuolena on se että dekoodauksessa syntyy virhe $\mathbf{x} - \mathbf{m}_i$
- Virhe on sitä pienempi mitä pienempi on kriteeri J ja mitä enemmän koodivektoreita käytetään.

Yhteenveto

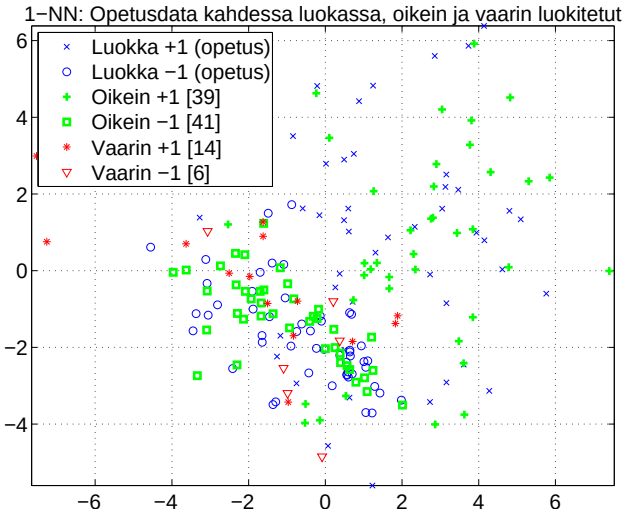
Yhteenveto

Tässä luvussa käsiteltiin luokittelua ja ryhmittelyä. Luokittelussa tunnemme hahmovektorien luokan, minkä avulla luokitin (erotinfunktio, hahmoalue) muodostetaan. Tämän jälkeen luokitin antaa uudelle datapisteelle luokkatiedon. Algoritmeista esiteltiin lähimmän naapurin luokitin (kNN) ja Bayes-optimaalinen luokitin jatkuvalle ja binääridatalle.[3mm]

Ryhmittelyssä luokkainformaatiota ei ole. Datasta muodostetaan ryppäitä (klustereita), joissa hahmovektorit ovat keskenään samankaltaisia. Ne voidaan sitten käsitellä “luokkina” tai niiden “koodivektorien” avulla voidaan vähentää datan määrää. Luvussa esiteltiin hierarkkinen ryhmittely ja c-means-ryhmittelyalgoritmi.

Liite: Kuvia esitykseen, luku 6

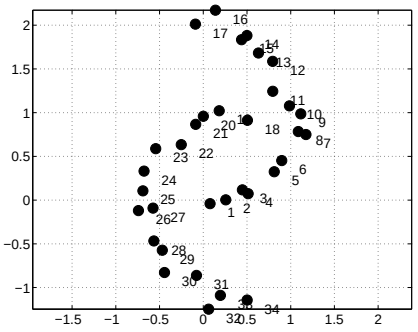
Esimerkki luokitteluvirheestä Opetusdatasta hahmoalueet kNN-luokittimelle, k=1



Kuva 6.12: kNN-luokitin kun $k = 1$. Kahden luokan opetusaineiston x ja o sinisellä. Testiaineiston 100 näytettä: “vihreä plus” luokiteltu rastiksi oikein, “vihreä neliö” luokiteltu oikein palloksi, “punainen tähti” on rasti joka luokiteltu virheellisesti palloksi, “punainen kolmio” on ympyrä joka luokiteltu virheellisesti rastiksi. Luokitteluvirhe (14+6)/100 = 20%. Takaisin kalvoihin

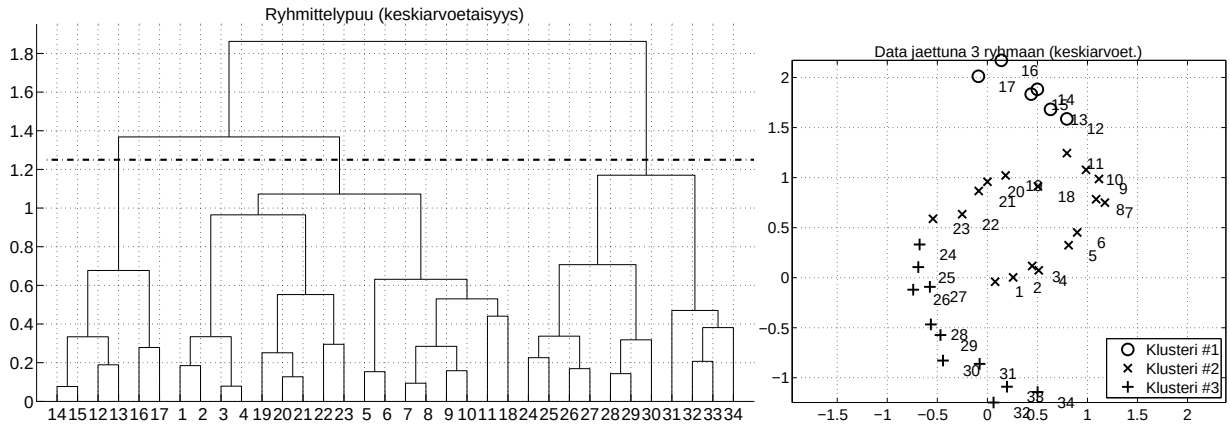
Hierarkkinen ryhmittely: etäisyysmitan vaikutus

Olkoon käytössä kuvan mukainen data:

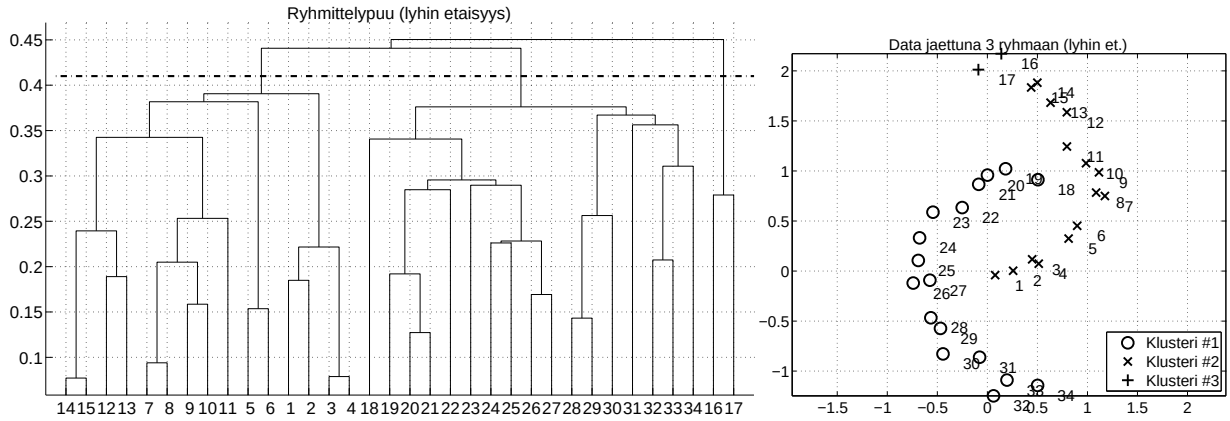


Kuva 6.13: Ryhmittelyesimerkin data.

Tehdään tälle datalle hierarkkinen ryhmittely käyttäen ryhmien välisen etäisyydet mittaamiseen (a) keskiarvomittaa (“average”), (b) pienintä (“single”), (c) suurinta (“complete”) etäisyyttä. Ryhmittelyn jälkeen katkaistaan puu kolmen ryhmän (oksan) kohdalta ja piirretään muodostuneet ryhmät.

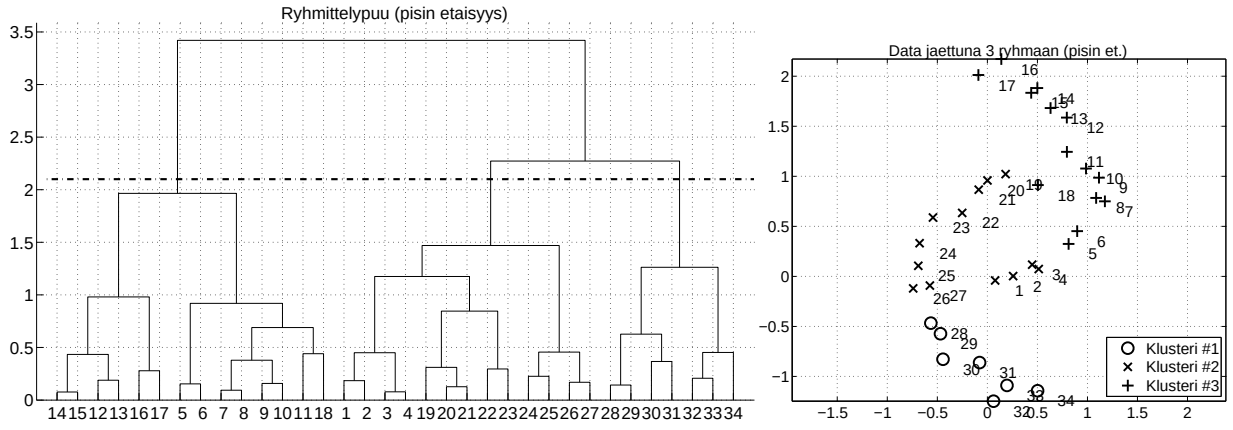


Kuva 6.14: Keskiarvoetäisyys (“average”): Pyöreähköt ryhmät.



Kuva 6.15: Lyhin etäisyys (“single”): Pitkulaiset ryhmät.

Takaisin kalvoihin



Kuva 6.16: Pisin etäisyys (“complete”): Pyöreähköt ryhmät

7 Itseorganisoiva kartta

7.1 Itseorganisoiva kartta (SOM)

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.
- 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

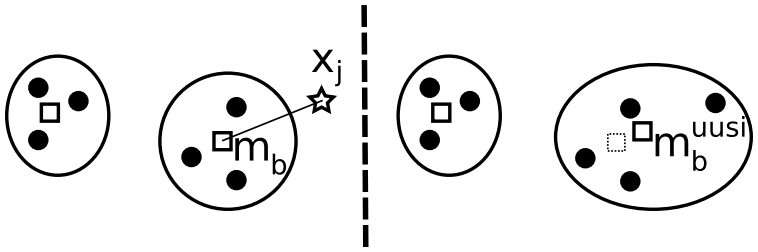
  $$b = b(\mathbf{x}_j) = argmin_{i=1,\ldots,c} \|\mathbf{x}_j - \mathbf{m}_i\|$$

  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-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:

$$\begin{aligned}
\mathbf{m}_b^{uusi} &= \frac{1}{n_b+1}[\sum_{\mathbf{x}\in C_b} \mathbf{x} + \mathbf{x}_j] && (7.1)\\
&= \frac{1}{n_b+1}[\mathbf{x}_j + n_b\mathbf{m}_b] && (7.2)\\
&= \frac{1}{n_b+1}[\mathbf{x}_j + (n_b+1)\mathbf{m}_b - \mathbf{m}_b] && (7.3)\\
&= \mathbf{m}_b + \frac{1}{n_b+1}[\mathbf{x}_j - \mathbf{m}_b] && (7.4)
\end{aligned}$$

- 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 7.1: c-means-algoritmin päivitysaskel

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

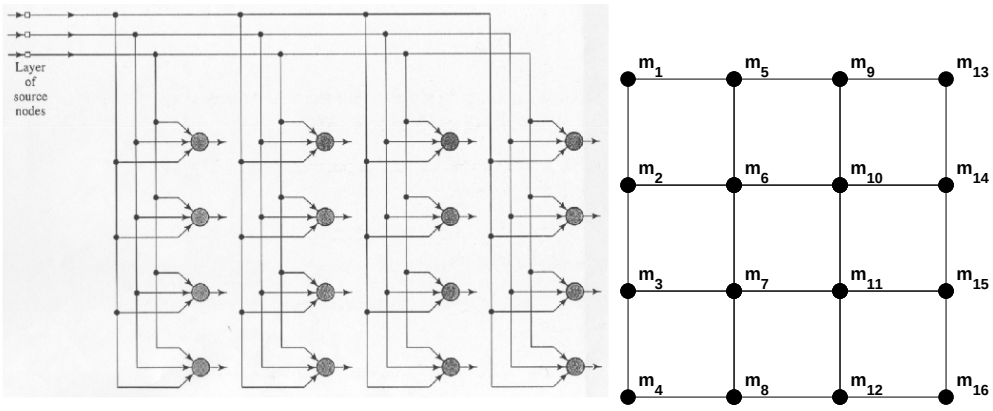
SOM-algoritmin perusidea Voittajan yhteistyö naapurien kanssa

- 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?



Kuva 7.2: Kaksiulotteinen SOM-verkko: neuronin m_{11} naapureita ovat m_7, m_{10}, m_{12} ja m_{15}

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*
- 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).
- 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

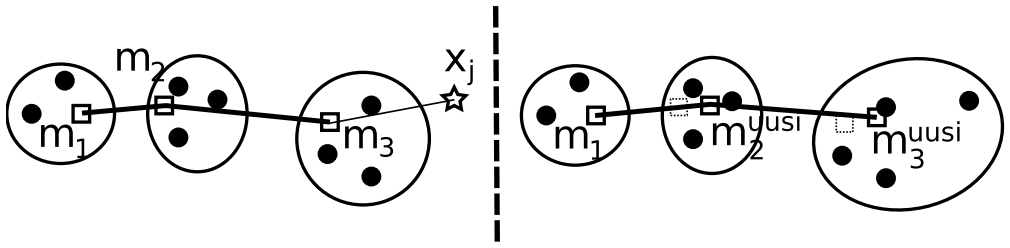
- Entäs *toinen kysymys: Miten naapureiden keskipistevektorit liikkuvat?*

- Merkitään kaavassa (7.4) olevaa kerrointa $1/(n_b + 1)$ merkinnällä α, ja nyt saadaan SOM-päivityskaava painovektoreille:

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

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

- SOM 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 7.3: SOM-kartan (1D) päivittyminen: sekä voittaja että sen *naapuri* päivittyvät

- 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?*

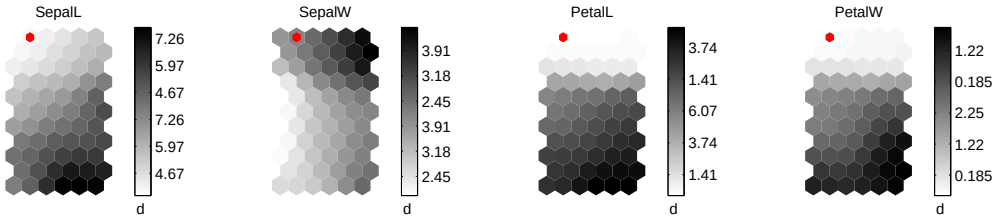
- Kaksi asiaa:

  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



Kuva 7.4: 2-ulotteisen SOM-kartan neljä kompenttitasoa (4-ulotteiset painovektorit), kun syötteenä 4-ulotteista dataa. Karttahila ja painovektorit komponenteittain

  - näyttämällä itse SOM-karttahilan ja kussakin neuronissa sen painovektorin
  - 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

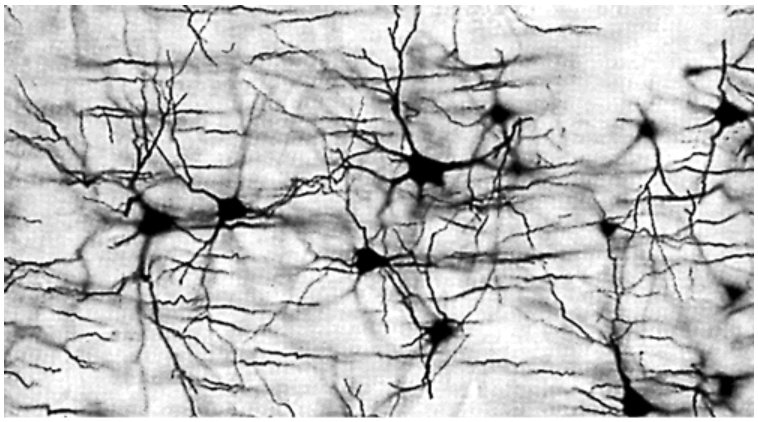
7.2 Itseorganisoivan kartan yhteys biologiaan

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

7.3 Itseorganisoivan kartan suppenevuus 1-ulotteisessa tapauksessa

Suppenevuus 1-ulotteisessa tapauksessa



Kuva 7.5: Hermosoluja

- Katsotaan nyt edellä esitettyä SOM-algoritmia:

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

- 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.

- 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.

- 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

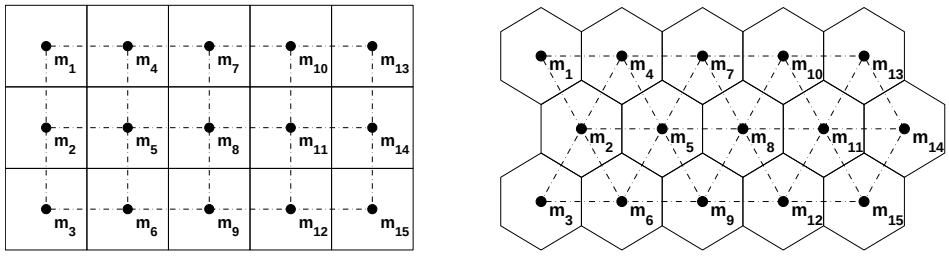
$$\begin{aligned} m_{b-1}^{uusi} - m_{b-2}^{uusi} &= m_{b-1} + \alpha(x - m_{b-1}) - m_{b-2} \\ &= m_{b-1} - m_{b-2} + \alpha(x - m_{b-1}) \end{aligned}$$

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 $m_{b+2}^{uusi} - m_{b+1}^{uusi} > 0$.

7.4 Käytännön valintoja

Käytännön valintoja

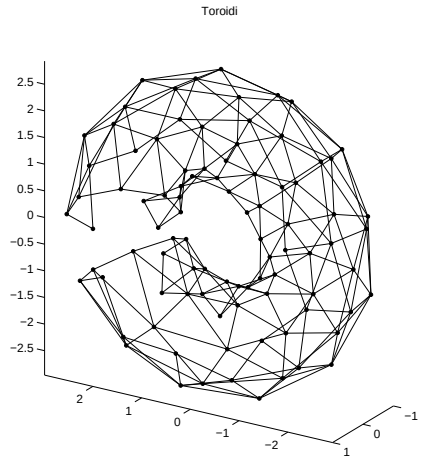
- Yleisimmät hilat ovat 1- tai 2-dimensioisia, koska kartta on silloin helppo visualisoida
- 2-dimensioisella hilalla on kaksi perusrakennetta: *suorakaidehila* ja *kuusikulmiohila*



Kuva 7.6: SOM:n suorakaidehila ja kuusikulmiohila

- Kuusikulmio- eli heksagonaalihilan etu on se, että kullakin yksiköllä on 6 naapuria jotka kaikki ovat yhtä etäällä siitä
- Reunoilla naapurusto on epätäydellinen, paitsi jos hila ajatellaan kierretyksi "munkkirinkeliksi" (toruspinnaksi), jolloin se on jatkuva eikä mitään reunoja ole
- 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

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



Kuva 7.7: SOM-hila “munkkirinkilinä”

kirjoitetaankin tämä seuraavasti:

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

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

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

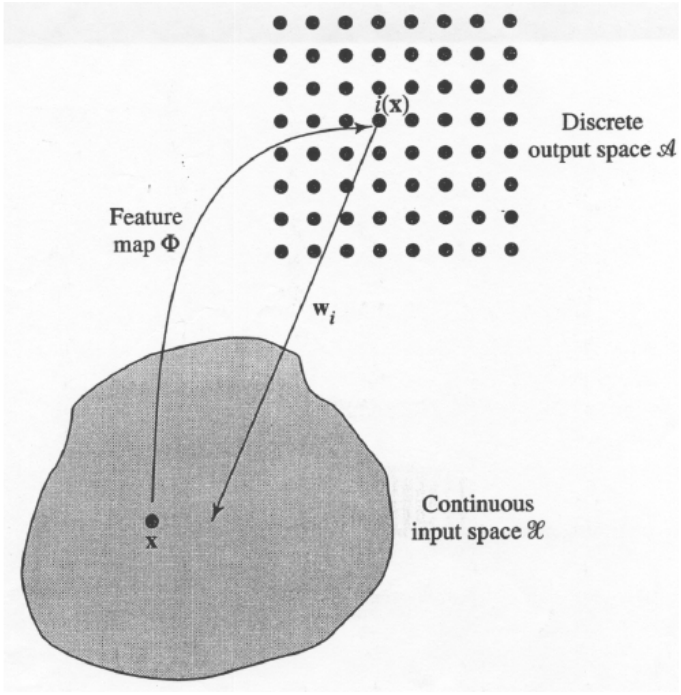
- Jotta kaavat olisivat aivan yhdenpitävät, on oltava myös $h(b,b) = 1$ eli pitää ajatella että b itsekin kuuluu omaan naapurustoonsa
- Nyt naapuruston valintaa voidaan “pehmentää” valitsemalla funktioksi $h(k,b)$ sileämpi funktio, esim. gaussinen $h(k,b) = e^{-\beta d^2(k,b)}$ missä $d(k,b)$ on yksikön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

7.5 Mihin SOM:ia käytetään?

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 R^d ja karttahilan välille



Kuva 7.8: SOM: kuvaus syöteavaruuden ja karttahilan välillä

- Kukin syöteavaruuden piste $\mathbf{x}$ (esim. kukin opetusjoukon vektori) kuvautuu hilaan siihen yksikköön joka on $\mathbf{x}$: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 $\mathbf{x}$
- 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
- 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 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.

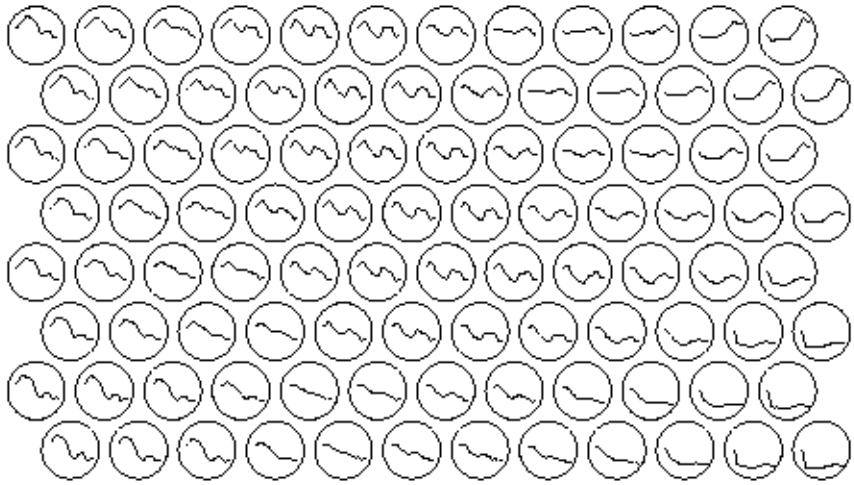
Yhteenveto

Tässä luvussa esiteltiin itseorganisoiva kartta (SOM), jota voidaan käyttää korkeaulotteisen datan ryhmittelyyn ja visualisointiin.

SOM:n keksijä on akateemikko Teuvo Kohonen.

Liite: Kuvia esitykseen, luku 7

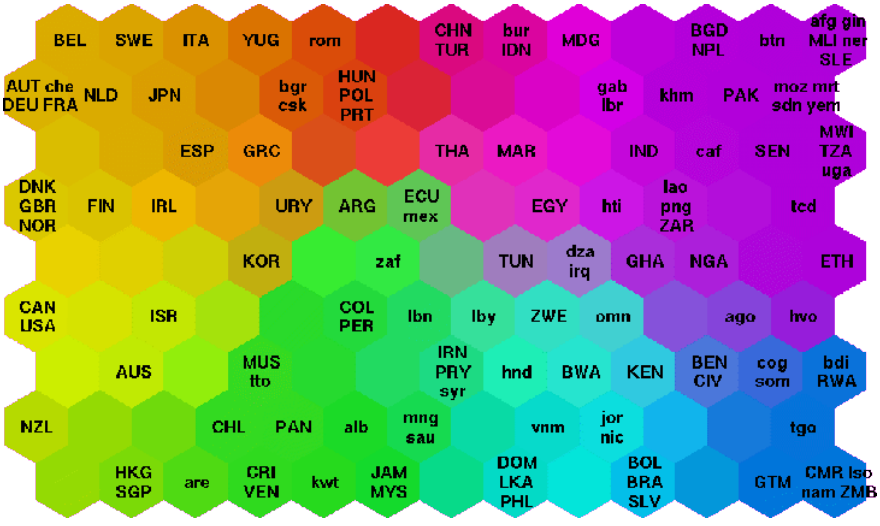
“Foneemikirjoitin”



Kuva 7.9: Erilaiset puheen spektrit kuvautuvat SOM-karttapinnalle. Neuronissa on piirrettynä äännettä vastaava spektrikuvio.

Takaisin kalvoihin

“Köyhyyskartta”

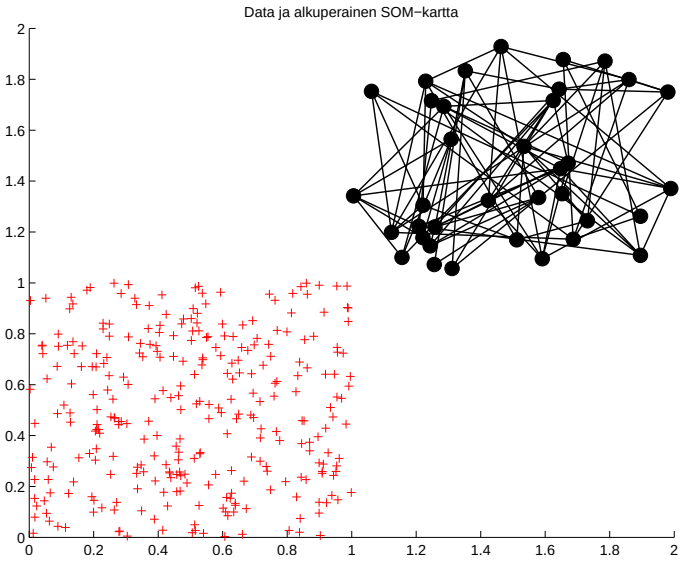


Kuva 7.10: Valtioiden taloudelliset indikaattorit SOM-karttapinnalla. Korkeaulotteinen data on saatu visualisoitua mielekkääksi 2-ulotteiseksi kuvaksi. Vierekkäisiin neuroneihin kuvautuu samankaltaisia datavektoreita (maita).

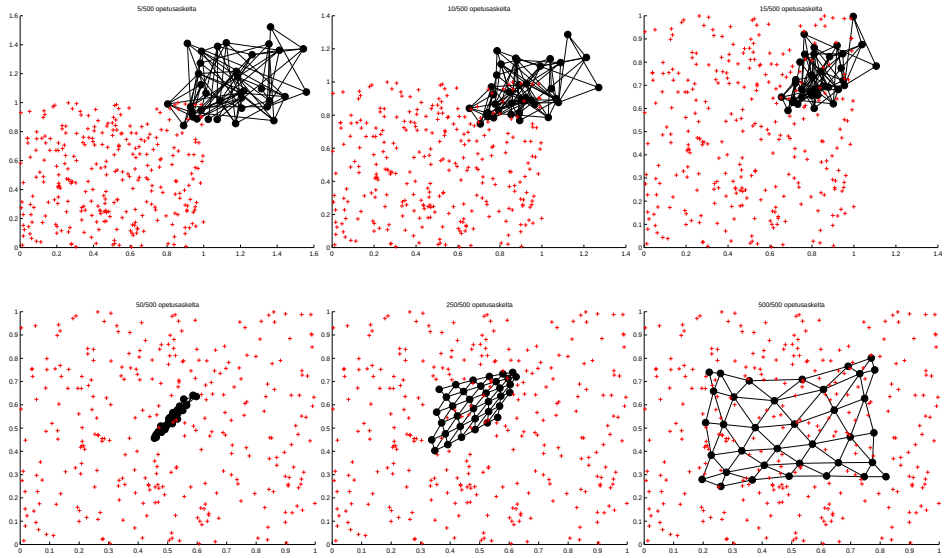
Takaisin kalvoihin

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

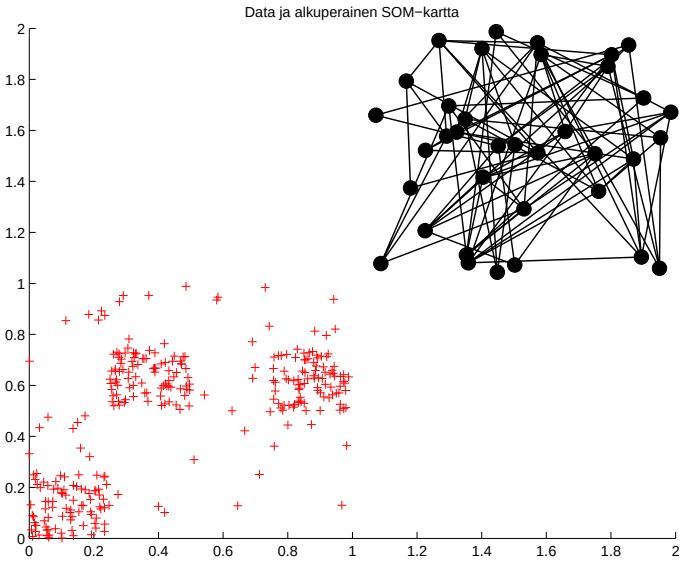
Alkutilanne: datanäytteitä (+) yksikköneliössä ja 6×6 -kokoinen suorakulmaisen hilan SOM-kartta järjestäytymättömänä. Mustat pallot painovektoreita $\mathbf{m}_j$.



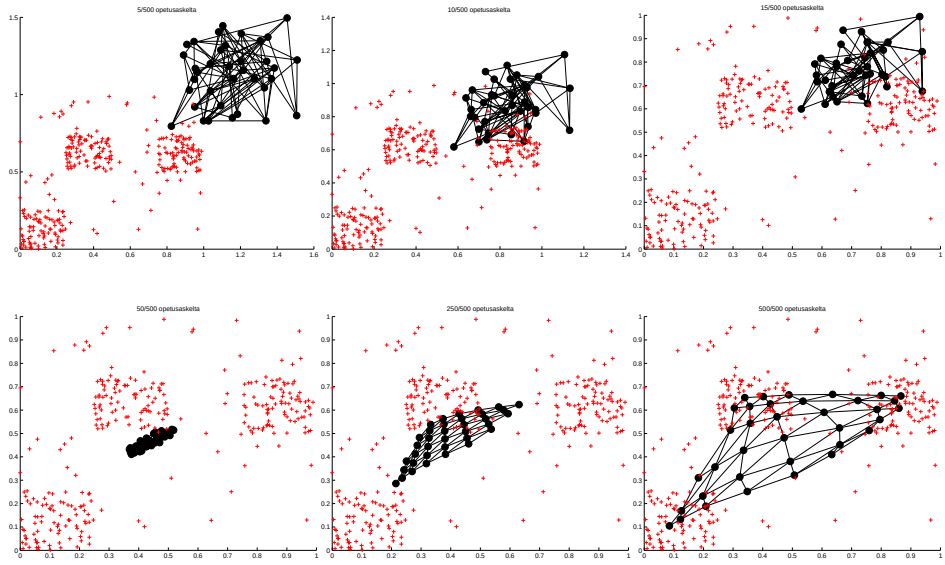
Opetuksen aikana kartta levittäytyy ja järjestyy opetusjoukon alueelle. Kuvat 5, 10, 15, 50, 250 ja 500 kierroksen jälkeen.



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



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



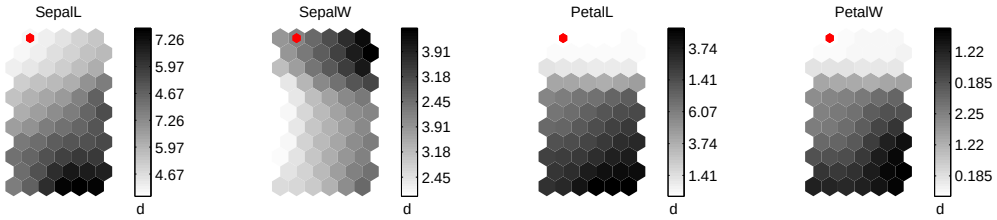
Lähde: SOM Toolboxin `som_demo1`. Takaisin kalvoihin

SOM:n komponenttitasoesitys

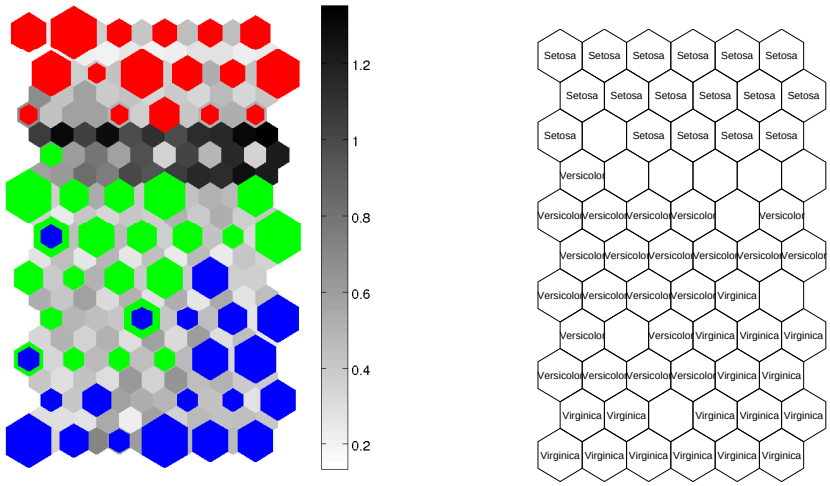
Datana tässä esimerkissä on käytetty klassista Fisherin liljadataa (Iris), jossa 150 neliulotteista havaintoa kolmesta liljalajista (’Setosa’,’Virginica’, ’Versicolor’). Mitatut ominaisuudet aluslehden (’Sepal’) pituus (’L’) ja leveys (’W’) sekä terälehden (’Petal’) vastaavat.

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

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



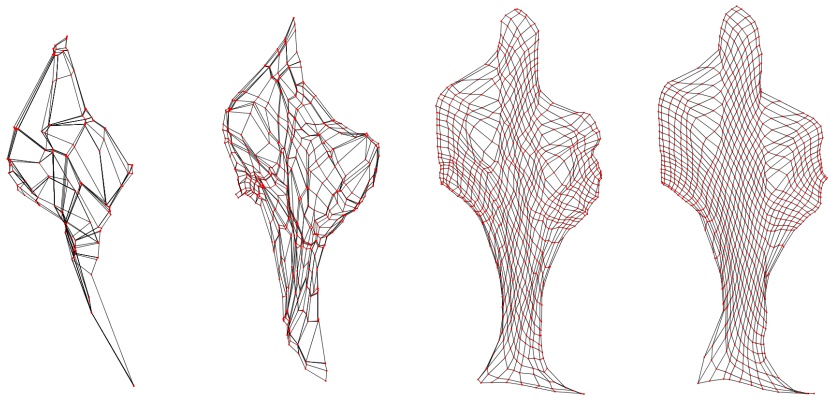
Painovektorikartat komponenteittain eli ns. komponenttitasot. Yhden painovektorin $\mathbf{m}_{12} \in R^4$ kohta on osoitettu punaisella pisteellä. Kyseisen kohdan arvot ovat pieniä muuttujille 'SepalL', 'PetalL' ja 'PetalW', mutta verrattain suuri muuttujalle 'SepalW'.



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

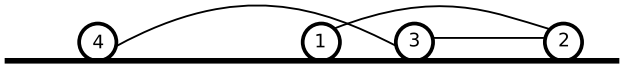
Elastinen verkko: "Kohosen kaktus"

Takaisin kalvoihin

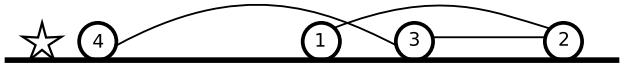


Kuva 7.11: “Kohosen kaktus”: kolmiulotteinen data ja kaksiulotteinen SOM. Kartta pyrkii täyttämään kolmiulotteisen muodon taipumalla datan mukaan.

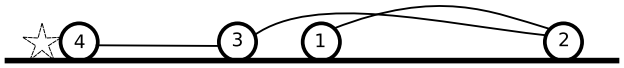
1D-kartan järjestyminen sopivalla 1D-datalla x



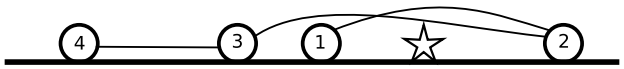
Yksiulotteiset painovektorit m_4, m_1, m_3 ja m_2 suoralla. Tavoitteena etsiä sellainen datasyöte x_1, x_2, . . . että painovektorit tulevat järjestykseen.



Opetusvektori (piste) x vasemmalla. Lähimpänä m_4.



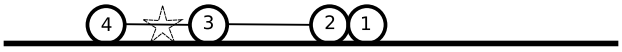
Voittaja m_4 ja sen naapuri m_3 liikkuvat.



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

8 Diskreettejä menetelmiä laajojen 0-1 datajoukkojen analyysiin

Kurssin loppuosa

- Kurssin alkuosassa muuttujat olivat pääsääntöisesti reaalilukuarvoisia. Kahdessa viimeisessä luvussa tarkastellaan *diskreettejä hahmoja*.
- Luku 8: Diskreettejä menetelmiä laajojen 0-1 datajoukkojen analyysiin
  - Kattavat joukot ja niiden etsintä tasoittaisella algoritmilla
- Luku 9: Relevanttien sivujen etsintä verkosta: satunnaiskulut verkossa
  - Linkkikeskukset ja auktoriteetit ("hubs and authorities") -algoritmi
- Kutakin asiaa vain raapaistaan; kaikki merkittäviä tutkimusalueita

8.1 Suuret 0-1 datajoukot

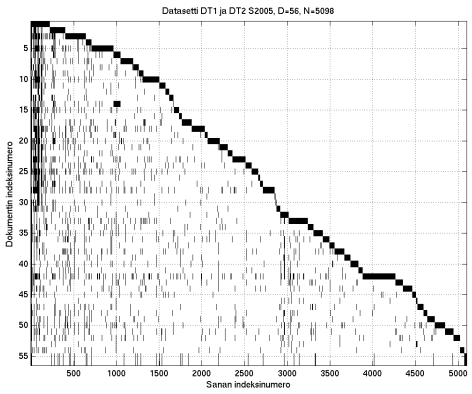
Suuret 0-1 datajoukot

- Paljon rivejä (muuttujia), paljon sarakkeita (havaintoja)
- 0-1 dataa: esiintymädataa. Esiintyykö jokin tietty ilmiö tietyssä havainnossa?
- Matriisin (taulun) D alkio $D(m, h)$ kertoo, esiintyykö muuttujan m kuvaama ilmiö havainnossa h eli $D(m, h) = 1$ tai $D(m, h) = 0$

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots & 1 & 0 \end{bmatrix}$$

0-1-datan esimerkki Sanojen esiintyminen dokumenteissa

- Havainto: dokumentti
- Muuttuja: (perusmuodossa oleva) sana
- Data $D(s, d) = 1$: esiintyikö sana s ainakin kerran dokumentissa d?

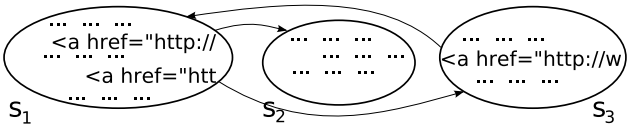


Kuva 8.1: Sana-dokumentti-matriisi tietokoneharjoituskierrokselta 5. Musta juova tarkoittaa, että sana s on dokumentissa d eli $D(s,d) = 1$. Mukana syksyllä 2005 palautettujen harjoitustöiden (kaksi aihetta) tekstiaineisto: 5098 eri sanaa ja 56 dokumenttia, joista kaksi tahallisesti generoituja ("kopioita").

0-1-datan esimerkki Webbisivut

- Havainto: webbisivu
- Muuttuja: webbisivu
- $D(s,p) = 1$ jos ja vain jos sivulta s on linkki sivulle p

(s,p)	s_1	s_2	s_3
s_1	0	1	1
s_2	0	0	0
s_3	1	0	0



0-1-datan esimerkki Nisäkäslajien esiintyminen luonnossa

- Havainto: 50x50 km ruutu r
- Muuttuja: Nisäkäslaji l
- $D(l,r) = 1$ jos ja vain jos laji l esiintyi ruudussa r

0-1-datan esimerkki Osamolekyylien esiintyminen molekyyleissä

- Havainto: Molekyyli m (tyypillisesti suuri)
- Muuttuja: Molekyyli p (pienempi)
- $D(p, m) = 1$ jos ja vain jos molekyyli p on osa molekyyliä m

NSF 0-1-dokumenttiesimerkki

NSF 0-1-dokumenttiesimerkki Lähdeaineisto

- Abstraktitietokanta: 128000 abstraktia jotka kuvaavat NSF:n rahoittamia perustutkimushankkeita `http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html`
- Esimerkki abstraktin tietokentistä
- Esimerkki abstraktista

NSF 0-1-dokumenttiesimerkki Datan esitys

- Tekstidokumentit (abstraktit) muutettuna 0-1-taulukkomuotoon, jossa 1 = esiintyy ainakin kerran
- Sarakkeet vastaavat dokumentteja
- Rivit vastaavat sanoja
- Esimerkkisanoja indekseineen: . . ., 73:abscissa, 74:absence, 75:absent, 76:absolute, 77:absolutely, 78:absorb, 79:absorbed, . . .

	#1	#2	#3	. . .
327:additional	1	0	0	. . .
⋮	⋮	⋮	⋮	⋮
11339:genetic	1	1	0	. . .
⋮	⋮	⋮	⋮	⋮
26457:studies	1	1	1	. . .
⋮	⋮	⋮	⋮	⋮

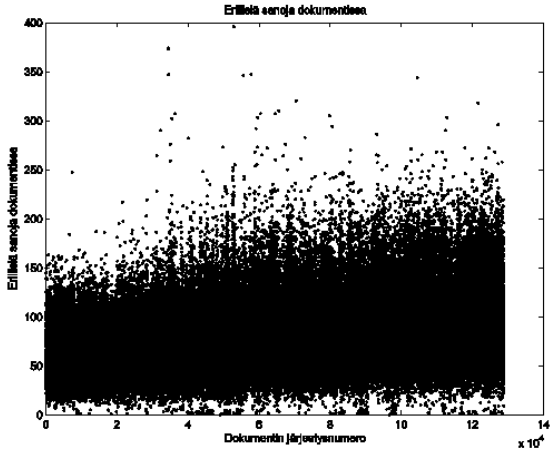
NSF 0-1-dokumenttiesimerkki Miksi tällainen esitysmuoto?

- Tiedonhaun kannalta dokumentin sisältävien sanojen luettelo on monesti riittävä
- Määrämuotoista dataa on helpompi käsitellä kuin vaihtelevan mittaisia tekstijonoja

NSF 0-1-dokumenttiesimerkki Perustilastoja

- Dokumentteja: 128000, sanoja: 30800; ei mitenkään erityisen suuri dokumenttijoukko
- Taulussa $30800 \times 128000 = 3942400000 \approx 4 \cdot 10^9$ alkiota
- Ykkösiä datassa 10449902 eli $0.002650 = 0.265$ % kaikista alkioista; ei kannata esittää nollia
- Noin 81 erilaista sanaa / dokumentti (1 = sana esiintyy ainakin kerran)
- Kukin sana esiintyy keskimäärin 340 dokumentissa
- Jakauma vino: jotkin sanat esiintyvät usein, toiset hyvin harvoin
- Joitakin useimmin esiintyviä sanoja on jätetty pois ("blacklist")

NSF 0-1-dokumenttiesimerkki Kuva #1: Erilaisten sanojen lukumäärä dokumenteittain

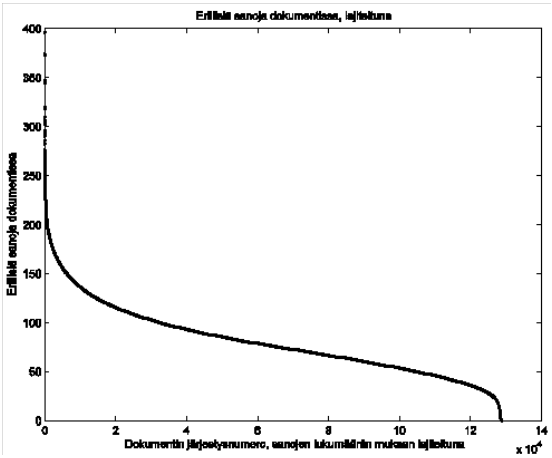


Kuva 8.2: *x-akseli*: Dokumentin järjestysnumero, yhteensä $12.8 \cdot 10^4$ dokumenttia. *y-akseli*: Erilaisten sanojen lukumäärä dokumentissa.

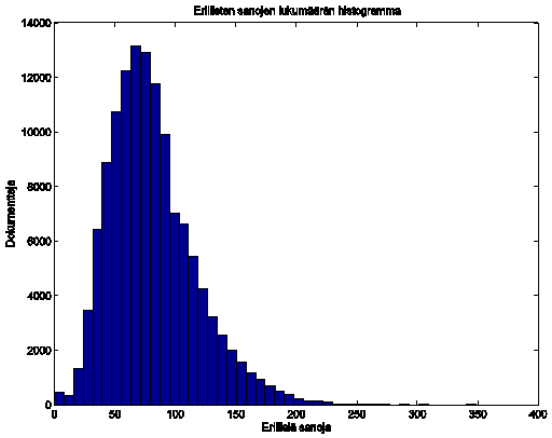
NSF 0-1-dokumenttiesimerkki Kuva #2: Erilaisten sanojen lukumäärä dokumenteittain

NSF 0-1-dokumenttiesimerkki Kuva #3: Erilaisten sanojen lukumäärä dokumenteittain

NSF 0-1-dokumenttiesimerkki Kuva #4: Kuinka monessa dokumentissa kukin sana esiintyy?



Kuva 8.3: *x-akseli*: Dokumentin järjestysnumero sanojen lukumäärän mukaan lajiteltuna. *y-akseli*: Erilaisten sanojen lukumäärä dokumentissa

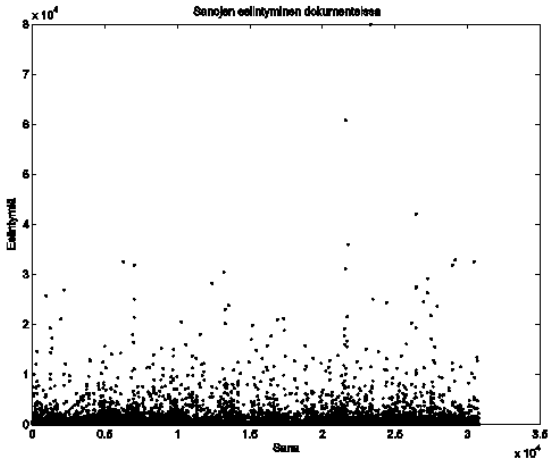


Kuva 8.4: Erilaisten sanojen lukumäärä dokumentissa histogrammina. Dokumentin pituus vaihtelee pääosin noin 30 – 180 (eri) sanan välillä.

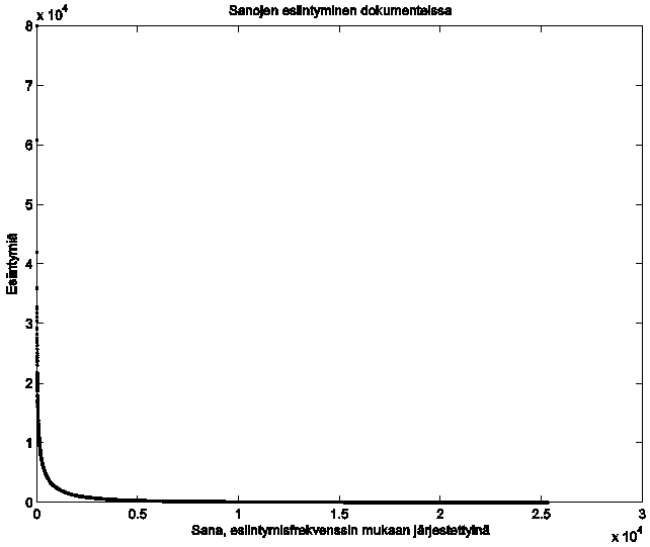
NSF 0-1-dokumenttiesimerkki Kuva #5: Kuinka monessa dokumentissa kukin sana esiintyy?

NSF 0-1-dokumenttiesimerkki Kuva #6: Kuinka monessa dokumentissa kukin sana esiintyy?

NSF 0-1-dokumenttiesimerkki Kuva #7: Kuinka monessa dokumentissa kukin sana esiintyy?



Kuva 8.5: *x-akseli*: Sanan indeksinumero $(1, \ldots, 30800)$. *y-akseli*: Esiintymiä $\times 10^4$ $(1, \ldots, 80000)$

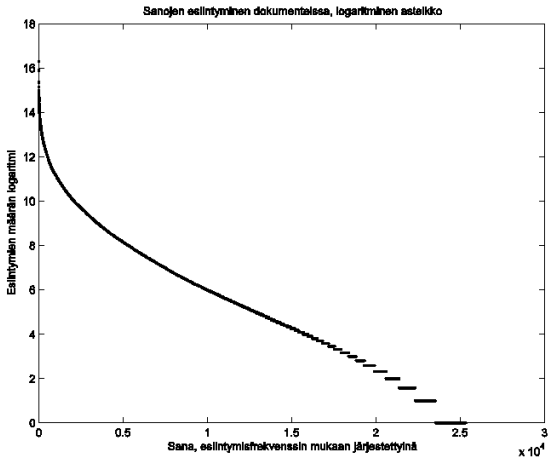


Kuva 8.6: Sama kuin edellä, mutta nyt järjestettynä esiintymisfrekvenssin mukaisesti

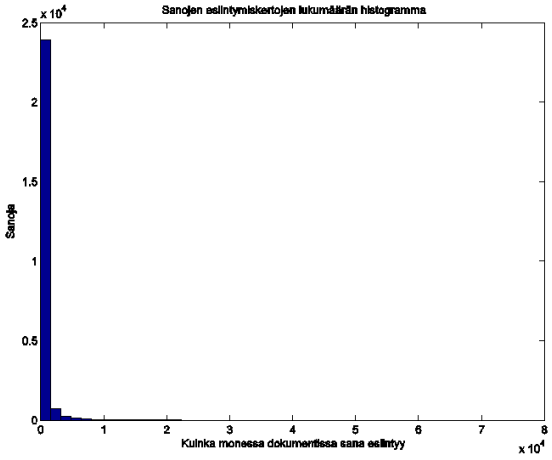
Datan kuvailua

0-1-datajoukko verkkona

- Verkko: joukko solmuja, joista osa on yhdistetty toisiinsa kaarilla
- 0-1-datajoukosta on helppo tehdä kaksijakoinen (“bipartite”) verkko: muuttujat ja havainnot ovat solmuja, ja muuttujan ja havainnon välillä on kaari jos ja vain jos vastaava datan arvo on 1

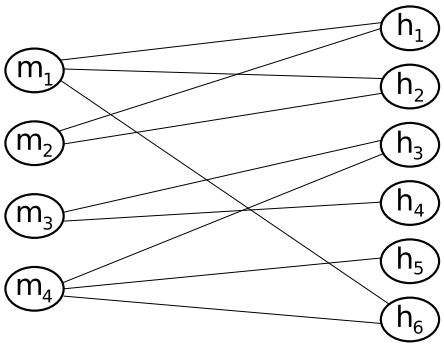


Kuva 8.7: Sama kuin edellä, mutta nyt esiintymisfrekvenssi (y-akseli) 2-kantaisena logaritmina. Miksi kuvaaja on osalta ($\sim 4000, \ldots, 16000$) lähes suora?



Kuva 8.8: Sanojen esiintymiskertojen lukumäärän histogrammi

	h_1	h_2	h_3	h_4	h_5	h_6
m_1	1	0	1	0	0	1
m_2	1	1	0	0	0	0
m_3	0	0	1	1	0	0
m_4	0	0	1	0	1	1

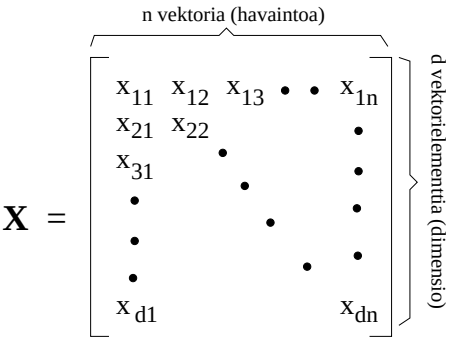


Mitä tällaisesta datasta voisi etsiä?

- Mitä datassa on?
- Minkälaisia ryhmittymiä datan muuttujilla ja havainnoilla on?
- Miten muuttujat riippuvat toisistaan?
- Jne.
- Muuttujien ja havaintojen astelukujen analyysi (kuinka monta kaarta muuttujasta lähtee, eli kuinka monessa havainnossa muuttuja on mukana): power laws – hyvin aktiivinen tutkimusalue
- Jatkossa: *miten etsitään pieniä kiinnostavia muuttujajoukkoja*

8.2 Usein esiintyvien muuttujakombinaatioiden etsintä: kattavat joukot

Usein esiintyvien muuttujakombinaatioiden etsintä Merkintöjä



- d-ulotteinen havaintovektori $\mathbf{x}$; sen alkiot $x_1, x_2, ..., x_d$
- Useat havaintovektorit $\mathbf{x}(1), \ldots, \mathbf{x}(n)$
- Havaintovektorin $\mathbf{x}(j)$ muuttujan i arvoa merkitään $x(i, j)$ (kuvassa x_{ij})

Kattavat joukot Määritelmä

- Jos muuttujia on esim. 30000, kaikkien parittaisten korrelaatioiden etsintä ei ole mahdollista
- Usein yhdessä esiintyvät muuttujat?
- Etsi kaikki muuttujaparit (a, b) siten että on olemassa ainakin N havaintoa $\mathbf{x}(i)$, jolla $x(a, i) = 1$ ja $x(b, i) = 1$
- *Yleisemmin*: etsi kaikki muuttujajoukot $\{a_1, a_2, \ldots, a_k\}$ siten että on olemassa ainakin N havaintoa $\mathbf{x}(i)$, jolla $x(a_1, i) = 1$ ja $x(a_2, i) = 1$ ja ... ja $x(a_k, i) = 1$
- Kutsutaan tällaista muuttujajoukkoa $\{a_1, a_2, \ldots, a_k\}$ *kattavaksi* ("frequent set")

Kattavat joukot Esimerkki kattavasta joukosta

- Olkoon tutkittavana neljä kauppakassia, joissa on tai ei ole appelsiineja (a), banaaneja (b) ja omenia (c).
- Kassi #1: 5 appelsiinia, 2 banaania. Kassi #2: 4 banaania. Kassi #3: 1 appelsiini, 2 banaania, 1 omena. Kassi #4: 8 appelsiinia, 2 omenaa.
- Koodataan esiintyminen kassissa 1:llä ja poissaolo 0:lla.

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

- Kynnysarvolla $N = 2$ kattavia joukkoja ovat $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$ ja $\{a, c\}$. Esimerkiksi appelsiinit (a) ja omenat (c) esiintyvät yhdessä ($\{a, c\}$) vähintään kahdessa kassissa.

Kattavat joukot Miten paljon kattavia joukkoja on?

- Jos datassa on pelkkiä ykkösiä, niin kaikki muuttujaparit tai muuttujaosajoukot täyttävät ehdon
- Ongelman mielekkyys edellyttää, että data on harvaa. (Tämä onkin tyypillistä. Edellisessä esimerkissä sekä kauppakasseja että tuotteita käytännössä tuhansittain, mutta yksi asiakas ostaa kerrallaan vain pienen määrän tuotteita. Harvassa matriisissa on siis nollia huomattavasti enemmän kuin ykkösiä.)
- Koneoppimisterminologialla: etsitään usein esiintyviä positiivisia konjunktioita ($A \wedge B$)

Kattavat joukot Miten kattavia muuttujajoukkoja etsitään?

- Triviaali lähestymistapa: käydään läpi kaikki muuttujien osajoukot
- d muuttujaa $\Rightarrow 2^d$ muuttujaosajoukkoa
- Jos muuttujia on esim. 100, niin osajoukkoja on liikaa
- Täytyy siis toimia jotenkin nokkelammin
- *Perushavainto: jos muuttujajoukko $\{a_1, a_2, \ldots, a_k\}$ on kattava, niin kaikki sen osajoukot ovat kattavia.*
- *Kääntäen: $\{a_1, a_2, \ldots, a_k\}$ voi olla kattava vain jos jokainen sen osajoukko on kattava.* Osajoukkojen kattavuus on siis välttämätön muttei riittävä ehto.

Kattavat joukot Esimerkki kattavien muuttujajoukkojen etsinnästä

- Oletetaan, että muuttujajoukot $\{a\}, \{b\}, \{c\}, \{e\}, \{f\}, \{g\}$ (joukon koko 1) esiintyvät tarpeeksi usein eli niistä on vähintään N havaintoa
- Silloin pari (i, j) on *mahdollisesti* kattava joukko, kun $i, j \in \{a, b, c, e, f, g\}$. Ehdokaspareja on $\binom{6}{2} = \frac{6!}{4! \cdot 2!} = 15$
- Oletetaan, että ehdokkaista vain parit $\{a, b\}$, $\{a, c\}$, $\{a, e\}$, $\{a, f\}$, $\{b, c\}$, $\{b, e\}$, $\{c, g\}$ ovat kattavia (koko 2) eli ne esiintyvät vähintään N kertaa yhdessä
- Luodaan kahden kokoisista kattavista kolmen kokoiset joukot. Pudotetaan pois ne joukot, joissa on mukana kahden kokoisia ei-kattavia joukkoja. Näin saadaan 3-kokoiset ehdokasjoukot.
  - Esimerkiksi kattavista muuttujapareista $\{a, b\}$ ja $\{a, f\}$ saadaan $\{a, b, f\}$, jonka alijoukko $\{b, f\}$ ei ole kattava, jolloin joukkoa $\{a, b, f\}$ ei hyväksytä ehdokkaaksi.
- Silloin $\{a, b, c\}$ ja $\{a, b, e\}$ ovat *ainoat mahdolliset* kolmen kokoiset kattavat joukot, koska niiden *kaikki* alijoukot ovat kattavia.
- Oletetaan, että $\{a, b, c\}$ kattava (koko 3).
- Lisäksi jo nyt tiedämme, että mitään neljän (tai isomman) kokoista kattavaa joukkoa *ei* voi olla, koska $\{a, b, c, e\}$:n alijoukoista mm. $\{a, c, e\}$ ei ole kattava. Toisin päin: ei-kattavaa joukkoa $\{a, c, e\}$ ei voi saada kattavaksi lisäämällä siihen alkioita.
- Kuten edellä jo pääteltiin, neljän kokoisia kandidaatteja ei synny, joten etsintä loppuu.
- Luetellaan lopuksi kaikki kattavat joukot $\{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$: $\{\{\}, \{a\}, \{b\}, \{c\}, \{e\}, \{f\}, \{g\}, \{a, b\}, \{a, c\}, \{a, e\}, \{a, f\}, \{b, c\}, \{b, e\}, \{c, g\}, \{a, b, c\}\}$

- Yllä “oletetaan” tarkoittaa siis, että “datasta on laskettu”. Tässä esimerkissä varsinaista dataa (datamatriisi $\mathbf{X}$) ei ole annettu.

Algoritmi pseudokoodina

8.3 Tasoittainen algoritmi kattavien joukkojen etsintään

Tasoittainen algoritmi Periaate

- Yksinkertainen mutta tehokas algoritmin kattavien joukkojen etsintään on *tasoittainen algoritmi* (“levelwise algorithm”, “apriori”)
- Etsitään ensin yhden kokoiset kattavat joukot, sitten kahden jne.
- Hyödynnetään edellä tehtyä perushavaintoa: muuttujajoukko voi olla kattava vain jos sen kaikki osajoukot ovat kattavia.

Tasoittainen algoritmi Joukon alkioiden lkm=1

- Kattavat joukot, jossa on yksi alkio: muuttuja a, jolla on vähintään N arvoa 1 datassa, ts. vähintään N havaintoa (saraketta) i, jolla $x(a, i) = 1$. Nämä on helppo etsiä lukemalla data kertaalleen läpi ja laskemalla kunkin muuttujan esiintymisfrekvenssi.

Tasoittainen algoritmi Joukon alkioiden lkm=2

- Kun yhden kokoiset kattavat joukot tunnetaan, niin muodostetaan kahden kokoiset *ehdokasjoukot* (“candidate”)
- Muuttujajoukko $\{a, b\}$ on ehdokasjoukko, jos sekä $\{a\}$ että $\{b\}$ ovat kattavia
- Käydään data uudestaan läpi ja lasketaan kullekin ehdokasjoukolle kuinka monessa havainnossa on sekä a- että b-muuttujan kohdalla 1.
- Näin löydetään kahden kokoiset kattavat joukot.

Tasoittainen algoritmi Joukon alkioiden lkm=k

- Oletetaan, että tunnetaan kaikki k:n kokoiset kattavat joukot; olkoon tämä kokoelma $\mathcal{C}_k$
- Muodostetaan kokoa $k + 1$ olevat *ehdokasjoukot*: joukot, jotka saattaisivat olla kattavia
  - Ehdokasjoukko on sellainen muuttujajoukko, jossa on $k + 1$ alkiota ja jonka kaikki osajoukot ovat jo kattavia. Riittää tarkastaa, että kaikki k alkiota sisältävät osajoukot ovat kattavia.

  - Otetaan kaikki joukkoparit $A = \{a_1, \ldots, a_k\}$ ja $B = \{b_1, \ldots, b_k\}$ kokoelmasta $\mathcal{C}_k$, lasketaan niiden yhdiste $A \cup B$, tarkistetaan, että yhdisteessä on $k+1$ alkiota ja että sen kaikki k:n alkion osajoukot ovat kattavia (ts. kuuluvat kokoelmaan $\mathcal{C}_k$).

- Kun $k+1$:n kokoiset ehdokasjoukot on laskettu, niin käydään data läpi ja lasketaan kullakin ehdokasjoukolla, monessako sarakkeessa ehdokasjoukon kaikki muuttujat ovat =1.
- Näin löydetään kokoa $k+1$ olevat kattavat joukot.
- Algoritmia jatketaan kunnes uusia ehdokasjoukkoja ei enää löydy.

Algoritmi pseudokoodina

Esimerkki tasoittaisen algoritmin tuloksesta

NSF-esimerkki tasoittaisen algoritmin tuloksesta Algoritmeja saatavilla

- Jatketaan saman NSF-datan kanssa
- Monia algoritmeja löytyy valmiina, esim. `http://fimi.ua.ac.be/` tai `http://adrem.ua.ac.be/~goethals/software/`
- Tietokoneharjoituskierroksella T5 käytetään Bart Goethalsin `apriori`-ohjelmaa kassakuittien yleisimpien tuotteiden ja kurssin harjoitustyödokumenttien yleisten sanojen etsintään
- Laskuharjoituksessa H5/2 esitellään algoritmin laskennan kompleksisuutta

NSF-esimerkki tasoittaisen algoritmin tuloksesta Kattavien joukkojen etsintä datajoukosta kynnysarvolla $N = 10000$

- Kynnysarvolla $N = 10000$ haku tuottaa yhteensä 250 kattavaa joukkoa aineiston 30800:sta muuttujasta (sanasta), jotka peräisin 128000 dokumentista
- 9 erilaista kolmen sanan joukkoa löytyy ainakin 10000 dokumentista

koko	ehdokkaita	kattavia joukkoja
1	30800	134
2	8911	107
3	79	9
4	0	0

NSF-esimerkki tasoittaisen algoritmin tuloksesta Kattavien joukkojen etsintä datajoukosta kynnysarvolla $N = 2000$

- Pienemmällä kynnysarvolla $N = 2000$ kattavia joukkoja tulee yhteensä 16040 ja ehdokasjoukkoja vielä runsaasti enemmän
- Löytyy yksi kuuden sanan joukko, joka löytyy ainakin 2000 dokumentista

koko	ehdokkaita	kattavia joukkoja
1	30800	1171
2	685035	7862
3	105146	6098
4	5813	889
5	92	19
6	1	1

NSF-esimerkki tasoittaisen algoritmin tuloksesta Kattavien joukkojen lukumäärä kynnysarvon N funktiona

- Kynnysarvo N: ykkönen (ykköset) täytyy löytyä vähintään N:stä sarakkeesta

kynnysarvo N	kattavia joukkoja
10000	250
5000	1539
2000	16040
1000	96223
$\vdots$	$\vdots$

NSF-esimerkki tasoittaisen algoritmin tuloksesta Kuuden kokoinen kattava joukko kynnysarvolla $N = 2000$

- `egrep '21582|21614|23313|24416|26454|29137' words.txt`

```
21582 program
21614 project
23313 research
24416 science
26454 students
29137 university
```

8.4 Riippumattomuus kattavissa joukoissa

Kattavat joukot Mitä muuttujajoukon kattavuus merkitsee?

- Jos muuttujajoukko esiintyy usein, voiko se johtua sattumasta?
- Olkoot a ja b muuttujia, ja olkoon datassa n_a havaintoa, joissa a esiintyy ja n_b havaintoa, joissa b esiintyy
- Kaikkiaan n havaintoa
- $p(a = 1)$: todennäköisyys sille, että satunnaisesti valitussa havainnossa on $a = 1$
- $p(a = 1) = n_a/n$ ja $p(b = 1) = n_b/n$

Kattavat joukot Riippumattomuus

- Kuinka monessa havainnossa a ja b esiintyvät yhtäaikaa, jos niiden oletetaan olevan riippumattomia?
- $p(a = 1 \wedge b = 1)$: todennäköisyys sille, että havainnossa on sekä $a = 1$ että $b = 1$ ($\wedge$ = ja)
- Jos havainnot riippumattomia niin:

$$\begin{aligned} p(a = 1 \wedge b = 1) &= p(a = 1)p(b = 1) \\ n_{ab}/n &= (n_a/n)(n_b/n) = n_a n_b/n^2 \end{aligned}$$

NSF-esimerkki jatkuu

NSF-esimerkki riippumattomuudesta Esimerkki riippumattomuudesta

- a = 'computational': $n_a = 7803$
- b = 'algorithms': $n_b = 6812$
- Riippumattomuus: 'computational' ja 'algorithms' esiintyvät satunnaisessa sarakkeessa todennäköisyydellä

$$\frac{n_a}{n}\frac{n_b}{n} = \frac{7803}{128000}\frac{6812}{128000} = .003244$$

- Odotusarvo tällaisten sarakkeiden lukumäärälle on

$$n\frac{n_a}{n}\frac{n_b}{n} = \frac{n_a n_b}{n} = 415$$

- Datassa tällaisia sarakkeita on $n_{ab} = 1974$ kappaletta.
- Merkitseekö tämä mitään? (Palataan tähän pian.)

NSF-esimerkki riippumattomuudesta Ylimäärin esiintyvien parien etsintä awk-skriptinä

- Syötteenä NSF-datan sanojen ja sanaparien frekvenssit
- Käyttö: `cat bag-of-words.txt | ./this.awk`

```
#!/usr/bin/awk -f
BEGIN {N=128000}
{
if (NF==2) {
        f[$1]=$2;}
if (NF==3) {
        p[$1, $2] = $3;}
}
END {
        for (u in f) {
                for (v in f) {
                   if (p[u, v]>0) {
                        print u, v, f[u], f[v], p[u,v], p[u,v]/(f[u]*f[v]/N)
                   }
                }
    }
}
```

NSF-esimerkki riippumattomuudesta Karkea analyysi millä pareilla $p(a = 1\&b = 1)/p(a = 1)p(b = 1)$ **on suuri?**

- Lasketaan suhde sanaparien todellisen esiintymisen ja sattumanvaraisen esiintymisen välillä

$$\frac{p(a = 1\&b = 1)}{p(a = 1)p(b = 1)} = \frac{n_{ab}/n}{(n_a/n)(n_b/n)} = \frac{n \cdot n_{ab}}{n_a \cdot n_b}$$

- Sarakkeet: (1) sanapari (a, b), (2) n_a, (3) n_b, (4) n_{ab}, (5) $\frac{p(a=1\&b=1)}{p(a=1)p(b=1)}$
- Näissä esimerkeissä suhde $\frac{p(a=1\&b=1)}{p(a=1)p(b=1)}$ on suuri:

```
fellowship postdoctoral 1458 3021 1028 29.8741
film thin               2275 3122 1431 25.789
dissertation doctoral   2272 2818 1135 22.6912
business innovation     4417 3514 2689 22.1754
polymer polymers        2768 2540 1208 21.9926
microscopy scanning     2925 2163 1079 21.8298
carolina north          1407 4687 1024 19.8756
poorly understood       2065 4100 1224 18.5049
...
```

NSF-esimerkki riippumattomuudesta Muuttujapareja löytyy...

- Muutamia muuttujapareja esimerkkinä
- Pari nro 100

  `held workshop 5417 4890 1644 7.94409`
- Pari nro 500

  `gene identify 4897 8450 1230 3.80477`
- Pari nro 1000

  `engineering manufacturing 15201 3991 1334 2.81457`

8.5 Kattavien joukkojen merkitsevyyden tutkiminen

Kattavien joukkojen merkitsevyyden tutkiminen

- Ajatellaan datan generointia *toistokokeena* (kts. binomijakauma)
- Tehdään n havaintoa (NSF-data: $n = 128000$)
- Kullakin havainnon kohdalla "onnistutaan" (tuotetaan havainto, jossa on sekä $a = 1$ että $b = 1$ todennäköisyydellä $p = n_a n_b / n^2$)
- Onnistumisten lukumäärä on binomijakautunut $Bin(n, p)$ parametrein n ja $n_a n_b / n^2$
- Kuinka todennäköistä on, että saadaan vähintään n_{ab} onnistumista?
- Kuinka pieni on todennäköisyys, että saisimme odotusarvosta niin paljon poikkeavan tuloksen?
- Jos tämä todennäköisyys on pieni, niin löydetty hahmo kertoo jotakin kiinnostavaa.

Kattavien joukkojen merkitsevyyden tutkiminen Binomijakauman hännän todennäköisyyden arviointi

- Binomijakauman hännän todennäköisyyden arviointi:

$$\sum_{i=n_{ab}+1}^{n} B(n, i, p),$$

  missä $B(n, i, p)$ on todennäköisyys, että n:n toiston kokeessa saadaan tasan i onnistumista, kun onnistumisen todennäköisyys on p:

$$B(n, i, p) = \binom{n}{i} p^i (1 - p)^{n-i}.$$

- Perinteiset approksimaatiot (normaaliapproksimaatio, Poisson-approksimaatio) eivät välttämättä suoraan sovi tähän (todennäköisyys p on pieni, mutta onnistumisten määrä on suuri)

Kattavien joukkojen merkitsevyyden tutkiminen Chernoffin raja

- Olkoon X onnistumisten lukumäärä toistokokeessa, jossa n koetta ja todennäköisyys onnistumiselle on p. Tällöin onnistumisen lukumäärän odotusarvo on np.
- Todennäköisyys $Pr(X > (1 + \delta)np)$ on rajoitettu (Chernoffin raja):

$$Pr(X > (1 + \delta)np) < \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^{np} < 2^{-\delta np}.$$

- (Katso lisää paperilaskareista H5/2)

NSF-esimerkki jatkuu

NSF-esimerkki kattavan sanaparin merkitsevyydestä Simulointitestaus

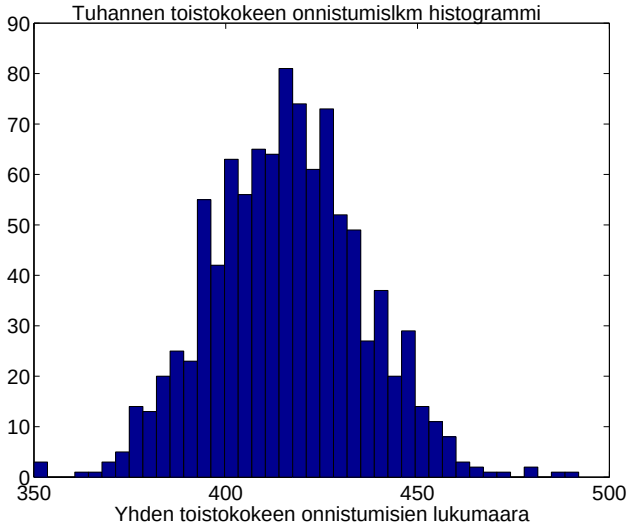
- NSF-data:
  - a = 'computational': $n_a = 7803$
  - b = 'algorithms': $n_b = 6812$
  - yhteiset havainnot $n_{ab} = 1974$
  - riippumattomuusoletuksella laskettu $p(a = 1)p(b = 1) = n_a n_b / n^2 = 0.003244$
- Satunnaistus: kokeillaan miten käy kun heitetään rahaa $n = 128000$ kertaa
- Onnistumistodennäköisyys $p = 0.003244$
- Lasketaan onnistumisten määrä

NSF-esimerkki kattavan sanaparin merkitsevyydestä Simulointitestaus Matlab-ohjelmana

```
p = 0.003244;
count=zeros(10,1);    % alustus
for a=1:10
   for i=1:128000     % hidas tapa
      if rand(1,1)<p
         count(a)=count(a)+1;
      end
   end
end

count=zeros(1000,1);  % alustus
for a=1:1000          % nopeampi tapa Matlabissa
   count(a) = sum(rand(128000,1)<p);
end
hist(count,40);
```

NSF-esimerkki kattavan sanaparin merkitsevyydestä Simuloinnin tulos



Todennäköisyys saada 1974 onnistumista lienee siis pieni; mutta kuinka pieni?

NSF-esimerkki kattavan sanaparin merkitsevyydestä Verrataan Chernoffin rajaan

- Toistokoe $X \sim Bin(n, p)$, jossa onnistumisen lukumäärän odotusarvo on np

$$Pr(X > (1+\delta)np) < \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{np} < 2^{-\delta np}.$$

- Nyt $p = .003244$, odotusarvo $np = 415$, havaittu arvo $X = 1974$.
- $X > (1+\delta)np$ eli $\delta = (X - np)/np = 3.757$
- Todennäköisyys on varsin pieni:

$$\begin{aligned} Pr(X > 1974) &= Pr(X > (1+3.757)415) \\ &< (\frac{42.8}{1667})^{415} \approx 0.0257^{415} < 10^{-660} \end{aligned}$$

- $2^{-\delta np} \approx 2^{-3.757*415} \approx 10^{-469}$

Muita huomioita

Moninkertaisen testauksen ongelma

- Oletetaan, että löydetään 100 kiinnostavaa muuttujaparia (a, b), ja tutkitaan kaikilla pareilla onko n_{ab} merkittävästi suurempi (tai pienempi) kuin riippumattomuusoletuksen antaman arvio $n_a n_b/n$.
- Saadaan tulokseksi esim., että parilla (a, b) näin suuri ero esiintyy sattumalta todennäköisyydellä 0.01.
- Todennäköisyys, että 100 parista jollakin tulee tällainen tulos on

$$1 - (1 - 0.01)^{100} \approx 0.63$$

(miksi?)

- Tutkittaessa usean hahmon esiintymäfrekvenssien satunnaisuutta pitää ottaa huomioon se, että tarkastelemme useita hahmoja.
- Bonferroni-korjaus: jos tutkitaan M:n hahmon esiintymistodennäköisyyksiä, niin sattumalta esiintymisen todennäköisyydet tulee kertoa M:llä.
- Edellä (NSF-esimerkki) yksittäisen hahmon todennäköisyys oli niin pieni, että korjaus ei muuta asiaa.
- Suurin osa löydetyistä muuttujapareista esiintyy niin usein yhdessä, että tämän tapahtuman esiintyminen sattumalta on erittäin epätodennäköistä.

Entä suuremmat kattavat joukot?

- Oletetaan, että muuttujajoukko $\{a, b, c\}$ esiintyy datassa n_{abc} kertaa (ts. datassa on n_{abc} saraketta, joilla muuttujat a, b ja c ovat kaikki 1.
- Onko tämä enemmän kuin satunnaisessa tapauksessa voisi olettaa?
- Mikä on satunnainen tapaus tässä tilanteessa?
- Esim. $n = 1000$ ja $n_a = n_b = n_c = 500$. Jos muuttujat ovat riippumattomia, niin n_{abc} on luultavasti noin 125.
- Jos $n_{abc} = 250$, niin $\{a, b, c\}$ on jotenkin epätavallinen kokoelma.
- Tämä voi johtua siitä, että esim. a ja b ovat vahvasti toisistaan riippuvia. (Jos $n_{ab} = 500$, niin sen jälkeen $n_{abc} = 250$ on odotettavaa.)
- Mikä on sovelias nollahypoteesi? Esim. ns. maksimientropiahypoteesi.

Miten vaikeaa kattavien joukkojen etsintä on?

- Annettuna 0-1 data ja kynnysarvo N
- Etsittävä kaikki kattavat joukot
- Vastaus voi olla hyvin suuri
- Käytännössä tasoittainen algoritmi on riittävän tehokas

Muunlaisten usein esiintyvien hahmojen etsintä

Muunlaisten usein esiintyvien hahmojen etsintä

- Samoja periaatteita voidaan käyttää myös muihin tehtäviin
  - Usein esiintyvien episodien etsintä
  - Usein esiintyvien aliverkkojen etsintä
  - Usein esiintyvien alipuiden etsintä
- Sama tasoittaisen algoritmin perusidea toimii

Yhteenveto

Yhteenveto

Tässä luvussa esiteltiin erityyppisen diskreetin aineston muuttamista 0-1-dataksi. Tätä voidaan käsitellä ja analysoida tehokkaasti mm. tasoittaisella algoritmilla.

Liite: Kuvia esitykseen, luku 8

NSF-data Esimerkki abstraktin kentistä
Takaisin kalvoihin

```
Title       : Mathematical Sciences: Structure and Rigidity of Graphs with Applications to
               Network Models of Materials
Type        : Award
NSF Org     : DMS
Latest
Amendment
Date        : July 5,  1994
File        : a9412017

Award Number: 9412017
Award Instr.: Standard Grant
Prgm Manager: Michael H. Steuerwalt
              DMS  DIVISION OF MATHEMATICAL SCIENCES
              MPS  DIRECT FOR MATHEMATICAL & PHYSICAL SCIEN
Start Date  : July 1,  1994
Expires     : June 30,  1998       (Estimated)
Expected
Total Amt.  : $67877             (Estimated)
Investigator: Deborah S. Franzblau   (Principal Investigator current)
Sponsor     : Rutgers Univ New Brunswick
              ASB III, 3 Rutgers Plaza
              New Brunswick, NJ  08901    732/932-0150

NSF Program : 1271      COMPUTATIONAL MATHEMATICS
Fld Applictn: 0000099   Other Applications NEC
              21        Mathematics
Program Ref : 9161,9162,9263,AMPP,
```

NSF-data Esimerkki abstraktista
Takaisin kalvoihin

```
Abstract    :
   9412017  Franzblau       In this project, the principal investigator aims to
   obtain  the following results: (1) an implementation of a practical
   combinatorial algorithm to compute bounds on degrees of freedom  of a network
   (in 3 or more dimensions), (2) new combinatorial  conditions for rigidity or
   simple formulas for degrees of freedom  in families of graphs, (3) new, easily
   computed measures of  medium-range order in network models of glasses.  The
   methods  employed include those of combinatorial optimization, graph  theory,
   and discrete algorithm design.  Problems on network  models are addressed
   largely through computer experiments, and  make use of algorithms developed and
   implemented by the  investigator.       This project has two related aims, and
   is intended to  contribute new results both to mathematics and materials
   science.  The first aim is to address key open problems in the mathematical
   theory of rigidity, including computing the degrees of freedom of  a network
   (also called a graph).  This theory has a long history,  which includes the
   work of Maxwell (1864) on determining whether  a "scaffold" made of rigid bars
   and movable joints is itself  rigid. The second aim is to address basic issues
   on network  models of solids; such models (also called ball-and-stick  models),
   in which points represent atoms and connections between  points represent
   chemical bonds, are often studied to better  understand the properties of both
   crystalline solids and glassy  materials.  One focus is to characterize the
   relationship between  the structure of a network model and its rigidity, and
   the other  is to find useful measures which capture this network structure.
   The work therefore leads to a better understanding of materials  properties.
```

Tasoittainen algoritmi

1: $k \leftarrow 1$ ▷ Init level $k = |X|$
2: $C_k \leftarrow \{ \{ a \} \mid a \in \text{variables} \}$ ▷ Init candidate sets C_1
3: **while** $C_k \neq \emptyset$ **do**
4: counter[X] $\leftarrow 0$ for all X ▷ Init counters for each candidate
5: **for** observation in data **do** ▷ Count frequencies of candidates
6: **for** X in C_k **do** ▷ Check if all variables in X are present
7: good $\leftarrow$ True

```
8:              for var in X do
9:                  if observation[var] = 0 then
10:                     good ← False
11:             if good then
12:                 counter[X] ← counter[X] + 1
13:     F_k ← ∅                                   ▷ Init frequent sets F_k at level k
14:     for X in C_k do                           ▷ Select frequent candidates
15:         if counter[X] ≥ N then                ▷ Threshold N
16:             F_k ← F_k ∪ { X }
17:     C_{k+1} ← ∅                               ▷ Init candidate sets C_{k+1} at level k + 1
18:     for A in F_k do                           ▷ Generate next candidates
19:         for B in F_k do
20:             X ← A ∪ B
21:             if |X| = k + 1 then
22:                 good ← True
23:                 for var in X do               ▷ Ignore sets with nonfreq. subsets
24:                     if X \ { var } not in F_k then
25:                         good ← False
26:                 if good then
27:                     C_{k+1} ← C_{k+1} ∪ { X }
28:     k ← k + 1                                 ▷ Return to row 3
```

koodi: Jouni Seppänen.

Takaisin kalvoihin: esimerkki

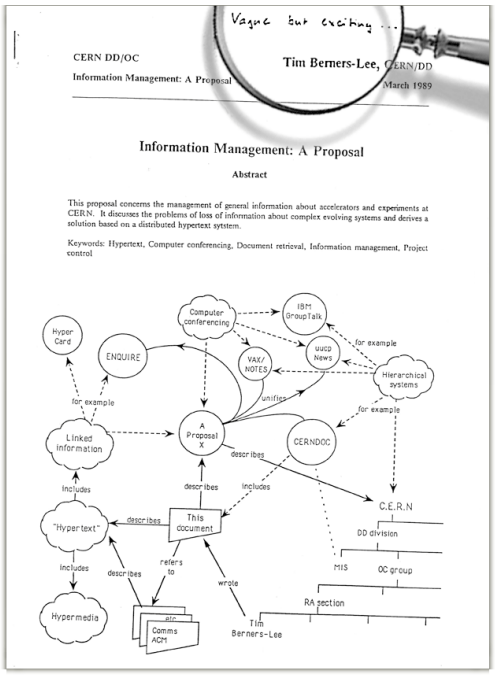
Takaisin kalvoihin: algoritmin kuvaus

9 Relevanttien sivujen etsintä verkosta: satunnaiskulut verkossa

9.1 Webin lyhyt historia

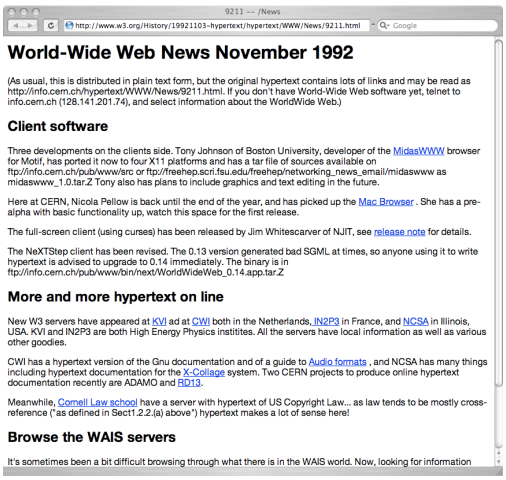
Webin lyhyt historia 1989: Tim Berners-Lee

- Maaliskuussa 1989 Tim Berners-Lee jätti ehdotuksen CERNin tiedonhallintajärjestelmäksi
- Esimiehen kommentti: “epämääräinen mutta kiinnostava...”
  Proposal.html



Webin lyhyt historia 1992: ensimmäisiä selaimia

- Syksyllä 1992 info.cern.ch kertoi uusista palvelimista ja sivuista
  http://www.w3.org/History/...



Webin lyhyt historia Linkkikokoelmia

- Jerry and David's Guide to the World Wide Web. Myöhemmin nimellä Yahoo; iso käsin ylläpidetty lista webbiosoitteita.

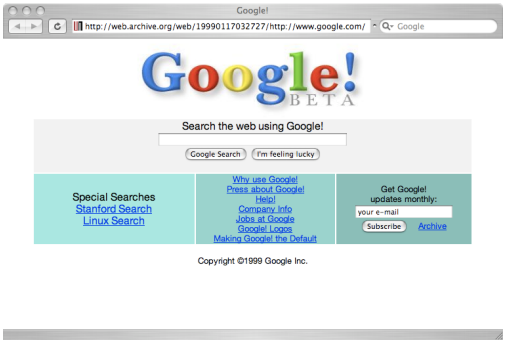
  vanhoja sivustoja

Webin lyhyt historia Altavista oli 1990-luvun Hakukone

- Digital teki Altavistan mainostaakseen servereitään: "Näillä voi indeksoida vaikka koko webin."
- Altavistalla webbisivujen etsiminen muuttui paljon helpommaksi, mutta vastaukset tulivat melko satunnaisessa järjestyksessä.

Webin lyhyt historia 1998: Google

- Google keksi, miten hakutulokset järjestetään hyödyllisyyden mukaan.



9.2 Etsintä verkosta

Etsintä verkosta

- Miten löydetään verkosta hyviä sivuja, jotka kertovat vaikkapa tiedon louhinnasta?
- Ensimmäiset menetelmät skaalautuivat huonosti:
  - info.cern.ch: ilmoitustaulu uusista sivuista
  - Yahoo: käsin toimitettu hakemisto
- Annettuna kysely “data mining”
- Miten löydetään automaattisesti sivut, joilla nuo sanat esiintyvät (tai esiintyvät peräkkäin)?
- Miten löydetään näistä sivuista hyödyllisimmät?

Etsintä verkosta Sanahaku

- *Merkkijonomenetelmät* (“string algorithms”): paljon hauskoja algoritmeja
- Esivalmisteluna käydään koko verkko läpi seuraamalla linkkejä
- Kullakin sanalla X tallennetaan lista sivuista, joilla X esiintyy
- Kyselyyn “X ja Y” vastattessa käydään kummankin sanan osoitelistat läpi ja palautetaan ne osoitteet, jotka esiintyvät molemmissa.
- Käytännössä hankalampaa kuin näyttää paperilla: dataa on paljon, vastaus pitäisi saada alle sekunnissa.
- Paljon mielenkiintoista tekniikkaa, katso esim.

  `http://research.google.com/pubs/papers.html`

Etsintä verkosta Relevanssi

- “Results 1 – 10 of about 50,300,000”
- Vanha tapa ratkaista ongelma: käyttäjä tarkentaa kyselyä lisäämällä ovelasti valittuja sanoja ja operaattoreilla AND, OR, NOT, NEAR.
- Miten valitaan ne sivut, jotka näytetään ensimmäiseksi? Heuristiikkoja:
  - Sanojen esiintymisfrekvenssi (huono idea)
  - Sivu on hyvä, jos siihen viitataan paljon
  - Sivu on hyvä lähde asiasta X, jos X mainitaan sivuun viittaavissa linkeissä (eli *ihminen* on tehnyt jo arvioinnin)

9.3 Keskukset ja auktoriteetit

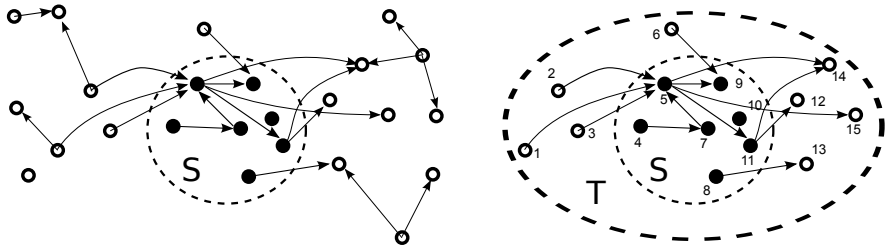
Keskukset ja auktoriteetit

- “Hubs and authorities”: Jon M. Kleinberg: Authoritative Sources in a Hyperlinked Environment, IBM Research Report 1997; SODA 1998

  http://scholar.google.fi/ $\rightarrow$
- Hyvillä auktoriteetti-sivuilla on paljon *yhteisiä* linkittäjiä eli keskuksia.
- Hyvä keskus osoittaa hyvää arvostelukykyä linkittämällä hyviin auktoriteetteihin

Keskukset ja auktoriteetit Etsitään ensin parhaiden sivujen ja linkkien joukot

- Etsitään ensin sivut, joilla "X ja Y" esiintyy
- Otetaan näistä heuristiikkojen perusteella esim. 200 parasta; olkoon tämä sivujen *ydinjoukko* S
- Muodostetaan joukko T: S ja sivut jotka viittaavat johonkin joukon S sivuun ja sivut joihin jokin S:n sivu viittaa (kuva 193)



Kuva 9.1: Keskukset ja auktoriteetit. Ydinjoukko S laajennettuna sivuilla, joihin tai joista on linkki.

- Tarkastellaan joukkoa T verkkona: solmuina sivut, kaarina linkit (suunnattuja kaaria)
- Olkoon E verkon T kaarien joukko: $(u, v) \in E$ kun sivulta u on linkki sivulle v
- Pelkkä sivuun osoittavien linkkien määrä ei ole kovin hyvä relevanssin mittari

Keskukset ja auktoriteetit Periaate

- *Periaate*:
  - Hyvä keskus osoittaa hyviin auktoriteetteihin
  - Hyviin auktoriteetteihin tulee linkkejä hyvistä keskuksista
- Kehämääritelmä?

Keskukset ja auktoriteetit Hyvien keskusten ja auktoriteettien laskenta iteratiivisesti

- Kehämääritelmästä selvitään iteratiivisella menetelmällä (vrt. c-means, Luku 6)
- Kullekin joukon T sivulle s määritellään keskuspaino k_s ja auktoriteettipaino a_s
- Tarvitaan jotkin alkuarvot:

$$a_s = k_s = \frac{1}{\sqrt{n}}, \quad \forall s \in T$$

- Iteratiiviset päivityssäännöt:

$$a_s \leftarrow \sum_{t \in T,\ (t,s) \in E} k_t, \qquad k_s \leftarrow \sum_{t \in T,\ (s,t) \in E} a_t$$

- Eli sivun s auktoriteettipainoa varten summataan kaikki keskuspainot niistä sivuista, joista on linkki sivulle s. Sivun s keskuspaino päivitetään vastaavasti summaamalla sivun s linkkaamien sivujen auktoriteettipainot

- Joka iteraatiossa skaalataan painojen neliöiden summat ykkösiksi

$$\sum_{s \in T} a_s^2 = 1, \qquad \sum_{s \in T} k_s^2 = 1$$

Keskukset ja auktoriteetit Hyvien keskusten ja auktoriteettien laskenta matriiseilla

- Edellä saadut iterointikaavat voidaan kirjoittaa helposti myös matriisimuodossa

- Ajatellaan painoja vektoreina

$$\mathbf{a} = (a_s)_{s \in T}, \qquad \mathbf{k} = (k_s)_{s \in T}$$

  ja verkko matriisina M

$$M = (1, \text{ jos } (s,t) \in E, \text{ muuten } 0)$$

- Nyt iteraation päivityssäännöt voidaan kirjoittaa:

$$\mathbf{a} \leftarrow M^T \mathbf{k}, \qquad \mathbf{k} \leftarrow M\mathbf{a}$$

- Ottaen huomioon skaalauksen ykköseksi voidaan kirjoittaa koko iterointisääntö:

$$\mathbf{a} \leftarrow \frac{M^T \mathbf{k}}{\|M^T \mathbf{k}\|}, \qquad \mathbf{k} \leftarrow \frac{M\mathbf{a}}{\|M\mathbf{a}\|}$$

- Ajetaan iteraatiota läpi muutama kierros, kun alustuksena $\mathbf{1}$

$$\begin{cases} \mathbf{a} \leftarrow \frac{1}{\|M^T\mathbf{1}\|} M^T\mathbf{1} \\ \mathbf{k} \leftarrow \frac{1}{\|M\mathbf{1}\|} M\mathbf{1} \end{cases} \quad \begin{cases} \mathbf{a} \leftarrow \frac{1}{\|\ldots\|} MM^T\mathbf{1} \\ \mathbf{k} \leftarrow \frac{1}{\|\ldots\|} M^TM\mathbf{1} \end{cases} \quad \begin{cases} \mathbf{a} \leftarrow \frac{1}{\|\ldots\|} M^TMM^T\mathbf{1} \\ \mathbf{k} \leftarrow \frac{1}{\|\ldots\|} MM^TM\mathbf{1} \end{cases}$$

- Siis i:n iteraation jälkeen

$$\mathbf{a} = \frac{1}{\|\ldots\|}(M^TM)^i\mathbf{1}$$

- Pieni esimerkki

- Matriisi $M^T M$ on symmetrinen, joten sillä on reaaliset ominaisarvot ja -vektorit ja se voidaan diagonalisoida
$$M^T M = V D^n V^T$$
missä $D = diag(\lambda_1, \ldots, \lambda_n)$
- Teknisellä oletuksella $\lambda_1 > \lambda_2$ saadaan
$$(M^T M)^i \mathbf{1} = V D^i V^T \mathbf{1} \approx V \cdot diag(\lambda_1^i, 0, \ldots, 0) \cdot V^T \mathbf{1} = (\lambda_1^i \mathbf{v}_1^T \mathbf{1}) \mathbf{v}_1$$
- Siis $\mathbf{a}$ on matriisin $M^T M$ suurinta ominaisarvoa vastaava ominaisvektori
- Vastaavasti $\mathbf{k}$ on matriisin $M M^T$ ominaisvektori
- Ominaisvektorit voitaisiin esim. Matlabissa funktiolla `[V, D] = eig(M'*M)`

Keskukset ja auktoriteetit Tuloksia Kleinbergin paperista

- Auktoriteetteja (ylempi kuva)
- Toinen sovellus (alempi kuva): Samankaltaisten sivujen löytäminen: sen sijasta, että aloitettaisiin sanahaun löytämistä sivuista, aloitetaan johonkin sivuun linkittävistä sivuista.

(java) Authorities

.328 http://www.gamelan.com/	*Gamelan*
.251 http://java.sun.com/	*JavaSoft Home Page*
.190 http://www.digitalfocus.com/digitalfocus/faq/howdoi.html	*The Java Developer: How Do I...*
.190 http://lightyear.ncsa.uiuc.edu/~srp/java/javabooks.html	*The Java Book Pages*
.183 http://sunsite.unc.edu/javafaq/javafaq.html	*comp.lang.java FAQ*

(censorship) Authorities

.378 http://www.eff.org/	*EFFweb - The Electronic Frontier Foundation*
.344 http://www.eff.org/blueribbon.html	*The Blue Ribbon Campaign for Online Free Speech*
.238 http://www.cdt.org/	*The Center for Democracy and Technology*
.235 http://www.vtw.org/	*Voters Telecommunications Watch*
.218 http://www.aclu.org/	*ACLU: American Civil Liberties Union*

(www.honda.com) Authorities

.202 http://www.toyota.com/	*Welcome to @Toyota*
.199 http://www.honda.com/	*Honda*
.192 http://www.ford.com/	*Ford Motor Company*
.173 http://www.bmwusa.com/	*BMW of North America, Inc.*
.162 http://www.volvocars.com/	*VOLVO*
.158 http://www.saturncars.com/	*Welcome to the Saturn Web Site*
.155 http://www.nissanmotors.com/	*NISSAN - ENJOY THE RIDE*
.145 http://www.audi.com/	*Audi Homepage*

9.4 PageRank

PageRank

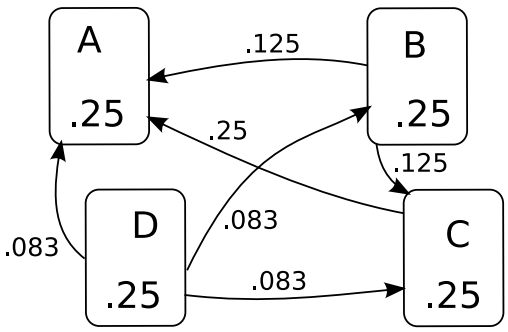
- Sergey Brin and Larry Page, 1998 (Google)

  The Anatomy of a Large-Scale Hypertextual Web Search Engine

  Wikipedia: PageRank
- Sivu on relevantti, jos relevantit sivut viittaavat siihen.
- Kehämääritelmä? (Jälleen tehdään iteratiivinen menetelmä)

PageRank Laskenta

- Jos sivulta t on linkit sivuille s_1, s_2, ..., s_k, sivun t relevanssista r välittyy $1/k$ jokaiselle näistä sivuista
- Esimerkiksi universumin neljä sivua, niiden nykyiset relevanssit, linkit ja välittyvät relevanssit



- Siis relevanssivektori (“PageRank”) $\mathbf{r} \leftarrow F\mathbf{r}$, missä matriisin F määrittelee

$$F_{s,t} = \begin{cases} 1/\deg(t), & \text{jos } t\text{:stä on linkki } s\text{:ään} \\ 0, & \text{muuten} \end{cases}$$

  jossa $\deg(t)$ on linkkien määrä eli k yllä

- Sopivilla oletuksilla $\mathbf{r}$ on F:n suurinta ominaisarvoa vastaava ominaisvektori, joten iteraatio konvergoi

- Toinen tulkinta: Satunnainen surffailija aloittaa joltain sivulta ja seuraa linkkejä satunnaisesti. Sivun relevanssi on todennäköisyys, jolla surffailija päätyy sivulle (pitkän ajan kuluessa).

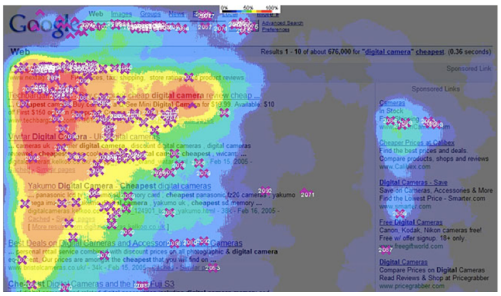
- Entä jos...

- Jos sivusta ei ole linkkejä minnekään, lisätään linkit kaikkialle. Tässä n on sivujen määrä.

$$F_{s,t} = \begin{cases} 1/\deg(t), & \text{jos } t\text{:stä on linkki } s\text{:ään} \\ 1/n, & \text{jos } t\text{:stä ei ole linkkejä minnekään} \\ 0, & \text{muuten} \end{cases}$$

- Lisätään linkkejä saman tien kaikkialle: sallitaan surffaajan joskus kyllästyä (lisäksi vältytään teknisiltä hankaluuksilta konvergenssitodistuksessa) Pieni esimerkki

$$F_{s,t} = \begin{cases} \frac{1-\epsilon}{\deg(t)} + \frac{\epsilon}{n}, & \text{jos } t\text{:stä on linkki } s\text{:ään} \\ 1/n, & \text{jos } t\text{:stä ei ole linkkejä minnekään} \\ \epsilon/n, & \text{muuten} \end{cases}$$



Altavistan käyttäjät jaksoivat lukea tuloksia monelta sivulta, Googlen käyttäjät katsovat vain muutamaa ensimmäistä

Yhteenveto

Yhteenveto

Tässä luvussa keskusteltiin merkitsevien webbisivujen löytämisestä sivujen linkityksen perusteella.

Keskusten ja auktoriteettien algoritmissa periaattena on, että hyvät keskukset linkittävät hyviin auktoriteetteihin ja vastaavasti hyviin auktoriteetteihin tulee linkkejä hyvistä keskuksista.

PageRank-algoritmi laskee sivujen relevanssia “satunnaisen surffailijan” tavoin. Algoritmi on käytössä Googlessa.

Liite: Kuvia esitykseen, luku 9

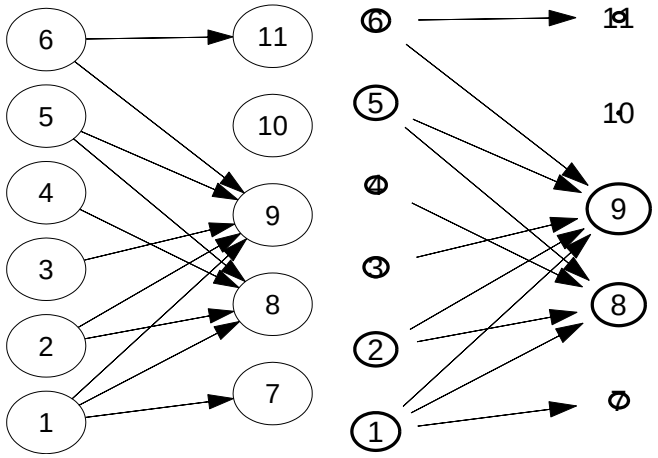
Esimerkki keskuksista ja auktoriteeteista

Tutkitaan oheista verkkoa (kuva 9.2),

jonka kaarien matriisi M, $M(i,j) = 1$, jos olemassa kaari (s_i, s_j).

Oikeanpuoleisessa kuvassa vasemman puolen solmujen koko viittaa laskettuun keskuspainoon ja oikean puolen solmujen koko auktoriteettipainoon.

Seuraavilla kalvoilla Matlab-koodi ja tulokset.



Kuva 9.2: Keskukset ja auktoriteetit. (a) Lähtötilanne. (b) Keskuspainot $k_1, \ldots, k_6$ ja auktoriteettipainot $a_7, \ldots a_{11}$

Esimerkki keskuksista ja auktoriteeteista Matlab-koodi

```
M = [ 0 0 0 0 0 0 1 1 1 0 0; <M-matriisi> ]; % M(i,j)=1 <=> s_i-->s_j

n = size(M,1);
a = ones(n,1)/sqrt(n);  % auktoriteettipainot
k = ones(n,1)/sqrt(n);  % keskuspainot

for s = 1 : 20
  aold = a;    kold = k;
  anew = M'*k; knew = M*a;     % paivitys
  a = anew/sqrt(sum(anew.^2)); % skaalaus
  k = knew/sqrt(sum(knew.^2));

  disp(['Kierros: ' num2str(s)]);
  disp(['Muutos:  ' num2str(sum((a-aold).^2)) ', ' num2str(sum((k-kold).^2))]);
  disp(['    #         aukt      kesk       alku']);
  disp([(1:n)' a k ones(n,1)/sqrt(n)])
  pause;
end
```

Esimerkki keskuksista ja auktoriteeteista Tulokset

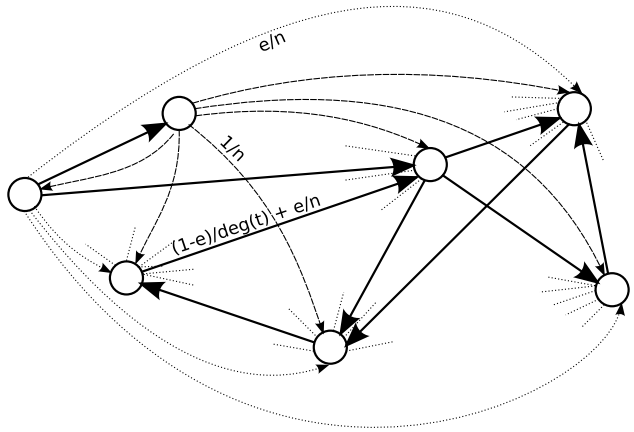
```
Kierros: 20
```

```
Muutos:  1.2917e-15, 1.911e-14
    #          aukt       kesk        alku
    1.0000          0    0.5583    0.3015
    2.0000          0    0.4877    0.3015
    3.0000          0    0.2659    0.3015
    4.0000          0    0.2219    0.3015
    5.0000          0    0.4877    0.3015
    6.0000          0    0.3043    0.3015
    7.0000     0.1985         0    0.3015
    8.0000     0.6241         0    0.3015
    9.0000     0.7479         0    0.3015
   10.0000          0         0    0.3015
   11.0000     0.1082         0    0.3015
```

Takaisin kalvoihin

Esimerkki PageRank:sta

$$F_{s,t} = \begin{cases} \frac{1-\epsilon}{\deg(t)} + \frac{\epsilon}{n}, & \text{jos } t\text{:stä on linkki } s\text{:ään} \\ 1/n, & \text{jos } t\text{:stä ei ole linkkejä minnekään} \\ \epsilon/n, & \text{muuten} \end{cases}$$



Kuva 9.3: PageRank-esimerkki. Varsinaiset linkit paksummalla viivalla ja nuolella.

Takaisin kalvoihin

Hakemisto