Luku 6. Hahmontunnistuksen perusteita

T-61.2010 Datasta tietoon, syksy 2011

professori Erkki Oja

Tietojenkäsittelytieteen laitos, Aalto-yliopisto

17.11.2011

Tämän luvun sisältö

Tämä luku käydään kahdella luennolla: ensimmäisellä luokittelu ja toisella ryhmittelyanalyysi.

1 Hahmontunnistuksen perusteita
- Luokittelu
- Lähimmän naapurin luokitin (kNN)
- Bayes-optimaalinen luokitin
- Ryhmittelyanalyysi
- Hierarkkinen ryhmittely
- c-means ryhmittelyalgoritmi (k-means, KM)

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

Luokittelu (2)

- 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

Luokittelu (3)

- .. ja paljon muuta. Kyseessä on selkeä kasvuala.

Hahmoalueet, erotinfunktio

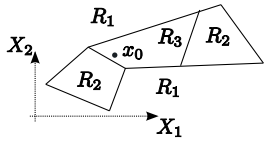
- *Hahmo* on tunnistettavan kohteen numeerinen esitystapa, joka voi olla vektori, matriisi, puu, graafi, merkijono, ...
- Tässä suppeassa esityksessä keskitytään taas vektoreihin: ajatellaan siis että kukin kohde on esitettävissä vektorina $\mathbf{x} = (x_1, \ldots, x_d)^T$ (pisteenä d-ulotteisessa 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?

Hahmoalueet, erotinfunktio (2)

- 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.

Hahmoalueet, erotinfunktio (3)

- Esimerkki: Kuvassa on kolme hahmoaluetta, joista R_2 kahdessa osassa. Alueiden keskinäiset leikkaukset tyhjiä, yhdiste koko $\mathbb{R}^2$. Kuvassa siis $\mathbf{x}_0$ kuuluu alueeseen R_3 joten sen luokka on ω_3



Kuva: Hahmoalueet

- Alueet ovat mielivaltaisen muotoisia, ja voi olla vaikeaa laskennallisesti päätellä mihin alueeseen $\mathbf{x}$ kuuluu

Hahmoalueet, erotinfunktio (4)

- 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

Hahmoalueet, erotinfunktio (2)
Luokitteluvirheen minimointi

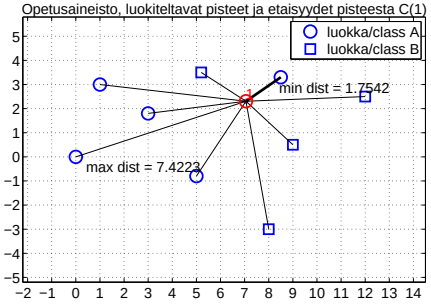
- *Testijoukkoa* voidaan käyttää luokitteluvirheen laskemiseen: annetaan testijoukko ilman luokkatietoa luokittimille, ja lasketaan kuinka usein luokitin päätyi väärään lopputulokseen ▸ Esimerkki
- Koska opetusjoukko on äärellinen, ei yleensä voida muodostaa virheetöntä luokitinta, eli kun uusia hahmovektoreita luokitellaan, menee luokka joskus väärin

Lähimmän naapurin luokitin (kNN)

- Idea on hyvin yksinkertainen: olkoon uusi luokiteltava vektori $\mathbf{x} \in \mathbb{R}^d$, jolle pitäisi siis löytää “oikea” luokka ω
- Käytettävissä opetusjoukko $(\mathbf{x}, \omega)_j$ eli tunnetaan valmiiksi luokiteltua (M luokkaa) dataa $\mathbf{x}_{\omega_i}(j) \in \mathbb{R}^d$: $\{\mathbf{x}_{\omega_1}(1), \ldots, \mathbf{x}_{\omega_1}(n_1)\}, \ldots, \{\mathbf{x}_{\omega_M}(1), \ldots, \mathbf{x}_{\omega_M}(n_M)\}$.
- 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

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

Lähimmän naapurin luokitin (kNN) (2)

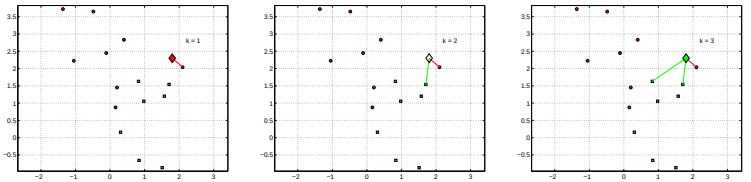


Kuva: Euklidinen etäisyys lähimpään naapuriin ≈ 1.75.

- Neliöjuurta ei käytännössä tarvitse ottaa, koska se ei vaikuta kun vertaillaan etäisyyksiä keskenään

Lähimmän naapurin luokitin (kNN) (3)

- Luokitellaan **x** siihen luokkaan, mihin *enemmistö naapureista kuuluu*. Tasapelin sattuessa heitetään lanttia.



Kuva: 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.

Lähimmän naapurin luokitin (kNN) (4)

- Usein luokittelussa kaksi luokkaa $M = 2$. Tällöin k kannattaa valita parittomaksi ($k = 1, 3, 5, \ldots$)
- 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

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 **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”

Bayes-optimaalinen luokitin (2)

- 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

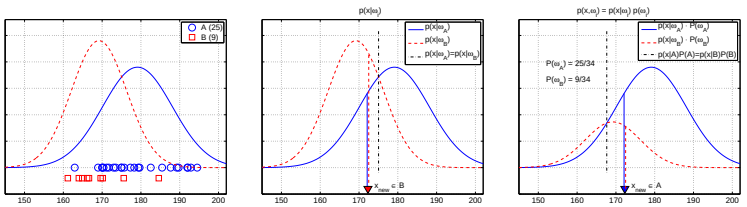
Bayes-optimaalinen luokitin (3)

- 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.

Bayes-optimaalinen luokitin (4)

- 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-optimaalinen luokitin (5)



Kuva: (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.

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

- Muistetaan moniulotteisen normaalijakauman tiheysfunktio (Luku 5):

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

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

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

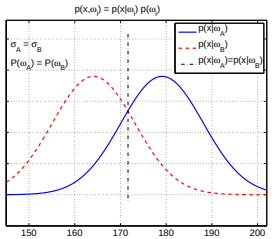
- Nähdään että jos kovarianssimatriisit ovat samat (tai oletetaan samoiksi), kaikki termit K_j ovat samat ja voidaan tiputtaa vertailussa pois.

Bayes-luokitin normaalijakautuneelle datalle (2)
Millaiseksi sääntö muodostuu jos $p(\mathbf{x}|\omega_j)$:t ovat normaalijakautuneita?

- 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 $\mathbf{x} \in \omega_2$.

Bayes-luokitin normaalijakautuneelle datalle (3)
Millaiseksi sääntö muodostuu jos $p(\mathbf{x}|\omega_j)$:t ovat normaalijakautuneita?

- Huomaa että vasen puoli on $\mathbf{x}$:n *lineaarinen* funktio, päätöspinta luokkien ω_1 ja ω_2 välillä on *hypertaso*.
- Esimerkki 1D:ssä:



Kuva: 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 $P(x_i = 1|\omega_1) = p,\ P(x_i = 1|\omega_2) = q,\ p > q.$

Bayes-luokitin binääridatalle (2)

Millainen sääntö binääridatalle?

- 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 $\Pi_{i=1}^{8} p^{x_i}(1-p)^{1-x_i}$

Bayes-luokitin binääridatalle (3)

Millainen sääntö binääridatalle?

- 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:

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

Bayes-luokitin binääridatalle (4)

Millainen sääntö binääridatalle?

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 $g_1(\mathbf{x}|\omega_1) > g_2(\mathbf{x}|\omega_2)$

$$\sum_i x_i > 0.661d$$

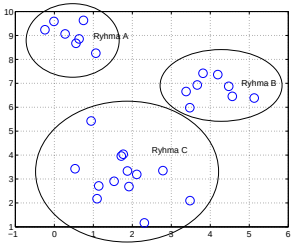
Tämän voi tulkita niin että “kuulut luokkaan yksi, jos vastasit ’samaa mieltä’ vähintään 66.1 prosenttiin kysymyksistä”.

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.

Ryhmittelyanalyysi (2)

- 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: Datan ryhmittely kolmeen joukkoon.

Ryhmittelyanalyysi (3)

- $\Rightarrow$ 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: Miten ryhmittelisit tämän datan?

Ryhmittelyanalyysi (4)

- 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.

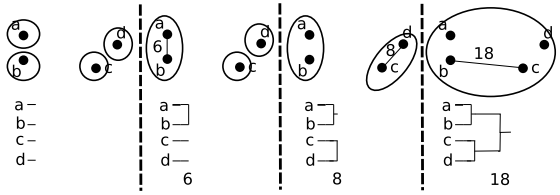
Ryhmittelyanalyysi
Sovelluksia

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

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).

Hierarkkinen ryhmittely (2)
Algoritmi

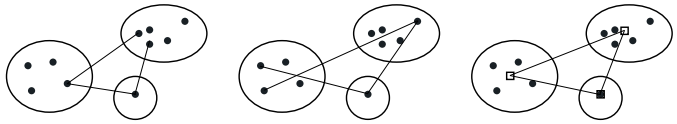


Kuva: Hierarkkinen ryhmittely käyttäen lyhintä etäisyyttä. Alussa neljä datapistettä on neljässä ryhmässä. Yhdistetään kaksi lähintä ryhmää etäisyydellä/kustannuksella 6. Piirretään ryhmittelypuuta. Jatketaan kunnes kaikki pisteet yhdessä ryhmässä.

- Edellisessä esimerkissä ryhmittelypuussa oli hyppy yhden ja kahden klusterin välissä: kaksi klusteria olisi luonnollinen valinta

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: Ryhmien välisen etäisyyden määräytyminen: “single”, “complete”, “average”

Hierarkkinen ryhmittely (2)
Ryhmien yhdistäminen

- 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

- 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.

Dynaaminen ryhmittely (2)

- Järkevä ryhmittelykriteeri on nyt:

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

- 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

Dynaaminen ryhmittely (3)

- Mitä siis voidaan tehdä?

c-means ryhmittelyalgoritmi (k-means, KM)

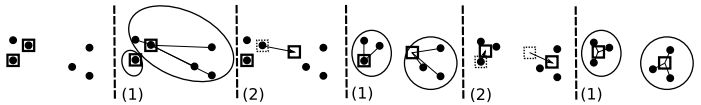
- Tunnetuin dynaaminen ryhmittelymenetelmä on ns. *c-means* -algoritmi (eli c:n keskipisteen algoritmi; vaihtoehtoinen nimi k-means, KM):
  1. Valitse satunnaisesti c pistettä joukosta $\mathbf{x}(1), ..., \mathbf{x}(n)$ pisteiksi $\mathbf{m}_1, ..., \mathbf{m}_c$;
  2. Toista kunnes pisteet $\mathbf{m}_1,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

c-means ryhmittelyalgoritmi (k-means, KM) (2)

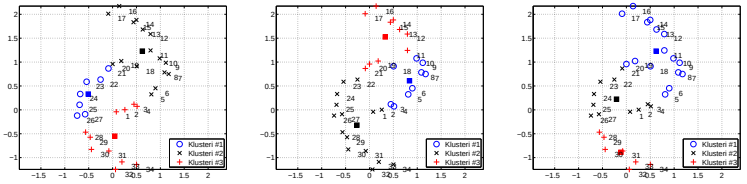
- Algoritmi kuvallisesti:



Kuva: 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.

c-means ryhmittelyalgoritmi (k-means, KM) (3)

- Koska aloituspisteet arvotaan, voi eri ajokerroilla tulla erilaisia tuloksia



Kuva: 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

$$J_i^{uusi} = J_i - \|\mathbf{x} - \mathbf{m}_i\|^2 \quad (2)$$

$$J_k^{uusi} = J_k + \|\mathbf{x} - \mathbf{m}_k\|^2. \quad (3)$$

c-means ryhmittelyalgoritmi (k-means, KM) (2)
Mitä algoritmissa tapahtuu?

- 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

c-means ryhmittelyalgoritmi (k-means, KM) (3)
Mitä algoritmissa tapahtuu?

- (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 $\mathbf{x} - \mathbf{m}_i$

c-means ryhmittelyalgoritmi (k-means, KM) (2)
Yhteys koodausteoriaan

- Virhe on sitä pienempi mitä pienempi on kriteeri J ja mitä enemmän koodivektoreita käytetään.

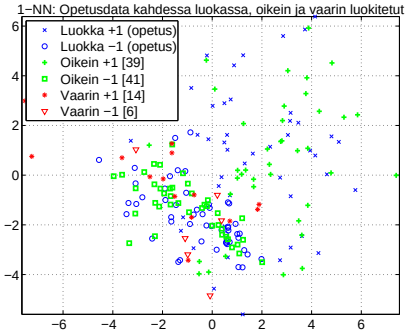
Yhteenveto

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.

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.

Esimerkki luokitteluvirheestä

Opetusdatasta hahmoalueet kNN-luokittimelle, k=1

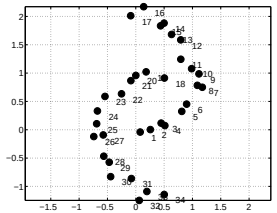


Kuva: kNN-luokitin kun $k = 1$. Kahden luokan opetusaineiston x ja o sinisellä. Testiaineiston 100 näytettä: “vihreä plus” luokiteltu rastiksi oikein, “vihreä neliö” luokiteltu oikein palloksi, “punainen tähti” on rasti joka luokiteltu virheellisesti palloksi, “punainen kolmio” on ympyrä joka luokiteltu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

Takaisin kalvoihin

Hierarkkinen ryhmittely: etäisyysmitan vaikutus

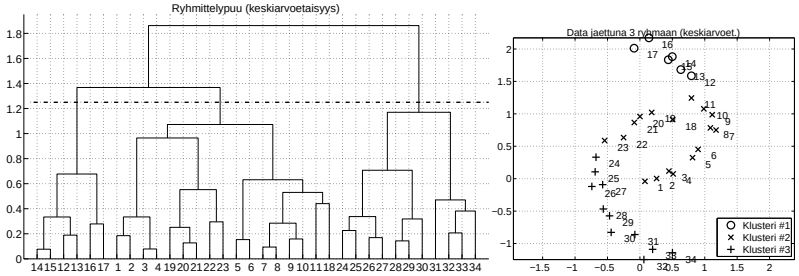
Olkoon käytössä kuvan mukainen data:



Kuva: Ryhmittelyesimerkin data.

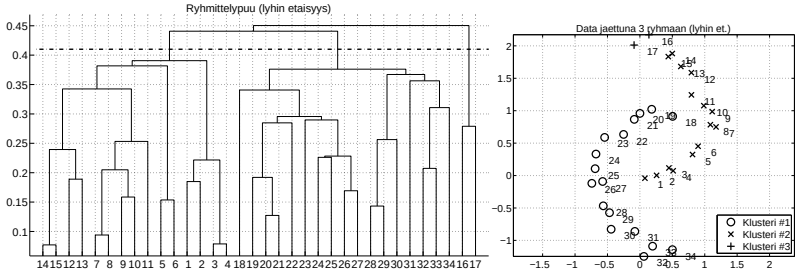
Tehdään tälle datalle hierarkkinen ryhmittely käyttäen ryhmien välisen etäisyydet mittaamiseen (a) keskiarvomittaa (“average”), (b) pienintä (“single”), (c) suurinta (“complete”) etäisyyttä. Ryhmittelyn jälkeen katkaistaan puu kolmen ryhmän (oksan) kohdalta ja piirretään muodostuneet ryhmät.

Hierarkkinen ryhmittely: etäisyysmitan vaikutus (2)



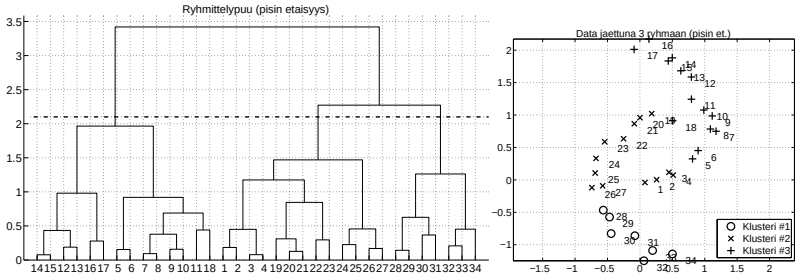
Kuva: Keskiarvoetäisyys (“average”): Pyöreähköt ryhmät.

Hierarkkinen ryhmittely: etäisyysmitan vaikutus (3)



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

Hierarkkinen ryhmittely: etäisyysmitan vaikutus (4)



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

Takaisin kalvoihin