Luku 7
Monimuuttujadatan kuvaamisesta

Seuraavassa ei puhuta ainoastaan monimuuttujadatan visualisoinnista kuin menetelmistä, joilla etsitään muuttujien välisiä korrelaatioita ja muuttuja-avaruudessa olevia samankaltaisten pisteiden muodostamia ryhmiä. Menetelmät soveltuvat parhaiten etsivän data-analyysin vaiheeseen, jolloin ei ole vielä selvää kuvaa eri muuttujien ja havaintojen välisistä riippuvuuksista. Kun riippuvuudet ovat joko teoreettisesti tai havaintojen analyysin kautta löydetty, voidaan paremminkin keskittyä suoraan havaintojen mallinnukseen (esim. tietyn mallin parametrien määrittämiseen). Tämän luvun menetelmät voivat kuitenkin olla hyödyksi myöhemminkin 'laadunvalvonnassa' – siis esim. tarkistettaessa ettei aineisto sisällä yllättäviä poikkeavia tai suorastaan virheellisiä havaintopisteitä.

7.1 Moniulotteinen skaalaus

Moniulotteinen skaalaus (*multidimensional scaling*) on yhteisnimitys menetelmille, joiden avulla havaintopisteiden keskinäinen sijainti yritetään selvittää, kun ainoastaan pisteiden väliset etäisyydet tunnetaan. Lähtökohtana on olemassaoleva mitta havaintopisteiden samanlaisuudelle tai erilaisuudelle (*similarity* <> *dissimilarity*). Jos pisteitä on n kappaletta, voidaan pisteiden väliset etäisyydet toteuttaa ainakin jollakin pisteiden sijoittelulla $n-1$ ulotteiseen avaruuteen. Käytännössä pisteet pyritään sijoittamaa alempiulotteiseen avaruuteen, useimmiten kaksiulotteiseksi piirrokseksi, jolloin annetut pisteiden samanlaisuudet toteutuvat ainoastaan likimain. Ratkaisuun voidaan luonnollisesti tehdä mielivaltaisia ortogonaalisia rotaatioita etäisyyksien siitä muuttumatta. Samoin origin paikka ei ole kiinnitetty, joskin origoksi voidaan valita esimerkiksi pisteiden massakeskipiste. Moniulotteinen skaalaus on hyödyllinen lähinnä silloin, jos havainnot muodostavat ratkaisussa selviä rakenteita, esim. selvästi erillisiä pistejoukkoja.

7.1.1 Klassinen skaalaus

Klassisessa skaalauksessa (*classical scaling*) pisteiden erilaisuuksen oletetaan olevan normaaleja euklidisia etäisyyksiä. Jos pisteiden paikat olisivat tunnettu matriisi X, voitaisiin etäisyydet laskea matriisin $B = XX^T$ avulla. Matriisin B elementit ovat muotoa $b_{rs} = \sum x_{rj}x_{sj}$, joten pisteiden s ja r välinen etäisyys on

$$d_{rs}^2 = \sum_{j=1}^{p} (x_{rj} - x_{sj})^2 = b_{rr} + b_{ss} - 2b_{rs}.$$

Tehtävänä on löytää matriisi B ja faktoroida se. Ratkaisu ei ole yksikäsitteinen, joten lisäehtona kiinnitetään origon sijainti

$$\sum_{r=1}^{n} x_{rj} = 0, \quad \forall j\,.$$

Matriisi B saadaan kaavasta

$$b_{rs} = -\frac{1}{2}\,[d_{rs}^2 - d_{r\cdot}^2 - d_{\cdot s}^2 + d_{\cdot\cdot}^2\,],$$

missä alaindeksi '$\cdot$' tarkoittaa keskiarvoa kyseisen indeksin yli.

Matriisi X saadaan laskettua matriisin B ominaisarvohajotelman avulla,

$$X = [f_1, f_2, ..., f_k],$$

missä kukin vektori f_i on matriisin B i:s ominaisvektori, jonka pituudeksi on valittu vastaavan ominaisarvon neliöjuuri

$$f_i = \sqrt{\lambda_i}\,\frac{\vec{a}_i}{|\vec{a}_i|}\,.$$

Klassinen skaalaus on itse asiassa identtinen myöhemmin esiin tulevan pääkomponenttianalyysin (PCA) kanssa, joka tosin vaatii varsinaisen datamatriisin, eikä pelkkiä pisteiden välisiä etäisyyksiä. (Jos PCA analyysi suoritetaan käyttäen havaintojen korrelaatiomatriisia, saadaan vastaava tulos klassisella skaalauksella normittamalla kunkin muuttujan varianssi ykköseen).

7.1.2 Järjestysasteikkoskaalaus

Nimensä mukaan järjestysasteikkoskaalauksessa (*ordinal scaling*) käytettävissä on ainoastaan tieto havaintopisteiden järjestyksestä eri muuttujien suhteen. Ratkaisussa on myös kuvan skaalaus huonosti määrätty.

Ratkaisu voidaan löytää iteratiivisesti. Merkitään annettuja pisteiden eroja δ ja ratkaisusta saatuja d. Sovitetaan näiden välille regressiosuora, ja luetaan siitä ennusteet $\hat{d}_{rs}$ pisteiden välisille etäisyyksille. Ratkaisu saadaan minimoimalla jännitettä (stress) S

$$S = \sum_{r<s} (d_{rs} - \hat{d}_{rs})^2 \,/\, \sum_{r<s} d_{rs}^2\,.$$

Ratkaisu on löytynyt, kun S on riittävän pieni (esim. 0.05, vaikka mitään yleispätevää rajaa on mahdotonta antaa).

Järjestysasteikkoskaalausta käytetään esim. sosiaalisissa tieteissä, missä muuttujat ovat usein järjestyslukuja. Sitä voidaan soveltaa kuitenkin myös silloin, kun pisteiden erilaisuuksille on olemassa jatkuvat lukuarvot, mutta erojen metriikasta ei ole varmuutta.

7.1.3 Moniulotteinen skaalaus *R*-ohjelmistossa

Kirjaston `MASS` rutiini `sammon` suorittaa moniulotteisen skaalauksen lähtien liikkeelle pisteiden eroja kuvaavasta matriisista. (Huom. kirjastossa `multiv` on samanniminen rutiini, mutta se tarvitsee alkuperäisen datamatriisin!). Ratkaisua haetaan optimointimenetelmällä, ja ajo on syytä toistaa useamman kerran. Näin nähdään, kuinka luotettava saatu tulos on, ja voidaan valita mahdollisesti erilaisista ratkaisuista paras.

Esimerkki 7.1. *Sammon mapping*-rutiinin soveltaminen hii.tab tiedoston muuttujien `T`, `FWHM`, `VLSR` (tiedoston sarakkeet 3, 5 ja 7) muodostamaan aineistoon.

```
library(MASS)
x _ read.table("hii.tab", header=TRUE)
m _ cbind(x[,3], x[,5], x[,7])
s _ sammon(dist(m))
plot(s$points, type="n")
text(s$points, labels=as.character(x[,1]))
```

Samasta kirjastosta löytyy myös toinen vastaava rutiini, `isoMDS`. Kuten rutiini `sammon` se suorittaa 'ei-metrisen' skaalauksen. Termi *non-metric scaling* tarkoittaa tässä samaa kuin *ordinal-scaling*, eli rutiini ei tee mitään oletusta metriikasta, jota käytetään kuvaamaan pisteiden erilaisuutta.

Samanlaiseen kaksiulotteiseen esitykseen pyrkivät myös nk. itseorganisoituvat kartat (SOM, self organizing maps), joista tunnetuin lienee Kohosen kehittämä neuroverkkosovellus. *R*-kirjasto `GeneSOM` sisältää menetelmän toteutuksen, mutta esim. visualisointirutiinit ovat vielä puutteelliset.

7.2 Rypäsanalyysi

Rypäsanalyysi (*cluster analysis*) on lähinnä etsivän data-analyysin menetelmä, jolla pyritään tunnistamaan ryhmät keskenään samankaltaisia havaintoja.

7.2.1 Algoritmeista

Moniulotteinen skaalaus tai muut projektiomenetelmät (esim. myöhemmin esiteltävä pääkomponenttianalyysi) voivat jo paljastaa havaintopisteiden keskittymiä, joita ei voi havaita projektioefektien vuoksi alkuperäisten muuttujien välisistä hajontakuvioista. Varsinaisesti tähän käytetään kuitenkin rypäsanalyysiä (*cluster analysis*). Rypäsanalyysin tehtävänä on löytää monimuuttujadatasta havaintojen muodostamia keskittymiä, kun käytössä on tieto eri havaintopisteiden välisistä etäisyyksistä. Rypäsanalyysi tunnistaa (mahdollisesti ennalta määrätyn suuruisen) joukon ryppäitä, ja ilmoittaa mihin ryppääseen kukin havaintopisteistä kuuluu. Rypäsanalyysi kuuluu etsivään data-analyysiin, ja voi olla hyvinkin valaisevaa silloin, kun ennalta ei tiedetä eri havaintojen välisistä suhteista.

Rypäsanalyysi voi edetä ylhäältä alaspäin, jolloin havaintoaineisto jaetaan yhä pienempiin ja pienempiin ryhmiin, kunnes lopulta jokainen ryhmä sisältää ainoastaan yhden havainnon. Kun joka askeleella pidetään muistissa mihin suurempaan ryhmään kukin ryhmä kuuluu, päädytään hierarkiseen esitykseen. Kukin havainto kuuluu ryhmään a, joka on osa ryhmää b jne. Toiset algortimit etenevät päinvastaiseen suuntaa, keräten eri tasoilla yhä suuremman joukon pisteitä yhtenäisiksi ryppäiksi. Jaottelu perustuu tiedolle pisteiden välisistä etäisyyksistä d tai paremminkin pisteiden erilaisuudesta.

Rypäsanalyysin tulos esitetään usein graafisesti puudiagrammina eli dendrogrammina. Puun juuri kuvaa koko aineistoa ja yksittäiset havaintopisteet ovat puun lehtinä. Kullakin tasolla alaryhmät yhdistetään, kun niiden välinen etäisyys alittaa tietyn rajan. Dendrogrammi implikoi tietyn etäisyyden d^* myös liitettävien alaryhmien jäsenille.

Ehkä yksinkertaisin rypäsalgoritmi on 'yhden linkin' menetelmä (*singe-link method*). Kullakin tasolla asetetaan raja d^*, ja samaan ryppääseen jaotellaan kaikki pisteet a, b, c, ..., joille pätee, että vastaavat etäisyyden d_{ab}, d_{bc}, ... ovat kaikki pienempiä kuin d^*. Algoritmi voi aloittaa yksittäisistä havaintopisteistä, ja yhdistää ryppäitä aina, kun joidenkin niihin kuuluvien pisteiden etäisyys alittaa senhetkisen rajan. Algoritmia kutsutaan myös lähimmän naapurin menetelmäksi. Menetelmän suurin ongelma on se, että se tuottaa helposti pitkiä linkkiketjuja, eivätkä ryppäät keskity selvästi eri osiin avaruutta. Muuten algoritmi käyttäytyy hyvin, eivätkö esim. pienet muutokset havainnoissa johda suuriin muutoksiin rypäsanalyysin tuloksissa.

Minimum spanning tree on sukua yhden linkin rypäsalgoritmille. Se ei tuota listaa ryppäistä, vaan ainoastaan kaikki pisteet yhdistävän verkon, jossa ei ole yhtään rengasta. Kukin pisteitä yhdistävä jana vastaa pisteiden välistä etäisyyttä, ja yhdistettävät pisteet valitaan niin, että janojen pituuksien summa saavuttaa minimin.

Muita rypäsanalyysin menetelmiä ovat mm.

- kaukaisimman naapurin menetelmä (*complete-link, furthest-neighbour*): kahden ryppään etäisyys määräytyy niiden keskenään kaukaisimpien havaintojen mukaan
- keskipistemenetelmä (*centroid-method*): etäisyys lasketaan ryppäiden keskipisteiden mukaan
- ryhmäkeskiarvo (*group-average method*): ryhmien etäisyys on summa niiden jäsenten pareittaisista etäisyyksistä
- Wardin menetelmä: ryppäitä yhdistetään peräjälkeen niin, että niiden sisäinen etäisyyksien neliösumma kasvaa mahdollisimman hitaasti
- Wishartin menetelmä: etsitään pisteitä, joiden ympärille piirretyn R-säteisen hyperpallon sisällä on vähintään k muuta pistettä. Sädettä R kasvatetaan, kunnes kaikki havainnot ovat sen sisällä.
- ositusalgoritmit (*partitioning*): ryppäiden lukumäärä on tavallisesti annettu, ja kaikki pisteet jaetaan eri ryppäisiin. Pisteitä voidaan vielä siirtää ryppäästä toiseen esim. näiden sisäisen varianssin pienentämiseksi.

Erilaisissa vertailuissa ovat hyvin menestyneet lähinnä yhden linkin menetelmä sekä Wardin menetelmä. Eräät lähteet kehottavat rypäsanalyysin sijasta mieluummin tyytymään havaintojen graafiseen tarkasteluun, esim. pääkomponenttianalyysin avulla

7.2.2 Rypäsanalyysin rutiinit *R*-ohjelmistossa

Rypäsanalyysin rutiineja on *R*:n kirjastoissa `cclust` ja `cluster`. Lisää löytyy kirjastoista `class` ja `rpart`, jotka keskittyvät nimenomaan havaintojen osittamiseen ja luokitteluun (*classification*).

Kokeillaan esimerkkinä `cclust` kirjaston samannimistä rutiinia. Kyseessä on *k-means* algoritmi, joka tarvitsee parametrinä etsittävien ryppäiden lukumäärän. Menetelmä luo aluksi annetun määrän ryppäitä, ja minomoi sitten niiden sisäistä etäisyyksien neliösummaa siirtäen havaintoja ryppäästä toiseen. Seuraavassa esimerkissä datasta etsitään kaksi rypästä ja koordinaattiakselien suuntaisissa projektiokuvissa eri ryppäisiin kuuluvat havainnot väritetään eri värein. Rutiini etsii aina parametrinä annetun määrän ryppäitä – riippumatta siitä, mitä datassa itse asiassa on!

Esimerkki 7.2. Kahden ryppään etsiminen taulukon `sgp. tab` muuttujien `UB`, `BV` ja `z` muodostamasta aineistosta. Rutiinina cclust kirjaston ohjelma `cclust`.

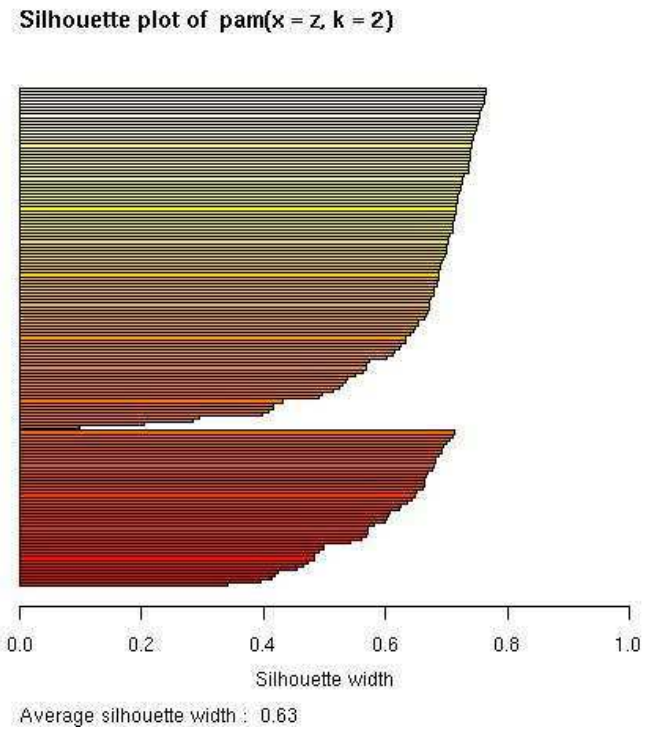
```
library(cclust)
x _ read.table("sgp.tab", header=TRUE)
x2 _ cbind(x$UB, x$BV, x$z)
c _ cclust(x2, 2)
plot(x2[,1:2], col=c$cluster)
plot(x2[,2:3], col=c$cluster)
plot(cbind(x2[,1],x2[,3]), col=c$cluster)
```

Kirjastossa `cluster` on useita erilaisia algoritmeja, joista esimerkiksi seuraavassa `pam` (*partitioning around medoids*). Algoritmi minimoi ryppäiden sisäisiä etäisyyksien summia (*ei* neliösummia kuten edellinen `cclust` algoritmi).

Esimerkki 7.3. Kahden ryppään etsiminen itse generoidusta aineistosta. Rutiinina kirjaston `cluster` ohjelma `pam`.

```
x _ c(rnorm(100, 0, 10), rnorm(50, 30, 15))
y _ c(rnorm(100, 0, 12), rnorm(50, 40, 15))
z _ cbind(x, y)

c _ pam(z,2)
summary(c)
plot(c)
```

Edellisessä esimerkissä komento `plot(c)` tuottaa sekä kaksiulotteisen kuvan, johon ryppäät on merkitty. Lisäksi piirretään nk. silhuettipiirroksen, jossa on vaakapylväs kullekin havainnolle. Pylväs on sitä pidempi, mitä varmemmin havainto on voitu sijoittaa tiettyyn ryppääseen. Lähellä nollaa olevat arvot tarkoittavat, että havainto voi yhtä hyvin kuulua johonkin toiseen ryppääseen.



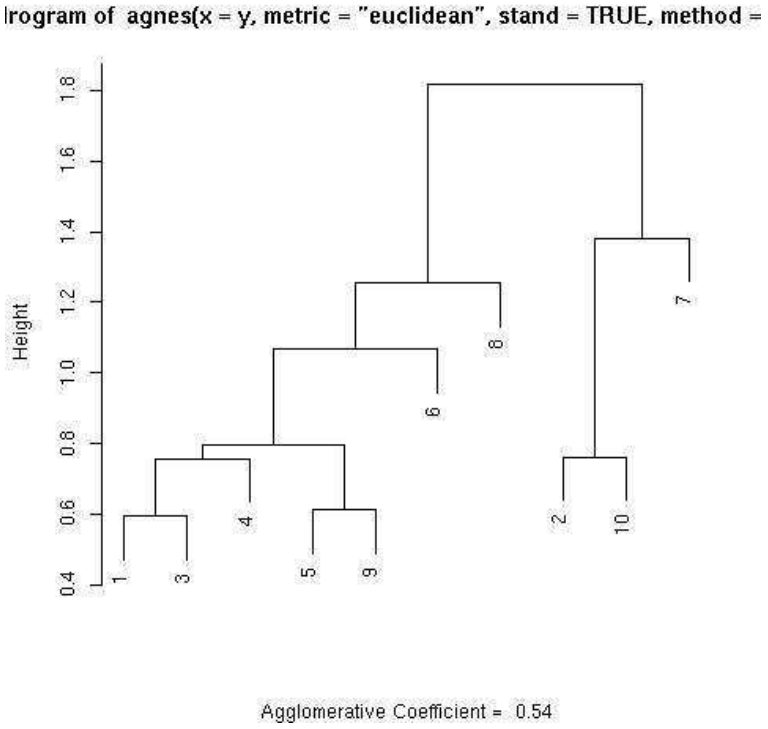
Kuva 7.1. Silhuettikuva. Mitä pidempi vaakapylväs, sitä luotettavammin yksittäinen havainto on voitu sijoittaa tiettyyn ryppääseen.

Edellä ryppäiden lukumäärä oli edeltä valittu. Koska algoritmit etsivät vain annetun määrän ryppäitä ($\sim$aineiston luokittelu), analyysi ei tuottanut myöskään dendrogrammia. Lopuksi kokeillaan yhtä hierarkista rypäsalgoritmia, kirjaston `cluster` rutiinia `agnes`. Itse asiassa ko. rutiinissa on valittavissa useita algoritmia, joissa ryppäiden etäisyyksiä mitataan joko lähimpien (*single-link*) tai etäisimpien (*furthest neighbour*) parien avulla, tai kolmantena vaihtoehtona käyttäen kaikkien parien keskimääräistä etäisyyttä. Wardin menetelmä on myös valittavissa. Seuraavassa esimerkissä käytetään yhden-linkin algoritmia eli verrataan vain ryppäiden lähimpiä naapureita.

Esimerkki 7.4. Hierarkinen rypäsalgoritmi agnes, jossa valittu käytettäväksi yhden linkin algoritmi.

```
x _ (read.table("sgp. tab", header=TRUE))[1:30,]
y _ cbind(x$z, x$UB)
a _ agnes(y, metric="euclidean",stand=TRUE, method="single")
plot(a)
```

Edellä valittiin aineistoon ainoastaan 30 ensimmäistä kohdetta, jotta tulosten graafisesta esityksestä saadaan vielä selvää. Komento `plot(a)` tuottaa jälleen kaksi kuvaa. Tällä kertaa näistä ensimmäinen ('banner') näyttää vaakasuuntaisena histogrammina tason, jolla kukin havainto esiintyy dendrogrammin lehtenä. Toinen piirros on varsinainen dendrogrammi, joka on esitetty myös seuraavassa kuvassa. Havainnot on identifioitu niiden järjestysnumeroilla.



Kuva 7.2. Dendrogrammi, joka esittää rutiinin `agnes` tekemää rypäsanalyysiä.

7.3 Pääkomponenttianalyysi

Tässä kappaleessa käytetään pääkomponenttianalyysiä (*Principal Component Analysis*, PCA) uuden koordinaatiston etsimiseen niin, että havainnot voidaan mahdollisimman hyvin havainnollistaa kaksiulotteisessa kuvassa. Tämän lisäksi pääkomponenttianalyysiä voidaan käyttää muuttujien välisten korrelaatioiden tutkimiseen. Tähän palataan vielä luvussa 8.1.

PCA-analyysi olettaa, että käsiteltävät muuttujat ovat kaikki samanarvoisia (ei erikseen riippuvia ja riippumattomia muuttujia), ja mieluusti myös skaalaltaan vertailukelpoisia. PCA suorittaa koordinaattiakselien ortogonaalisen kierron siten, että havaintopisteiden varianssi on suurin ensimmäisen akselin suhteen, ja jäännösvarianssi on aina suurin mahdollinen seuraavan koordinaattiakselin suhteen. Uudet muuttujat ovat myös täysin korreloimattomia. Kukin uusi koordinaattiakseli on lineaarikombinaatio alkuperäisistä muuttujista. Analyysi siis paljastaa ne muuttujien kombinaatiot, jotka parhaiten 'selittävät' havainnot. Toisaalta PCA hajotelmaa voidaan käyttää havaintojen tietosisällön tiivistämiseen jättämällä hajotelmasta pois viimeiset koordinaattiakselit, joiden suuntaan data sisältää enää vain vähän vaihtelua. Korreloimattomien muuttujien tapauksessa PCA analyysistä ei ole hyötyä - uudet koordinaattiakselit vastaavat tällöin suoraan alkuperäisiä muuttujia, ainoastaan varianssin perusteella uudelleen järjestettyinä. PCA ei sisällä varsinaista tilastollista mallia eikä siten tee oletuksia esim. muuttujien jakaumista. Kyseessä on ainoastaan matemaattinen operaatio koordinaatiston kierrolle.

Olkoon X havaintovektori

$$X^T = [X_1, ..., X_p],$$

joka koostuu p eri muuttujasta tehdyistä havainnoista (huom: tässä tavallisuudesta poikkeavasti muuttujat matriisin *riveinä* eikä sarakkeina!). Oletetaan, että havaintojen keskiarvo on μ, ja kovarianssimatriisi Σ. Tehtävänä on löytää uudet, korreloimattomat muuttujat (koordinaattiakselit) Y_i, niin että varianssi maksimoidaan aina seuraavan akselin suuntaan. Akseleita Y_i kutsutaan pääkomponenteiksi. Merkitään

$$Y_j = a_{1j} X_1 + ... + a_{\mathrm{pj}} X_p = a_j^T X\,,$$

jossa a_{ij} ovat määritettävät kertoimet, joille asetetaan lisäehto

$$a_j^T a_j = 1. \tag{7.1}$$

Tämä takaa muunnoksen ortogonaalisuuden, jolloin pisteiden väliset etäisyydet säilyvät muunnoksesta huolimatta. Lasketaan ensimmäinen pääkomponentti maksimoimalla varianssi

$$\sigma^2(Y_1) = \sigma^2(a_1^T X) = a_1^T \Sigma a_1, \tag{7.2}$$

edelleen vaatien että $a_1^T a_1 = 1$. Yleisesti funktion f maksimoiminen ehdon $g(x_1, ..., x_p) = c$ vallitessa onnistuu käyttäen Lagrangen kertoimia, λ,

$$\frac{\partial f}{\partial x_i} - \lambda \frac{\partial g}{\partial x_i} = 0, \quad i = 1, ..., p \tag{7.3}$$

Ehto voidaan kirjoittaa myös muotoon $\partial L / \partial x = 0$, ottamalla käyttöön uusi funktio

$$L(x) = f(x) - \lambda [g(x) - c]. \tag{7.4}$$

Ensimmäinen pääkomponentti saadaan tämän mukaisesti yhtälöstä

$$\frac{\partial}{\partial a_1}[L(a_1)] = \frac{\partial}{\partial a_1}[a_1^T \Sigma a_1 - \lambda(a_1^T a_1 - 1)] = 2\Sigma a_1 - 2\lambda a_1 = 0, \tag{7.5}$$

eli toisin sanoen

$$(\Sigma - \lambda I)\, a_1 = 0. \tag{7.6}$$

Lineaarialgebrasta tiedetään, että yhtälöllä on ei-triviaali ratkaisu ainoastaan, kun

$$|\Sigma - \lambda I| = 0, \tag{7.7}$$

eli λ on matriisin Σ ominaisarvo. Matriisilla on (yleensä) p ominaisarvoa, jotka oletetaan tässä järjestetyiksi $\lambda_1 > \lambda_2 > \ldots > \lambda_p$. Edellisestä yhtälöstä ja normalisointivaatimuksesta $a_1^T a_1 = 1$ seuraa, että varianssi on

$$\sigma^2(a_1^T X) = a_1^T \Sigma a_1 = a_1^T \lambda I a_1 = \lambda. \tag{7.8}$$

Etsitty Lagrangen kerroin on siis matriisin Σ suurin ominaisarvo (*eigenvalue*), ja ensimmäinen pääkomponentti on $a_1^T X$, missä a_1 on tätä vastaava matriisin Σ ominaisvektori (*eigenvector*).

Muiden pääkomponenttien johtaminen tapahtuu samaan tapaan. Lisäehtona on korreloimattomuus edellisten komponenttien kanssa,

$$\mathrm{Cov}(Y_2, Y_1) = \mathrm{Cov}(a_2^T X, a_1^T X) = E[a_2^T (X - \mu)(X - \mu)^T a_1] = a_2^T \Sigma\, a_1 = 0. \tag{7.9}$$

Seuraavan komponentin johdossa tarvitaan kaksi Lagrangen kerrointa, jolloin

$$L(a_2) = a_2^T \Sigma a_2 - \lambda(a_2^T a_2 - 1) - \delta(a_2^T a_1 - 0). \tag{7.10}$$

Lopputulos on, että seuraava λ_i on aina seuraavaksi suurin matriisin Σ ominaisarvo, ja a_i on tätä vastaava ominaisvektori. Kirjoitetaan ominaisvektorit matriisiksi, $A = [a_1, \ldots, a_p]$, ja pääkomponentit ovat $Y = A^T X$. Pääkomponentit ovat määritelmän mukaan korreloimattomia, ja Y:n kovarianssimatriisin diagonaalielementit ovat λ_i. Edelleen, koska kyseessä on lineaarimuunnos, voidaan kirjoittaa

$$\Lambda = A^T \Sigma A \iff \Sigma = A \Lambda A^T. \tag{7.11}$$

PCA antaa siis hajotelman alkuperäisten havaintojen kovarianssimatriisille. Edellä nähtiin, että matriisin Σ ominaisarvot kertovat suoraan eri pääkomponentteja vastaavien varianssien suuruuden. Tämän perusteella ensimmäiset k pääkomponenttia selittävät osan

$$\sum_{j=1}^{k} \lambda_j \Big/ \sum_{j=1}^{p} \lambda_j \tag{7.12}$$

koko varianssista. Tämän perusteella voidaan alkuperäisen havaintomatriisin informaatiota tiivistää - hajotelmaan otetaan mukaan ainoastaan ne ensimmäiset pääkomponentit, jotka vielä selittävät merkittävän osan koko varianssista.

Edellä pääkomponenttihajotelma laskettiin käyttäen havaintojen kovarianssimatriisia. Jos alkuperäiset muuttujien skaalat poikkeavat huomattavasti toisistaan, saadaan esim. ensimmäiseksi pääkomponentiksi likimain se muuttuja, jonka kohdalla arvojen hajonta on suurinta. Tästä syystä voi olla järkevää normittaa muuttujat ennen PCA analyysiä, eli jakaa kunkin muuttujan havainnot ko. muuttujan keskihajonnalla. Käytännössä tämä tarkoittaa PCA hajotelman laskemista kovarianssimatriisin sijasta korrelaatiomatriisista. Näin saadut pääkomponentit eivät enää ole samoja. Yleisemminkin PCA analyysin lopputulos riippuu aina muuttujien skaalauksesta. Myös tekijöiden a_i etumerkki on valintakysymys, ja tavallisesti valitaan positiiviset a:n arvot.

Pääkomponenttien Y keskiarvo ei ole yleensä suoraan nolla, mutta usein näihin lisätään vakiot, niin että keskiarvo saadaan nollaksi

$$Y = A^T (X - \bar{x}\ ). \tag{7.13}$$

Tässä $\bar{x}$ on otoskeskiarvojen muodostama vektori. Yksittäiselle havainnolle x_r nk. *component score* on

$$y_r = A^T (x_r - \bar{x}\ ). \tag{7.14}$$

Tämä kertoo pisteen sijainnin ominaisvektorien muodostamassa uudessa koordinaatistossa. Käänteinen muunnos on

$$X = AY + \bar{x}\,. \tag{7.15}$$

Joskus ominaisvektoreiden a_j sijasta käytetään vektoreita $a^* = \lambda_j^{1/2} a_j$. Näiden vektoreiden pituus on siis λ_j eikä 1. Olettaen, että matriisi X on keskistetty, on käänteismuunnos

$$X = AY = A\Lambda^{1/2}\Lambda^{1/2}Y = A\Lambda^{1/2}Y^* = CY^*\,. \tag{7.16}$$

Matriisin C elementtejä kutsutaan nimellä *component loadings* - analogisesti faktorianalyyttisen mallin *factor loadings* kanssa. Ne kertovat, miten alkuperäiset muuttujat selitetään uusien koordinaattiakselien (Y^*) avulla. Toisaalta voidaan kirjoittaa korrelaatiomatriisi

$$\mathrm{Corr}(Y\,, X) = \Lambda^{1/2}\, A^T = C^T. \tag{7.17}$$

Siis, kun C on laskettu korrelaatiomatriisista, mittaavat matriisin elementit korrelaatioita uusien komponenttien ja alkuperäisten muuttujien välillä. Nämä ovat nk. komponenttikorrelaatioita (*component correlations*).

Pääkomponenttihajotelmaa käytetään usein havaintomatriisin oleellisimpien piirteiden esittämiseen. Jos suuri joukko kovarianssimatriisin (tai korrelaatiomatriisin) ominaisarvoista on pieniä, nämä voidaan korvata nollilla, jolloin laskettu hajotelma antaa tiivistetyn (likimääräisen) esityksen havainnoista. Yleisimmässä tapauksessa jätetään jäljelle vain kaksi ensimmäistä pääkomponenttia, jolloin havainnot voidaan esittää kaksiulotteisena hajontakuviona, jossa koordinaattiakseleina ovat ensimmäiset kaksi pääkomponenttia. Muunnoksen jälkeen halutaan tarkastella itse asiassa kahta seikkaa: miten uudet koordinaattiakselit suhtautuvat alkuperäisiin muuttujiin, ja miten havaintopisteet suhtautuvat uusiin koordinaattiakseleihin.

Havaintopisteiden sijainnin ensimmäisten pääkomponenttien muodostamassa koordinaatistossa saadaan selville laskemalla havaintojen *component score* arvot (siis $A^T x_r$). Näin saadusta kuvasta voidaan yrittää löytää esim. poikkeavia arvoja tai kuvan eri osiin keskittyviä havaintojoukkoja. Tällaisesta analyysistä käytetään yleisesti termejä pääkomponenttianalyysi (*principal co-ordinate analysis*) tai klassinen skaalaus (*classical scaling*). Kyseisessä projektiossa nähdään suurin mahdollinen havaintopisteiden hajonta. Jos havainnoissa on selvää rakennetta ja/tai poikkeavia arvoja, ne nähdään todennäköisimmin juuri tästä kuvasta.

R:ssä pääkomponenttihajotelma löytyy mm. kirjastosta `pcurve`. Rutiinin nimi on `pca`. Argumentteina ovat data-matriisi, sekä tieto siitä, pitääkö matriisi keskistää ja/tai skaalata ennen analyysiä. Jos matriisi skaalataan, tehdään analyysi käyttäen korrelaatiomatriisia kovarianssimatriisin sijaan (ja tulokset ovat erilaisia). Rutiini palauttaa listan, jossa ovat *component loading*-arvot (havaintopisteiden koordinaatit pääkomponenttien muodostamassa koordinaatistossa), diagonaalimatriisi ominaisarvoista sekä ominaisvektorit sisältävä matriisi.

Seuraavassa esimerkissä käytetään kuitenkin toista, kirjastosta `mva` löytyvää rutiinia `princomp`. Tämä on hieman monipuolisempi kuin kirjaston `pcurve` vastaava rutiini. Tarkastellaan esimerkkinä tiedostoa `hii.tab`, josta otetaan analysoitavaan matriisiin kaikki muuttujat galaktisia koordinaatteja lukuunottamatta. Lasketaan pääkomponenttihajotelma, ja esitetään havainnot kahden ensimmäisen pääkomponentin muodostamassa koordinaatistossa. Kirjasto `princomp` sisältää myös rutiinin `biplot`, ja tämän avulla voidaan samaan kuvaan piirtää myös alkuperäisiä muuttujia kuvaavat vektorit. Nämä näyttävät, miten ensimmäiset pääkomponentit korreloivat kunkin muuttujan kanssa.

Esimerkki 7.5. Tiedoston `hii.tab` pääkomponenttianalyysi kovarianssimatriisin ja korrelaatiomatriisin pohjalta. Komennon biplot avulla nähdään havaintojen jakauma kahden ensimmäisen pääkomponentin muodostamassa koordinaatistossa sekä muuttujien korrelaatiot näiden pääkomponenttien kanssa.

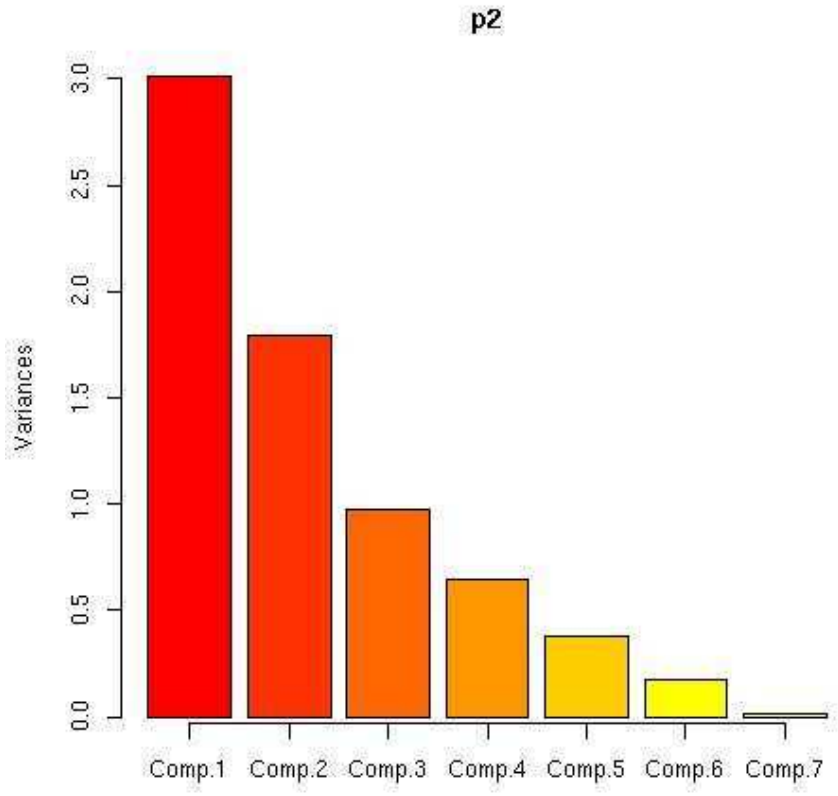
```
library(mva)

# x[,3:9] => muuttujat T dT FWHM dFWHM VLSR dVLSR RMS
x  _ read.table("hii.tab", header=TRUE)
m  _ as.matrix(x)[,3:9]
p1 _ princomp(m)
plot(p1)              # histogrammi variansseista
biplot(p1)            # biplot

# toistetaan sama käyttäen korrelaatiomatriisia
p2 _ princomp(m, cor=TRUE)
plot(p2)
```

```
biplot(p2)
```

Esimerkissä komennot `plot(m)` näyttävät pääkomponentteja vastaavien ominaisarvojen suuruuden histogrammina. Tässä tapauksessa pääkomponentteja on kaikkiaan 7 kappaletta (= muuttujien lukumäärä). Jos analyysi suoritetaan käyttäen havaintojen kovarianssimatriisia, selittävät ensimmäiset kaksi ensimmäistä pääkomponenttia lähes kaiken havaintomatriisin vaihtelun

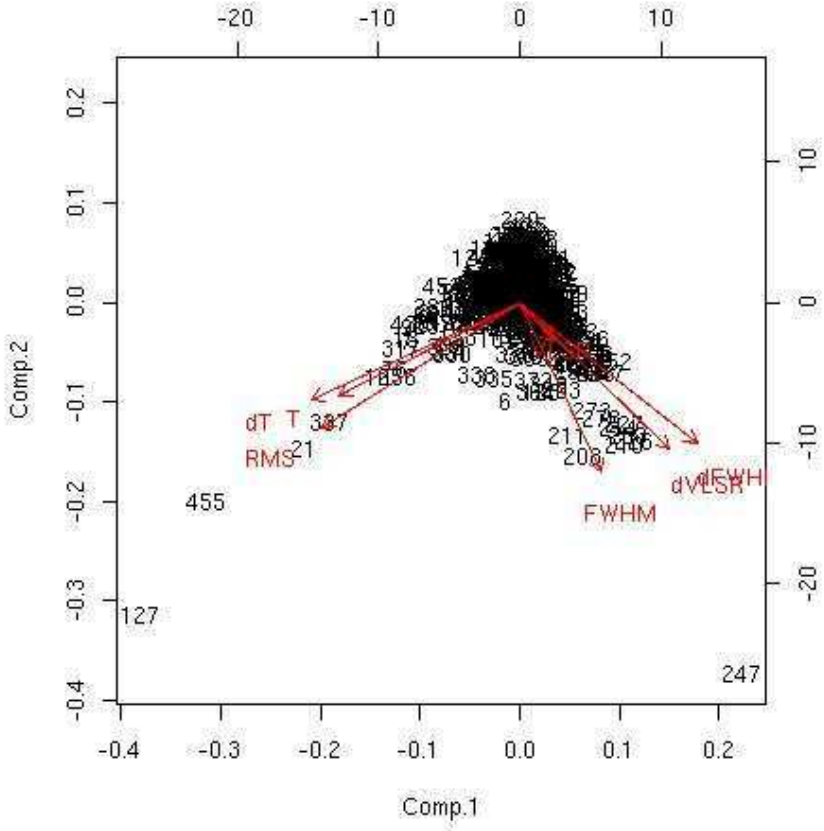
```
> sum(p1$sdev[1:2])/sum(p1$sdev)
[1] 0.944643
```

Tämä johtuu kuitenkin ainoastaan muuttujien erilaisista skaaloista, jolloin erityisesti antennilämpötila saa muita suurempia numeerisia arvoja. Kun analyysi toistetaan käyttäen korrelaatiomatriisia, on ominaisarvoja kuvaava piirros seuraavan kuvan mukainen.



Kuva 7.3. HII-alueista tehtyjen havaintojen kovarianssimatriisin ominaisarvot. Nämä ovat suoraan verrannollisia siihen osuuten kokonaisvarianssista minkä vastaavat pääkomponentit selittävät.

Tässä tapauksessa kaksi ensimmäistä pääkomponenttia selittävät ainoastaan 51% kokonaisvarianssista. Seuraavassa kuvassa on kovarianssimatriisista lasketun pääkomponenttihajotelman antama *biplot*.



Kuva 7.4. Kirjaston `mva` rutiinin `biplot` tuottama piirros. Tässä ovat samassa kuvassa esitettyinä alkuperäisten havaintovektoreiden (=havaintojen indeksit) sekä alkuperäisten muuttujien (=vektorit) projektiot kahden ensimmäisen pääkomponentin muodostamaan koordinaatistoon.

Kuvasta näkyy ensinnäkin alkuperäisten muuttujien ryhmittyminen (tässä projektiossa) kahteen ryhmään (`dT, T, RMS` toisaalla ja toisaalla `FWHM, dVLSR, dFWHM`). Pareittaiset korrelaatiot nähtäisiin tietysti suoraan myös normaalista korrelaatiodiagrammasta. Ensimmäiset kaksi pääkomponenttia selittävät ainoastaan

puolet kokonaisvaihtelusta, joten pääkomponenttianalyysi ei sen vuoksi toimi tässä tapauksessa erityisen hyvin. Edellä analyysiin otettiin mukaan myös suureet kuten `dT, dVLSR`, jne. joilla ei ole lähteiden kannalta fysikaalista merkitystä. Tällainen tarkastelukin on usein tarpeen, jotta datan ominaisuuksista saataisiin riittävän monipuolinen kuva. Tässä tapauksessa kuva 7.4 kertoo mm. että spektrien kohina korreloi havaitun viivan intensiteetin kanssa (heteroskedastinen aineisto!), ja että esim. säteisnopeuden virhearviot kasvavat (odotetusti) viivanleveyden kasvaessa. Nämä ovat seikkoja, jotka mahdollisessa jatkomallinnuksessa pitää ottaa huomioon. Uusi projektio voi paljastaa myös uusia poikkeavia havaintoja (esim. havainnot 127 ja 247), jotka eivät välttämättä näkyisi yhtä selvästi korrelaatiodiagrammoissa.

7.4 Muita menetelmiä

Parhaimpien projektiosuuntien eysimiseksi on myös PCA:n lisäksi useita ei-lineaarisia menetelmiä. Aiemmin kuvaillut *Sammon mapping* ja SOM toimivat pisteiden etäisyyksien pohjalta, eikä niiden tarjoamassa projektiokuvassa havaintojen keskinäinen sijainti ole välttämättä aivan oikea. Varsinaisissa projektiomenetelmissä kyse on sensijaan nimenomaan oikean koordinaatiston kierron löytämisestä. Esimerkiksi aiemmin vastaan tulleessa *Gobi* ohjelmistossa on muiden projektiomenetelmien lisänä automaattinen projektion etsintärutiini (*projection pursuit*). Oikean projektion etsintä voidaan toteuttaa iteratiivisesti niin, että maksimoidaan jotakin haluttua suuretta. Esimerkiksi varianssin maksimointi ei välttämättä tuota samaa kuvaa kuin pääkomponenttianalyysi - siis jos maksimointia ei suoriteta ensin yhden ja vasta sen jälkeen toisen akselin suhteen.

Kirjasto `pcurve` tarjoaa PCA analyysin ei-lineaarisen yleistyksen, jolla voidaan etsiä havaintopisteiden välistä kulkeva käyrä, jonka suhteen havaintojen varianssi on pienin. Rutiini piirtää kaksiulotteisen projektiokuvan ratkaisusta ja havaintopisteiden poikkeamat käyrästä, jolloin x-akselilla on muuttujana etäisyys käyrän alkupisteestä.

Muita aihepiiriin liittyviä rutiineja löytyy mm. kirjastoista `CoCoAn` (correspondence analysis), `PTAk` (Principal Tensor Analysis), `maptree` (regression tree plots ym.) ja `rpart` (recursive partitioning).

Luku 8
Komponenttianalyysi

8.1 Pääkomponenttihajotelma

Tässä palataan pääkömponenttihajotelman käyttöön. Tällä kertaa sitä ei käytetä ainoastaan visualisoinnin apuna, vaan laskettujen ominaisvektorien ja pääkomponenttien katsotaan esittävän havainnoissa olevia komponentteja (faktoreita), joiden lineaarikombinaationa havaintomatriisi selitetään.

PCA laskee hajotelman

$$X = AY\,,$$

jossa matriisin Y vektorit ovat keskenään ortogonaalisia. Pääkomponenttien lukumäärä on yhtä suuri kuin pienempi matriisin X rivien lukumäärä, jonka ajatellaan tässä vastaavan myös havaittujen muuttujien lukumäärää. Kovarianssimatriisin (tai korrelaatiomatriisin) ominaisarvot λ_i kertovat suoraan suhteellisen osuuden havaintojen vaihtelusta, jonka vastaava pääkomponentti selittää. Graafisesti λ_i arvoja tarkasteltiin *scree*-piirroksen avulla. Jos ominaisarvoista useat ovat pieniä, voidaan vastaava osa hajotelmasta jättää huomiotta, ja havaintomatriisi voidaan approksimoida ehkä vain muutaman pääkomponentin avulla. Tässä tapauksessa voidaan tehdä hypoteesi, että jäljelle jäävillä pääkomponenteilla on myös jokin fysikaalinen tulkinta. Monimutkaisen havaintomatriisin sijasta riittää tarkastella vain muutamaa pääkomponettia, ja sitä, miten ne ilmenevät kussakin yksittäisessä havainnossa.

Otetaan esimerkiksi spektriviivahavainnoista tehdyt kartat (tai vain joukko erillisiä spektrihavaintoja). Havaintomatriisi koostuu spektreistä, jolloin matriisin yksi dimensio vastaa yksittäisessä spektrissä esiintyvien spektrikanavien lukumäärää. Toinen dimensio on spektrien lukumäärä, eli matriisin kukin sarake sisältää yhdessä kartan pisteessä mitatun spektrin. Tehdään matriisin pääkomponenttihajotelma, ja poistetaan hajotelmasta se osa, jota vastaavat ominaisarvot λ_i ovat 'pieniä'. Jäljelle jäävät pääkomponenttien (matriisin Y sarakkeiden) voidaan tulkita kuvaavan spektrien perusmuotoja, joiden lineaariyhdistelynä kukin havaittu spektri voidaan likimääräisesti esittää. Matriisin A elementit kertovat puolestaan sen, kuinka voimakkaana kukin näistä spektrin perusmuodoista esiintyy kussakin havaitussa spektrissä.

Pääkomponenttien esittämille spektrimuodoille ei yleensä löydy suoraa fysikaalista tulkintaa. Jos esim. kartassa esiintyy kaksi eri säteisnopeudella olevaa lähdettä, ovat näiden lähteiden spektrit lineaariyhdistelmiä pääkomponenttien esittämistä muodoista (ja päinvastoin). Matemaattisestihan

$$AY = AR^{-1}RY = A^*Y^*, \tag{8.1}$$

missä R voi olla mikä tahansa käänteismatriisin omaava matriisi. Tällaisia muunnoksia kutsutaan rotaatioiksi, ja käytännössä alkuperäiseen hajotelmaan voidaan tehdä ääretön määrä erilaisia muunnoksia. Haettu fysikaalisesti oikea esitys on ainoastaan yksi näistä.

Pääkomponenttihajotelman käytöllä saavutetaan kuitenkin myös tiettyjä etuja. Hajotelma antaa sovituksen, jossa (esimerkin tapauksessa) matriisin Y spektrikomponenttien muodot ovat täysin vapaita eli ne estimoidaan lähtien ainoastaan data-matriisin sisältämästä tiedosta. Sovitus tehdään globaalisti, kaikkia havaintoja hyväksi käyttäen. Yksittäisissä spektreissä nämä peruspiirteet voivat hävitä kohinaan, mutta pääkomponenttihajotelma voi silti pystyä poimimaan ne esiin. Riittää, että kohinaisia spektrejä on riittävän monta, ja spektrien peruskomponenttien voimakkuuden vaihtelevat eri spektreissä. PCA voi siis löytää havainnoista heikkoja piirteitä, joita ei voida nähdä yksittäisistä havainnoista. Toisaalta saatujen peruspiirteiden fysikaalinen tulkinta on vaikeaa lukuisista mahdollisista rotaatioista johtuen.

PCA:ta voidaan käyttää myös yksinkertaiseen havaintojen esityksen tiivistämiseen. Jos 'tärkeiden' pääkomponenttien lukumäärä on paljon pienempi kuin alkuperäisten muuttujien lukumäärä, voidaan havaintojen oleellinen sisältö (ilman kohinaa) esittää PCA:n avulla tiiviimmin. Kriittinen kysymys on tietenkin, milloin ominaisarvot λ_i ovat niin pieniä, että ne edustavat vain kohinaa ja voidaan korvata nollilla. Kysymykseen ei ole olemassa tyhjentävää vastausta, mutta esim. *scree-plot* antaa tähän hyvän arvion. Kun ominaisarvot piirretään suuruusjärjestyksessä, pienenevät lukuarvot aluksi nopeasti. Jossakin vaiheessa pieneneminen hidastuu, ja lopussa ominaisarvot muodostavat lineaarisesti laskevan sarjan. Lineaarinen osuus aiheutuu pelkästä kohinasta, joten viimeinen merkittävä ominaisarvo löytyy juuri ennen (tai 'hieman ennen') lineaarisen osuuden alkua.

Eräs huomionarvoinen seikka on se, että PCA analyysissä havaintomatriisi tavallisesti (ja joskus automaattisesti) keskistetään ennen analyysin tekoa. Samalla katoaa informaatio eri muuttujien absoluuttisista arvoista – ainoastaan muuttujien väliset kovarianssit ovat oleellisia.

Pääkomponenttianalyysiä on käytetty mm. neuroverkkomenetelmien yhteydessä, jolloin spektrien luokitteluun käytetään ei koko spektriaineistoa, vaan siitä lasketun pääkomponenttihajotelman esim. paria ensimmäistä ominaisarvoa vastaavaa osuutta. Kun spektrien peruskomponentit (ominaisvektorit) on kiinnitetty, riittää yksittäisen spektrin kuvaamiseen muutaman kertoimen joukko.

Esimerkki 8.1. *Russeil D., Juvela M., Lehtinen K., Mattila K., Paatero P. 2002,* Kinematics and morphology of L1642*, A&A*

- PCA hajotelman sovellus spektrikarttojen analyysiin.

Esimerkki 8.2. *Cabanac R.A., de Lapparent V., Hickson P., 2002,* Classification and redshift estimation by principal component analysis*, A&A 389, 1090*

- PCA analyysi spektrien luokittelun apuna.

8.2 Singulaariarvohajotelma

Singulaariarvohajotelma laskee matriisihajotelman

$$X = USV^T, \tag{8.2}$$

jossa matriisit U ja V ovat ortogonaalisia. U:n sarakkeita kutsutaan vasemmanpuoleisiksi *singulaarivektoreiksi*, ja vastaavasti matriisin V sarakkeet (V^T rivit) ovat oikeanpuoleisia singulaarivektoreita. Sekä matriisin U että matriisin V sarakkeet ovat ortonormaaleja (sarakkeen pistetulo itsensä kanssa on 1 ja muiden sarakkeiden kanssa nolla). Matriisi S on diagonaalimatriisi, jonka diagonaalielementit s_i ovat matriisin X *singulaariarvoja*.

Singulaariarvohajotelmalla on lukuisia käyttökohteita, erityisesti koska se voidaan aina laskea luotettavasti (joskin hitaammin kuin eräät muut matriisihajotelmat). Sitä voidaan käyttää esimerkiksi matriisin asteen (tai *efektiivisen* asteen) määrittämiseen. Matriisin aste on yhtäsuuri kuin pienempi sen lineaarisesti riippumattomien rivien ja sarakkeiden lukumääristä. Singulaariarvot ovat matriisin S diagonaalilla pienenevässä järjestyksessä, ja matriisin aste on yhtä suuri sen (merkittävästi) nollasta poikkeavien singulaariarvojen lukumäärä.

effective degree

Singulaariarvohajotelmaa voidaan käyttää matriisin X tietosisällön tiivistämiseen, täsmälleen samaan tapaan kuin pääkomponenttihajotelmaa. Matriisin singulaariarvot kertovat suoraan, kuinka paljon kokonaisvaihtelusta vastaava osa hajotelmasta (siis vastaava matriisin U sarake ja matriisin V^T rivi) selittää. Jos tietystä indeksistä eteenpäin singulaariarvot korvataan nollilla, antaa jäljelle jäävä osa hajotelmasta pienimmän neliösumman sovituksen datamatriisiin.

Matriisin kääntäminen onnistuu erittäin helposti, jos SVD hajotelma on tiedossa. Yhtälöryhmän

$$Ax = b \iff USV^T x = b$$

ratkaisu on

$$x = (USV^T)b = VS^{-1}U^T b$$

(ortogonaaliselle matriisille matriisin transpoosi on samalla käänteismatriisi). SVD hajotelmaa voidaan siis käyttää myös tavallisten lineaaristen yhtälöryhmien ratkaisemiseen. Se on kuitenkin hidas ($\sim mn^2 + \frac{11}{2}n^2$ operaatiota, jossa m ja n ovat matriisin A dimensiot), joten yleensä käytetään nopeampia menetelmiä. Jos yhtälöryhmässä on enemmän yhtälöitä kuin muuttujia, ei yhtälöryhmällä (yleensä) ole ratkaisua (ylideterminstinen yhtälöryhmä). Tässä tapauksessa löytyy pienimmän neliösumman ratkaisu, kun matriisista S otetaan mukaan ainoastaan se osa, jossa singulaariarvot ovat positiivisia. Silloin voidaan käänteismatriisikin S^{-1} laskea (diagonaalimatriisin käänteismatriisi saadaan kirjoittamalla diagonaalille matriisielementtien käänteisluvut – itse asiassa siis asetetaan $0^{-1} \to 0$). Yleisemmin voidaan tietysti tyytyä pienempäänkiin määrään singulaariarvoja (typistetty eli *truncated* SVD hajotelma). SVD:tä käyttäen saadaan pienimmän neliösumman ratkaisu myös vajaa-asteiselle yhtälöryhmälle. Tässä tapauksessa pienimmän neliösumman ratkaisu ei ole yksikäsitteinen, mutta SVD:n avulla saadaan se ratkaisu, jolle ratkaisuvektorin pituus, $|x|$, on pienin.

Singulaariarvohajotelmaa voidaan käyttää poikkeavien havaintojen etsimiseen. Palataan spektriesimerkkiin, ja oletetaan, että matriisin X rivit esittävät havaittuja spektrejä. Jos havainnot voidaan esittää muutaman komponentin yhdistelynä, on nollasta merkittävästi poikkeavia singulaariarvoja vain muutama. Matriisin V^T rivit kuvaavat spektrien peruskomponentteja, ja matriisin U kukin sarake kertoo, kuinka paljon vastaavaa komponenttia on kussakin havaitussa spektrissä. Jos spektrien joukossa on poikkeavia spektrejä, poikkeavuus on esitettävä uusien matriisin V^T rivien avulla. Jos pienempiä singulaariarvoja vastaavassa matriisin U sarakkeessa on suuria arvoja vain muutaman spektrin kohdalla, tiedetään näiden poikkeavan muista, ja vastaava matriisin V^T rivi kertoo, minkälainen poikkeama on.

SVD esittää havitut spektrit lineaariyhdistelynä matriisin V^T riveistä. Jos spektriprofiili on kaikissa spektreissä identtinen (ainoastaan amplitudi vaihtelee), voidaan havainnot esittää (kohinaa lukuunottamatta) yhdellä matriisin V^T rivillä ($\sim$ profiilit) ja yhdellä matriisin US sarakkeella ($\sim$ amplitudit). Jos yksittäisessä spektrissä on poikkeama, vaatii sen esitys uuden V^T:n rivin ja U:n sarakkeen mukaanottamista. Näitä vastaava, tässä tapauksessa järjestyksessä toinen singulaariarvo kertoo poikkeaman suuruuden. Yleisessä tapauksessa tilanne ei ole aina näin selkeä, vaan eri spektrien poikkeavuudet ja kohina sekoittuvat esityksessä. Joka tapauksessa SVD on hyödyllinen väline poikkeavuuksien etsimiseen (tai sen varmistamiseen, ettei niitä esiinny). Samoin kuin PCA sovitus tehdään käyttäen koko havaintoaineistoa, joten esiin voi tulla myös kohinan alle jääviä piirteitä, jotka kuitenkin toistuvat riittävän monessa spektrissä, niin että ne voidaan nähdä SVD hajotelmasta. Piirteet eivät välttämättä ole peräisin havaittavasta kohteesta, vaan voivat olla esim. havaintolaitteiston aiheuttamia systemaattisia virheitä.

Tarkastellaan esimerkkinä SVD-hajotelman käyttöä datamatriisin sisältämän tiedon tiivistämiseen. Tässä merkityksessä SVD-hajotelmaa voitaisiin käyttää vaikkapa lajittelutehtävän esikäsittelyvaiheena.

Esimerkki 8.3. Singulaariarvohajotelman soveltaminen kuvan analysointiin. Poistamalla hajotelmasta pienimpiä singulaariarvoja vastaava osa, saadaan alkuperäiselle kuvalle approksimaatio, jossa esim. kohinan osuus on pienentynyt.

```
par(ask=TRUE)
library(pixmap)
pic _ read.pnm("CygA2.pnm")
s   _ svd(pic) ;
D   _ diag(s$d) ;              # vektori s$d -> diag.matriisi
N   _ dim(D)[1] ;
D2  _ D ;
D2[30:N, 30:N] _ 0.0 ;         # poistetaan 'merkityksetön' osa
B _ s$u %*% D2 %*% t(s$v) ;    # approksimaatio
image(z=B, col=heat.colors(200))
```

Edellä SVD hajotelma laskettiin tiedostosta luetulle kuvalle. Jos pienimpien singulaariarvojen sijasta poistetaan suurimmat, saatu kuva kertoo lähinnä kuvan kohinan ominaisuuksista.

PCA ja SVD hajotelmien yksi ongelma on se, ettei niissä voida ottaa huomioon yksittäisten havaintojen virhearvioita. Sen sijaan kullekin havaintomatriisin sarakkeelle ja riville voidaan asettaa oma painonsa (olettaen normaalijakautuneet virheet), yksinkertaisesti skaalaamalla ne termillä σ_k^{-2}, missö σ_k on kyseisen rivin tai sarakkeen yhteinen virhearvio. Skaalaus voidaan toteuttaa kertomalla havaintomatriisi molemmin puolin diagonaalimatriiseilla. Vasemman puoleinen kerroinmatriisi sisältää rivien skaalakertoimet, oikeanpuoleinen sarakkeiden skaalakertoimet. Typistetyn SVD hajotelman tapauksessa kerrotaan havaintomatriisi kerroinmatriiseilla D ja $\overline{D}$,

$$X = USV^T + E \Longrightarrow X' = DX\bar{D} = USV^T + E'. \tag{8.3}$$

Virhematriisi E on mukana, koska hajotelma on vajaa-asteinen (osa singulaarivektoreista on poistettu). Tästä saadaan lauseke alkuperäisen virhematriisin avulla

$$X = D^{-1}USV^T\overline{D}^{-1} + D^{-1}E'\overline{D}^{-1} \Longrightarrow E' = DE\overline{D}\ . \tag{8.4}$$

SVD etsii ratkaisun, joka minimoi matriisin E' normin $\|E'\|_2$ (alaindeksi 2 tarkoittaa, että normi lasketaan elementtien neliösummana – kyseessä on normaali, nk. Frobeniuksen normi)

$$\min\|E'\| = \min\|DE\overline{D}\ \| = \min\Big(\ \sum\sum\ D_{i,i}^2\, E_{i,j}^2\, \overline{D}_{j,j}^2\ \Big)^{1/2}. \tag{8.5}$$

Skaalaus on (pienimmän neliösumman ongelmalle) täydellinen, jos

$$\sigma_{i,j} = D_{i,i}^{-1}\ \overline{D}_{j,j}^{-1}, \tag{8.6}$$

eli jokainen virhearvio voidaan esittää ainoastaan matriisin X sarakkeesta ja toisaalta matriisin rivistä riippuvan tekijän tulona. Skaalaus voidaan toteuttaa samoin PCA rutiinin yhteydessä.

Spektriesimerkissä matriisin X rivit esittivät yksittäisiä spektrejä. Jos käytettävissä on ainoastaan virhearviot yksittäisten spektrien kohinatasosta, saadaan oikea, painotettu pienimmän neliösumman ratkaisu pelkän riviskaalauksen avulla ($D_{i,i}{=}\sigma_i^{-2}$, missä σ_i on i:nnen spektrin eli i:nnen rivin kohinataso). Jos kaikissä spektreissä esiintyy erityisen kohinaisia kanavia, voidaan ne puolestaan ottaa huomioon sarakeskaalauksella: X kerrotaan oikealta puolen diagonaalimatriisilla $\overline{D}$. Jos esim. huonot kanavat esiintyvät ainoastaan osassa spektrejä, ei virhearvioita enää voida ottaa täysin huomioon – sarakeskaalaus vaikuttaa aina saman verran kaikkiin spektreihin. Erityisen ongelmallisia ovat *outlier* tyyppiset virheet. Yksi ainoa virheellinen lukema vaatii joko koko sarakkeen/rivin painoarvon pudottamista tai virheellisen lukeman eliminoimista esim. interpoloinnilla. Jos poikkeavia arvoja on paljon, tulee tilanne hyvin hankalaksi, sillä painotuksen jälkeen suuri osa datan sisältämästä informaatiosta on hukattu. Toisaalta, tällöin ei ehkä muutenkaan ole järkevää suorittaa nimenomaan pienimmän neliösumman sovitusta!

8.3 Faktorianalyyttinen malli

Varsinaista faktorianalyysiä käsitellään tässä lähinnä yleissivistyksen vuoksi, sillä sen käyttöalue esim. PCA analyysin ja yleisen mallinnuksen välimaastossa ei ole kovinkaan suuri.

Varsinainen faktorianalyyttinen malli kirjoitetaan muodossa

$$X_j = \lambda_{j1} f_1 + \ldots + \lambda_{jm} f_m + e_j. \tag{8.7}$$

Yhtälö tarkoittaa, että muuttujaa X_j mallinnetaan lineaarikombinaationa muuttujista f_i, joita kutsutaan **faktoreiksi**. Nämä ovat oletettuja piilomuuttujia, joiden olemassaolo havainnoista pyritään kaivamaan esiin. Tavallisesti pitää piilomuuttujien lukumäärä olettaa pienemmäksi kuin havaittujen muuttujien lukumäärä. Faktorit vaikuttavat kuhunkin muuttujaan painolla λ_{ji}, jota kutsutaan termillä *factor loading*. Faktorien keskinäinen normitus hoidetaan tavallisesti niin, että factorien f_i varianssi on yksi. Lisäksi malliin sisältyy termi e_j, joka sisältää faktorien selittämättä jättämän, havaintokohtaisen vaihtelun – faktorien f_i ei tarvitse selittää kaikkea havaintomatriisissa esiintyvää vaihtelua. Tekijöistä f_i käytetään myös nimeä *common factor*, ja e_j:stä nimeä *specific factor* (koska se on 'havaintokohtainen'). Havaintomatriisin kokonaisvarianssi jakautuu osiin

$$\mathrm{Var}(X_j) = \sum_{k=1}^{m} \lambda_{jk}^2 + \Psi_j, \tag{8.8}$$

Ensimmäisestä termistä käytetään nimitystä *communality*, sillä se kuvaa yhteisten faktorien selittämää osuutta. Jälkimmäinen termi seuraa havaintokohtaisista termeistä e_j. Tavallisesti faktorit oletetaan korreloimattomiksi. Havaintomatriisin kovarianssimatriisi on

$$\Sigma = \Lambda\Lambda^T + \Psi\,. \tag{8.9}$$

Määritelmän mukaan Ψ on diagonaalimatriisin, joten faktorien on selitettävä muuttujien väliset kovarianssit täydellisesti. Tämä yhtälö määrittelee faktorianalyyttisen mallin, kunhan lisäksi asetetaan luonnollinen vaatimus matriisin Ψ termien ei-negatiivisuudesta.

Eräs tapa mallin parametrien laskemiseksi on nk. *principal factor* menetelmä. Siinä ensimmäinen faktori valitaan niin, että se selittää mahdollisimman suuren osan yhteisvarianssista. Menetelmä vaatii arviota kommunaliteeteista. Jos faktorien oletetaan selittävän kaiken varianssin, menetelmä palautuu pääkomponenttien laskemiseen. Eräs toinen menetelmä faktorien laskemiseksi on *maximum likelihood* menetelmä, jossa matriisin $\Lambda^T \Psi^{-1} \Lambda$ vaaditaan olevan diagonaalinen. Koska matriisi painotetaan kovarianssimatriisilla, ratkaisu ei tässä riipu muuttujien skaalauksesta.

Ratkaisun laskemisen jälkeen malliin kohdistetaan usein vielä **rotaatioina** tunnettuja muunnoksia. Nämä voivat olla tavallisia ortogonaalisia koordinaatiston kiertoja tai ei-ortogonaalisia muunnoksia, joiden jäljiltä faktorien väliset kulmat (korrelaatiot) voivat muuttua. Rotaatiomenetelmät perustuvat erilaisiin kriteereihin hyvästä ratkaisusta. Esimerkiksi `varimax` pyrkii tekemään *factor loading* arvoista joko mahdollisimman isoja tai mahdollisimman pieniä. Tuloksena olisi ratkaisu, jossa alkuperäiset muuttujat riippuisivat mahdollisimman pienestä joukosta laskettuja faktoreita. Toinen yleinen rotaatioalgoritmi on `promax`, joka ei tee ortogonaalista rotaatiota, vaan voi myös muttaa faktoreiden välisiä korrelaatioita etsiessään 'parempaa' ratkaisua. Molemmat rotaatiomenetelmät löytyvät R:n kirjastosta `mva`. Samoja menetelmiä voidaan käyttää myös PCA:n ja SVD:n tulosten edelleenkäsittelyyn.

Faktorianalyysi muistuttaa hyvin paljon pääkomponenttimenetelmää, mutta sisältää muutamia tärkeitä eroja. Ensinnäkin, PCA on matemaattinen muunnos ilman mitään tilastollista mallia. Faktorianalyyttinen malli voi sisältää tarkan mallin, ja varsinaisten faktoreiden ei esimerkiksi oleteta selittävän kaikkea havaintoarvojen vaihtelua. Näiden ansiosta faktorimalli voi olla myös riippumaton muuttujien skaalauksesta, toisin kuin PCA.

Faktorianalyysissä on toisaalta useita ongelmia (joista osa on yhteisiä PCA:n kanssa). Mallin kirjoittaminen vaatii lukuisia oletuksia (vaikkapa kommunaliteetin arvo, tai alkuoletus piilomuuttujista ja niiden lukumäärästä). Jos faktorien lukumäärää m muutetaan, lasketut faktorit voivat olla kokonaan erilaisia, kuin pienemmällä m:n arvolla lasketut. Lopuksi, alkuperäiseen malliin voidaan tehdä mielivaltaisia rotaatioita, jotka voivat jälleen muuttaa ratkaisun kokonaan erilaiseksi. Monista valinnoista johtuen lopputulos riippuu helposti analysoijan subjektiivisesta näkemyksistä. Näistä ongelmista johtuen PCA analyysiä pidetään useimmiten (?) parempana vaihtoehtona.

8.4 PMF-malli ja Multilinear Engine

PMF on lyhennys sanoista **Positive Matrix Factorization**, ja kyse on samantapaisesta mallista kuin edellisen luvun faktorianalyysissä,

$$X = GF + E. \tag{8.10}$$

Matriisi X on $n \times m$ elementin havaintomatriisi. Matriisit G ja F koostuvat kokonaan tuntemattomista, ja matriisien dimensiot ovat $n \times r$ ja $r \times m$. Luku r on 'faktorien' lukumäärä eli faktoroinnin aste, joka on valittava periaatteessa etukäteen. Seuraavassa X:n sarakkeiden oletetaan vastaavan eri muuttujia. Matriiseja G ja F kutsutaan faktorimatriiseiksi. Matriisin F i:s sarake ja matriisin G i:s rivi kuvaavat yhdessä osan havaintomatriisin sisällöstä. F:n rivejä kutsutaan jatkossa 'faktoreiksi', ja matriisin G vastaavan sarakkeen kertoimet ovat näiden painokertoimia, jotka ilmoittavat kuinka paljon ko. faktoria sisältyy yksittäisiin havaintohin (= matriisin X riviin). Matriisi E sisältää sovituksen jäännöspoikkeamat.

PMF malli poikkeaa esim. normaalista pääkomponenttianalyysistä ainakin neljässä oleellisessa kohdin. Ensinnäkin (kuten faktorianalyyttisessä mallissa) malli sisältää jäännöspoikkeamat, eli faktoreiden ei oleteta selittävän havaintomatriisia täydellisesti. Toisekseen faktoreiden lukumäärä ei ole kiinnitetty, vaan se voidaan vapaasti valita ($1 \leqslant r \leqslant \min(n, m)$). Kolmanneksi, ratkaisua etsitään minimoimalla painotettua jäännösneliösummaa

$$\min_{G,F} \sum_{i,j} \left(\frac{E_{i,j}}{\sigma_{i,j}} \right)^2 = \min_{G,F} \sum_{i,j} \left(\frac{X_{i,j} - (GF)_{i,j}}{\sigma_{i,j}} \right)^2. \tag{8.11}$$

Virheiden oletetaan siis olevan normaalijakautuneita. Tässä on oleellista se, että neliösumma painotetaan havaintojen yksilöllisten virhearvioiden $\sigma_{i,j}$ avulla. PCA ja SVD analyysissä painotus voitiin tehdä ainoastaan skaalaamalla joko rivejä tai sarakkeita, mutta havaintoarvojen yksilöllinen painottaminen ei ollut mahdollista. Analyysin lähtökohtana ovat havaintomatriisi X ja havaintojen virhearvioiden muodostama matriisi Σ (symbolista huolimatta tämä **ei ole** havaintojen kovarianssimatriisi!). Neljänneksi, faktoriarvoille voidaan asettaa positiivisuusehtoja. Voidaan esimerkiksi vaatia, että kaikkien faktorielementtien (siis matriisien G ja F elementtien) on oltava ei-negatiivisia. Osa faktoriarvoista voidaan myös sitoa nollaan.

PMF mallin ratkaisu voidaan periaatteessa löytää yleisillä optimointialgoritmeilla, joskin jo positiivisuusehdot hankaloittavat optimointia. Varsinainen ongelma liittyy yhtälön 8.10 muotoon. Yhtälö ei sisällä ainoastaan hyvin suurta joukkoa tuntemattomia, vaan myös tuntemattomien *tuloja*. Perinteisesti tällaisia ongelmia on yritetty ratkaista nk. *alternating regression* algoritmeilla. Näissä toinen matriiseista (G tai F) pidetään vakiona, ja toisen matriisin elementtejä muutetaan niin että jäännösneliösumma pienenee. Seuraavalla iteraatioaskeleella osat vaihtuvat, ja korjaus tehdään toiseen matriisiin. Tällainen iteraatio suppenee kuitenkin hyvin hitaasti. Pentti Paatero (HY Fysikaalisten tieteiden laitos) on kehittänyt tehokkaampia PMF mallin ratkaisurutiineja, ja nykyisellään voidaan tavallisella tietokoneella laskea melko helposti ainakin joitakin kymmeniä tuhansia elementtejä sisältävien matriisien faktorointeja.

PMF malli on monissa tapauksissa fysikaalisesti järkevä. Havaintojen painotus mahdollistaa kaiken havaintoaineiston käyttämisen, kunhan virhearviot ovat kunnossa. Esimerkiksi PCA:ssa voisivat ensimmäiset pääkomponentit määräytyä yksittäisten epätarkkojen havaintojen perusteella, koska virhearvioita ei voida

käyttää optimaalisesti. Toisaalta positiivisuusvaatimus on paikallaan, kun havaitaan suureita, jotka ovat luonnostaan positiivisia (esim. intensiteetti tai lukumäärä). Positiivisuusehdon avulla ongelman ratkaisuun tuodaan itse havaintojen ulkopuolista *a priori* tietämystä.

Erityisen oleellisia positiivisuusehdot ovat rotaatioiden kannalta. Faktorianalyysissähän saatuun ratkaisuun voitiin tehdä muunnoksia, jotka muuttavat täysin alkuperäiset faktorit. Sama pätee yhtälössä 8.10. Voidaan valita mikä tahansa eisingulaarinen matriisi T, jolloin $GF = GT^{-1}TF = G^*F^*$ on toinen ratkaisu, jonka jäännösmatriisi E on identtinen ensimmäisen ratkaisun kanssa. Rotaatiot voidaan jakaa alkeisoperaatioihin, joissa operaatiot kohdistuvat matriisin G kahteen sarakkeeseen ja matriisin F kahteen riviin. Jos G:ssä sarake i lisätään (vakiolla kerrottuna) sarakkeeseen j, pitää F:ssä rivistä i vähentää rivi j samalla vakiolla kerrottuna. Sama pätee toisin päin niin, että jokaisella askeleella lasketaan joko matriisin G sarakkeiden tai matriisin F rivien erotus. Jos molemmille matriiseille pätee positiivisuusehto, useimmat rotaatiot estyvät, ja ratkaisu voi olla jopa yksikäsitteinen.

Esimerkki 8.4.

Juvela M., Lehtinen K., Paatero P. 1996, The use of Positive Matrix Factorization in the analysis of molecular line spectra*, MNRAS 280, 616-626*

Russeil D., Juvela M., Lehtinen K., Mattila K., Paatero P. 2002, Kinematics and morphology of L1642*, A&A (submitted)*

Emissiospektriviivojen tapauksessa positiivisuusehdot voidaan asettaa molemmille faktorimatriiseille. Jos matriisin X rivit vastaavat havaittuja spektrejä, sisältävät ratkaisussa matriisin F rivit näiden spektrien perusprofiilit, joissa intensiteetin ovat luonnollisesti ei-negatiivisia. Matriisin G kukin sarake kertoo, kuinka paljon vastaavaa peruskomponenttia sisältyy yksittäiseen havaittuun spektriin. Myös nämä ovat luonnostaan ei-negatiivisia. Ensimmäisessä artikkelissa PMF analyysiä käytettiin tutkittaessa Thumbprint-globulista tehtyjä molekyyliviivahavaintoja; jälkimmäisessä kohteena oli laajempi pilvi, Lynds 1642. PMF analyysin avulla havaituista spektreistä voidaan löytää yhtenäiset nopeusrakenteet ja erotella mahdollisesti jopa osittain samalle näkösäteelle osuvat mutta eri säteisnopeuden omaavat pilven osat toisistaan.

P. Paatero on kehittänyt PMF ohjelman pohjalta yleisemmän bi- ja trilineaarisia ongelmia ratkaisevan **Multilinear Engine** (ME) ohjelman. Yhtälö 8.10 on tyypillinen bi-lineaarinen ongelma, ja trilineaarisessa ongelmassa on vastaavasti kyse tyyppiä

$$X = GWF + E \tag{8.12}$$

olevasta yhtälöstä. Tuntemattomia sisältäviä matriiseja on kolme kappaletta, mutta matriisien kaikkien elementtien ei ole oltava tuntemattomia. Itse asiassa jo PMF ohjelmassa voitiin käyttää apuna kolmatta kerroinmatriisia, jonka avulla voidaan toteuttaa lisäehtoja kuten vaikkapa F:n vaakavektoreiden monotonisesti kasvavat arvot.

ME sisältää itse asiassa kaksi osaa. Ensinnäkin ME:n sisältämän kielen avulla toteutetaan ohjelma, jossa määritellään malli ja sille asetettavat reunaehdot. Ohjelmassa kuvataan myös tietojen luku tiedostoista, mahdollinen esikäsittely sekä vastaavasti mallinsovituksen jälkeinen jälkikäsittely ja tulosten tallentaminen.

Malli määritellään käyttäen yleistä taulukkonotaatiota. Esimerkiksi yhtälön 8.10 mukainen malli määriteltäisiin seuraavaan tapaan (`n1-n3` ovat matriisien dimensioita)

```
for> j3=1:1:n3;
   for j2=1:1:n2;
      equ X[j2,j3], errmod=em;
      for> jp=1:1:np;
         term> pos; @G[jp, j2]; @F[j3,jp]; term!;
      for!;
   for!;
for!;
```

Virhemallin määrittely on osa yhtälöiden määrittelyä (edellä muuttujan `errmod` arvo). Itse asiassa virheet oletetaan tavallisesti normaalijakautuneiksi, mutta virhearvion suuruus voidaan laskea eri tavoin lähtien esim. havaintoarvoista tai sovitetuista arvoista lähtien. ME ohjelma sisältää myös vankan toimintamoodin (*robust mode*), jolloin tietyn määrän sovituksesta poikkeavat arvot saavat jatkoiteraatioilla pienemmän painoarvon. Ratkaisu riippuu siten tavallista pienimmän neliösumman ratkaisua vähemmän havainnoissa olevista virheellisistä arvoista.

Edellä kuvatun nk. pääyhtälön lisäksi malliin voidaan sisällyttää apuyhtälöitä. Nämä ovat erittäin monipuolinen työkalu. Apuyhtälöissä voidaan käyttää mitä tahansa vakioita, tiedostoista luettuja arvoja tai mallissa esiintyviä faktoriarvoja. Näiden avulla voidaan esim. toteuttaa ratkaisulle asetettavia monotonisuusehtoja tai vaikkapa pakottaa tietyn faktorin muoto lähelle annettua *a priori* versiota. Apuyhtälöiden lisäksi voidaan käyttää faktoriarvojen sitomista, kuitenkin hieman monipuolisemmin kuin PMF ohjelman tapauksessa. Jokainen faktorimatriisien elementti voidaan sitoa annettuihin arvoihin, tai kullekin faktorielementille voidaan määritellä sallittu ylä– ja/tai alaraja. Erikoistapauksena näihin sisältyvät luonnollisesti esim. positiivisuusehdot. Esimerkiksi seuraavat apuyhtälöt määräävät, että ensimmäisen faktorin (taulukon `F` ensimmäinen sarake) muodon on noudatettava taulukossa `INITF` annettua *a priori* muotoa. Vaatimus ei ole ehdoton, vaan sen painoarvo suhteessa esim. pääyhtälöihin määräytyy annettujen kertoimien `c1-c3` perusteella. Yhtälölle määritellään (tässä tapauksessa) virhearvio $\sigma = c1 + c2\sqrt{|x|} + c3|x|$, jossa x on yksittäinen havaintoarvo. Virheen laskentakaavan muoto määräytyy muuttujan `errmod` arvosta.

```
equ>  INITF[1,jp] , C1=1.0e-4, C2=0.0, C3=1.0e-2, errmod=-12;
  term> pos; @F[1,jp] ; term!;
equ!;
```

ME ohjelmaa voidaan käyttää mm. sokeaan dekonvoluutioon (*blind deconvolution*) tai tuntemattomien komponenttien erotteluun havainnoista (*component separation*). Näissä on oleellista se, että erotettavien komponenttien muoto on ennalta tuntematon. Jos komponenttien muoto olisi ennalta tiedossa, olisi kyseessä suoraviivainen mallinnusongelma, jonka ratkaisu löydettäisiin esim. aiempien esimerkkien mukaisesti uskottavuuden maksimoinnilla. Kaavan 8.10 tapaisissa matriisifaktorointiin perustuvissa menetelmissä ongelmana on päinvastoin vapaiden parametrien suuri lukumäärä, ja ratkaisut, jotka eivät rotaatioiden vuoksi ole yleensä yksikäsitteisiä. ME ohjelman sallimat rajoitukset faktoriarvoille sekä apuyhtälöiden käyttö antavat hyvän mahdollisuuden asettaa rajoja hyväksyttävälle ratkaisulle. Apuyhtälöillä voidaan esittää esim. *a priori* arvaus faktorien muodoille tai komponenttien jakaumille, jolloin myös ratkaisu on automaattisesti yksikäsitteinen. Jos apuyhtälöt on asetettu oikein (lisäehtojen oikea 'varmuus'), ratkaisu on *paras* ratkaisu ottaen huomioon kaiken tiedon, mitä on käytettävissä.

Esimerkki 8.5. Planck satelliitti tulee kartoittamaan koko taivaan kymmenellä taajuudella välillä 30-860 GHz. Taajuusväli on valittu niin, että eri säteilykomponentit esiintyvät eri kanavissa eri voimakkuuksilla, jolloin komponentit voidaan periaatteessa erottaa eri taajuuden karttoja vertaamalla. Jos datan avulla halutaan tutkia vaikkapa pölyemission spektriä tietyssä kohteessa, ei erottelun lähtökohta voi olla oletus komponenttien tunnetuista spektreistä.

Ongelma voidaan ratkaista esimerkiksi ME-ohjelman avulla, jolloin kaikki käytettävissä oleva lisäinformaatio voidaan sisällyttää malliin.

Pääyhtälöt ovat samat kuin edellä esitetyssä esimerkissä. Lisäinformaatio sisällytetään toisaalta faktorielementtejä koskevien side-ehtojen ja toisaalta apuyhtälöiden kautta. Kaikkien komponenttien jakaumille (G-matriisi) voidaan periaatteessa asettaa positiivisuusvaatimus. Kosmisen taustasäteilyn tapauksessa tästä ei kuitenkaan ole käytännön hyötyä, sillä sen vaihtelut ovat häviävän pieniä verrattuna absoluuttitasoon. Useimpien komponenttien spektreille (matriisin F rivit) voidaan myös asettaa positiivisuusehdot. Poikkeuksena on Sunyajev-Zeldovich efekti, jossa spektrissä esiintyy myös negatiivinen osa. Kaikkien komponenttien spektrien muoto on likimääräisesti tiedossa, ja tämä tieto otetaan käyttöön apuyhtälöiden kautta. Kuten aiemmassa esimerkissä, matriisin F rivit sidotaan annettuihin mallispektreihin (*templates*). Apuyhtälöiden virhearvioiden avulla kuvataan *a priori* tiedon epävarmuus, niin että sovituksessa spektrien muodot voivat poiketa jonkin verran annetuista malleista. Joillekin komponenteille myös emission jakaumasta on olemassa tietoa. Esimerkiksi terminen pölyemissio on kartoitettu 100μm aallonpituudella ($\sim$ 3000GHz). Tämä kertoo jotakin komponentin jakaumasta pienemmillä taajuuksilla, mutta ekstrapolaatio on epävarma. Tämä tieto voitaisiin sisällyttää malliin apuyhtälöiden avulla. On kuitenkin yksinkertaisempaa lisätä nämä apuhavainnot havaintomatriisiin X, jolloin ne on sovitettava yhdessä uusien havaintojen kanssa. Yhtälöitä voidaan vielä muuttaa niin, että kunkin komponentin mallikartta tulee sovitettavaksi vain yhden faktorin avulla.

Luku 9

Muita menetelmiä, joita ei ehditä kunnolla käsittelemään, mutta jotka mainitaan niin että niistä voidaan kysyä jotakin kokeessa

9.1 ICA – Independent Component Analysis

Siinä missä esim. PCA tyytyy laskemaan muuttujien variansseja uuden koordinaatiston suhteen riippumattomien komponenttien analyysi, ICA, perustuu käsitteelle komponenttien (faktorien) tilastollisesta riippumattomuudesta. Periaatteessa kaksi komponenttia ovat riippumattomia, jos niiden kaikki momentit ovat toisistaan riippumattomia. Käytännössä algoritmit voivat tarkastella kuitenkin esim. vain jakaumien vinoutta.

Seuraava esimerkki havainnollistaa PCA ja ICA analyysien eroa. Esimerkki on otettu suoraan R:n paketin `FastICA` dokumentaatiosta.

9.2 Bayesin menetelmä

Tähän mennessä kurssin kaikki menetelmät (PMF lukuunottamatta) ovat perustuneet pelkästään annetun havaintoaineiston ominaisuuksiin. PMF:n (ja yleisemmin ME:n yhteydessä) annettiin lisäehtoja, kuten faktorien positiivisuus, joilla rajattiin mahdollisia ratkaisuja. Yleisemminkin haetusta ratkaisusta on jo olemassa jokin ennakkokäsitys, joka voi perustua esim. aiempiin havaintoihin tai teoriaan. Jo aiemmin todettu Bayesin kaava antaa formalismin, jolla tällainen a priori tieto voidaan tuoda osaksi mallin sovitusta.

Etukäteistieto ratkaisusta esitetään **etukäteisjakauman** (*prior density*) avulla, jota merkitään $p(\theta)$. Kyseessä on siis määritettävän parametrin θ arvioitu todennäköisyysjakauma. Havaintojen mukanaan tuomaa lisäinformaatiota kuvataan todennäköisyydellä $L(y|\theta)$, josta tässä käytetään nimeä **uskottavuusfunktio** (*likelihood function*). ML menetelmässä tyydyttäisiin maksimoimaan uskottavuutta. Bayesin kaavan mukaan **jälkikäteisjakauman** (*posterior density*) $p(\theta|y)$ voidaan osittaa näiden kahden termin tuloksi

$$p(\theta|y) \propto p(\theta)\, L(y|\theta). \tag{9.1}$$

niistä voidaan kysyä jotakin kokeessa

Etsitty ratkaisu, Bayesin estimaatti, määräytyy tämän lausekkeen maksimikohdan perusteella. Etukäteisjakauman $p(\theta)$ valinta on helppo tehdä, jos se perustuu esim. aiempiin havaintoihin. Tässä voidaan käyttää kuitenkin myös epätarkempaa tietoa, esim. oletetuista sallituista rajoista parametrille θ. Äärimmäisessä tapauksessa parametrin θ jakaumasta ei ole ennalta mitään tietoa. Tällöin etukäteisjakauma on vakio, ja menetelmä palautuu tavalliseen uskottavuuden maksimointiin.

Esimerkki 9.1. Arvioidaan satunnaismuuttujan x oletusarvoa, kun käytössä on aluestava arvio $\bar{x} = 5.0 \pm 1.0$, ja uudet havainnot koostuvat pisteistä {4.3, 5.5, 4.8}. Kaavan 9.1 mukaisesti ratkaisu löytyy maksimoimalla etukäteisjakauman, $N(5.0,\ 1.0)$, ja uskottavuusfunktion $L(y|\mu)$ tuloa. Jotta tilanne olisi mahdollisimman yksinkertainen, oletetaan että luvut noudattavat normaalijakaumaa, ja jakauman keskihajonta on tunnettu, $\sigma = 1.0$. Jos σ olisi tuntematon, voitaisiin myös sen etukäteisjakauma tietenkin ottaa mukaan.

Maksimoitava lauseke on

$$p(\theta)\,L(y|\mu) \;\propto\; N(5.0, 1.0) \;\times \Pi_{i=1}^{3} \exp[\,-\frac{1}{2}(\frac{x_i - \hat{x}}{\sigma})^2\,].$$

Numeerinen ratkaisu antaa tulokseksi 4.90, joka on luonnollisesti etukäteisarvion 5.0 ja havaintojen keskiarvon 4.87 välissä.

Edellä kuvatun formalismin kautta 'mutu' tietoa voi käyttää kaikessa mallinsovituksessa. Vapaamuotoisemmin sitä voi tuoda mukaan sakkofunktioiden avulla – samaa menetelmää käytetään yleisesti reunaehtojen toteuttamiseen (siis rajoitetuissa optimointitehtävissä). Esimerkiksi toteamus 'θ pitäisi olla selvästi suurempi kuin 10' voitaisiin koodata kertomalla uskottavuusfunktio vaikkapa normaalijakauman kertymäfunktiolla `pnorm(`θ`, a, b)` – varsinaiset arvot reunan sijainnille (`a` ~ 10) ja vyöhykkeen leveydelle (esim. `b`=1.0) riippuisivat siitä, kuinka varmana väitettä pidettäisiin ja miten sana 'selvästi' tulkittaisiin.

9.3 Maksimientropiamenetelmä

Maksipientropiamenetelmä (ME).

9.4 Aallokehajotelmat

Aallokehajotelma (wavelet decomposition).