# Korkeauloitteisen datan dimensionaalisuuden pienentäminen

Ilja Sidoroff

11. elokuuta 2010

Itä-Suomen yliopisto
Tietojenkäsittelytiede
Pro gradu -tutkielma

# Tiivistelmä

Tietokoneella käsitellään usein dataa, jossa datavektoreiden ulottuvuudet ovat korkeita. Korkeauloitteisissa avaruuksissa työskenneltäessä kohdataan monia vaikeuksia: dataan visualisoiminen on hankalaa, datan tilavaatimukset sekä algoritmien laskennallinen vaativuus kasvavat ja etäisyysmittojen diskriminoiva teho myös heikkenee eksponentiaalisesti korkeauloitteisissa avaruuksissa.

Ongelmien ratkaisuun on kehitelty erilaisia dimensionaalisuuden vähentämis- ja analysointimenetelmiä. Vanhoja, hyvin tunnettuja lineaarisia menetelmiä ovat pääkomponenttianalyysi ja monidimensioskaalaus. Näiden rinnalle on luotu epälineaarisia menetelmiä kuten ydinpääkomponenttianalyysi, Isomap ja Laplace-ominaiskuvaus.

Tässä työssä esitellään joitakin keskeisiä ja edustavia dimensionaalisuuden vähentämis- ja analysoimismenetelmiä ja testataan niitä puhedatasta muodostettujen MFCC-vektoreiden analysointiin. Dimensionaalisuuden analysointimenetelmillä analysoidaan MFCC-vektoreiden luontaista ulottuvuutta ja dimensionaalisuuden vähentämismenetelmillä muodostetaan kaksiulotteisia projektioita vokaali- ja frikatiiviäänteistä. Lisäksi testataan sitä, miten dimensionaalisuuden vähentäminen vaikuttaa tukivektorikoneella tehtävän äänteiden luokittelun tarkkuuteen.

Tulokset osoittavat, että dimensionaalisuuden vähentämismenetelmät ovat käyttökelpoisia MFCC-vektoreiden visualisoimisessa ja luokittelussa. Dimensionaalisuuden vähentäminen vaikuttaa joissakin tapauksissa heikentävästi luokittelutarkkuuteen, mutta yleisesti ottaen dimensionaalisuuden vähentämismenetelmät pystyvät säilyttämään luokittelussa tarvittavan rakenteen datassa.

***ACM-luokat*** (ACM Computing Classification System, 1998 version): E.0, I.5, I.m

***Avainsanat: dimensionaalisuuden vähentäminen, dimensioanalyysi, hahmontunnistus, koneoppiminen***

# Sisältö

# 1 Johdanto

Data-analyysissä käsitellään datajoukkoja, jotka koostuvat useista havainnoista. Kukin havainto koostuu puolestaan yhdestä tai useammasta muuttujasta. Yhden muuttujan datajoukko voi esimerkiksi olla sarja lämpötilamittauksia tietyssä paikassa tietyillä ajanhetkillä. Jos samasta paikasta mitataan lämpötilan lisäksi myös ilmanpaine ja tuulen nopeus, saadaan datajoukko, joka koostuu monen muuttujan havainnoista. Jos yksittäisten muuttujien arvot ovat numeerisia, voidaan havainnot esittää muuttujien määrän mukaan $n$-uloitteisen avaruuden pisteinä tai $n$-ulotteisina vektoreina $\mathbf{x} \in \mathbb{R}^n$.

Datajoukkojen muuttujien määrä eli dimensionaalisuus on usein korkea. Esimerkiksi kuva-analyysissä $32 \times 32$-kokoista harmaasävykuvaa voidaan käsitellä yksittäisenä 1024-ulotteisen avaruuden vektorina [61]. Geeni- ja proteenidataa käsiteltäessä yksittäisten vektorien koko voi olla yli 10000 [14].

Korkea dimensionaalisuus aiheuttaa useita käytännön ongelmia[13]:

1. Yli kolmiuloitteisen datan visualisointi on hankalaa.
2. Data vaatii runsaasti säilytystilaa.
3. Datan ulottuvuuden kasvaessa algoritmien aikavaativuus kasvaa pahimmassa tapauksessa eksponentiaalisesti.
4. Datapisteiden keskinäisten etäisyyksien hajonta pienenee, mikä heikentää etäisyyspohjaisten luokittelu- ja ryhmittelymenetelmien toimivuutta.

Korkeadimensionaalisten datajoukkojen tapauksessa on kuitenkin usein niin, etteivät kaikki muuttujat ole toisistaan riippumattomia, vaan datajoukko voitaisiin projisoida tai upottaa pienempiuloitteiseen avaruuteen siten, etteivät sen ominaisuudet muutu. Pienintä tällaista ulottuvuutta kutsutaan datan aidoksi tai luontaiseksi dimensionaalisuudeksi (*intrinsic dimensionality*), tai luontaiseksi ulottuvaisuudeksi.

Menetelmiä, jotka pyrkivät selvittämään datan luontaisen dimensionaalisuuden, nimitetään luontaisen dimensionaalisuuden analysointimenetelmiksi (*intrinsic dimensionality analysis*). Dimensionaalisuuden analysointimenetelmiin liittyy läheisesti toinen menetelmäryhmä, dimensionaalisuuden vähentämis- tai

redusiontimenetelmät (*dimensionality reduction*), joiden pyrkimyksenä on pienentää datan ulottuvuutta siten, että kullekin sovellukselle olennaiset piirteet datasta säilyvät muuttumattomina.

Dimensionaalisuuden vähentämisessä kohdedimensio voidaan valita periaatteessa mielivaltaisesti. Käytännössä kuitenkin sovellus, johon dataa käytetään määrää kohdedimension. Jos korkeauloitteista dataa halutaan visualisoida, on kohdedimensio yleensä 2 tai 3. Mikäli datan luontainen dimensionaalisuus tunnetaan, voi se usein olla luonteva kohdeulottuvuus. Kohdeulottuvuus voidaan valita myös siten, että käytettävien algoritmien ajoaika- tai tilavaatimukset saadaan halutulle tasolle.

Dimensionaalisuuden vähentämismenetelmiä on kehitetty useita erilaisia [44, 12]. Mikään yksittäinen menetelmä ei ole yksiselitteisesti kaikkia muita menetelmiä parempi. Tämä johtuu osittain siitä, että dimensionaalisuutta vähentäessä voidaan pyrkiä säilyttämään joitakin datan ominaisuuksia, mutta kaikkien ominaisuuksien säilyminen ei onnistu. Niinpä erilaiset menetelmät voivat pyrkiä esimerkiksi minimoimaan datan keskineliövirhettä, pyrkiä säilyttämään pisteiden välisiä etäisyyksiä tai naapuruussuhteita.

Tässä työssä esitellään keskeisimpiä dimensionaalisuuden analysointi- ja redusointimenetelmiä, painottuen redusointimenetelmiin. Aluksi tutustutaan datan dimensionaalisuuden määrittelyyn, minkä jälkeen esitellään erilaisia dimensionaalisuuden vähentämismenetelmiä. Viimein tutkitaan miten esitellyt menetelmät toimivat puhedatasta muodostettujen vektoreiden tapauksessa.

# 2 Dimensionaalisuus

Ennen varsinaisten dimensionaalisuuden vähentämismenetelmien esittelyä tässä luvussa käydään läpi, miten dimensionaalisuus määritellään. Määrittelyn lisäksi tutustutaan hieman korkeauloitteisen datan ominaisuuksiin ja käydään läpi joitakin tapoja arvioida datan dimensionaalisuus.

## 2.1 Dimensionaalisuuden määritely ja arvioiminen

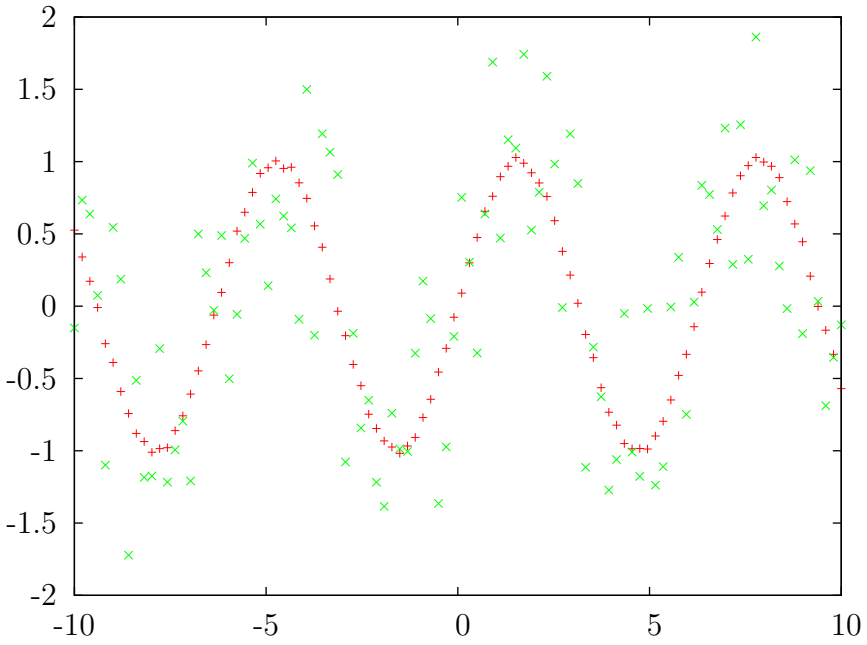
Dimensionaalisuus voidaan määritellä yksinkertaisesti geometristen objektien avulla. Piste on nollauloitteinen objekti, suora yksiulotteinen, neliö kaksiulotteinen, kuutio kolmiulotteinen ja niin edelleen. Tätä määritelmää voidaan soveltaa myös datajoukkojen tapauksessa. Jos datajoukko $X$ koostuu mittauksista, jotka voidaan esittää $n$-uloitteisina reaalilukuvektoreina ($\mathbf{x} \in \mathbb{R}^n$), on datan dimensionaalisuus $n$.

Usein on kuitenkin niin, etteivät datavektorien kaikki $n$ komponenttia ole tarpeellisia datan yksikäsitteiseen esittämiseen. Tällöin datajoukon luontainen dimensionaalisuus (*intrinsic dimensionality*) on *pienin määrä vapaita muuttujia, joilla joukko voidaan kuvata* [13]. Matemaattisesti tämä voidaan ilmaista siten, että datan luontainen dimensionaalisuus $m$ on pienimmän sellaisen $\mathbb{R}^n$:n aliavaruuden dimensio, joka riittää säilyttämään alkuperäisen joukon kaikki ominaisuudet.

Yksinkertainen dimensionaalisuuden määritelmä ei ole aina yksikäsitteinen. Esimerkkinä kuvassa 1 on kaksi joukkoa datapisteitä, jotka sijaitsevat kaksiuloitteisessa avaruudessa likimääräisesti sinikäyrällä. Toinen joukko sisältää runsaasti kohinaa.

Datapisteiden ja data-avaruuden ulotteisuus on selvästi kaksi. Toisaalta, jos kohinaa ei oteta huomioon datan kuvaamiseen riittää ainoastaan yksi parametri, jos tiedetään sen jakautuvan funktion $f(x) = sin(x)$ mukaan. Tällöin datan luontainen ulottuvuus on yksi, mutta jos kohina lisääntyy tarpeeksi, datan kuvaamiseen tarvitaan kuitenkin kaksi muuttujaa.

Yksinkertainen dimensionaalisuuden määritelmä ei ole kaikissa tilanteissa myöskään matemaattisesti ristiriidaton. Tämän osoitti ensimmäisenä italiainen

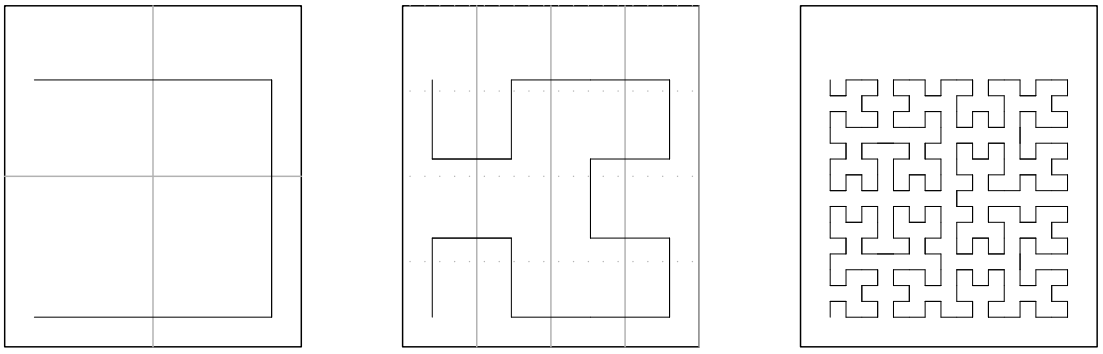



Kuva 1: Kaksi sinifunktion generoimaa datajoukkoa. Sinifunktion voi erottaa toisen joukon generoivaksi funktioksi helposti, mutta toiselle, kohinaiselle joukolle tämä ei ole yhtä helppoa.

matemaatikko Giuseppe Peano, joka konstruoi käyrän, joka kuvautuu injektiivisesti yksikköväliltä yksikköneliölle [50]. Tunnetuin graafinen esitys tällaisista tason täyttävistä käyristä on peräisin David Hilbertiltä [35]. Kuvassa 2 on esitetty Hilbertin käyrän kolmen ensimmäistä iteraatiota. Kun iteraatioiden määrä kasvaa rajatta, täyttää käyrä koko tason. Funktio, joka määritellään yhdellä vapaalla muuttujalla määrittää siis kaksiulotteisen objektin. Nämä esimerkit osoittavat, että dimensionaalisuus määriteltynä pelkkien vapaiden muuttujien avulla ei vastaa aina geometrista intuitiota.

## 2.2 Topologinen dimensio

Dimensionaalisuus voidaan määritellä täsmällisesti topologian avulla. Topologisia dimensionaalisuuden määritelmiä on useita hieman erilaisia, kuten esimerkiksi *pieni ja suuri induktiivinen dimensio* sekä *Lebesguen peitedimensio*. Nämä eroavat toisistaan lähinnä teknisten seikkojen osalta. Riittävän säännöllisissä topologisissa avaruuksissa induktiiviset dimensiot ja Lebesguen peitedimensio eivät eroa toisistaan [25]. Yleensä, kun puhutaan topologisesta dimensiosta (*topological di-*



$n = 2$ $n = 3$ $n = 4$

Kuva 2: Kaksiuloitteisen Hilbertin käyrän approksimaatioita. Kun $n$ lähenee ääretöntä, täyttää käyrä tason.

*mension*), tarkoitetaan Lebesguen peitedimensiota. Näin tehdään myös jatkossa tässä työssä.

Topologinen dimension tarkka matemaattinen määritelmä löytyy useista topologian oppikirjoista [25]. Seuraavassa käytetään näistä teoksista yleisesti löytyvää avoimen joukon määritelmää. Huomattavaa on, että topologisen dimension määritelmässä ei tarvita metriikkaa tai etäisyysmittaa[1], vaan dimensio määritellään avoimien joukkojen avulla.

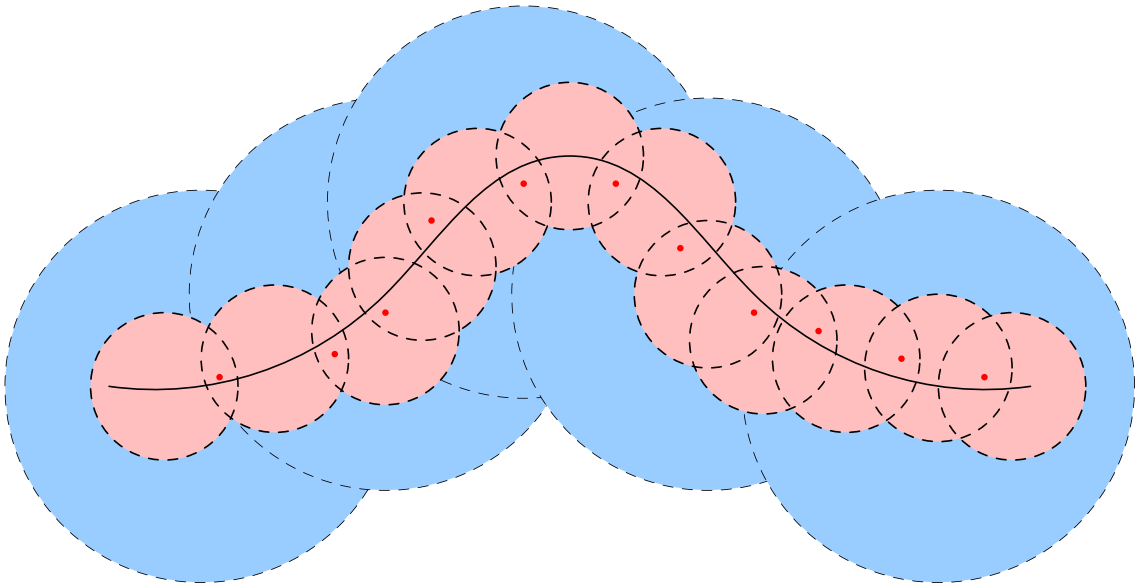
Olkoon $X$ jokin topologinen avaruus ja $S$ sen osajoukko $S \subset X$, missä $S$ voi olla esimerkiksi käyrä tai muu geometrinen olio. Joukon $S$ peite on kokoelma $C$ kaikista $X$:n avoimista osajoukkoista, joiden yhdiste sisältää $S$:n. Peitteen $C$ *hienonnus* on sellainen peite $C'$, että jokainen peitteen alkio $C'$ sisältyy jonkin peitteeseen $C$ kuuluvusta joukoista. Sanotaan, että topologisen avaruuden $X$ dimensionaalisuus on $m$, jos

1. $X$:n jokaisella peitteellä $C$ on hienonnus $C'$, jossa jokainen $X$:n piste esiintyy korkeintaan $m + 1$:ssä $C'$:n joukossa

2. $m$ on pienin edellisen ehdon täyttävistä kokonaisluvuista

Kuvassa 3 on esimerkki topologisen dimension määrittämisestä. Siniset, suuremmat ympyrät ovat kuvassa olevan käyrän eräs peite. Punaiset, pienemmät ym-

[1]E.W. Weisstein. *Metric*. MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com, 21.4.2010.



Kuva 3: Käyrän peite ja sen hienonnus. Käyrä voidaan peittää siten, että kukin käyrän osan peittämiseen tarvitaan korkeintaan kaksi toisensa leikkaavaa joukkoa.

pyrät ovat tämän peitteen hienonnus. Kuvassa oleva käyrä voidaan peittää siten, että korkeintaan kaksi peitteen osajoukkoa leikkaa toisensa (leikkausalueet on merkitty kuvaan punaisilla pisteillä), vaikka peittettä hienonnettaisiin rajatta. Näin ollen käyrän topologinen dimensio on yksi.

Topologinen dimensio vastaa aikaisemmin mainittua datan luontaista dimensionaalisuutta [52]. Topologinen dimensio ei ota huomioon sitä, minkälaiseen avaruuteen data on upotettuna, vaan määrittelee datan dimensionaalisuuden ainoastaan sen oman rakenteen avulla. Topologisesta dimensiosta käytetään joskus myös nimitystä *lokaali dimensio* (*local dimension*) vastakohtana *globaalille dimensiolle*, joka tarkoittaa koko data-avaruuden ulottuvaisuutta [13].

Topologisen dimensionaalisuuden määrittely algoritmisesti ei ole yksinkertaista, mutta erilaisia menetelmiä tähän on esitetty useita [13]. Varhaisin näistä on on *Fukunaga-Olsenin* algoritmin nimellä tunnettu menetelmä [30]. Menetelmä perustuu pääkomponenttianalyysiin, joka esitellään tarkemmin seuraavassa luvussa. Fukunaga-Olsenin algoritmissa data jaetaan pieniin osiin, joista kullekin osan kovarianssimatriisin ominaisarvot lasketaan. Normalisoituja ominaisarvoja verrataan valittuun kynnysarvoon jonka ylittävien ominaisarvojen määrä määrittää datan ulottuvaisuuden. Fukunaga-Olsenin algoritmin heikkoutena on se, että ei ole selvää miten kynnysarvo pitää valita [13]. Menetelmä on myös laskennallises-

ti vaativa. Yhden alueen paikallisten ominaisarvojen laskemisen aikavaativuus on $O(n^2)$ ja jotta menetelmä toimii, datan tiheyden täytyy olla riittävä [68].

Paikallisen pääkomponenttianalyysin ohella toinen tapa määrittää datan topologinen dimensio, on käyttää datapisteiden naapurustoja ulottuvaisuuden määritellyssä. Naapurustomenetelmistä ensimmäinen oli *Trunkin menetelmä*, jossa kukin datapisteen $\mathbf{x}$ $k$:sta lähimmäistä naapurista muodostetaan aliavaruus, ja tämän aliavaruuden ja pisteen $k+1$:n naapurin välinen kulma lasketaan. Datan luontainen dimensionaalisuus on se $k$:n arvo, jolle kulma on kaikkien datapisteiden osalta suurempi kuin annettu kynnysarvo [64].

Trunkin menetelmän ongelma on sama kuin Fukunaga-Olsenin: ei ole selvää miten kynnysarvo pitäisi määritellä, myös naapuruston koko $k$ on riippuvainen datan ominaisuuksista. Pettis ja kumppanit kehittivät oman muunnoksensa Trunkin menetelmästä, jossa kynnysarvoa ei tarvita. Dimensionaalisuus $d$ määräytyy yhtälöstä [52]

$$d = \frac{\langle r_k \rangle}{(\langle r_{k+1} \rangle - \langle r_k \rangle)k} \tag{1}$$

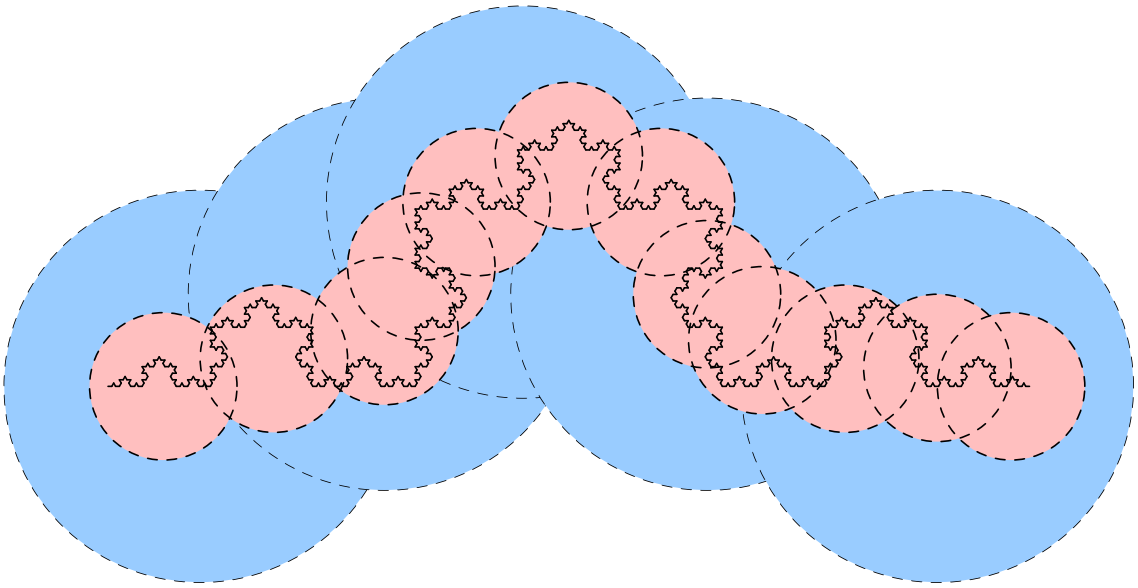
jossa $\langle r_k \rangle$ on pisteen $\mathbf{x}$ ja sen $k$:n lähimmän naapurin etäisyyksien keskiarvo. Naapuruston koon valinta vaikuttaa dimensionaalisuusarvioon, niinpä Verveer ja Duin esittivät seuraavan yhtälön, jossa naapuruston koon vaikutusta yritetään lieventää [68]:

$$d = \left( \sum_{k=k_{min}}^{k_{max}-1} (\langle r_k \rangle + 1 - \langle r_k \rangle_k)^2 \right)^{-1} \left( \sum_{k=k_{min}}^{k_{max}-1} \frac{(\langle r_{k+1} \rangle - \langle r_k \rangle)\langle r_k \rangle}{k} \right) \tag{2}$$

Edellä esitellyt naapuruston ominaisuuksiin perustuvat topologisen dimensionaalisuuden määritelymenetelmät eivät käytännössä toimi kovinkaan hyvin [13, 68].

## 2.3 Fraktaalidimensio

Datan dimensionaalisuutta määriteltäessä käytetään usein hyväksi kaaosteoriasta alkunsa saanutta *fraktaalidimension käsitettä* (*fractal dimension*) [23]. Kaaosteoriassa käsitellään yleensä dynaamisten systeemien tuottamaa fraktaalista dataa.



Kuva 4: Fraktaalinen Kochin käyrä ja sen peitteitä. Topologinen dimensio ei ole riittävän tarkka määrittämään fraktaalisten käyrien ulotteisuutta.

Tällainen data esiintyy tyypillisesti – mutta ei aina – aikasarjoina, esimerkiksi sydänkäyriä, pörssikursseja tai eläinpopulaatiotietoja voidaan käsitellä fraktaalimenetelmin [13, 37].

Fraktaalinen data ei ole aina topologisessa mielessä sileää. Kuvassa 4 on esitetty fraktaalinen Kochin käyrä. Jos kuvan mittakaavaa suurennetaan, säilyy käyrän mutkikkuus samanlaisena mittakaavasta huolimatta. Tämän vuoksi topologinen dimensionaalisuusarvio tuottaa liian suuren tuloksen – käyrää ei voi peittää siten, että vain kaksi peitettä leikkaisi toisensa.

Fraktaalidimensio voidaan määritellä usealla eri tavalla. Ensimmäinen määritelmä on peräisin Hausdorffilta [34]. Määritellään ensin datajoukolle $X$ kokoelma peitteitä seuraavasti:

$$\Gamma^d(r) = \inf_{s_i} \sum_i (r_i)^d \tag{3}$$

jossa joukko $X$ on peitetty joukoilla $s_i$, joista kunkin säde on $r_i$ (kaikilla $r_i < r$). Yhtälö 3 määrittää siis joukon $X$ minimaalisen peitteen, samoin kuten topologisen dimension tapauksessa. *Hausdorffin mitta* (*Hausdorff measure*) määritellään

$$\Gamma^d = \lim_{r \to 0} \Gamma^d(r) \tag{4}$$

Hausdorff todisti, että jokaiselle joukolle pätee, että $\Gamma^d = \infty$, kun $d > d_H$ ja $\Gamma^d = 0$, kun $d < d_H$. Luku $d_H$ on datajoukon *Hausdorffin dimensio* (*Hausdorff dimension*) [34].

Käytännössä Hausdorffin dimensio on hankala määrittää laskennallisesti, joten sen sijaan voidaan käyttää *laatikkodimensiota* (*Box counting dimension*) tai *Minkowski-Bouligand-dimensiota*, joka on yleensä sama kuin Hausdorffin dimensio[2]. Se määritellään käyttämällä datajoukon peitteenä mielivaltaisten joukkojen sijaan laatikonmuotoisia joukkoja. Jos $v(r)$ on niiden $r$-sivuisten laatikoiden lukumäärä, joka tarvitaan peittämään joukko $X$, niin laatikkodimensio $d_B$ on

$$d_B = \lim_{r \to 0} \frac{\ln v(r)}{\ln 1/r} \tag{5}$$

Algoritmin mutkikkain osa on pitää kirjaa pienenevien laatikoiden sisällä olevista datapisteistä, mutta laatikkodimension laskenta on käytännössä mahdollista (esim. [32]). On kuitenkin olemassa fraktaalidimensio, joka on algoritmisesti hieman yksinkertaisempi kuin laatikkodimensio. *Korrelaatiodimensio* (*correlation dimension*) [33] määritellään seuraavasti. Olkoot $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ datavektoreita ($\mathbf{x} \in \mathbb{R}^d$). *Korrelaatiointegraali* $C(r)$ määritellään:

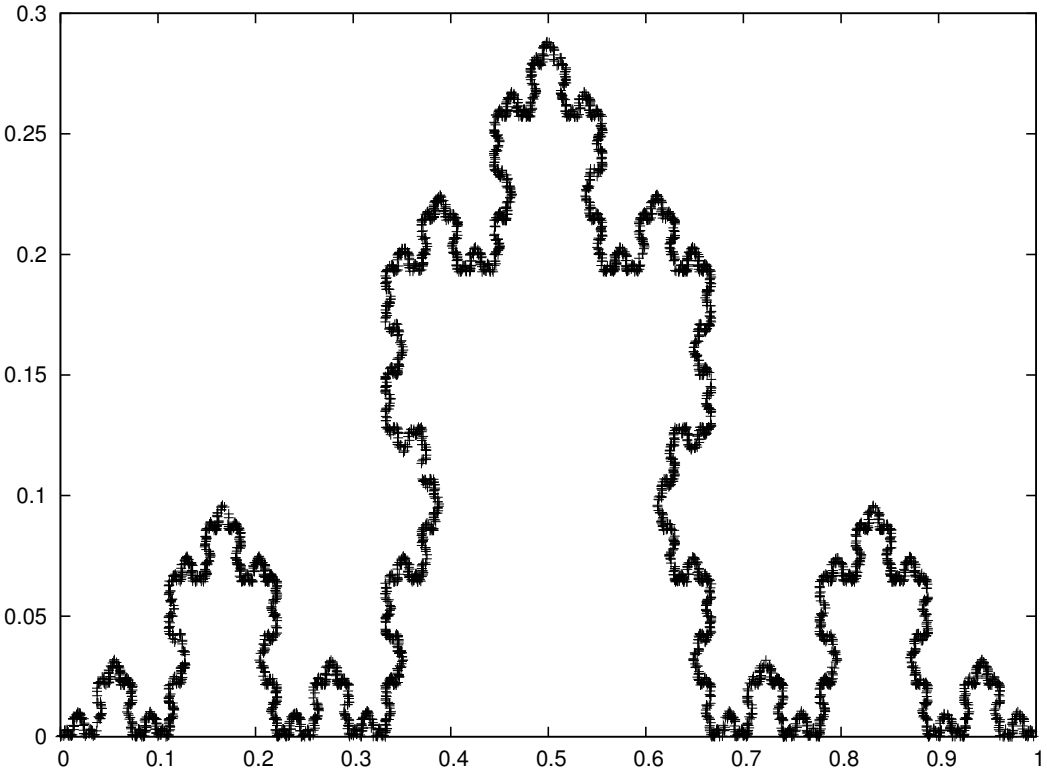
$$C(r) = \lim_{n \to \infty} \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} I(\|\mathbf{x}_j - \mathbf{x}_i\| \leq r) \tag{6}$$

Jossa $I(x)$ on indikaattorifunktio ($I(x) = 1$, joss x on tosi, muuten $I(x) = 0$; korrelaatiointegraalin tapauksessa jos vektorit $\mathbf{x}_i$ ja $\mathbf{x}_j$ ovat korkeintaan $r$, on $I = 1$). Korrelaatiodimensio $d_C$ on korrelaatiointegraalin raja-arvo:

$$d_C = \lim_{r \to 0} \frac{\ln C(r)}{\ln r} \tag{7}$$

Korrelaatiodimensio on aina laatikkodimension alaraja, mutta käytännössä kohi-

[2]E.W. Weisstein. *Minkowski-Bouligand dimension.* MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com, 21.4.2010



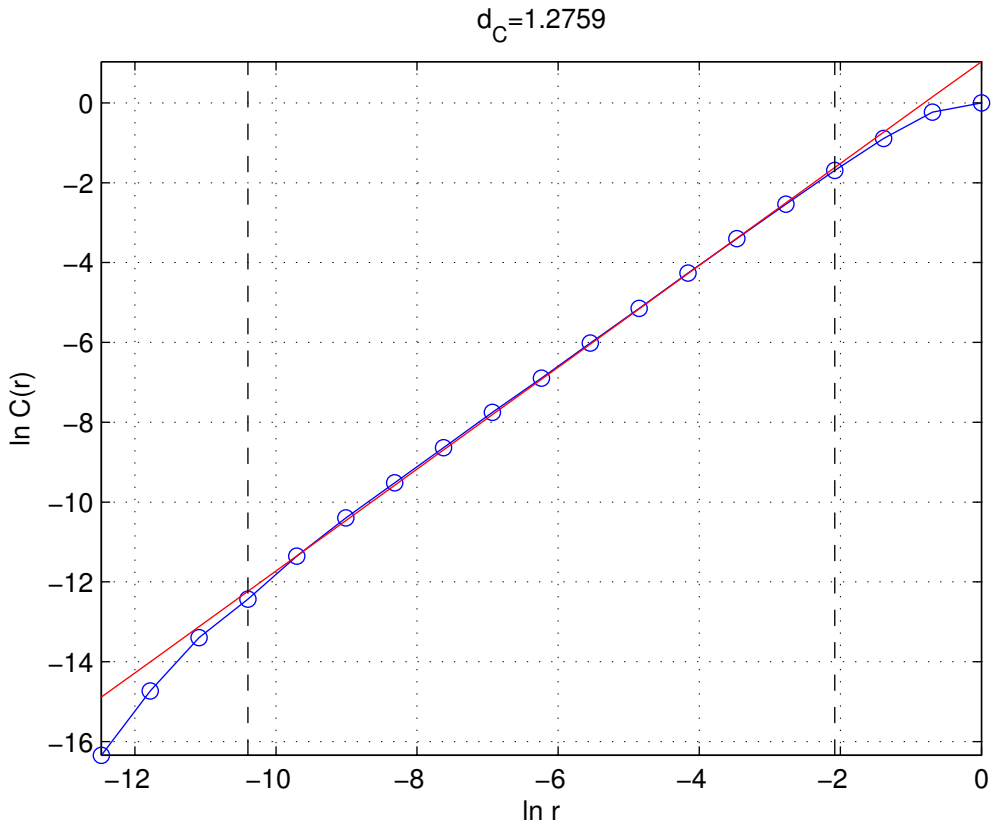

Kuva 5: Kochin käyrä, jota on näytteistetty 5000 pisteellä.

naisen datan tapauksessa dimensiot eivät eroa toisistaan [13]. Korrelaatiodimension etuna laatikkodimensioon nähden on algoritmisesti helppo toteutus.

Käytännössä dataa on käytettävissä aina äärellinen määrä, eikä kaavojen 5 ja 7 raja-arvoja voida siten laskea suoraan. Likiarvot raja-arvoille saadaan tulostamalla – esimerkiksi korrelaatiodimension tapauksessa – käyrä $\ln C(r)$ $\ln r$:n funktiona ja mittaamalla saadun käyrän lineaarisen osan kulmakerroin [43].

Esitetyt fraktaalidimensiot ovat topologisen dimension yleistyksiä, ja esimerkiksi ei-fraktaaliselle datalle Hausdorffin dimensio on sama kuin Lebesguen peitedimensio [34]. Fraktaalisia dimensioita voidaan näin ollen käyttää myös ei-fraktaalisen datan dimensionaalisuuden määrittelyyn. Fraktaalidimension erityispiirteenä on, että jos data on fraktaalista, dimensionaalisuuden ei tarvitse olla kokonaisluku. Esimerkiksi kuvan 4 Kochin käyrän laskennallinen dimensio on $\ln 4/\ln 3 \approx 1.26$.

Muita tunnettuja fraktaalidimensioita ovat *informaatiodimensio* (*information dimension*) [55] ja *kapasiteettidimensio* (*capacity dimension*) [49]. Näistä ensimmäinen määrittää todennäköisyysjakaumien dimensionaalisuutta.



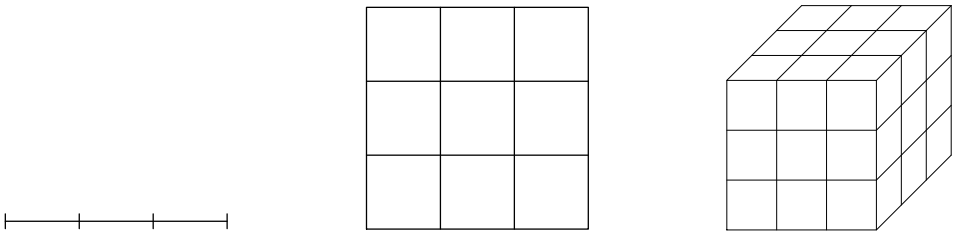

Kuva 6: Korrelaatiodimension määrittäminen Kochin käyrälle käyrän tangentin $\frac{\ln C(r)}{\ln r}$ kulmakertoimena.

Jälkimmäinen puolestaan muistuttaa korrelaatiodimensiota, mutta on laskennallisesti vaativampi. *Pakkausnumeromenetelmä* (*Packing numbers*) on hiljattain esitetty tapa laskea kapasiteettidimensio [43].

Kuvassa 5 on esitetty 5000 pisteellä näytteistetty Kochin käyrä. Kuvasta 6 näkyy käyrän dimensionaalisuuden määritys. Pisteiden äärellisestä määrästä johtuen dimensioarvio 1.28 eroaa hieman laskennallisesta dimensionaalisuudesta. Laatikkodimensio voidaan määritellä vastaavalla tavalla.

Fraktaalisen dimensionaalisuuden määrittelyn haittapuolena on se, että luotettavaan dimensionaalisuuden määritykseen vaaditaan paljon datapisteitä. On osoitettu [24], että dimensionaalisuuden $d$ ja datapisteiden määrän $n$ ja välillä pätee epäyhtälö:

$$d < 2 \log_{10} n \tag{8}$$




Kuva 7: Alueiden ($m$) määrä kasvaa eksponentiaalisesti ulottuvuuden $n$ kasvaessa.

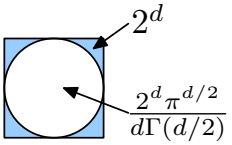
Luotettavaan $d$-ulotteisen dimensionaalisuuden määrittelyyn tarvitaan siis vähitään $10^{d/2}$ datapistettä.

Edellä esiteltyjen dimensionaalisuuden määritysmenetelmien lisäksi on ehdotettu myös muitakin tapoja datan dimensionaalisuuden määrittämiseen. *Geodeettinen minimaalinen virittävä puu* (*geodesic minimum spanning tree*, GMST) perustuu datasta muodostetun naapurustograafin analysointiin ja on sukua myöhemmin esiteltävälle Isomap-algoritmille [15]. Viimein on esitetty myös suurimman todennäköisyyden estimointiin (*maximum likelihood estimation*, MLE) perustuva menetelmä, jossa topologinen tai fraktaalinen dimensionaalisuuden arviointi on yhdistetty tilastotieteelliseen analyysiin [47]. Joihinkin dimensionaalisuuden redusointimenetelmiin liittyy läheisesti myös keino saada selville datan luontainen dimensionaalisuus. Näihin palataan tarkemmin redusointimenetelmien esittelyn yhteydessä.

## 2.4 Dimensionaalisuuden kirous

Johdannossa mainittujen seikkojen lisäksi korkeaulotteiseen dataan liittyy ongelma, jota kutsutaan *dimensionaalisuuden kirokseksi* (*Curse of dimensionality*). Termin otti ensimmäisenä käyttöön Bellman [8], joka tarkoitti sillä avaruuden tilavuuden eksponentiaalista kasvua dimensionaalisuuden kasvun myötä. Kuva 7 havannoillistaa ilmiötä. Kuvan avaruus on jaettu yhtä suuriin alueisiin, joiden lukumäärä kasvaa eksponentiaalisesti dimensionaalisuuden kasvaessa.

Tilavuuden kasvu korkeaulotteisissa avaruuksissa johtaa myös laskennallisiin ongelmiin, mikä johtuu siitä, että etäisyysmitan diskriminoiva kyky heikkenee kun dimensionaalisuus kasvaa. Tämä johtuu siitä, että suurin osa kappaleen, esimer-




Kuva 8: Yksikköpallo yksikkökuution sisällä kaksiulotteisessa avaruudessa ($d = 2$). Kun ulottuvuus kasvaa, kuution tilavuus dominoi pallon tilavuutta.

kiksi hyperkuution, tilavuudesta sijoittuu ohueen kerrokseen lähelle kappaleen ulkopintaa. Esimerkiksi kuvan 7 tapauksessa reunalla olevien alueiden lukumäärä dimensionaalisuuden kasvaessa on $3^n - 1$. Jos pisteet jakautuvat tasaisesti, on todennäköistä että ne sijoittuvat ulkoreunoille ja niiden etäisyys kappaleen keskipisteestä on suurin piirtein sama.

Asiaa voi havannollistaa vertaamalla yksikköpallon ja -kuution tilavuuksia. Yksikköpallon tilavuus ulottuvuuden $d$ funktiona on

$$\frac{2^d \pi^{d/2}}{d\Gamma(d/2)} \tag{9}$$

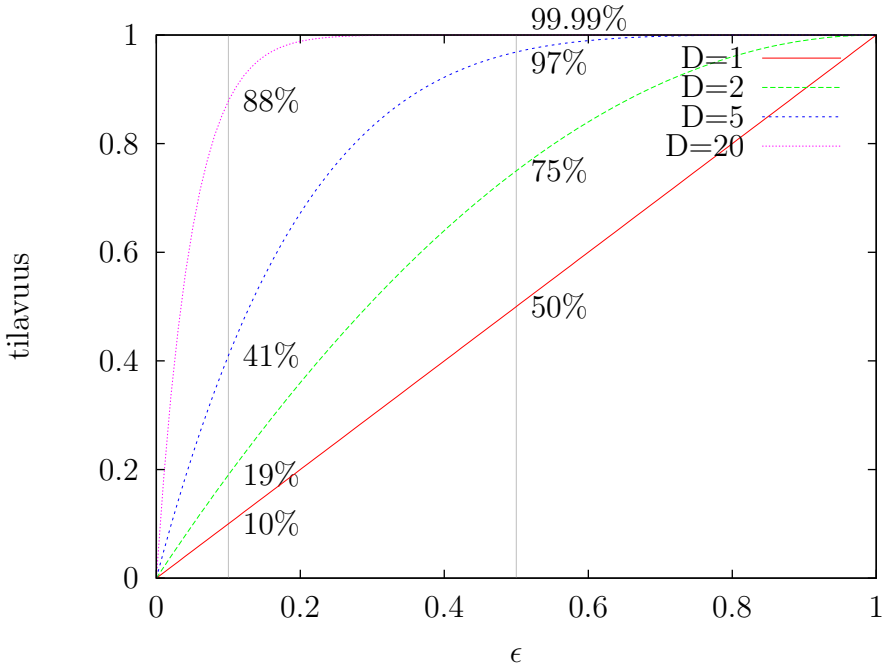
jossa $\Gamma$ on gammafunktio, joka määritellään kokonaisluvuilla kertomafunktion avulla: $\Gamma(d) = (n-1)!$. Yksikkökuution tilavuus puolestaan on

$$2^d \tag{10}$$

Kun avaruuden dimension $d$ kasvaa, dominoi hyperkuution tilavuus hyperpallon tilavuutta:

$$\lim_{d\to\infty} \frac{\pi^{d/2}}{d2^{d-1}\Gamma(d/2)} = 0 \tag{11}$$

Kuva 9 havainnollistaa samaa asiaa. Kuvaajat osoittavat miten suuri osa hyperpallosta, jonka säde on 1, sijaitsee säteen $1-\epsilon$ ulkopuolella. Pystyviivojen kohdille on merkitty joitakin yksittäisiä prosenttilukuja tilavuusjakaumasta. Ensimmäinen pystyviiva kertoo miten suuri osa tilavuudesta on säteen uloimmailla kymmenyksellä, toinen pystyviiva kertoo puolestaan tilavuudesta, joka jää säteen puolikkaan ulkopuolelle. Esimerkiksi ulottuvuuden ollessa 20, jo 88% pallon tilavuudesta sijaitsee pallon tilavuuden uloimmalla kymmenesosalla [11].




Kuva 9: Dimensionaalisuuden vaikutus tilavuusjakaumaan. Kuva osoittaa miten suuri osa yksikköpallon tilavuudesta on säteen $1 - \epsilon$ ulkopuolella pallon keskipisteestä (vrt. [11]).

Koska korkeauloitteisissa avaruuksissa kaikki pisteet näyttävät sijaitsevan suurin piirtein yhtä kaukana toisistaan, etäisyysmittojen lisäksi pisteiden naapuruussuhteet menettävät merkitystään [9]. Tämän vuoksi aidosti korkeauloitteisissa avaruuksissa etäisyyksiin tai naapuruussuhteisiin perustuvat luokittelu- tai ryhmittelyalgoritmit eivät yleensä toimi [47].

Suurin osa seuraavassa luvussa esiteltävistä dimensionaalisuuden vähentämisalgoritmeista kuuluu tähän joukkoon, joten niillä on vaikeuksia aidosti korkeadimensionaalisen datan kanssa. Todellinen data sijoittuu usein kuitenkin johonkin korkeaulotteisen datan pieniulotteiseen aliavaruuteen, eli on siis luontaiselta dimensionaalisuudeltaan pientä, jolloin algoritmeilla on mahdollisuus toimia.[3]

Datalla voi myös olla jonkinlaisia *sileysominaisuuksia* (*smoothness*), jolloin pieni muutos syötedatassa aiheuttaa vain pienen muutoksen analyysituloksessa. Siley-

[3] "There is a consensus in the high-dimensional data analysis community that the only reason any methods work in very high dimensions is that, the data are not truly high-dimensional." [47]

sominaisuuksia voidaan usein hyödyntää aidosti korkeadimensionaalisen datan käsittelyssä [11].

# 3 Dimensionaalisuuden vähentämismenetelmiä

Datan dimensionaalisuutta voidaan haluta pienentää monesta eri syystä. Jos datan luontainen dimensionaalisuus on pienempi, kuin sen avaruuden dimensionaalisuus, jossa data sijaitsee, dimensionaalisuuden vähentäminen helpottaa ja nopeuttaa datan laskennallista käsittelyä, kun käsillä olevan sovelluksen kannalta turhat dimensiot poistetaan datasta. Jos data on yli kolmiuloitteista, voidaan dimensionaalisuuden vähentämistä kahteen tai kolmeen ulottuvuuteen käyttää apuna datan visualisoimisessa. Aidosti korkeaulotteista dataa voidaan myös yrittää projisoida pienempiulotteiseen avaruuteen pelkästään laskennallisten aika- ja tilavaatimusten helpottamiseksi.

Erilaisia algoritmeja datan dimensionaalisuuden vähentämiseksi on esitetty kymmenittäin [44, 65]. Suosituimmista menetelmistä on lisäksi useita erilaisia variantteja. Menetelmät voidaan luokitella usealla eri tavalla. Eräs yleinen luokitteluperiaate on jako lineaarisiin ja epälineaarisiin menetelmiin.

Lineaariset dimensionaalisuuden vähentämismenetelmät löytävät datasta lineaarisia riippuvuussuhteita ja pystyvät säilyttämään ne myös pienemmissä ulottuvuuksissa. Epälineaariset menetelmät puolestaan pystyvät säilyttämään myös datan epälineaarisia ominaisuuksia [44]. Epälineaariset menetelmät ovat uudempia kuin lineaariset ja niiden erityispiirteenä on usein myös se, että datan oletetaan muodostavan matalaulotteisen jatkumon, joka dimensionaalisuutta vähennettäessä eräällä tavoin avataan.

Tässä työssä esiteltävät menetelmät on jaettu niiden toimintaperiaatteen mukaan. Ensimmäisen kategorian muodostavat pääkomponenttianalyysi ja muut euklidiseen etäisyyteen pohjautuvat menetelmät. Toiseen kategoriaan kuuluvat menetelmät, jotka perustuvat euklidisen etäisyyden sijaan geodeettiseen etäisyyteen. Geodeettistä etäisyyttä approksimoidaan graafeilla, joten näitä menetelmiä nimitetään graafipohjaisiksi. Osa menetelmistä ei käytä suoraan datapisteiden välisiä etäisyyksiä, vaan pyrkii tekemään dimensionaalisuuden vähentämisen datan topologisten ominaisuuksien perusteella; näitä menetelmiä nimitetään topologisiksi menetelmiksi. Viimeinen menetelmä, satunnaisprojektio, ei mahdu mihinkään edellämainittuun kategoriaan, joten se esitellään oman otsakkeensa alla.

Menetelmien toimintaa havannoillistetaan niiden esittelyn yhteydessä neljän esimerkkidatajoukon avulle, jotka ovat kuvassa 10. Kussakin keinotekoisessa datajoukossa on 5000 pistettä ja vain vähän kohinaa. Kolme ensimmäistä datajoukkoa, kierukka, tasorulla ja silmukka, ovat esimerkkejä kolmiulotteisessa avaruudessa sijaitsevista epälineaarisista datajoukoista, joiden luontainen ulottuvuus on kaksi. Nämä datajoukot ovat myös yhtenäisiä ja muodostavat jatkumon. Neljäs datajoukko on viiden separoituvan klusterin ryhmä, jolla testataan miten menetelmät toimivat sellaisen datan tapauksessa, jolle ei voida tehdä oletusta yhtenäisyydestä. Myöhemmin, luvussa 4 testataan esiteltyjä menetelmiä aidon puhedatan avulla.

Jatkossa käytetään seuraavia merkintöjä ja ilmauksia: Alkuperäinen data sijaitsee $d$-uloitteisessa *data-avaruudessa* ja se projisoidaan tai upotetaan $p$-uloitteiseen *projektioavaruuteen* ($p < d$). Alkuperäiseen dataan voidaan viitata kokonaisuudessaan matriisina $\mathbf{X}$ $(n \times d)$, jonka yksittäinen rivi on datavektori $d$-ulotteinen $\mathbf{x}_i$. Dimensioltaan vähennetty data on $p$-ulotteisista vektoreista $\mathbf{y}_i$ koostuva matriisi $\mathbf{Y}$ $(n \times p)$.

## 3.1 Etäisyyspohjaiset menetelmät

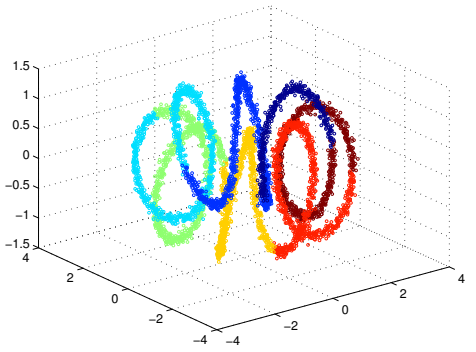
### 3.1.1 Pääkomponenttianalyysi

*Pääkomponenttianalyysi* (*Principal Component Analysis*, PCA) on dimensionaalisuuden vähentämisessä ja luontaisen dimensionaalisuuden analysoinnissa yleisimmin käytetty menetelmä. Menetelmä kehitettiin viime vuosisadan alussa [51] ja tunnetaan myös nimillä *Karhunen-Loève*- ja *Hotelling*-muunnos [40], [36].

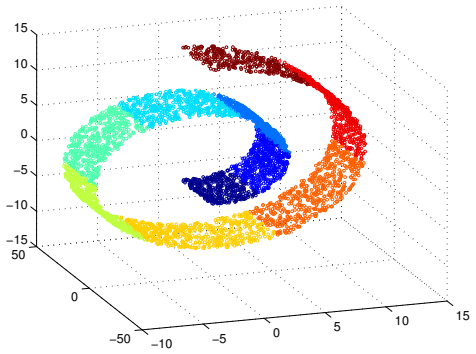
Pääkomponenttianalyysin perusajatuksena on löytää datalle sellainen projektio, jolle datan varianssi maksimoituu. Varianssi voidaan maksimoida seuraavasti.[4]

Alkuperäinen data $\mathbf{X}$ halutaan projisoida $p$-uloitteiseen avaruuteen siten, että sen varianssi maksimoituu. Tarkastellaan ensin projektiota yksiuloitteiseen avaruuteen. Yksiuloitteinen projektio tehdään vektorin $\mathbf{u}_1$ suuntaiseksi. Tällöin kukin datavektori $\mathbf{x}_i$ projisoituu skalaariksi $\mathbf{u}_1^T\mathbf{x}_i$. Projisoidun datan keskiarvo on

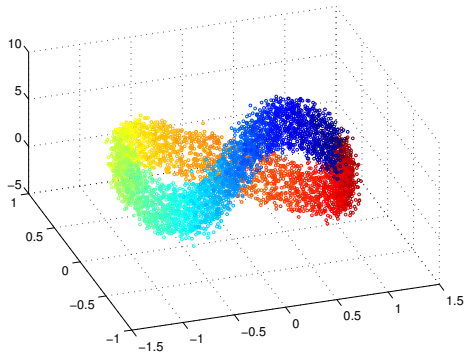
[4]Tässä seurataan Hotellingin tapaa formuloida pääkomponenttianalyysi [36]. Pearsonin alkuperäinen formulointi puolestaan pyrkii minimoimaan keskineliövirheen (*Mean Square Error*, MSE) [51].



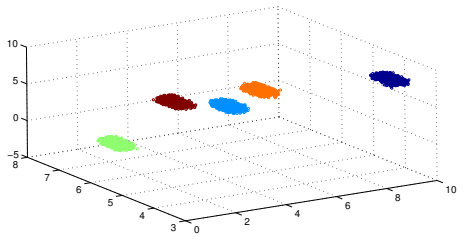
Kierukka



Tasorulla



Itsensä leikkaava silmukka



Klusterit: viisi separoituvaa ryhmää

Kuva 10: Esimerkkidatajoukkoja

$\mathbf{u}_1^T\bar{\mathbf{x}}$, jossa $\bar{\mathbf{x}}$ on alkuperäisen datan keskiarvo:

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \tag{12}$$

Projisoidun datan varianssi on puolestaan:

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{u}_1^T\mathbf{x}_i - \mathbf{u}_1^T\bar{\mathbf{x}})^2 = \mathbf{u}_1^T\mathbf{C}\mathbf{u}_1 \tag{13}$$

jossa $\mathbf{C}$ on datan kovarianssimatriisi:

$$\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T \tag{14}$$

Pyritään nyt maksimoimaan varianssi $\mathbf{u}_1$:n suhteen. Jotta $\|\mathbf{u}_1\|$ ei lähestyisi ääretöntä, oletetaan normalisointiehdoksi $\mathbf{u}_1^T\mathbf{u}_1 = 1$ ja liitetään se Lagrangen kertoimen ($\lambda_1$) avulla maksimoitavaan lausekkeeseen:

$$J(\mathbf{u}_i) = \mathbf{u}_1\mathbf{C}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1) \tag{15}$$

Derivoimalla yhtälö $\mathbf{u}_1$:n suhteen havaitaan, että varianssin ääriarvo saavutetaan kun

$$\mathbf{C}\mathbf{u}_1 = \lambda_1\mathbf{u} \tag{16}$$

mikä tarkoittaa sitä, että $\mathbf{u}_1$ on matriisin $\mathbf{C}$ ominaisvektori ja Lagrangen kerroin $\lambda_1$ sitä vastaava ominaisarvo. Jos lauseke kerrotaan puoliksi vasemmalta $\mathbf{u}_1^T$:lla huomataan, että koska $\mathbf{u}_1^T\mathbf{u}_1 = 1$, saadaan varianssi lausekkeesta

$$\mathbf{u}_1^T\mathbf{C}\mathbf{u}_1 = \lambda_1 \tag{17}$$

joka matriisien ominaisarvoyhtälön peruusteella maksimoituu, kun $\mathbf{u}_1$:ksi valitaan se ominaisvektori, jolla on suurin ominaisarvo $\lambda_1$. Tätä ominaisvektoria kutsutaan ensimmäiseksi pääkomponentiksi [11]. Koska projektio halutaan tehdä yksiuloitteisen avaruuden sijasta $p$-uloitteiseen avaruuteen, otetaan lopulliseen projektioon mukaan $p$ kappaletta ominaisarvoiltaan suurimpia ominaisvektoreita.

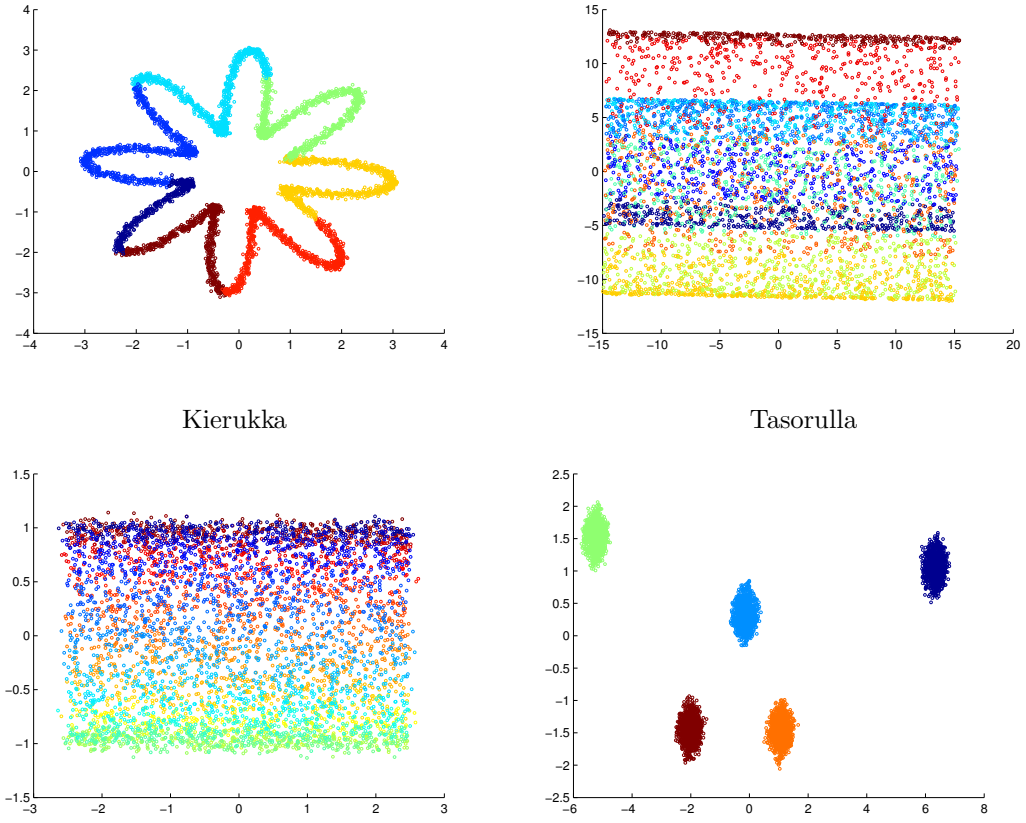
Algorimi 1 esitettää pääkomponenttianalyysin algoritmisessa muodossa. Pääkomponenttianalyysin laskennallinen vaativuus on $O(d^2n) + O(d^3)$ [11]. Algoritmin laskennallisesti vaativin osa on kovarianssimatriisin ominaisarvojen laskeminen ominaisarvohajotelmasta. Sen aikavaatimus on $O(d^3)$, mutta mikäli projektioulottuvuus $p$ on paljon pienempi kuin datan alkuperäinen ulottuvuus, voidaan käyttää potenssi-iteraatiomenetelmää (*power iteration*), jossa lasketaan vain halutut $p$ ominaisarvoa ja -vektoria. Potenssi-iteraatiomenetelmän aikavaativuus on $O(d^2n)$ [31]. Dimensionaalisuudesta johtuvan aikavaatimuksen vuoksi normaali pääkomponenttianalyysi voi olla laskennallisesti liian vaativa, jos data on erittäin korkeauloitteista [11, 29].

**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio

1. Keskitä data $\mathbf{X}$ siten, että datan keskiarvovektori $\mu$ on origossa (Tämä askel ei ole välttämätön, koska datan keskiarvo vaikuttaa ainoastaan kovarianssimatriisin ominaisarvojen suuruuteen, ei niiden keskinäiseen järjestykseen [58].)
2. Laske datan kovarianssimatriisi $\mathbf{C} = \mathbf{X}\mathbf{X}^T$
3. Laske kovarianssimatriisin $\mathbf{C}$ ominaisarvot $(\lambda_1, \ldots, \lambda_d)$ ja ominaisvektorit $(\mathbf{u}_1, \ldots, \mathbf{u}_d)$ ominaisarvohajotelmasta $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
4. Valitse $p$ kappaletta ominaisarvoiltaan suurinta ominaisvektoria ja muodostetaan näistä muunnosmatriisi $\mathbf{P}$ siten, että ominaisvektorit muodostavan matriisin $\mathbf{P}$ sarakkeet
5. Tee datan projektio $\mathbf{Y} = \mathbf{X}\mathbf{P}$

**Algoritmi 1:** Pääkomponenttianalyysi

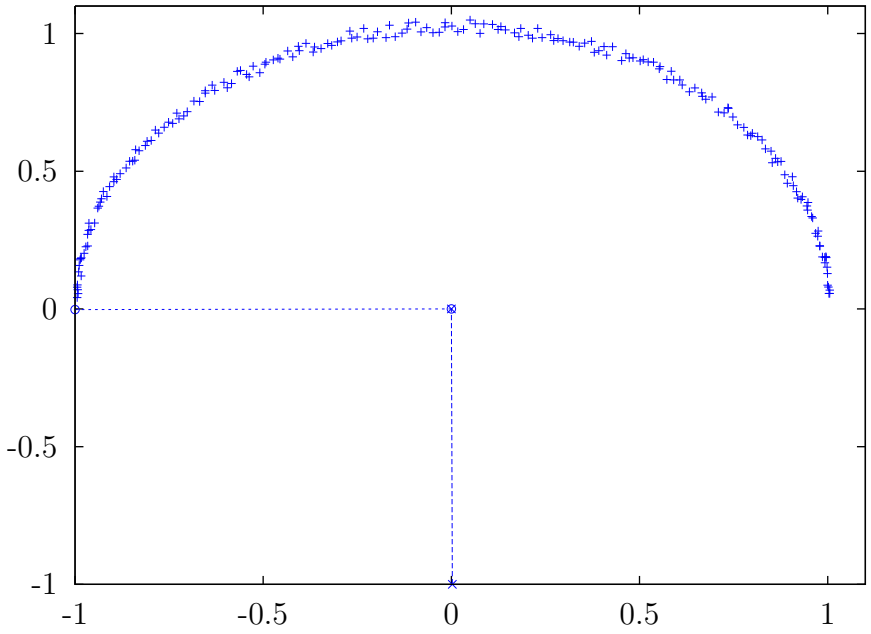
Kuvassa 11 on esitetty pääkomponenttianalyysin tuottama kaksiuloitteinen projektio kahdelle esimerkkidatajoukolle. Pääkomponenttianalyysi tuottaa etäisyydet hyvin säilyttävän projektion kierukkamaiselle datalle, koska datan varianssin suunta sattuu olemaan sopiva. Myös klusteridata säilyttää hyvin muotonsa. Sen sijaan kohtien rulla- ja silmukkadata näyttävät litistyvän, eivätkä joukkojen keskiosan pisteet säilytä keskinäisiä etäisyyksiään, koska datajoukkojen varianssi ja muoto eivät kohtaa.




Kuva 11: Pääkomponenttianalyysi esimerkkidatalle

Kuvat osoittavat pääkomponenttianalyysin heikkouden. Koska pääkomponentit, joiden mukaan projektio tehdään, ovat aina suorakulmaisia toisiinsa nähden, löytyy datasta ainoastaan lineaarisia rakenteita.

Pääkomponenttianalyysia voidaan käyttää datan projisoimisen lisäksi myös datan dimensionaalisuuden arviointiin. Pääkomponenttianalyysin mukaan datan luontainen uloitteisuus on datan kovarianssimatriisin nollasta poikkeavien ominaisarvojen lukumäärä. Pääkomponenttianalyysin lineaarisuus on heikkous myös dimensionaalisuuden arvioinnissa. Kuvassa 12 on esitetty ympyrän kaarelle sijoittuvaa dataa, jonka luontainen uloitteisuus on 1. PCA:n mukaan datalla on kuitenkin kaksi nollasta poikkeavaa ja miltei yhtä suurta ominaisarvoa. Menetelmä siis yliarvioi datan dimensionaalisuuden, koska se ei pysty erottamaan datan epälineaarisia riippuvuuksia.




Kuva 12: Puoliympyrän muotoiselle datalle löytyy kaksi yhtä suurta pääkomponenttia

### 3.1.2 Monidimensioskaalaus

*Monidimensioskaalaus* (*Multidimensional scaling*, MDS) on pääkomponenttianalyysin ohella toinen yleisesti käytetty menetelmä, jota on erityisesti käytetty psykologian ja yhteiskuntatieteiden alalla korkeaulotteisen datan visualisoimisessa [12]. Monidimensioskaalaus on epälineaarinen menetelmä, josta on olemassa myös lineaarinen variantti, joka on ekvivalentti pääkomponenttianalyysin kanssa. Lineaarisesta variantista käytetään nimitystä *klassinen skaalaus* tai *klassinen monidimensioskaalaus* (*classical multidimensional scaling*) [63]. Algoritmi 2 esittää klassisen skaalauksen. Se poikkeaa pääkomponenttianalyysistä korkeamman laskennallisen vaativuutensa osalta. Pääkomponenttianalyysissa lasketaan ominaisarvohajotelma kovarianssimatriisille, jonka koko on $d \times d$ kun klassisessa skaalauksessa ominaisarvohajotelma lasketaan puolestaan $n \times n$-kokoiselle Grammin matriisille.

Monidimensioskaalauksen perusajatuksena on löytää datalle sellainen konfiguraatio, joka säilyttää alkuperäisten datapisteiden väliset etäisyydet. Uuden pistekonfiguraation hyvyyttä mitataan jännitefunktiolla (*stress*). Yksinkertaisin jännitefunktio on *Kruskalin jännite* tai raaka jännite (*raw stress*) [42]:

**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio

1. Keskitä data ja muodosta Grammin matriisi $\mathbf{S} = \mathbf{X}^T\mathbf{X}$
2. Laske ominaisarvohajotelma $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
3. Tee datan projektio $\mathbf{Y} = \mathbf{I}_{p\times n}\mathbf{\Lambda}^{1/2}\mathbf{U}^T$

**Algoritmi 2:** Klassinen monidimensioskaalaus

$$S_{Krus} = \sum_{i<j}(d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d(\boldsymbol{y}_i - \boldsymbol{y}_j))^2 \tag{18}$$

jossa $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$ on pisteiden etäisyys alkuperäisessä avaruudessa ja $d(\boldsymbol{y}_i - \boldsymbol{y}_j)$ etäisyys kohdeavaruudessa. Toinen yleinen jännitefunktio on Sammonin kuvaus (*Sammon's mapping*) [39]:

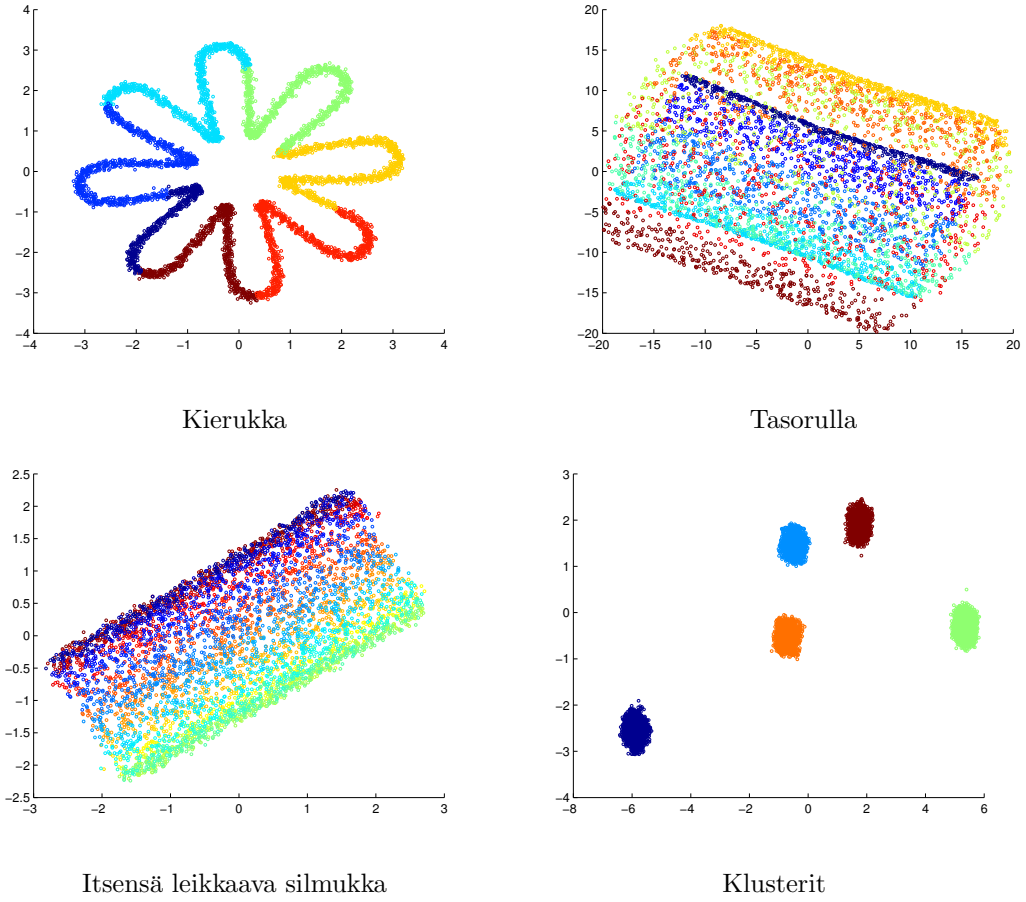
$$S_{Samm} = \sum_{i<j} \frac{(d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d(\boldsymbol{y}_i - \boldsymbol{y}_j))^2}{d(\boldsymbol{x}_i, \boldsymbol{x}_j)} \tag{19}$$

Sammonin jännitefunktio poikkeaa Kruskalin funktiosta siten, että pienemmillä etäisyyksillä on suurempi paino kuin suurilla. Erilaisia jännitefunktioita, jotka poikkeavat toisistaan etäisyyksien painotuksen suhteen on useita [12]. Etäisyyksien lisäksi jännitefunktiossa voidaan käyttää esimerkiksi etäisyyksien logaritmeja (*multiscale*) [53]. Toinen esimerkki hieman erilaisesta lähestymistavasta monidimensioskaalaukseen on jännitefunktiosta muodostettu bayesilainen malli [48].

Etäisyyspohjaiset jännitefunktiot eroavat toisistaan sen suhteen, millainen paino erilaisille etäisyyksille annetaan. Sen vuoksi yleinen jännitefunktio voidaan formuloida Kruskalin funktiosta lisäämällä kullekin etäisyydelle painoarvotekijä:

$$S = \sum_{i<j} w_{ij}(d(\boldsymbol{x}_i, \boldsymbol{x}_j) - d(\boldsymbol{y}_i - \boldsymbol{y}_j))^2 \tag{20}$$

Kun datalle on valittu sopiva jännitefunktio, pyritään se minimoimaan käyttämällä jotakin optimointimenetelmää. Jännitefunktio ei ole konveksi ja sillä




Kuva 13: Monidimensioskaalaus esimerkkidatalle (Sammonin jännitefunktio)

voi olla useita lokaaleja minimejä, joten skaalauksen optimaalisuutta ei voida taata [12].

Monidimensioskaalauksessa käytetään yleensä erityistä *SMACOF*-majorointialgoritmia (*Scaling by majorizing a complicated function*) [18]. *SMACOF*-algoritmissa (3) jännitefunktion sijaan minimoidaan sen konveksia approksimaatiota, joka takaa algoritmin konvergoitumisen, mutta ei optimaalisuutta [12]. Algoritmin aikavaativuus on $O(z(n^3 + pn^2))$, jossa $z$ on iteraatioiden määrä, ja koska yleensä $p \ll n$, voidaan aikavaativuuden sanoa olevan $O(zn^3)$ [5].

Kuvassa 13 on esitetty monidimensioskaalauksen tulos esimerkkidatalle. Monidimensioskaalaus säilyttää esimerkkidatajoukoilla pisteiden väliset etäisyydet hyvin (kierukka, klusterit), mutta samoin kuin pääkomponenttianalyysin tapauksessa, mahdollinen rakenne katoaa (tasorulla, silmukka).

**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio

$\mathbf{X}[0] \longleftarrow$ satunnainen k-dimensioinen aloituskonfiguraatio
$\mathbf{Z} \longleftarrow \mathbf{X}[0]$
$i \longleftarrow 0$
Laske jännite: $s[-1] = s[0] = S(\mathbf{X}[0])$
**while** $i = 0$ **or** $(s[i-1] - s[i]) > \epsilon$ **and** $i \leq$ maksimi-iteraatiomäärä **do**
$i \longleftarrow i + 1$
$\mathbf{X}[i] = n^{-1}\mathbf{B}(\mathbf{X}, \mathbf{X}[\mathbf{i-1}])\mathbf{X}[\mathbf{i-1}]$
$s[i] \longleftarrow S(\mathbf{X}[i])$
$\mathbf{Y} \longleftarrow \mathbf{X}[i]$
**end while**
**return** $\mathbf{Y}$ {skaalauksen tulos}

$\mathbf{B}(\mathbf{X}, \mathbf{Z})$:n elementit:

$$b_{ij} = \begin{cases} -\frac{w_{ij}d(x_i,x_j)}{d(z_i,z_j)} & \text{jos } i \neq j \text{ ja } d(z_i,z_j) \neq 0 \\ 0 & \text{jos } i \neq j \text{ ja } d(z_i,z_j) = 0 \end{cases}$$

$$b_{ii} = -\sum_{j=1, j\neq i}^{n} b_{ij}$$

**Algoritmi 3:** SMACOF-algoritmi

### 3.1.3 Käyrälineaarinen komponenttianalyysi

Pääkomponenttianalyysissa muodostetaan projektio pienempään dimensioon koko datajoukon varianssin perusteella. Monidimensioskaalauksessa puolestaan minimoidaan projektion kaikkien pisteiden etäisyyksien muutoksia. Jotkut datajoukot voivat kuitenkin olla sellaisia, että voi olla järkevää keskittyä optimoimaan vain tiettyä osaa pisteiden välisistä etäisyyksistä. *Käyrälineaarinen komponenttianalyysi* (*Curvilinear Component Analysis*, CCA) [20] perustuu tähän ajatukseen. Menetelmä muistuttaa monidimensioskaalausta, mutta keskittyy minimoimaan ainoastaan sellaisten pisteiden välistä etäisyyttä, jotka ovat lähellä toisiaan projektioavaruudessa.

CCA:ssa optimoidaan seuraavaa virhefunktiota:

$$E = \frac{1}{2} \sum_i \sum_{j \neq i} (d(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{y}_i - \mathbf{y}_j))^2 F(d(\mathbf{y}_i, \mathbf{y}_j), \sigma) \tag{21}$$

Tekijä $F$ erottaa CCA:n monidimensioskaalauksesta. Se määrittää millaisen painoarvon pisteet saavat virhefunktiossa ja voidaan määritellä esimerkiksi seuraavasti:

$$F(d(\mathbf{y}_i, \mathbf{y}_j), \sigma) = \begin{cases} 1 & \text{jos } d(\mathbf{y}_i, \mathbf{y}_j) \leq \sigma \\ 0 & \text{jos } d(\mathbf{y}_i, \mathbf{y}_j) > \sigma \end{cases} \tag{22}$$

Tällaisessa tapauksessa CCA jättää virhefunktion laskennassa huomiotta sellaiset pisteet jotka ovat projektioavaruudessa kaukana (kauempana kuin etäisyyden $\sigma$ päässä) toisistaan. Tämä mahdollistaa tietynlaisten pintojen aukirepimisen [44].

CCA:n toinen eroavuus monidimensioskaalaukseen on virhefunktion optimointimenetelmä. Tavallisessa stokastisessa gradienttilaskeutumismenetelmässä (*stochastic gradient descent*) kukin yksittäinen vektori $\mathbf{y}_i$ liikkuu kaikkien muiden vektoreiden $\mathbf{y}_j$ vaikutuksesta. CCA:n virhefunktion optimoinnissa kiinnitetään paikalleen yksittäinen vektori $\mathbf{y}_i$ ja kaikkia muita vektoreita $\mathbf{y}_j$ liikutetaan ottamatta huomioon niiden keskinäistä vuorovaikutusta.

Stokastisessa gradienttilaskeutumisessa projisoitujen vektoreiden muutos saadaan seuraavasta lausekkeesta:

$$\Delta \mathbf{y}_i \approx -\nabla_i E \tag{23}$$

Jos virhefunktio kirjoitetaan osiensa summaksi seuraavasti:

$$E = \frac{1}{2} \sum_i \sum_{j \neq i} E_{ij} \tag{24}$$

CCA:n yksittäisen vektorin optimointi saadaan seuraavasta lausekkeesta, jossa $i$ on satunnaisesti valittu, kiinnitetty vektori ja $\alpha$ oppimistekijä

$$\Delta \mathbf{y}_j(i) = \alpha(t) \nabla_i E_{ij} = -\alpha(t) \nabla_j E_{ij} \tag{25}$$

Auki kirjoittettuna optimontiaskeleeksi saadaan [20]:

$$\Delta \mathbf{y}_j = \alpha(t) F(d(\mathbf{y}_i, \mathbf{y}_j), \sigma) d(\mathbf{x}_i, \mathbf{x}_j) \frac{\mathbf{y}_j - \mathbf{y}_i}{d(\mathbf{y}_i, \mathbf{y}_j)} \ \forall j \neq i \tag{26}$$

CCA:n optimointialgoritmi säästää aikaa, koska kunkin vektorin muutoksen laskemisen aikavaativuus on $O(n)$, [5] kun se tavallisessa gradienttilaskeutumisessa on $O(n^2)$ [20]. Erona gradienttilaskeutumiseen on myös se, virhefunktio $E$ voi hetkittäin kasvaa, vaikka se keskimääräisesti on laskeva [44, 20].

Nimestään huolimatta CCA-algoritmi ei ole sukua pääkomponenttianalyysille. Tekijät itse ilmoittivat algoritmin olevan parannettu versio Kohosen itseorganisoituvista kartoista [41, 20]. Alkuperäiseen algoritmiin kuuluu ensimmäisenä vaiheena on datan vektorikvantisointi. Jos vektorikvantisoitia käytetään, putoaa varsinaisen dimensioreduktion tilavaativuus $O(m^2)$:n ja aikavaativuus $O(m^2 p)$, missä $m$ on prototyyppien lukumäärä. Algoritmi 4 esittää CCA:n algoritmisessa muodossa.

Kuvassa 14 on esitetty CCA:n tulokset esimerkkidatalle. Oppimistekijän alkuarvona käytettiin lukua 0.5. Parametri $\sigma$, eli vaikutusalue oli kolme kertaa datajoukon keskihajonta. Kierukka- ja klusteridata projisoituvat hyvin kaksiuloitteiseen

[5] Tämä pätee kaikille muille vektoreille lukuunottamatta kiinnitettyä vektori $\mathbf{y}_i$:tä, jonka muunnoksen laskeminen vie edelleen ajan $O(n^2)$.

avaruuteen, mutta rulla- ja silmukkadatassa projektio ei säilytä alkuperäisen datan muotoa, vaikkakin paikalliset naapuruussuhteet säilyvät hyvin. Parametrin $\sigma$ säätäminen voi parantaa projektion laatua.

**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio
$\sigma$ – optimoitavan naapuruston koko
$\alpha$ – oppimisvakio

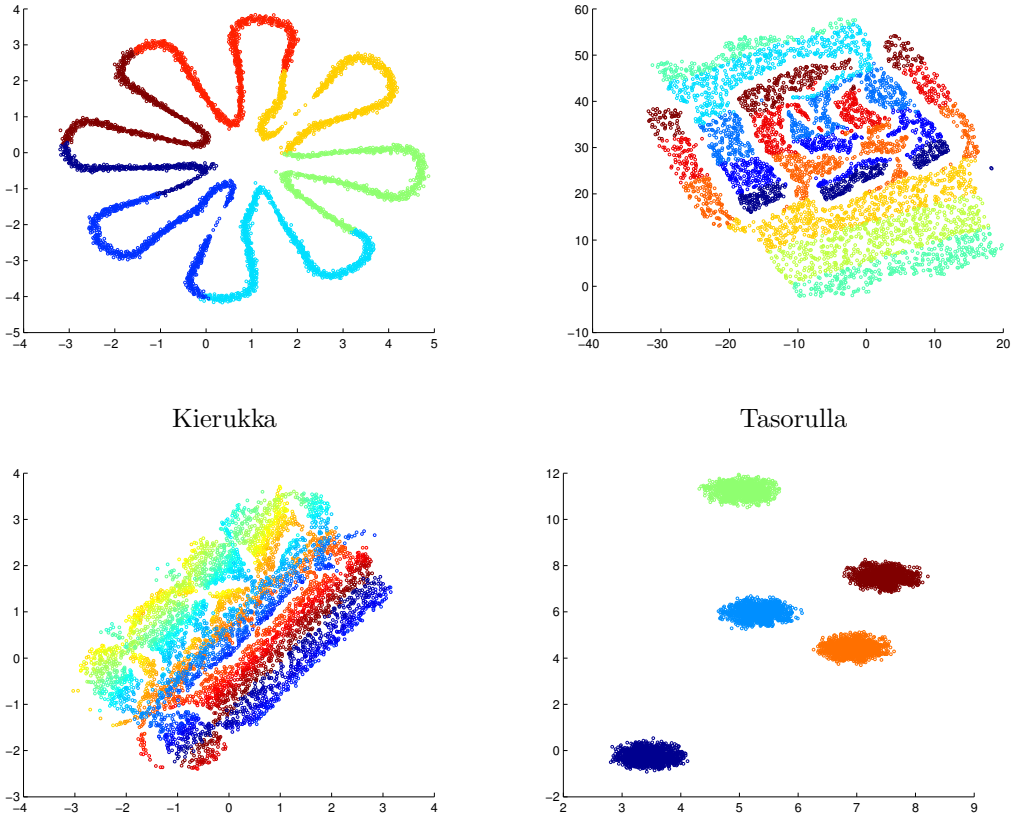
1. **Valinnainen askel:** Tee datalle vektorikvantisointi.
2. Lasken vektoreiden väliset etäisyydet.
3. Luo alustava satunnainen datakonfiguraatio $p$-uloitteiseen projektioavaruuteen, $q \leftarrow 1$.
4. Valitse oppimisvakio $\alpha$ ja naapuruston koko $\sigma$ epookin $q$ ajaksi.
5. Toista kunnes kaikki pisteet $\mathbf{x}(i)$ on valittu kerran: Valitse datapiste $\mathbf{x}(i)$ ja päivitä muut pisteet päivityssäännön (lauseke 26) mukaan.
6. $q \leftarrow q + 1$, jos virhefunktio ei ole konvergoitunut, palaa kohtaan 4 (uusi epookki).

**Algoritmi 4:** CCA-algoritmi

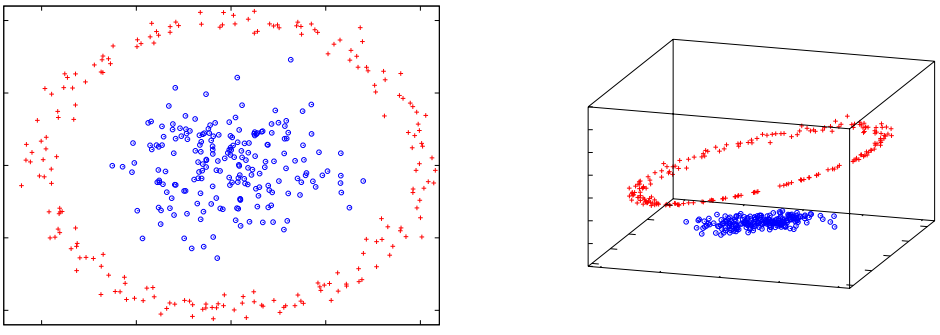
### 3.1.4 Ydinpääkomponenttianalyysi

Pääkomponenttianalyysin heikkous on, ettei sen avulla pystytä löytämään datassa olevia epälineaarisia riippuvuuksia. Menetelmää on pyritty laajentamaan eri tavoin myös epälineaariselle datalle sopivaksi. Eräs laajennoksista on *ydinmenetelmän* käyttö (*kernel method*).

Ydinmenetelmän ajatuksena on projisoida epälineaarinen data sellaiseen korkeampiuloitteiseen *piirreavaruuteen* (*feature space*), jossa dataan voidaan soveltaa lineaarisia menetelmiä. Kuvassa 15 on esimerkki kahdesta datajoukosta, jotka ovat lineaarisesti separoitumattomia. Kun data projisoidaan sopivaan korkeampiuloitteiseen piirreavaruuteen, separoituu se lineaarisesti.




Kuva 14: CCA esimerkkidatalle



Kuva 15: Projektio piirreavaruuteen funktion $\phi(\mathbf{x}) = (x_1, x_2, x_1^2 x_2^2)^T$ avulla. Vasemmalla alkuperäinen lineaarisesti separoitumaton data. Kun data on projisoitu piirreavaruuteen, on se hypertason avulla separoituva.

Koska datan käsittely on yleensä aina laskennallisesti vaativampaa korkeauloitteisissa avaruuksissa, ei suoran piirreavaruusprojektioiden tekeminen ole yleensä käytännöllistä. Tämän vuoksi dataa ei käsitellä piirreavaruudessa suoraan, vaan implisiittisesti piirreavaruuden pistetulojen avulla. Esimerkiksi seuraavalla kuvauksella piireavaruuteen

$$\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T \tag{27}$$

on ominaisuus, joka yhdistää piirreavaruuden pistetulon alkuperäiseen dataan:

$$\phi(\mathbf{x}_1)^T\phi(\mathbf{x}_2) = (\mathbf{x}_1^T\mathbf{x}_2)^2 \tag{28}$$

Alkuperäisestä datasta muodostetaan erityinen ydinmatriisi (*kernel matrix*), joka sisältää datan pistetulot piirreavaruudessa. Koska pistetulot lasketaan implisiittisesti alkuperäisen datan avulla, voi piirreavaruus olla jopa ääretönulotteinen algoritmien laskennallisen vaativuuden säilyessä matalana. Menetelmää kutsutaan *ydinsijoitukseksi* (*kernel substitution*, myös *kernel trick*) ja sitä voidaan käyttää kaikkien sellaisten algoritmien yhteydessä, jotka voidaan formuloida pistetulojen avulla [57, 4].

Pääkomponenttianalyysi voidaan formuloida piirreavaruudessa analogisesti tavallisen pääkomponenttianalyysin tavoin. Datan kovarianssimatriisi piirreavaruudessa on:

$$C = \frac{1}{n}\sum_{i=1}^{n}\phi(\mathbf{x_i})\phi(\mathbf{x_i})^T$$

jolle voidaan laskea ominaisarvot ja -vektorit periaatteessa analogisesti tavallisen pääkomponenttianalyysin tapaan[6]

Ydinpääkomponenttianalyysi pystyy erottamaan datasta epälineaarisen variansin. Laskennallisesti se on vaativampi kuin tavallinen pääkomponenttianalyysi, koska ydinmatriisi vaatii tilan $O(n^2)$. Myös ominaisarvojen laskeminen piirreavaruudessa on työläämpää kuin tavallisessa tapauksessa, koska ominaisarvot las-

[6]Datan keskittäminen piirreavaruudessa mutkistaa hieman algoritmin johtamista. Ks. yksityiskohdat esim. [57, 58, 11].

ketaan ydinmatriisista, joka on kooltaan $n \times n$. Ydinpääkomponenttianalyysissa täytyy lisäksi laskea matriisin kaikki ominaisarvot, joten potenssi-iteraatiomenetelmää ei voida soveltaa [11], jolloin algoritmin laskennallinen vaativuus nousee luokkaan $O(n^3)$.

**Syöte:** $X$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio

1. Laske ydinmatriisi: $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j), i, j = 1, \ldots, n$
2. Keskitä data piirreavaruudessa:
   $\mathbf{K} \leftarrow \mathbf{K} - \frac{1}{n}\mathbf{j}\mathbf{j}^T\mathbf{K} - \frac{1}{n}\mathbf{K}\mathbf{j}\mathbf{j}^T - \frac{1}{n^2}\mathbf{j}^T\mathbf{K}\mathbf{j}(\mathbf{j}\mathbf{j}^T)$
   ($\mathbf{j}$ =yksikkövektori) (Kaksoiskeskitys)
3. Laske ydinmatriisin $\mathbf{K}$ ominaisarvot ($\mathbf{\Lambda}$) ja ominaisvektorit ($\mathbf{V}$)
4. Laske projektiokertoimet: $\alpha_\mathbf{j} = \frac{1}{\sqrt{\lambda_i}}\mathbf{v}_j, j = 1, \ldots, k$
5. Tee projektio datalle: $\mathbf{y}_i = \sum_{i=1}^{n} \alpha_\mathbf{ij} k(\mathbf{x_i}, \mathbf{x})$

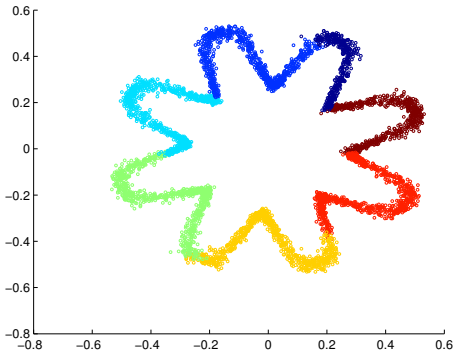
**Algoritmi 5:** Ydinpääkomponenttianalyysi

Algoritmin toinen ongelmallinen piirre liittyy käytettävän ytimen valintaan. Ydinmatriisi voi olla periaattessa mikä tahansa positiivinen semidefiniittimatriisi, mutta käytännössä data vaikuttaa ytimen toimivuuteen [58]. Yleisesti käytettyjä ytimiä ovat polynomiset ($k(\mathbf{x_1}, \mathbf{x_1}) = (\mathbf{x_1}^T\mathbf{x_2})^n$) ja gaussiset ($k(\mathbf{x_1}, \mathbf{x_1}) = e^{-\|\mathbf{x_1}-\mathbf{x_2}\|^2/2\sigma^2}$).

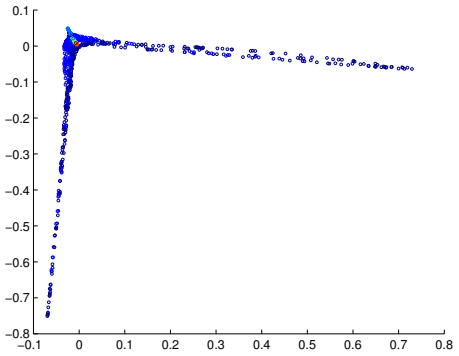
Kuvassa 16 on esitetty gaussilaisella ytimellä tehdyn ydinpääkomponenttianalyysin tulokset koedatajoukoille. Projektio onnistuu parhaiten kierukkadatalle, mutta klusteri- ja tasorulladata diskriminointi ei onnistu. Klustereiden kohdalla huomataan, että yksi klusteri kuvatuu päällekäin kolmen muun klusterin kanssa. Tasorulla puolestaan romahtaa yhdeksi pistejoukoksi. Ytimen valinta vaikuttaa tuloksiin. Kuva 17 osoittaa miten polynomisen ytimen käyttö muuttaa tuloksia.

## 3.2 Graafipohjaiset menetelmät

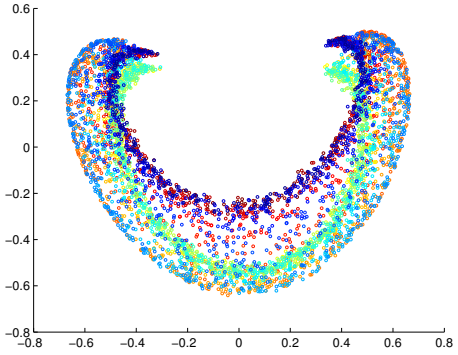
Edellä esitellyt etäisyyspohjaiset menetelmät eivät tee mitään oletuksia alkuperäisen datan muodosta ja ovat siksi yleisiä. Laskettu projektio perustuu ai-



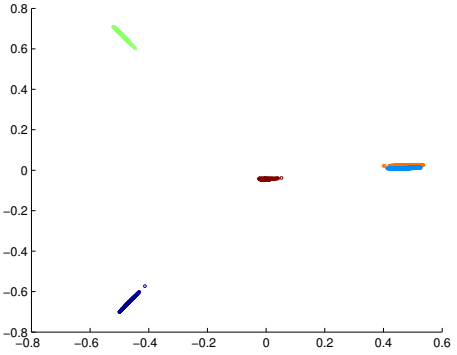
Kierukka



Tasorulla

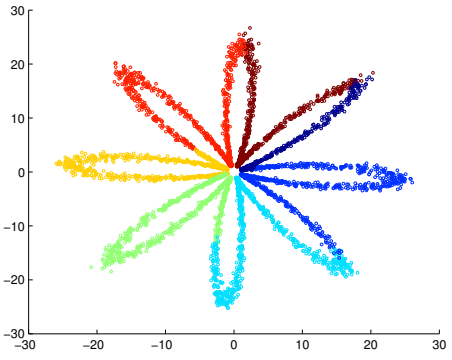


Itsensä leikkaava silmukka

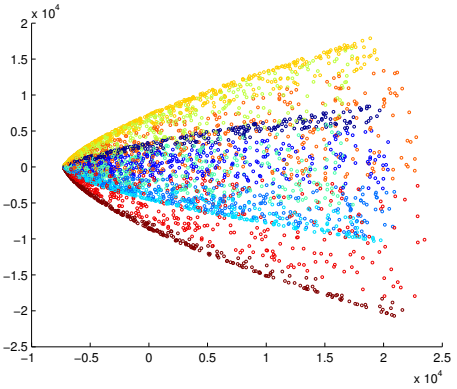


Klusterit

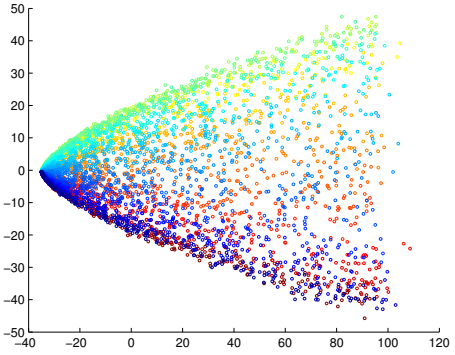
Kuva 16: Ydinpääkomponenttianalyysi gaussilaisella ytimellä



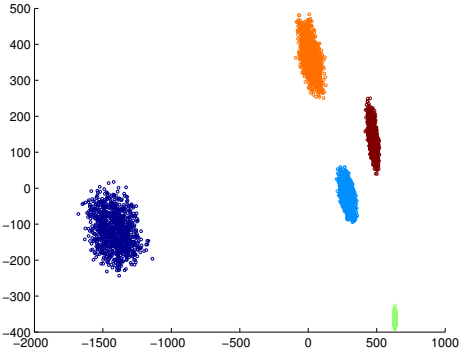

Kierukka




Tasorulla




Itsensä leikkaava silmukka




Klusterit

Kuva 17: Ydinpääkomponenttianalyysi polynomisella ytimellä

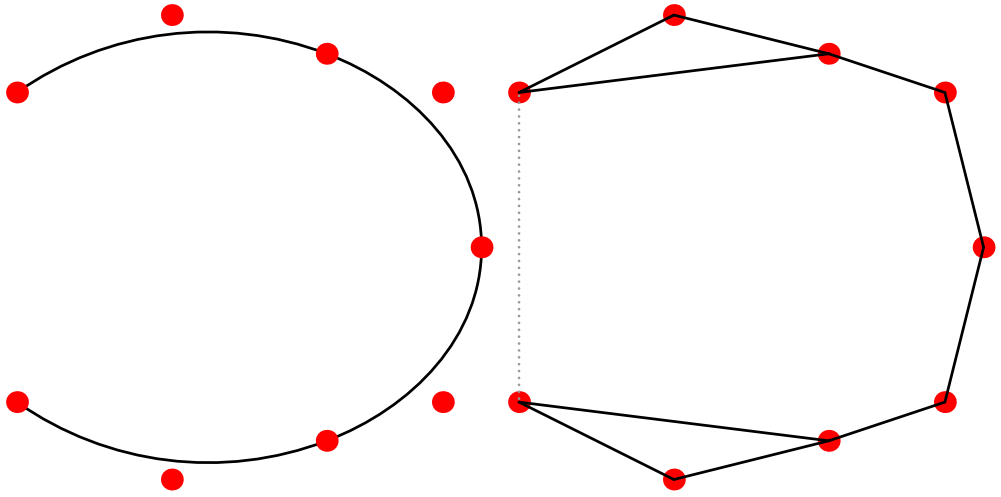
noastaan pisteiden välisiin euklidisiin etäisyyksiin. Monissa tapauksissa alkuperäinen data, vaikka sijaitseekin korkeauloitteisessa avaruudessa, on syntynyt sellaisten prosessien tuloksena, joiden vapausasteet ovat pienempiä kuin dataavaruus. Mikäli datan synnyttävän prosessin muutokset ovat jatkuvia, on tuloksena usein dataa, joka sijaitsee jonkinlaisella hyperpinnalla (*smoothness*) [11]. Esimerkkidatajoukoista tasorulla, itsensä leikkaava silmukka ja kierukka ovat tällaisia datajoukkoja.

Dimensioreduktiomenetelmät, jotka optimoivat datapisteiden välisiä euklidisia etäisyyksiä eivät ota huomioon mahdollisten pintojen olemassaoloa ja niiden tuottamat projektiot voivat hajottaa hyperpinnat, kuten osa aiemmista esimerkeistä osoittaa.

*Geodeettisten etäisyyksien* (*geodesic distance*) käyttäminen euklidisten etäisyyksien sijaan ottaa huomioon hyperpintojen olemassaolon. Geodeettinen etäisyys tarkoittaa pisteiden etäisyyden mittaamista jotakin pallopintaa pitkin, kun euklidinen etäisyys määrittää pisteiden välisen lyhimmän etäisyyden alkuperäisessä avaruudessa.

Kuvassa 18 havainnollistetaan kaksiulotteista geodeettista etäisyyden mittaamista. Kuvassa vasemmalla on joukko pisteitä, jotka sijaitsevat likimääräisesti ympyrän kaarella. Pisteiden väliset geodeettiset etäisyydet mitataan kaarta pitkin. Aitoja geodeettisiä etäisyyksiä voidaan approksimoida muodostamalla pisteiden välille naapurustograafi ja mittaamalla graafin kaarten pituudet. Kuvassa graafi on muodostettu naapuruston koolla $k = 2$. Jos naapuruston koko on liian suuri, graafiin muodostuu oikoteitä, kun pisteet jotka geodeettisesti eivät ole toistensa naapureita lasketaan sellaisiksi. Kuvassa oikealla kahden pisteen välinen katkoviiva kuvaa erästä oikotietä, joka muodostuu, jos naapuruston kooksi valitaan $k = 3$.

Geodeettisisten etäisyyksien käyttämisestä saatava hyöty on siinä, että menetelmissä voidaan ottaa huomioon sen hyperpinnan muoto, jolla data sijaitsee. Pisteiden väliset geodeettiset etäisyydet ovat lyhyitä silloin, jos pisteet sijaitsevat lähellä toisiaan tietyllä hyperpinnalla. Euklidisten etäisyyksien tapauksessa pisteet voivat olla lähellä toisiaan, mutta ne eivät silti ole naapureita samalla hyperpinnalla. Esimerkkidatajoukoista tasorulla havainnollistaa tilannetta parhaiten: rullan molemmissa päissä olevien pisteiden välinen euklidinen etäisyys on



Kuva 18: Geodeettisen etäisyyden määrittäminen. Kuvassa vasemmalla joukko pisteitä, jotka sijaitsevat likimääräisesti pallopinnalla. Oikealla pisteiden välille muodostettu naapurustograafi ($k = 2$). Graafin kaaret approksimoivat aitoja geodeettisia etäisyyksiä.

melko pieni, joten edellä käytettyjen euklidisia etäisyyksiä hyödyntävien menetelmien tapauksessa oli todennäköistä, että pisteet kuvautuivat projektiossa lähelle toisiaan.

Dimensionaalisuuden vähentämisalgoritmeissa geodeettisia etäisyyksiä voidaan käyttää suoraan joko euklidisten etäisyyksien korvaajina tai hyödyntää niiden avulla saatavaa informaatiota datapisteiden naapurustoista.

Seuraavassa esitellään kaksi menetelmää, *Isomap* ja *semidefiniittiupotus* (SDE), joissa hyödynnetään eri tavoin naapurustograafin avulla saatavia geodeettisten etäisyyksien approksimaatioita. Myös myöhemmin esiteltävät topologiset menetelmät hyödyntävät naapurustograafia.

### 3.2.1 Isomap

Isomap-algoritmi [61, 62] oli ensimmäisiä algoritmeja, joka pohjautui graafipohjaisiin geodeettisiin etäisyyksiin. Algoritmista on sittemmin kehitetty useita variantteja, seuraavassa esitellään alkuperäinen algoritmi. Pääajatuksena on yhdistää geodeettiset etäisyydet, tai oikeammin niiden graafipohjaiset approksimaatiot klassiseen monidimensioskaalaukseen.

Aluksi datasta muodostetaan naapurustograafi ottamalla kunkin datapisteen naapuriksi sen $k$ lähintä naapuria, tai ottamalla solmun naapureiksi kaikki $\epsilon$-säteisen hyperpallon sisälle sattuvat pisteet. Graafin kaaret saavat painoikseen pisteiden (solmujen) väliset etäisyydet.

Naapurustograafin kullekin solmulle lasketaan lyhimmät etäisyydet kaikkiin muihin solmuihin. Alkuperäisissä algoritmin versioissa lyhimmät etäisyydet lasketaan Floydin algoritmilla [28], aikavaativuus $\Theta(n^3)$, mutta nykyiset variantit käyttävät tehokkaampaa Dijkstran algoritmia, joka aikavaativuus on toteutuksesta riippuen $O(n^3)$ tai $O(n^2k)$ [21].

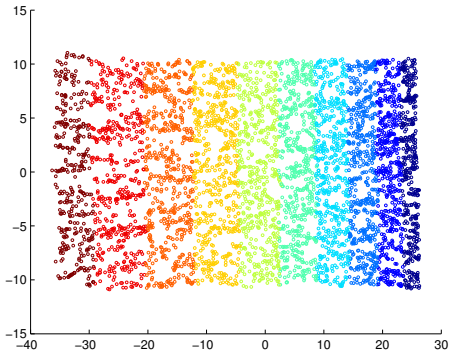
Saadusta $n \times n$ -kokoisesta lyhiempien etäisyyksien matriisista tehdään Grammin matriisi kaksoiskeskittämällä[7] se ja tämän jälkeen siitä lasketaan ominaisarvohajotelma, jonka avulla data projisoidaan kohdeavaruuteen. Ominaisarvohajotelman laskeminen on algoritmin vaativin osa, sen aikavaativuus on $O(n^3)$ [19]. Algoritmin viimeinen vaihe on sama kuin klassinen monidimensioskaalaus, mutta nyt euklidisten etäisyyksien sijaan käytetään graafietäisyyksiä. Algoritmi 6 esittää Isomapin vaiheet.

Isomap-algoritmista on esitetty joitakin variantteja. *C-Isomapissa* pyritään saamaan naapurustograafin etäisyydet vastaamaan tarkemmin geodeettisia etäisyyksiä korvaamalla kaarten painoina olevat pisteiden väliset euklidiset etäisyydet painotetuilla etäisyyksillä. C-Isomapin naapurustograafin etäisyyslauseke on $|\frac{\mathbf{x}_i - \mathbf{x}_j}{\sqrt{M(i)M(j)}}|$, jossa $M(i)$ on pisteen $\mathbf{x}_i$ ja sen $k$:n lähimmän naapurin etäisyyksien keskiarvo. *Landmark Isomapissa* puolestaan pyritään keventämään algoritmin laskennallista vaativuutta käyttämmällä laskennassa vain osaa datapisteistä ja interpoloimalla loput datapisteet [19].

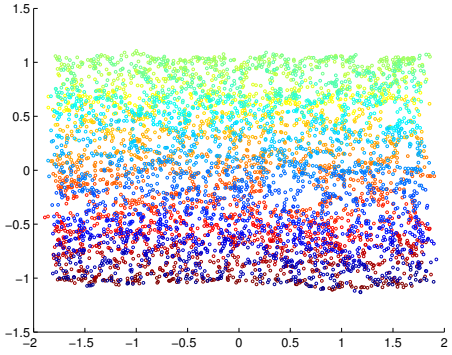
Kuvassa 19 on esitetty Isomap-algoritmin tuloksia esimerkkidatalle. Kuvissa on käytetty Isomapin Landmark-varianttia ja naapuruston kokona oli $k = 12$. Kierukkadata projisoituu hyvin kaksiuloitteiseen avaruuteen. Myös tasorulla avatuu kaksiulotteiseksi pinnaksi, mutta datapisteet jakautuvat projektiossa epätasaisesti ja pinta näyttää rakeiselta tai ikäänkuin reikäiseltä juustolta. Projektion tiheyden epätasainen jakautuminen johtuu siitä, etteivät graafietäisyydet pysty approksimoimaan geodeettisia etäisyyksiä tarpeeksi tarkasti. Kuvassa esiintyvät artifaktit ovat tyypillisiä Isomap-algoritmille [44].

[7] Kaksoiskeskittämistä käytetään etäisyysmatriisien keskittämisessä.

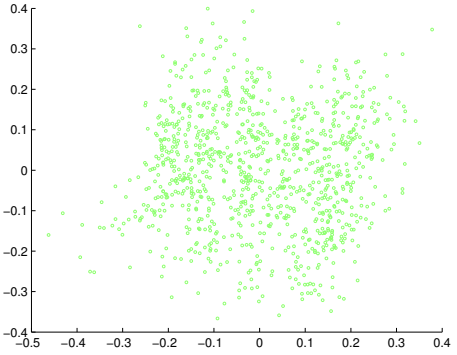

Kierukka




Tasorulla




Itsensä leikkaava silmukka




Klusterit

Kuva 19: Landmark Isomap-algoritmin naapuruston koolla $k = 12$

**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio
$k$ tai $\epsilon$ – graafin naapuruston koko

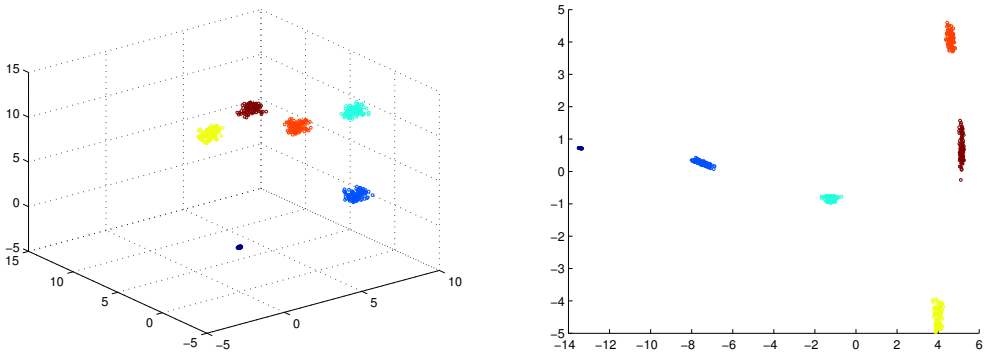
1. Muodosta datasta graafi joko $k$-lähimmän naapurin mukaan tai $\epsilon$-ympäristön mukaan
2. Painota graafin kaaret: kunkin kaaren paino on pisteiden välinen euklidisen etäisyys
3. Laske pisteiden väliset lyhimmät etäisyydet Dijkstran tai Floydin algoritmilla, neliöi ne ja talleta ne matriisiin $\mathbf{D}$
4. Muunna matriisi $\mathbf{D}$ Grammin matriisiksi $\mathbf{S}$ kaksoiskeskittämällä $\mathbf{D}$ (vrt. algoritmi 5, askel 2)
5. Laske $\mathbf{S}$:n ominaisarvohajotelma $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
6. Projisoi data $\mathbf{Y} = \mathbf{I}_{p\times n}\mathbf{\Lambda}^{1/2}\mathbf{U}^T$

**Algoritmi 6:** Isomap-algoritmi

Isomap-algoritmi ei pysty tuottamaan hyvää projektiota itsensä leikkaavalle silmukalle, mutta suurempi algoritmiin liittyvä ongelma tulee esiin klusteridatassa. Alkuperäinen testidata, joka koostuu 5000 datapisteestä ei muodosta yhtenäistä graafia vaikka naapuruston kokoa suurennettaan. Isomap-algoritmi toimii ainoastaan sellaiselle datan osajoukolle, josta voidaan muodostaa yhteinäinen graafi. Teoriassa naapuruston kokoa voitaisiin kasvattaa siten, että $k = n$, jolloin graafi on varmasti yhtenäinen. Tämä kasvattaa kuitenkin käytännössä algoritmin aikavaativuutta liikaa. Lisäksi tällöin menetetään graafietäisyyksien tuoma etu hyperpinnan mallintajina.

Huomattavaa on myös, että naapuruston koolla $k = 12$ myös muiden datajoukkojen naapurustograafeista jäi pois yksittäisiä datapisteitä joten aivan kaikille datapisteille ei löytynyt projektiota. Kuvassa 20 on esitetty Isomap-algoritmin tuloksia 500 datapisteestä koostuvalle klusterijoukolle. Yhteinäisen naapurustograafin saamiseksi käytetty naapuruston koko oli $k = 150$.

Isomapin ajatusta graafietäisyyksien käytöstä euklidisten etäisyyksien sijaan hyödyntää myös *käyrälineaarinen etäisyysanalyysi* (*Curvilinear Distance Ana-*



Alkuperäiset klusterit

Isomap, $k = 150$

Kuva 20: Isomap 500 alkion datajoukolle

*lysis*, CDA) [45, 46]. CDA on olennaisesti sama menetelmä kuin käyrälineaarinen komponenttianalyysi (CCA), mutta algoritmissa euklidiset etäisyydet on korvattu naapurustograafin etäisyyksillä.

### 3.2.2 Maksimaalisen varianssin avaaminen

Sekä Isomap että CDA perustuvat perinteisiin lineaarisiin dimensioreduktiomenetelmiin, mutta graafipohjaisten etäisyyksien käyttö mahdollistaa myös epälineaarisen datan käsittelyn. Perinteiset menetelmät pyrkivät optimoimaan joko datajoukon varianssia (PCA) tai projektiojännitystä (MDS). Dimensioreduktio-ongelmaa voi lähestyä myös *isometria*-käsitteen avulla, kuten *maksimaalisen varianssin avaamisessa* (*Maximum variance unfolding*, MVU) tehdään. Menetelmä tunnettiin aiemmin myös nimellä *semidefiniittiupotus* (*Semidefinite embedding*, SDE) [71, 73, 70, 72]. Tuloksena on ydinpääkomponenttianalyysia teoreettisesti muistuttava menetelmä, joka kuitenkin tuottaa visuaalisesti parempia tuloksia.

MVU:n perusajatuksena oleva isometria-käsite tarkoittaa etäisyydet säilyttävää kuvausta metristen avaruuksien välillä. Esimerkiksi kaksiulotteisen paperin tapauksessa isometrinen kuvaus on mikä tahansa fysikaalinen muutos, joka paperille voidaan tehdä repimättä tai puhkomatta sitä. Isometrinen kuvaus voidaan tehdä pistekohtaisesti rotaation ja translaation avulla.

Yleisesti ottaen isometria on ominaisuus, jota monet dimensioreduktio-

menetelmät pyrkivät säilyttämään. Klassinen monidimensioskaalaus pyrkii säilyttämään kaikkien pisteiden väliset euklidiset etäisyydet, graafipohjaiset monidimensioskaalauksen variantit (Isomap) puolestaan approksimoidaan pisteiden geodeettisia etäisyyksiä. Koska täydellisen isometrian saavuttaminen on yleensä vaikeaa tai mahdotonta, antavat menetelmät suuremman painoarvon kunkin pisteen naapurustolle, jolloin kyse on lokaalista isometriasta.

Isometrian topologinen määritelmä on metristen avaruuksinen välinen, mutta MVU määrittää käyttämänsä pistejoukkojen välisen lokaalin isometrian seuraavasti:

**Määritelmä.** Kaksi pistejoukkoa $\mathbf{X}$ ja $\mathbf{Y}$ ovat *k-lokaalisti isometriset*, jos jokaiselle datapisteelle $\mathbf{x}_i$ ja sen $k$:lle lähimmälle naapurille $\{\mathbf{x}_{j1}, \ldots, \mathbf{x}_{jk}\}$ on olemassa rotaatio ja translaatio, joka kuvaa ne täsmälleen pisteille $\mathbf{y}_i$ ja $\{\mathbf{y}_{j1}, \ldots, \mathbf{y}_{jk}\}$

Algoritmi, jolla mainittu lokaali isometria saavututetaan, johdetaan asettamalla rajoitteita matriisin $\mathbf{Y}$ vektoreille. Ensimmäinen rajoite on, että $\mathbf{x}_i$:n ja sen $k$:n naapurien kulmien ja etäisyyksien täytyy säilyä kuvauksessa vektori $\mathbf{y}_i$:ksi. Tämä voidaan ilmaista ehdolla

$$(\mathbf{y}_i - \mathbf{y}_j) \cdot (\mathbf{y}_i - \mathbf{y}_k) = (\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_i - \mathbf{x}_k) \tag{29}$$

kun $\mathbf{x}_j$ ja $\mathbf{x}_k$ ovat $\mathbf{x}_i$:n naapureita. Tämä takaa paikallisen isometrian, koska kolmen pisteen välille muodustuvan kolmion määrittävät yksi kulma ja kahden sivun pituus. Koska kolmio on täysin määritelty myös kun kaikkien kolmen sivun pituudet tunnetaan, voidaan paikallinen isometria määritellä yksinkertaisesti

$$\|\mathbf{y}_i - \mathbf{y}_j\|^2 = \|\mathbf{x}_i - \mathbf{x}_j\|^2 \tag{30}$$

aina kun $\mathbf{x}_i$ ja $\mathbf{x}_j$ ovat naapureita. MVU käyttää naapuruston muodostamiseen naapurustograafia Isomapin ja CDA:n tapaan. Toisena rajoitteena projektiojoukko keskitetään origoon:

$$\sum_i \mathbf{y}_i = \mathbf{0} \tag{31}$$

Varsinaisessa projektiossa tavoitteena on säilyttää paikallinen isometria (30),

mutta samalla maksimoida pisteiden väliset etäisyydet. Käytännössä tämän voi ajatella tarkoittavan kokoon taittuneen pinnan avaamista siten, ettei pinta repeydy. Maksimoitava funktio on seuraava:

$$\Phi = \frac{1}{2n} \sum_{ij}^{n} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \tag{32}$$

Funktion 32 optimointi on hankalaa, koska funktio ei ole neliömuodon vuoksi konveksi. Optimointifunktio voidaan kuitenkin kirjoittaa yksinkertaisempaan muotoon käyttämällä vektoreiden välisiä pistetuloja $\mathbf{K}_{ij} = \mathbf{y}_i \cdot \mathbf{y}_j$. Pistetuloja käyttämällä ehto 30 voidaan kirjoittaa muotoon:

$$\mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{jj} = \mathbf{D}_{ij} \tag{33}$$

jossa $\mathbf{D}_{ij} = \|\mathbf{x}_i \mathbf{x}_j\|^2$ on neliöity etäisyysmatriisi.

Ehto 31 puolestaan saa muodon:

$$\mathbf{0} = \left\| \sum_i \mathbf{y}_i \right\|^2 = \sum_{ij} \mathbf{y}_i \cdot \mathbf{y}_j = \sum_{ij} \mathbf{K}_{ij} \tag{34}$$

Jotta matriisi $\mathbf{K}$ voitaisiin tulkita ydinmatriisiksi, täytyy sen olla symmetrinen ja positiivinen semidefiniittimatriisi [58]. Tämä voidaan ilmaista ehdolla:

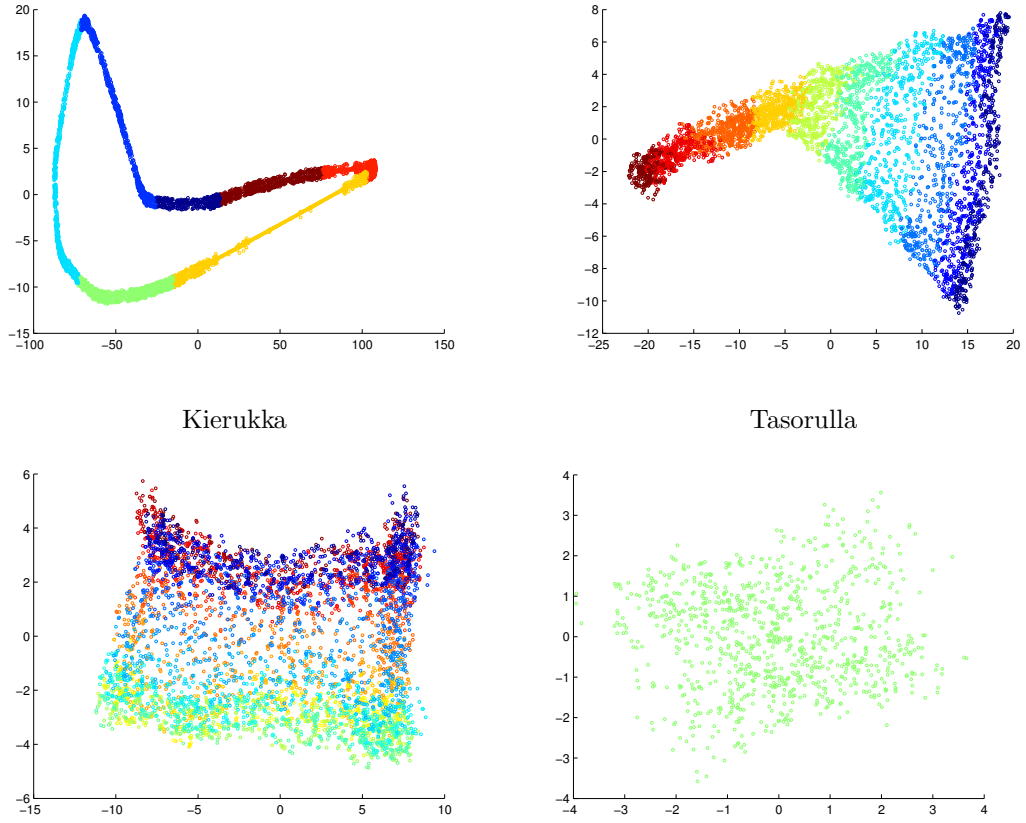
$$\mathbf{K} \succeq 0 \tag{35}$$

Optimoitava funktio 32 voidaan ilmaista ydinmatriisin avulla seuraavasti:

$$\Phi = \frac{1}{2n} \sum_{ij} (\|\mathbf{y}_i\|^2 + \|\mathbf{y_j}\|^2 + 2\mathbf{y}_i \cdot \mathbf{y}_j) = \sum_i \|\mathbf{y}_i\|^2 = \sum \mathbf{K}_{ii} = \mathrm{tr}(\mathbf{K}) \tag{36}$$

Maksimoitava funktio on siis ydinmatriisin jälki eli diagonaalineliöiden summa. Optimointiongelma voidaan kokonaisuudessaan ilmaista seuraavasti:

Maksimoi ydinmatriisin $\mathbf{K}$ jälki seuraavien rajoitteiden ollessa voimassa, kun $\eta_{ij} \in \{0, 1\}$ riippuen siitä ovatko $\mathbf{x}_i$ ja $\mathbf{x}_j$ naapureita:




Kuva 21: Maksimaalisen varianssin upotus naapuruston koolla $k = 12$

1. $\mathbf{K} \succeq 0$
2. $\sum_{ij} \mathbf{K}_{ij} = \mathbf{0}$
3. $\mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{ij} = \mathbf{D}_{ij}$ kaikille $i, j$ kun $\eta_{ij} = 1$

Tällainen optimointiongelma voidaan ratkaista *semidefiniittiohjelmointilla* (*semidefinite programming*, SDP) [66]. Semidefininiittiohjelmointi on lineaarisen ohjelmoinnin yleistys, jota voidaan käyttää erilaisten kombinatoristen ongelmien ratkaisuussa. SDP-ongelman ratkaiseminen on semidefiniittiupotuksen laskennallisesti vaativin osa. Sen laskennallinen vaativuus on $O(n^3(nk)^3)$ eli $O(n^6)$ [72].

Kun ydinmatriisi on optimoitu, saadaan datan projektio käyttämällä ydinmatriisin ominaisarvohajotelmaa:

$$\mathbf{Y} = \mathbf{I}\mathbf{\Lambda}^{1/2}\mathbf{U}^T \tag{37}$$

kun $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. Algoritmi 7 esittää maksimaalisen varianssin avaamisen kokonaisuudessaan.

**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio
$\sigma$ – optimoitavan naapuruston koko
$k$ graafin naapuruston koko

1. Laske datapisteiden väliset neliöidyt etäisyydet ja säilö ne matriisiin $\mathbf{D}$
2. Laske kunkin datapisteen $k$-lähintä naapuria
3. Luo $n \times n$-matriisi $\mathbf{K}$ semidefiniittiohjelmoinnin avulla, seuraavien rajotteiden mukaan:
   - $\mathbf{K} \succeq 0$
   - $\sum_{ij} \mathbf{K}_{ij} = \mathbf{0}$
   - $\mathbf{K}_{ii} - 2\mathbf{K}_{ij} + \mathbf{K}_{ij} = \mathbf{D}_{ij}$ kaikille $i, j$ kun $\eta_{ij} = 1$
4. Tee klassinen monidimensioskaalaus matriisille $\mathbf{K}$:
   - Laske $\mathbf{K}$:n ominaisarvohajotelma: $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
   - Laske datan projektio: $\mathbf{Y} = \mathbf{I}_{p\times n}\mathbf{\Lambda}^{(1/2)}\mathbf{U}^T$

**Algoritmi 7:** MVU-algoritmi

Maksimaalisen varianssin avaaminen voidaan nähdä eräänlaisena versiona ydinpääkomponenttianalyysista (KPCA), jota käsiteltiin luvussa 3.1.4. KPCA:ssa ongelmana on kullekin datajoukolle sopivan ytimen löytäminen. MVU:ssa ydinmatriisi optimoidaan kullekin datajoukolle sopivaksi [44].

Käytännössä MVU:n osana olevan semidefiniittiohjelmoinnin tekeminen on laskennallisesti liian vaativaa, jos dataa on paljon. Normaali MVU ei toimi 5000 datavektorin esimerkkidatan kanssa. Menetelmästä on kehitetty erilaisia variaatioita, joissa semidefiniittiohjelmointiongelman koko on pienempi. Seuraavassa on käytetty niin kutsuttua nopeaa MVU-algoritmia (*FastMVU*), jossa optimoitava matriisi on graafin vierusmatriisin ($n \times k$) sen Laplace-matriisia (ks. luku 3.3.2) [69]. Tällöin optimoivan matriisin koko on vain $d^2 \times d^2$.

Kuvassa 21 on esitetty nopean MVU:n tulokset mallidatajoukoille. Kuten muut-

kin graafietäisyyksiä käyttävät menetelmät, MVU soveltuu sellaisenaan vain yhtenäisille datajoukoille. Kuvan 21 klusteridatan upotus on siis kohdistunut vain datan suurimpaan yksittäiseen komponenttiin. Muuten MVU muistuttaa kyseisen datan kohdalla tuloksiltaan Isomap-algoritmin tuloksia.

## 3.3 Topologiset menetelmät

Edellä esitellyt menetelmät ovat pohjautuneet ajatukseen, että projektio pienempiuloitteiseen avaruuteen voidaan tehdä käyttämällä hyväksi tietoa pisteiden keskinäisistä etäisyyksistä, ja optimoimalla projektiossa näitä etäisyyksiä jollakin tavalla. Projektion pienempään ulottuvuuteen voi tehdä myös pyrkimällä säilyttämään datajoukon topologia etäisyyksien sijaan [44]. Edellä esillä ollut SDE otti jo lähtökohdakseen paikallisen isometrian säilyttämisen, mutta varsinainen optimointi tehtiin etäisyyksien perusteella.

Seuraavassa esitellään kaksi menetelmää, *paikallisesti lineaarinen upotus* (*Locally Linear Embedding*, LLE) ja *Laplace-ominaiskuvaus* (*Laplacean Eigenmaps*, LE), joiden perusajatuksena on datan topologian hyödyntäminen projektioiden tekemisessä.

### 3.3.1 Paikallisesti lineaarinen upotus

Paikallisesti lineaarisen upotuksen (LLE) [54, 56] teoreettisena lähtökohtana on muodostaa datalle niin sanottu *konformikuvaus* alkuperäisestä data-avaruudesta projektioavaruuteen. Yleisesti ottaen konformikuvaukset ovat kuvauksia, jotka säilyttävät paikallisesti pisteiden väliset kulmat [44]. LLE pyrkii tähän tavoitteeseen kolmen askeleen avulla. Ensin jokaiselle datapisteelle valitaan $k$-lähintä naapuria ($k$-naapuruston sijaan voidaan käyttää myös pisteen $\epsilon$-ympäristön pisteitä). Tämän jälkeen kukin datapiste esitetään naapureidensa lineaarikombinaationa. Pisteiden ja niiden lineaarikombinaatioiden välinen rekonstruktiovirhe voidaan esittää seuraavasti:

$$E(\mathbf{W}) = \sum_{i=1}^{n} \left\| \mathbf{x}_i - \sum_{j \in N_i} w_{i,j} \mathbf{x}_j \right\|^2 \tag{38}$$

jossa $N_i$ on pisteen $\mathbf{x}_i$ naapuri-indeksien joukko. Virhefunktion E argumenttina on matriisi $\mathbf{W}$, joka sisältää kunkin pisteen esityksessä käytetyn lineaarikombinaation komponenttien painokertoimet. Kun virhefunktio minimoidaan, minimoituu samalla rekonstruktiovirhe. Koska kunkin pisteen painokertoimia ($w_{i,j}$) käytetään vain kunkin yksittäisen pisteen $\mathbf{x}_i$ rekonstruktioon, täytyy minimointiin liittää seuraavat ehdot:

1. Vain pisteen naapurusto vaikuttaa sen rekonstruktiopainossa: $w_{i,j} = 0 \; \forall j \notin N_i$
2. Kunkin pisteen rekonstruktiopainojen summa on yksi: $\sum_{j=1}^{n} w_{i,j} = 1$

Minimoidun matriisin $\mathbf{W}$ painoilla on ominaisuus, että ne ovat kunkin datapisteen $\mathbf{x}_i$ kohdalla invariantteja rotaatioiden, skaalausten ja translaatioiden suhteen [54, 44]. Ne siis mallintavat kunkin pisteen topologista ympäristöä käyttämättä suoraan etäisyysmittaa, toisin kuin aikaisemmin esitetyt etäisyyspohjaiset algoritmit.[8] Projektiossa kohdeavaruuteen käytetään hyväksi tätä tosiasiaa: LLE olettaa, että rekonstruktiopainojen avulla tehtävä projektio pitää voimassa dataavaruuden topologiset suhteet.

Projektio tehdään käyttämällä alkuperäisten pisteiden rekonstruktiopainoja $w_{i,j}$ ja minimoimalla seuraavaa funktiota:

$$\Phi(\mathbf{Y}) = \sum_{i=1}^{n} \left\| \mathbf{y}_i - \sum_{j \in N_i} w_{i,j} \mathbf{y}_j \right\|^2 \tag{39}$$

Aikaisemmin lasketut painot $w_{i,j}$ pysyvät vakioina funktiota 39 minimoitaessa, vain projisoitavien pisteiden $\mathbf{y}_i$ sijainteja varioidaan.

Sekä funktion 38 että 39 minimointi voidaan tehdä suljetussa muodossa seuraavasti [56]. Rekonstruktiopainojen osalta kunkin yksittäisen pisteen rekonstruktiovirhe on funktion 38 perusteella:

$$\epsilon_i = \|\mathbf{x}_i - \sum_j w_j \mathbf{x}_j\|^2 = \|\sum_j (\mathbf{x}_i - \mathbf{x}_j)\|^2 = \sum_{jk} w_j w_k G_{jk} \tag{40}$$

[8] Koska pistetulot yhdistävät pisteiden väliset etäisyydet ja kulmat, on mahdollista että LLE voidaan formuloida myös etäisyyspohjaisena algoritmina.

jossa $G_{jk}$ on datapisteen *paikallisen* Grammin matriisin alkio:

$$G_{jk} = (\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_i - \mathbf{x}_k) \tag{41}$$

Jotta matriisista $\mathbf{G}_i$ ei tulisi singulaarista, voidaan se säännöllistää lisäämällä siihen yksikkömatriisin murto-osa:

$$\mathbf{G}_i \leftarrow \mathbf{G} + \frac{\Delta^2 \mathrm{tr}(\mathbf{G}_i)}{k}\mathbf{I}, \tag{42}$$

missä $\Delta$ on jokin reaaliluku, joka on pieni suhteessa matriisin $\mathbf{G}_i$ jälkeen $\mathrm{tr}(\mathbf{G}_i)$. Optimaalinen paino $w_j$ saadaan kääntämällä matriisi $\mathbf{G}$:

$$w_j = \frac{\sum_k \mathbf{G}^{-1}_{jk}}{\sum_{lm} \mathbf{G}^{-1}_{lm}} \tag{43}$$

Käytännössä raskasta matriisin kääntöoperaatiota ei tarvitse tehdä, vaan riittää ratkaista lineaarinen yhtälöryhmä $\sum_k \mathbf{G}_{jk} w_k = 1$ ja skaalata painot siten, että niiden summaksi tulee 1 [56]. Rekonstruktiopainojen laskemisen aikavaativuus kullekin datapisteelle on $O(dnk^3)$ ($k$ on kunkin pisteen naapureiden lukumäärä).

Projektio kohdeavaruuteen voidaan laskea seuraavasti. Muutetaan ensin funktio 39 neliömuotoon:

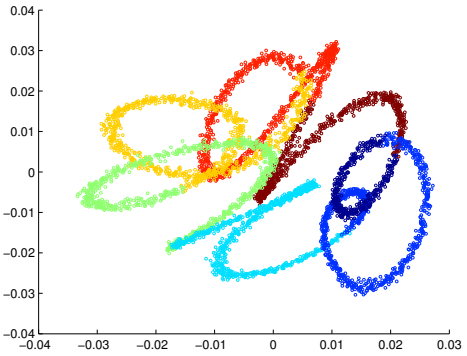
$$\Phi(\mathbf{Y}) = \sum_{ij} m_{ij}(\mathbf{y}_i \cdot \mathbf{y}_j) \tag{44}$$

jossa tekijät $m_{ij}$ ovat $n \times n$ kokoisen matriisin elementtejä, jotka saadaan rekonstruktiopainomatriisista ($W$) seuraavasti:

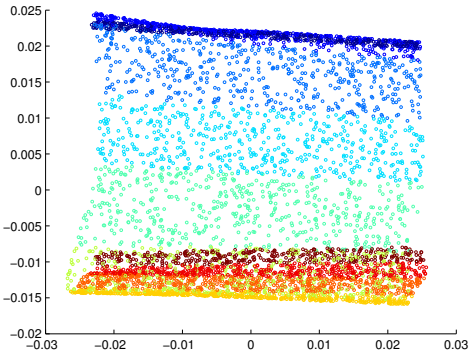
$$\mathbf{M} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W}) \tag{45}$$

Asetetaan projektiomatriisille $\mathbf{Y}$ kaksi rajoitetta:

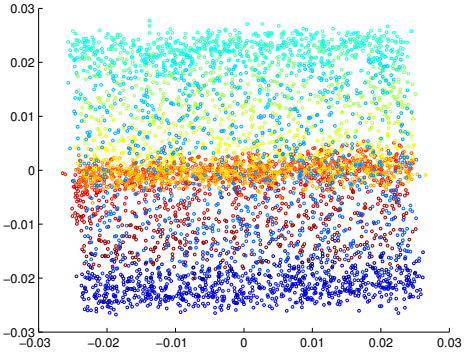
1. Matriisin täytyy olla keskitetty origoon: $\sum_i \mathbf{y}_i = \mathbf{0}$
2. Matriisin kovarianssimatriisin täytyy olla yksikkömatriisi: $\frac{1}{n}\sum_i \mathbf{y}_i \mathbf{y}_j^T = \mathbf{I}$



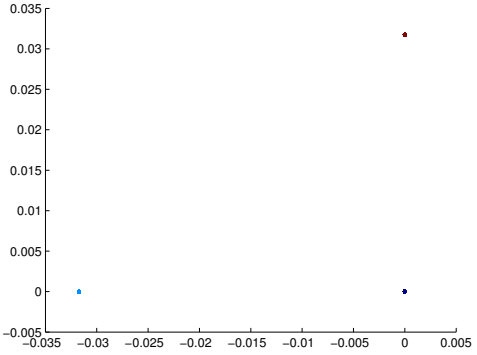

Kierukka




Tasorulla




Itsensä leikkaava silmukka




Klusterit

Kuva 22: Lokaali lineaarinen upotus

Rajoitteet poistavat ratkaisulta translaation ja rotaation vapausasteet. Tällöin funktion 39 minimointi voidaan tehdä laskemalla $p+1$ ominaisarvoltaan pienintä matriisin $\mathbf{M}$ ominaisvektoria [56, 44]. Näistä viimeisin, ominaisarvoltaan pienin, ominaisvektori hylätään ehdon 1 voimassapitämiseksi. Varsinainen projektio lasketaan samalla tavoin kuin pääkomponenttianalyysin tapauksessa, kertomalla alkuperäinen datamatriisi $\mathbf{X}$ lasketuista ominaisvektoreista muodostetulla matriisilla. Algoritmi 8 esittää paikallisesti lineaarisen upotuksen vaiheet kokonaisuudessaan.

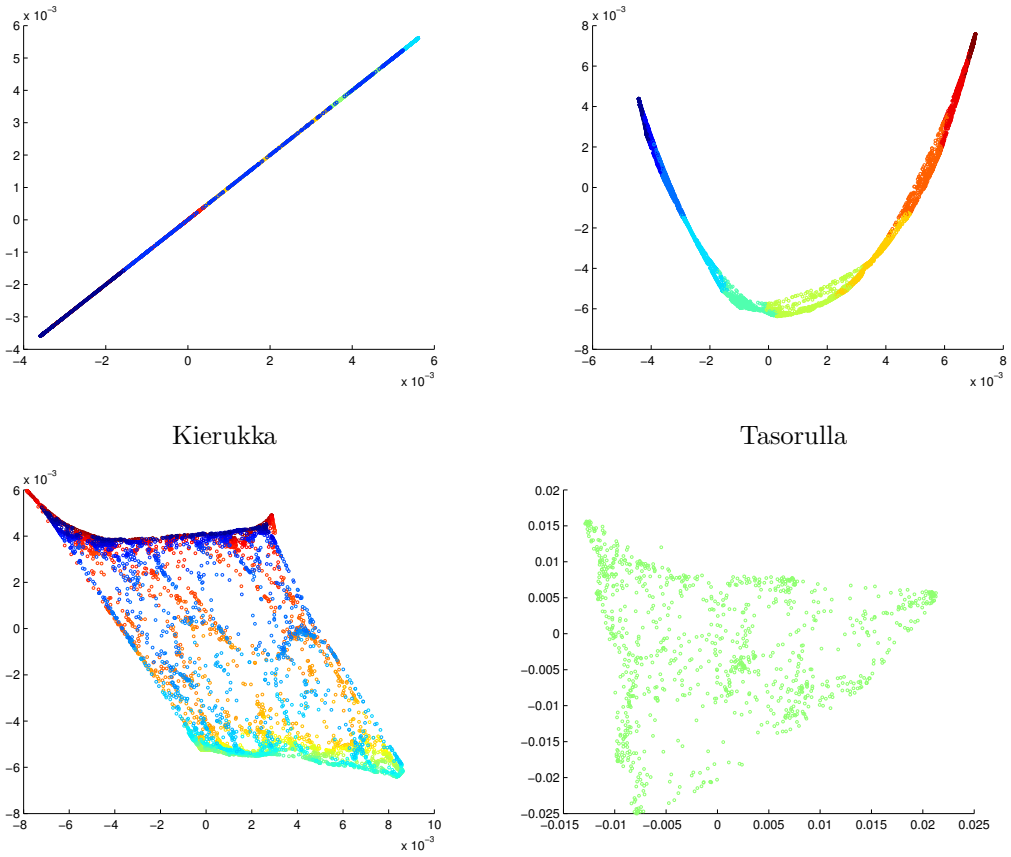
**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio
$\sigma$ – optimoitavan naapuruston koko
$k$ tai $\epsilon$ datapisteen naapuruston koko

1. Laske kullekin datapisteelle $\mathbf{x}_i$
   - Sen $k$ lähintä naapuria
   - Matriisi $\mathbf{G}_i$ yhtälöiden 41 ja 42 mukaan
   - Optimaaliset painot $w_{ik}$ yhtälön 43 mukaan
2. Laadi painojen $w_{ik}$ avulla matriisi $\mathbf{M}$ (yhtälö 45)
3. Laske ominaisarvohajotelma matriisille $\mathbf{M}$
4. Tee projektiomatriisi $\mathbf{P}$ hylkäämällä ominaisarvoltaan pienin matriisin $\mathbf{M}$ ominaisvektoreista ja ottamalla matriisin $\mathbf{P}$ sarakkeiksi seuraavat $p+1$ ominaisvektoria ominaisarvojärjestyksessä
5. Laske projektio $\mathbf{Y} = \mathbf{XP}$

**Algoritmi 8:** LLE-algoritmi

LLE:n laskennallisesti vaativin osuus on projektiovaiheen ominaisvektorien laskenta, joka vaatii ajan $O(pn^2)$ [56]. Matriisi $\mathbf{M}$ on kuitenkin tyypillisesti harva, mikä käytännössä nopeuttaa laskentaa.

Kuvassa 22 on esitelty LLE:n tuloksia esimerkkidatajoukoille. Pisteiden naapuruussuhteet säilyvät yleisesti ottaen hyvin, mutta pintojen muodot eivät; pinnat voivat olla alttiita venymiselle tai supistumiselle. Huomattavaa on, että vaikka LLE hyödyntää naapurustograafia, toimii se myös epäyhtenäiselle datalle. Klus-




Kuva 23: Laplace-ominaiskuvaus

teridatajoukon kukin klusteri on romahtanut yhdeksi pisteeksi, tosin klusterit ovat myös menneet osittain päällekäin.

Lokaali lineaarinen upotus on hyvä esimerkki dimensionaalisuuden vähentämisalgoritmista, jossa aikaisemmissa algoritmeissa esitetyt ajatukset (pääkomponenttianalyysin ominaisarvohajotelma, Isomapin graafirakenne) on yhdistetty uudella tavalla, tässä käyttämällä matemaattista konformikuvauksen käsitettä. Tällaisten menetelmien hyvänä puolena on se, että niillä on hyvä matemaattinen perusta. Haittapuolena puolestaan se, että algoritmit ovat vaikeita ymmärtää ja myös toteuttaa.

### 3.3.2 Laplace-ominaiskuvaus

Laplace-ominaiskuvaus (*Laplacian eigenmaps*, LE) on toinen datan topologisiin ominaisuuksiin perustuva dimensioreduktiomenetelmä [6, 7]. LE muistuttaa pal-

jon LLE:tä, mutta siinä missä LLE:n perusajatuksena on säilyttää kunkin pisteen ympäristö topologisesti samanlaisena sekä alkuperäisessä että projektioavaruudessa, on LE:n peruslähtökohta graafiteoreettinen [44].

LE:n lähtöhypoteesi on, että data $\mathbf{X}$ sijaitsee tarpeeksi sileällä hyperpinnalla, että se voidaan kuvata graafina $G = (V_N, E)$, joka säilyttää riittävällä tarkkuudella alkuperäisen datan ominaisuudet. Data pyritään projisoimaan pienempiuloitteeseen avaruuteen siten, että graafi $G$ säilyttää muotonsa myös projektiossa. Projektiossa graafin mallintamat pisteiden naapuruussuhteet siis säilyvät. Tämä voidaan ilmaista seuraavalla virhefunktiolla:

$$E = \frac{1}{2} \sum_{i,j=1}^{n} \|\mathbf{y}_i - \mathbf{y}_j\|^2 w_{i,j} \tag{46}$$

jossa termit $w_{i,j}$ ovat painomatriisin $\mathbf{W}$ alkioita. Painomatriisi $\mathbf{W}$ voidaan määritellä eri tavoin. Yksinkertaisin tapa on käyttää alkuperäisen datan vierusinformaatiota: paino $w_{i,j} = 1$ jos $\mathbf{x}_i$ ja $\mathbf{x}_j$ ovat naapureita. Menetelmän kehittäjät kuitenkin suosittelevat seuraavanlaisen gaussilaisen painomatriisin käyttöä [6]:

$$w_{i,j} = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2t^2}} \tag{47}$$

missä $t$ on parametri, jolla pisteiden välisten etäisyyksien painoa voidaan säätää. Tälläinen painomatriisi antaa kunkin pisteen lähiympäristössä oleville pisteille suuremman painoarvo kuin kaukaisille pisteille virhefunktion 46 optimoinnissa.

Otetaan seuraavaksi käyttöön matriisi $\mathbf{L}$, joka on graafin $G$ Laplace-matriisi[9], joka voidaan liittää painomatriisiin seuraavasti:

$$\mathbf{L} = \mathbf{W} - \mathbf{D} \tag{48}$$

jossa $\mathbf{D}$ on diagonaalimatriisi, jonka alkiot ovat painojen summia: $d_{ii} = \sum_{j=1}^{n} w_{ij}$. Matriisia $\mathbf{L}$ käyttäen virhefunktio 46 voidaan kirjoittaa muotoon [44]:

[9]Graafin Laplace-matriisi määritellään seuraavasti: Matriisin $\mathbf{L}$ alkio $l_{ij} =$
$$\begin{cases} \deg(v_i) & \text{jos } i = j \\ -1 & \text{jos } i \neq j \text{ ja } v_i \text{ on } v_j\text{:n naapuri} \\ 0 & \text{muutoin} \end{cases}$$

$$E = \mathrm{tr}(\mathbf{YLY}^T) \tag{49}$$

Funktion 49 minimointi on sama asia kuin graafin $G$ Laplace-matriisin ominaisarvoyhtälön $\lambda\mathbf{Df} = \mathbf{Lf}$ ratkaiseminen. Kun yhtälön ratkaisuun liitetään ehto $\mathbf{YDY}^T = \mathbf{I}$ poistuu yhtälöstä skaalaustekijä [6]. Toinen, ekvivalentti, tapa löytää projektio on normalisoida matriisi $\mathbf{L}$:

$$\mathbf{L} \leftarrow \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \frac{l_{ij}}{\sqrt{d_{ii}d_{jj}}} \tag{50}$$

Ja laskea normalisoidun matriisin $\mathbf{L}$ ominaisvektorit [44]. Ominaisarvoyhtälön pienin ratkaisu on yleensä 0, mikä vastaa kaikkien pisteiden romauttamista yhdeksi pisteeksi. Ei-triviaali projektio pienempään ulottuvuuteen saadaan samalla tavoin kuin LLE:ssä; valitsemalla projektion kerroinmatriisiin $p$-kappaletta ominaisarvotaan nollasta poikkeavaa ominaisvektoria ja kertomalla alkuperäinen datamatriisi kerroinmatriisilla. LE:n kaikki vaiheet on esitetty algoritmissa 9.

LE on laskennallisesti yhtä vaativa kuin LLE. Molemmissa menetelmissä muodostetaan $O(n^2)$ kokoinen naapurusto/painomatriisi ja tämän matriisin johdannaismatriisille tehdään ominaisarvohajotelma. Samoin kuten LLE:ssä, hajotettava matriisi on tyypillisesti harva, mikä nopeuttaa algoritmia käytännössä.

Kuvassa 23 on esitetty Laplacen ominaiskuvauksen tulokset esimerkkidatalle. Kuvista voidaan havaita LE:n taipumus romauttaa data. Sekä kierukka- että tasorulladata muuttuvat miltei yksiulotteisiksi, eivätkä pisteiden naapuruussuhteet kierukkadatassa edes säily. Klusteridatan kohdalla menetelmä toimii vain datan yhden yhtenäisen komponentin, yhden klusterin, kohdalla. LE- ja LLE-algoritmeista on myös kehitetty variaatio *Hessin paikallisesti lineaarinen upotus* (*Hessian locally linear embedding*, HLLE) [22], jossa LLE:ssä käytetty neliömuoto 39 korvataan Hessen matriisilla[10]. Hessen matriisi käyttö perustellaan puolestaan samalla tavoin kuin LE-algoritmissa.

[10]Hessen matriisi on funktion $f$ toisen kertaluvun derivaattojen matriisi.

**Syöte:** $\mathbf{X}$ – $n \times d$ -matriisi, jonka kukin rivi on yksi $d$-uloitteinen datavektori
$p$ – projektion dimensio
$\sigma$ – optimoitavan naapuruston koko
$k$ tai $\epsilon$ datapisteen naapuruston koko

1. Määritä kunkin datapisteen $\mathbf{x}_i$ $k$-lähintä naapuria (tai $\epsilon$-ympäristö)
2. Tee naapurustograafi $G$
3. Laske painomatriisi $\mathbf{W}$ yhtälön 47 avulla
4. Muodosta diagonaalimatriisi $\mathbf{D}$ laskemalla painomatriisin $\mathbf{W}$ rivisummat
5. Laske matriisi $\mathbf{L} = \mathbf{W} - \mathbf{D}$
6. Normalisoi matriisi $\mathbf{L}$: $\mathbf{L} \leftarrow \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$
7. Laske ominaisarvohajotelma matriisille $\mathbf{L}$: $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
8. Tee projektiomatriisi $\mathbf{P}$, kertomalla ominaisvektorit $\mathbf{D}^{1/2}$:lla, hylkäämällä ominaisarvoltaan pienin ominaisvektoreista (joka on nollavektori) ja ottamalla matriisin $\mathbf{P}$ sarakkeiksi seuraavat $p + 1$ ominaisvektoria ominaisarvojärjestyksessä
9. Laske projektio $\mathbf{Y} = \mathbf{X}\mathbf{P}$

**Algoritmi 9:** LE-algoritmi

## 3.4 Satunnaisprojektio

Edellä esitellyt menetelmät ovat pyrkineet hyödyntämään datapisteiden etäisyys- tai topologiainformaatiota. Aidosti korkeauloitteisen datan tapauksessa kaikki edellä esitellyt menetelmät voivat olla laskennallisesti liian raskaita. Toisaalta dimensionaalisuuden kirouksen vuoksi edellä esitellyt menetelmät eivät välttämättä toimi lainkaan. Seuraavassa esitellään vielä yksi menetelmä, joka toimii myös korkeaulotteiselle datalle. Haittapuolena on kuitenkin menetelmään liittyvä satunnaisuus, joka rajoittaa sen mahdollisia sovelluskohteita.

*Satunnaisprojektio* (*random projection*, RP) on kaikkia aikaisemmin esiteltyjä menetelmiä yksinkertaisempi. Satunnaisprojektio ei tarkoita datan täysin satunnaista uudelleensijoittelua kohdeavaruuteen vaan projektion suuntien satunnaista valitsemista. Menetelmän toimivuus perustuu matemaattiselle tosiseikalle, joka kantaa Johnson-Lindenstraussin lemman nimeä [38].

Teoreeman mukaan korkeauloitteisella datalla on dimensionaalisuuden kirouksen lisäksi data-analyysissä hyödyllisiä ominaisuuksia. Teoreeman tai lemman mukaan mikä tahansa $d$-ulotteisessa euklidisessa avaruudessa sijaitseva $n$-alkioinen datajoukko voidaan kuvata toiseen euklidiseen avaruuteen, jonka dimensio on $k = O(\epsilon^{-2} \log n)$ siten, etteivät pisteiden väliset etäisyydet muutu enempää kuin tekijän $1 \pm \epsilon$ $(0 < \epsilon < 1)$ verran [16]. Alkuperäinen lemma on seuraava:

Olkoon $\epsilon$ reaaliluku $0 < \epsilon < 1$ ja $n$ mikä tahansa kokonaisluku, ja $k$ positiivinen kokonaisluku, jolle pätee:

$$k \geq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n \tag{51}$$

Tällöin mille tahansa joukolle pistejoukolle $X$, jossa on $n$ avaruuden $\mathbb{R}^d$ pistettä, on olemassa kuvaus $f : \mathbb{R}^d \leftarrow \mathbb{R}^k$ siten että kaikille $u, v \in V$ pätee

$$(1 - \epsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon)\|u - v\|^2 \tag{52}$$

Tällainen kuvaus voidaan löytää polynomiaalisessa ajassa [38].

Koska edellä esitetyn Johnson-Lindenstraussin lemman mukaan yllä esitetyt ehdot täyttävät satunnaisprojektiot säilyttävät todennäköisesti datajoukon pistei-

den suhteelliset etäisyydet, voi satunnainenkin projektio paljastaa tai säilyttää esimerkiksi datassa olevia klustereita [16, 27].

Käytännössä satunnaisprojektio tehdään seuraavasti: esitetään $n$ kappaletta $d$-uloitteisia datapisteitä matriisin $\mathbf{X}$ riveinä. Jos data halutaan projisoida $k$-uloitteeseen avaruuteen, muodostetaan projektiomatriisi $\mathbf{P}$ esimerkiksi joillakin seuraavista tavoista:

1. Matriisin alkiot valitaan satunnaisesti normaalijakaumasta ([16])
2. Matriisin alkio $p_{ij} = \begin{cases} 1 & \text{todennäköisyydellä } 1/2 \\ -1 & .. \quad 1/2 \end{cases}$ ([2])
3. Matriisin alkio $p_{ij} = \sqrt{3} \times \begin{cases} 1 & \text{todennäköisyydellä } 1/6 \\ 0 & .. \quad 2/3 \\ -1 & .. \quad 1/6 \end{cases}$ ([2])

Datan projektio saadaan kertomalla alkuperäinen data projektiomatriisilla ja skaalaamalla se kohdedimension suhteen: $\mathbf{Y} = \frac{1}{\sqrt{k}}\mathbf{XP}$

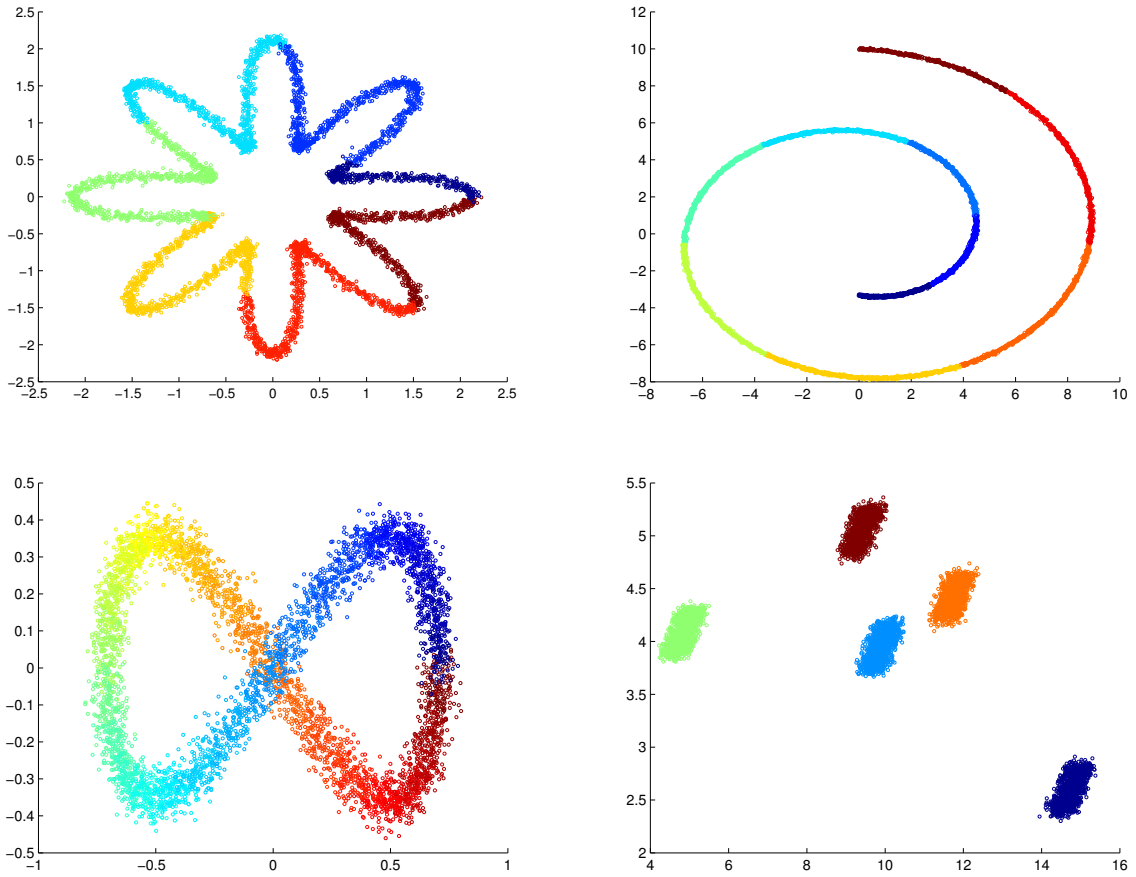
Ensimmäinen menetelmä tuottaa etäisyydet mielivaltaisen hyvin säilyttävän projektion vähintään todennäköisyydellä $\frac{1}{n}$, joten hyvän projektion tuottamiseen tarvitaan $O(n)$ projektiota. Hyvä projektio tarkoittaa täsmällisesti sellaista projektiota, joka säilyttää alkuperäisten ja projisoitujen pisteiden väliset etäisyydet tekijän $1 \pm \epsilon$ tarkkuudella. Tällöin kohdedimension $k$ ja virhetermi $\epsilon$:n vallitsee seuraava suhde [16]:

$$k \geq 4 * (\frac{\epsilon^2}{2} - \frac{\epsilon^3}{3})^{-1} \log(n)$$

Kaksi jälkimmäistä tapaa ovat laskennallisesti kevyempiä, koska projektiomatriisi on harva ja satunnaisuutta vaaditaan vähemmän. Menetelmät tuottavat seuraavan riippuvuuden kohdedimension ja parametrien välille: [2]

$$k \geq \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log(n)$$

Projektion tarkkuutta säätelevän parametrin $\epsilon$ lisäksi yhtälössä on hyvän projektion todennäköisyyttä säätävä parametri $\beta$. Hyvän projektion todennäköisyys on



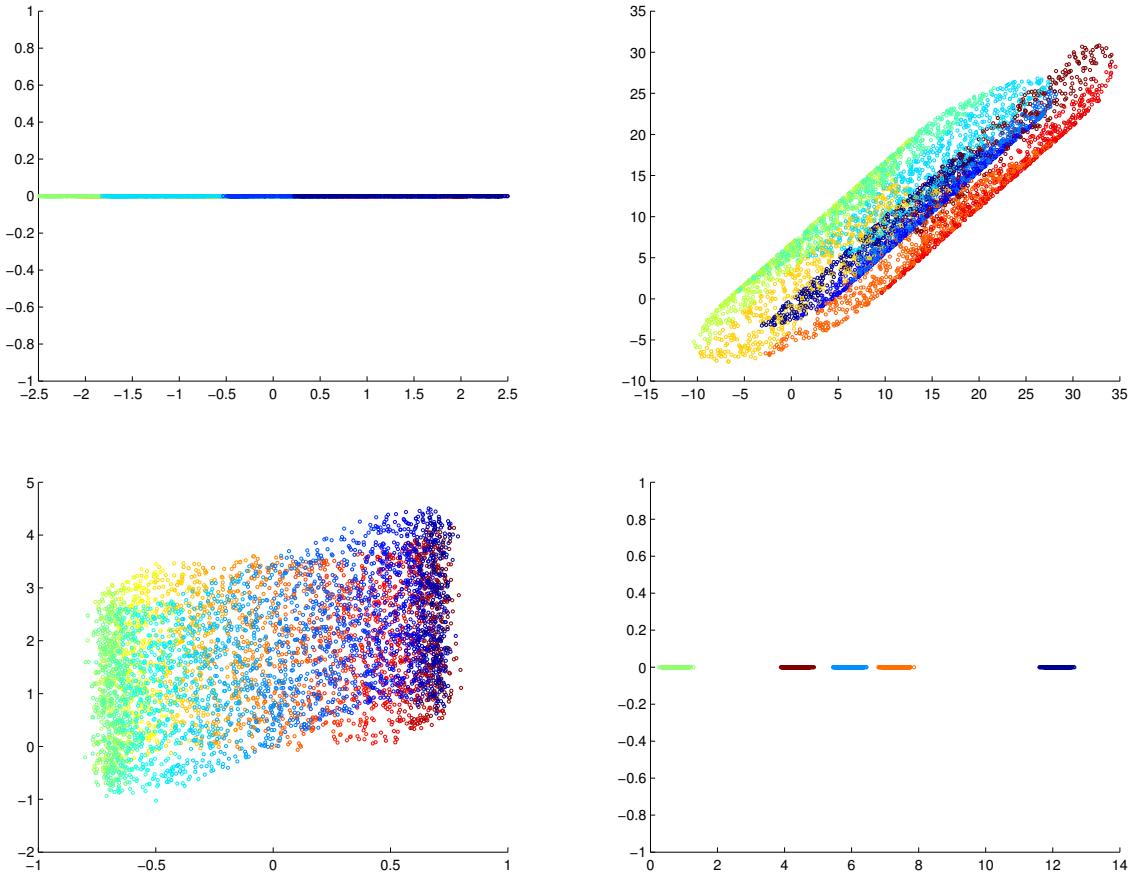
Kuva 24: Hyviä satunnaisprojektiota esimerkkidatalle

vähintään $1 - n^{-\beta}$.

Kuvassa 24 on esitetty onnistuneita satunnaisprojektioita esimerkkidatajoukoille (projektiomatriisin elementteinä $\pm 1$). Muihin dimensionreduktiomenetelmiin nähden satunnaisprojektion etuna on sen pieni aikavaativuus. Projektiomatriisin laatiminen vie ajan $O(pq)$ ja varsinainen projektio ajan $O(npq)$. Mutta kuten kuva 25 osoittaa satunnaisprojektion laatu ei ole aina hyvä. Satunnaisprojektion käytöstä korkeadimensionaalisen datan yhteydessä on kuitenkin saatu joitakin lupaavia tuloksia [10, 29, 3].

## 3.5 Yhteenveto

Edellä on esitelty joukko erilaisia dimensionaalisuuden vähentämismenetelmiä. Taulukkoon 1 on koottu yhteen tiedot menetelmien säädettävien parametrien




Kuva 25: Huonoja satunnaisprojektiota esimerkkidatalle

h

Taulukko 1: Dimensioreduktiomenetelmien aika- ja tilavaativuuksia. $n$ pisteiden lukumäärä, $d$ lähtödimensio, $p$ kohdedimensio, $z$ optimointifunktion iteraatioiden määrä

| **Menetelmä** | **Parametrit** | **Aikavaativuus** | **Tilavaativuus** |
|---|---|---|---|
| PCA | - | $O(d^3)$ | $O(d^2)$ |
| MDS | - | $O(zn^3)$ | $O(n^2)$ |
| CCA | oppimisvakio, naapurusto | $O(n^2p)$ | $O(n^2)$ |
| KPCA | ydin | $O(n^3)$ | $O(n^2)$ |
| Isomap | naapurusto | $O(n^2k)$ | $O(n^2)$ |
| (Fast)MVU | naapurusto | $O(n^3 + d^{12})$ | $O(n^2)$ |
| LLE | naapurusto | $O(pn^2)$ | $O(n^2)$ |
| LE | naapurusto, painomatriisi | $O(pn^2)$ | $O(n^2)$ |
| Satunnaisprojektio | - | $O(ndp)$ | $O(dp)$ |

lukumäärä, aika- ja tilavaativuudet. Menetelmien tilavaativuus on yleensä aina luokkaa $O(n^2)$, paitsi pääkomponenttianalyysin ja satunnaisprojektion tapauksissa, joiden tilavaativuus on pienempi. Myös aikavaativuudeltaan menetelmät eivät poikkea suuresti toisistaan. Karkeasti ottaen menetelmien aikavaativuus on luokkaa $O(n^3)$, parhaimmillaan $O(n^2)$, mikäli rajoitutaan tarkastelemaan vain datapisteiden lukumäärää $n$. Käytännössä algoritmien ajoajat poikkeavat toisistaan enemmän kuin pelkät aikavaativuusluokat kertovat.

Suurimmassa osassa menetelmiä joudutaan käyttämään datasta riippuvaisia kontrolliparametreja, kuten naapuruston kokoa tai ydinpääkomponenttianalyysin tapauksessa käytettävää ydintä. Parametrien valinta vaikuttaa tuloksiin suuresti. Etenkin ydinpääkomponenttianalyysin tapauksessa ytimen valinta on kriittinen. Monessa menetelmässä joudutaan määrittämään naapuruston koko. Tämän parametrin vaikutus ei näytä olevan yhtä kriittinen tulosten laadun suhteen, mutta suurempi naapuruston koko hidastaa algoritmien ajoaikoja.

Menetelmiä havainnollistettiin käyttämällä keinotekoista kolmiulotteista dataa, joka projisoitiin kahteen ulottuvuuteen. Menetelmien hyvyyttä ja huonoutta arvioitiin silmämääräisesti katsomalla säilyikö kolmiulotteisen pinnan muoto projektiossa ja miten pisteiden väliset naapuruussuhteet pysyivät ennallaan. Menetelmien objektiiviseksi arvioimiseksi ei ole olemassa mitään yksittäistä mittaa,

joka paljastaisi yksiselitteisesti mikä menetelmä on paras. Tämä johtuu osittain siitä, että kunkin menetelmän objektiivinen funktio, jota minimoidaan tai maksimoidaan on yleensä aina datan geometriasta tai jostakin muusta ominaisuudesta johdettu heuristinen funktio.

Pääkomponenttianalyysi esimerkiksi minimoi projektion keskineliövirhettä (MSE), mutta keskineliövirheen minimoiminen ei mahdollista epälineaaristen pintojen projisoimista pienempään ulottuvuuteen siten, että pinnan muoto säilyy.

Venna ja kumppanit ovat esittäneet erityisiä *luotettavuus* (*trustworthiness*) ja *jatkuvuus* (*continuity*) -mittoja, joilla voidaan arvioida menetelmien toimivuutta visualisoinnin yhteydessä [67]. Luotettavuus ja jatkuvuus mittaavaat kahdenlaisia virheitä projektiossa. Toisaalta sellainen piste, joka ei ole toisen pisteen naapuri alkuperäisessä avaruudessa, voi kuvautua tämän pisteen naapuriksi projektioavaruudessa. Luotettavuus mittaa tällaisia virheitä. Jatkuvuus puolestaan mittaa virheitä, jotka syntyvät, kun alkuperäisen avaruuden naapurit eivät ole enää naapureita projektioavaruudessa.

Tarkkaan ottaen luotettavuus määritellään seuraavana summana:

$$M_{trust}(k) = 1 - A(k) \sum_{i}^{n} \sum_{j \in U_k(i)} r(i,j) - k \tag{53}$$

$U_k(i)$ on niiden pisteiden joukko, jotka kuuluvat pisteen $\mathbf{x}_i$ $k$-kokoiseen naapurustoon projektioavaruudessa, mutteivat alkuperäisessä avaruudessa. $r(i,j)$ kertoo monenneksiko lähimpänä piste $\mathbf{x}_j$ on pistettä $\mathbf{x}_i$ alkuperäisessä avaruudessa. $A(k)$ on tekijä, joka skaalaa luotettavuusarvon yhden ja nollan välille:

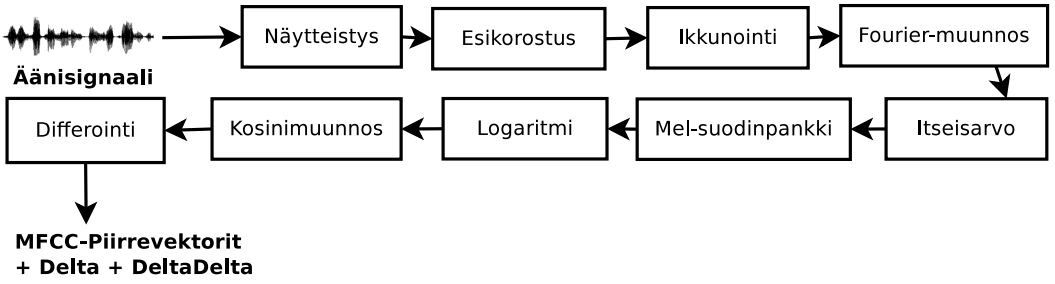
$$A(k) = \begin{cases} \frac{2}{nk(2n-3k-1)} & \text{jos } k < \frac{n}{2} \\ \frac{2}{n(n-k)(n-k-1)} & \text{jos } k \geq \frac{n}{2} \end{cases} \tag{54}$$

Vastaavasti jatkuvuus määritellään seuraavasti:

$$M_{cont}(k) = 1 - A(k) \sum_{i}^{n} \sum_{j \in V_k(i)} \hat{r}(i,j) - k \tag{55}$$

jossa $V_k(i)$ on niiden pisteiden joukko, jotka kuuluvat pisteen $\mathbf{x}_i$ $k$-kokoiseen naa-

purustoon alkuperäisessä avaruudessa, mutta eivät projektioavaruudessa. $\hat{r}(i,j)$ kertoo puolestaan monenneksiko lähimpänä piste $\mathbf{x}_j$ on pistettä $\mathbf{x}_i$ projektioavaruudessa. Skaalaustekijä $A(k)$ määritellään kuten yhtälössä 54. Luotettavuus- ja jatkuvuus-mittoja ei ole vielä ainakaan toistaiseksi käytetty yleisesti menetelmien laadun arvioinnissa.




Kuva 26: MFCC-piirrevektoreiden irroittaminen äänisignaalista.

# 4 Äännedatan dimensionaalisuuden vähentäminen ja luokittelu

Edellisessä luvussa esiteltiin joukko ulotteisuuden vähentämismenetelmiä ja havannoillistettiin niiden ominaisuuksia projisoimalla kolmeulotteista dataa kahteen ulottuvuuteen. Tällaiset esimerkit havainnollistavat hyvin menetelmien tyypillisiä piirteitä, mutta eivät välttämättä anna riittävää kuvaa menetelmien toimivuudesta todellisen datan tapauksessa. Seuraavassa testataan sitä, miten edellä kuvatut menetelmät soveltuvat todelliselle äännedatalle.

Testidatana käytetään puhedatasta eristettyjä äänteitä. Miesääni on lukenut häiriöttömissä olosuhteissa mikrofoniin suomenkielisiä lauseita. Puheesta on muodostettu joukko MFCC-vektoreita (*Mel-frequency Cepstral Coefficient*) [17].

MFCC-vektorit ovat puheen- ja puhujantunnistuksessa yleisesti käytettyjä piirrevektoreita, jotka pyrkivät mallintamaan äänen akustisia piireitä. MFCC-vektorit muodostetaan tekemällä äänisignaalille ensin Fourier-muunnos, käyttämällä mel-suodinpankkeja ja ottamalla viimein suodinpankkien signaalista kosinimuunnokset. MFCC-vektorien muodostus on esitetty tarkemmin kuvassa 26.

MFCC-vektoreiden yhteydessä käytetetään yleensä myös ns. delta- ja deltadelta-piirteitä, jotka muodostetaan kahden (delta) tai kolmen (deltadelta) peräkkäisen piirrevektorin erotuksena ja liitetään osaksi lopullista piirrevektoria. Delta- ja deltadelta-piirteet pyrkivät mallintamaan puheessa tapahtuvia muutoksia, mutta koska tässä tutkimuksessa pyritään tutkimaan yksittäisiä äänteitä, on delta-piirteet jätetty pois. Käytettävät MFCC-vektorit ovat 15-ulotteisia. Taulukossa 2 on esitetty tarkemmin MFCC-piirteiden luomisessa käytetyt parametrit.

Taulukko 2: MFCC-piirteiden luomisessa käytetyt parametrit

| **Parametri** | **Arvo** |
|---|---|
| Ikkunan koko | 20ms |
| Ikkunan siirros | 10ms |
| Fourier-muunnoksen koko | 1024 |
| Mel-filttereiden määrä | 27 |
| Ulotteisuus | 15 |

Taulukko 3: Datajoukkojen ominaisuudet

| **Datajoukko** | **Ulotteisuus** | **Vektorien lukumäärä** | **Luokkien lukumäärä** |
|---|---|---|---|
| Vokaalit | 15 | 4423 | 8 |
| Frikatiivit | 15 | 1437 | 3 |

Yksittäiset äänteet on eroteltu puheesta käyttämällä dataan liittyvää annotointia. Kukin yksittäinen äänne on hieman eripituinen ja koostuu eri määrästä yksittäisiä MFCC-vektoreita, esitellyt dimensionaalisuuden analysointi- ja redusointimenetelmät eivät sovellu suoraan saatavalle datalle. Tämän vuoksi kunkin äänteen muodostavista MFCC-vektoreista on otettu keskiarvo, joka toimii äänteen edustajana.

Käytössä on kaksi erilaista datajoukkoa. Toinen datajoukko koostuu suomen kielen kahdeksasta erilaisesta vokaalista ja siinä on 4423 datavektoria. Toinen datajoukko koostuu frikatiiviäänteistä /s/, /h/ ja /f/ ja siinä on 1437 datavektoria. Taulukossa 3 on yhteenveto käytettävien datajoukkojen ominaisuuksista. Taulukossa 4 on puolestaan esitetty kummankin joukon eri luokkien koot. Molemmissa joukoissa on yksi selvästi pienempi luokka (/ö/ ja /f/). Frikatiiveissa suurimman /s/-luokan koko on lähes kaksi kertaa niin suuri kuin luokan /h/, luokat ovat kooltaan muutoin samaa suuruusluokkaa.

Seuraavissa luvuissa testataan ensin dimensionaalisuuden arvioimismenetelmiä kahdelle esitetylle datajoukolle. Sen jälkeen tutkitaan, miten hyvin esitellyt dimensionaalisuuden vähentämismenetelmät sopivat datajoukkojen visualisoimiseen. Viimein testataan dimensionaalisuuden vähentämisen vaikutusta luokittelutarkkuuteen.

Taulukko 4: Eri äänteiden lukumäärä aineistoissa

| **Vokaali** | **Lukumäärä** |
|---|---|
| a | 687 |
| e | 674 |
| i | 677 |
| o | 647 |
| u | 615 |
| y | 401 |
| ä | 599 |
| ö | 123 |
| Yhteensä | 4423 |

| **Äänne** | **Lukumäärä** |
|---|---|
| s | 910 |
| h | 503 |
| f | 24 |
| Yhteensä | 1437 |

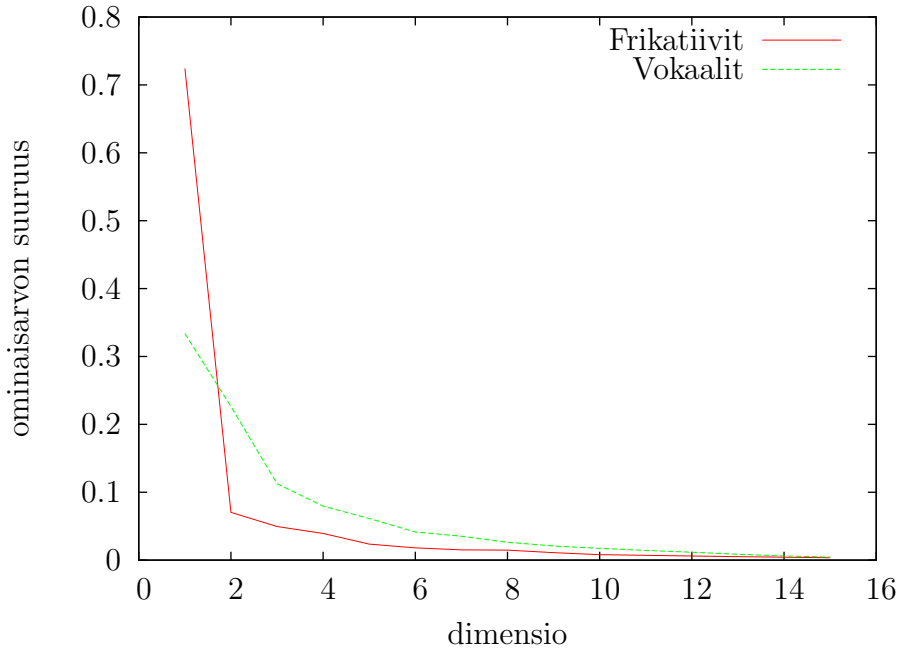
## 4.1 Dimensionaalisuuden arvioiminen

MFCC-vektoreiden luontaista dimensionaalisuutta on pyritty selvittämään aikaisemmin [59]. Tällöin käytettiin TIMIT-puhekorpuksesta [1] muodostettuja 12-ulotteisia MFCC-vektoreita, joiden dimensionaalisuutta analysoitiin käyttäen korrelaatiodimensiota sekä MDS- (Sammonin kuvaus) ja CCA-algoritmeja. Lisäksi pyrittiin löytämään sellainen dimensio, jossa MFCC-vektorit sopisivat parhaiten SOM-hilaan. MFCC-vektoreita ei oltu jaoteltu äänteiden mukaan.

Korrelaatiodimensioksi MFCC-vektoreille saatiin MFCC-parametreista riippuen tulos lukujen 5.0 - 5.7 väliltä. MDS- ja CCA-algoritmeja käytettiin puolestaan siten, että data projisoitiin pienempiin dimensiohin ja laskettiin rekonstruktiovirhe. Rekonstruktiovirheen ja dimensionaalisuuden suhteesta pyrittiin löytämään sellainen kohta, jossa virhe kasvaa merkittävästi ja tämän avulla yritettiin määrittää datan luontainen dimensio. Rekonstruktiovirhe kasvoi kuitenkin tasaisesti dimensionaalisuuden pienentyessä, eikä selkeää taitekohtaa löytynyt [59].

Äännedatan dimensionaalisuutta arvioidaan seuraavassa luvussa 2 esiteltyillä menetelmillä. Koska yksittäisten äänteiden pituudella ei dimensionaalisuuden arvioinnissa ole merkitystä, käytettiin pelkkiä MFCC-vektoreita, eikä kullekin äänteelle muodostettuja keskiarvoja, kuten myöhemmin äänteiden visualisoinnissa. Vokaalidatassa vektoreita oli 114774, frikatiividatassa 29048.

Kuvassa 27 on esitetty datajoukoista muodostetun kovarianssimatriisin norma-




Kuva 27: Äännedatan ominaisarvot

lisoitujen ominaisarvojen kuvaukset. Ominaisarvokuvausten perusteella voidaan sanoa, että merkittävä osa varianssista säilyy vokaaleilla jo 6-ulotteisessa projektiossa, frikatiiveilla puolestaan 5-ulotteisessa projektiossa. Jos ominaisarvon kynnysarvoksi laitetaan 0.025, saadaan datajoukkojen luontaisiksi dimensionaalisuuksiksi vokaaleilla 8 ja frikatiiveilla 9.

Dimensionaalisuuden arviointi lähimpien naapureiden menetelmällä ei jostain syystä tuota tulosta äännedatalle, vaan dimensionaalisuuden arvioksi molemmille joukoille tulee 0. Sen sijaan korrelaatiodimensio tuottaa vokaaleille arvon 6.4 ja frikatiiveille arvon 6.7. Fraktaalidimensioiden tapauksissa luotettava dimensionaalisuusarvio saadaan, kun dataa on käytettävissä $10^{15/2} = 10^{7.5} \approx 31000000$. Vähäisemmästä datan määrästä huolimatta korrelaatiodimension antama arvo on lähellä aiemmassa työssä esitettyä arviota [59] (5.0-5.7). Arvio sopii myös hyvin yhteen ominaisarvojen antamaan dimensionaalisuusarvioon (6-9).

Myös maksimaalisen todennäköisyyden (MLE) ja geodeettisten minimaalisten virittävien puiden (GMST) antamat dimensionaalisuusarviot laskettiin. Ne vaihtelivat 9.9 ja 12.4 välillä. Taulukkoon 5 on koottu eri menetelmien antamat dimensionaalisuusarviot.

Yhteenvetona voidaan sanoa, että ominaisarvojen ja korrelaatiodimension an-

Taulukko 5: Datajoukkojen luontaisen ulottuvuuden arvioita

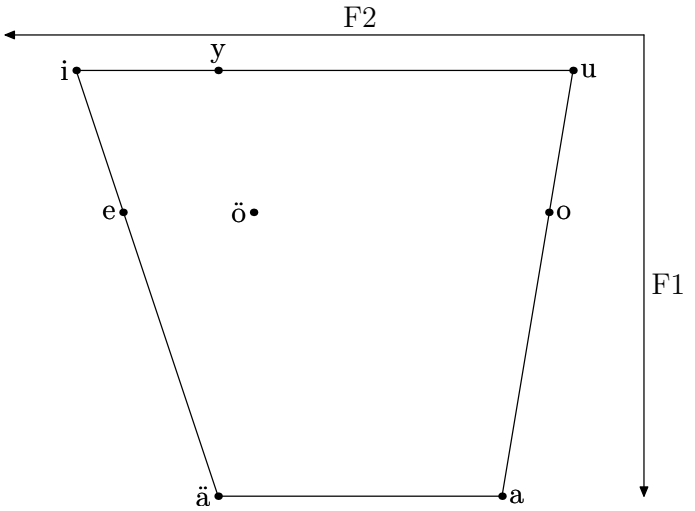
| **Menetelmä** | **Vokaalit** | **Frikatiivit** |
|---|---|---|
| Ominaisarvot (PCA) | 8 | 9 |
| Korrelaatiodimensio | 6.4 | 6.7 |
| MLE | 9.9 | 10.7 |
| GMST | 12.4 | 11.7 |

tamien arvojen perusteella testattavien vokaali- ja frikatiividatajoukkojen luontainen dimensionaalisuus sijoittuu todennäköisesti välille 6-9. Luotettavamman dimensionaalisuuden arvion saamiseksi voitaisiin tutkia datajoukkoja, joissa on enemmän dataa. Lisäksi olisi aiheellista tutkia useita erilaisia äänijoukkoja ja sitä miten MFCC-vektorien irrotuksessa käytettävät parametrit vaikuttavat niiden luontaiseen dimensionaalisuuteen.

## 4.2 Puhedatan kaksiulotteinen visualisointi

Dimensionaalisuuden vähentämismenetelmiä ei ole juurikaan käytetty puhedatan ominaisuuksien tutkimiseen. Muutamia aikaisempia tuloksia kuitenkin on. Samassa julkaisussa kuin käyrälineaarinen komponenttianalyysi (CCA) esiteltiin, testattiin myös lyhyesti sen käyttöä vokaaleiden kaksiuloitteiseen visualisointiin. Käytettävänä syötedatana oli MFCC-vektoreiden sijaan kaistasuodatettu äänisignaali yhdistettynä videokuvasta saatuun mittaustietoon suun muodosta ääntämishetkellä [20]. CCA:n verrokkimenetelminä testattiin samassa yhteydessä pääkomponenttianalyysia ja monidimensioskaalausta Sammonin jännitefunktiolla. Kaikki kolme menetelmää onnistuivat erottelemaan vokaalit omiksi klustereikseen kaksiuloitteisessa projektiossa, eikä menetelmien välillä ollut suurta eroa.

Toisessa, tuoreemmassa tutkimuksessa testattiin TIMIT-korpuksen vokaalien (englannin vokaaliäänteet /aa/, /uw/, /iy/) projisoimista kaksiulotteiselle tasolle käyttäen pääkomponenttianalyysia, Isomap-, LLE- ja LE-algoritmeja [26]. Toisin kuin tässä, projisoitavina vektoreina ei käytetty MFCC-vektoreita, vaan esikäsittelyn jälkeen kunkin äänteen äänisignaalille tehtiin Fourier-muunnos, joka muutettiin log-magnitudispektriksi.



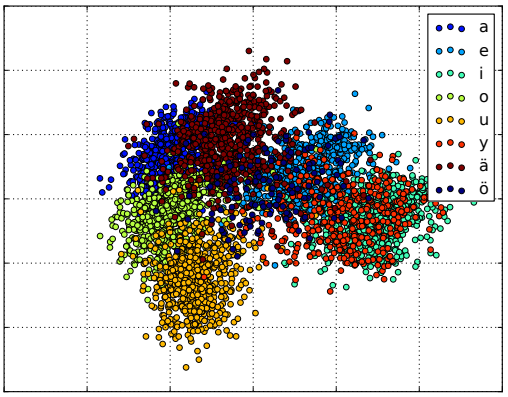

Kuva 28: Formanttien mukaan laadittu vokaalikartta

Vokaalit projisoituivat päällekkäin, mutta olivat kuitenkin selkeästi omina ryhminään. Menetelmiä arvioitiin numeerisesti laskemalla projisoitujen vokaaliryhmien väliset *Bhattacharyya*-etäisyydet. Tällä tavoin arvioituna PCA toimi algoritmeista huonoiten, LLE puolestaan parhaiten [26].

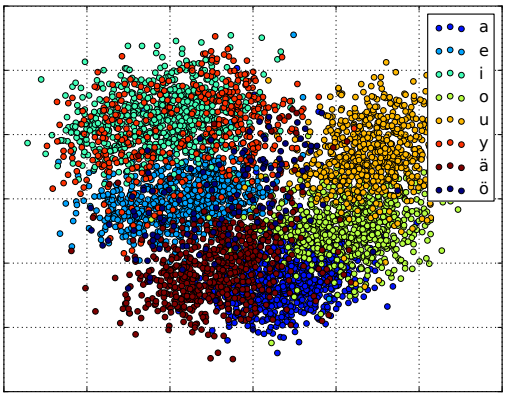
Seuraavassa tutkitaan miten dimensionaalisuuden vähentämismenetelmät soveltuvat MFCC-vektoreiden kaksiulotteisten projektioiden tekemiseen. MFCC-vektoreiden käytön etuna on se, että niitä käytetään yleisesti erilaisissa puheen- ja puhujantunnistuksen sovelluksissa. Mikäli dimensionaalisuuden vähentämismenetelmiä halutaan käyttää todellisissa sovelluksissa, on helpompaa käyttää syötteenä samaa dataa kuin sovelluksien mahdollisissa muissa osissa. Toisaalta MFCC-vektoreiden soveltuvuutta äänteiden ja puheen visualisointiin ei ole aikaisemmin lainkaan tutkittu.

### 4.2.1 Vokaalien projisointi

Foneetikot ovat perinteisesti asettaneet vokaalit kaksiulotteiselle vokaalikartalle, joka perustuu osittain puheessa olevien formanttien eli vahvistuneiden osavärähtelyalueiden arvoihin tai suun ja äänne-elinten muotoon äänteitä tuotettaessa [60]. Kuvassa 28 on esitetty suomen kielen vokaalikartta, joka perustuu kahteen ensimmäiseen formanttiin F1 ja F2. Vokaalikartta toimii verrokkina dimensionaalisuuden vähentämismenetelmien tuottamille vokaaliprojektioille.




PCA



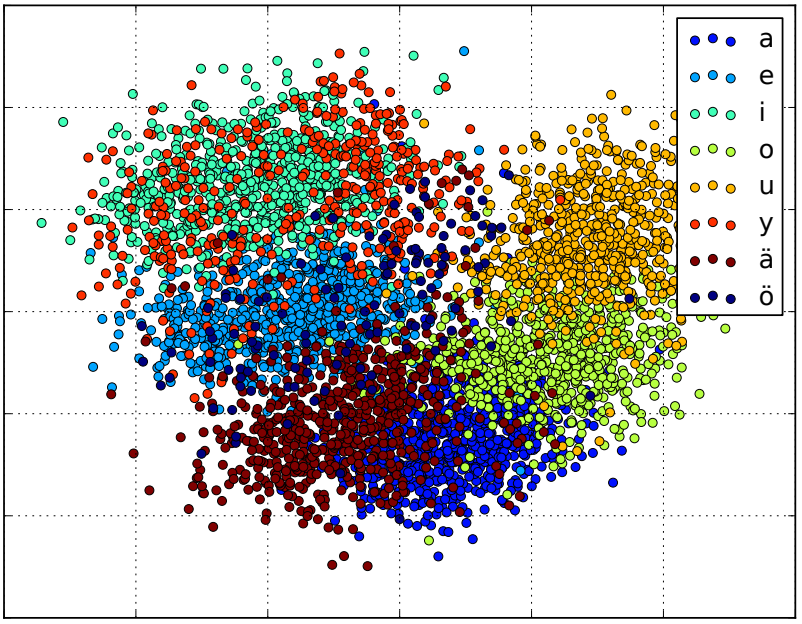

MDS

Kuva 29: Pääkomponenttianalyysi ja monidimensioskaalaus (Sammonin kuvaus) vokaaleille

Kuvassa 29 on esitetty pääkomponenttianalyysin ja monidimensioskaalauksen (jännitefunktiona Sammonin kuvaus) tulokset vokaalidatalle. Tulokset osoittavat sen, että myös lineaariset menetelmät onnistuvat jossain määrin projoisoimaan vokaalidatan kaksiulotteiseksi. Suurin osa vokaaleista projisoituu omaksi ryhmäkseen. Ainoastaan vokaalit /i/ ja /y/ projisoituvat selvästi päällekkäisiksi, mutta ne ovat lähellä toisiaan sekä ääntämyksellisesti että perinteisellä vokaalikartalla [60]. Vokaali /ö/ ei sen sijaan projisoidu selkeästi omaksi ryhmäkseen, vaan jää epämääräisesti projektion keskelle muiden ryhmien päälle. Tosin tämäkin vastaa /ö/:n asemaa vokaalikartalla. Pääkomponenttianalyysi ja monidimensioskaalaus tuottavat muuten melko samanlaiset projektiot, mutta pääkomponenttianalyysin vokaaliklusterit ovat tiiviimpiä.

Ehkä parhaan visualisoinnin vokaaleille tuottaa käyrälineaarinen komponenttianalyysi (kuva 30). Kuvassa kukin vokaali on projisoitunut omalle alueelleen, lukuunottamatta vokaalia /ö/, joka on jälleen muiden vokaalien keskellä. Mielenkiintoisesti vokaali /y/ projisoituu kahteen eri ryhmään ja muutama /u/ vokaali jää kauaksi omasta ryhmästään. Tämän tuloksen valossa olisi mielenkiintoista kokeilla, miten CCA:n graafietäisyyksiin pohjautuva variantti CDA [45] suoriutuisi vokaaliaineistosta.

Isomap-algoritmin antama projektio vokaaleille ei poikkea merkittävästi edellisten algoritmien tuloksista. Vokaalit /i/ ja /y/ ovat jälleen päällekkäisiä ja vokaali

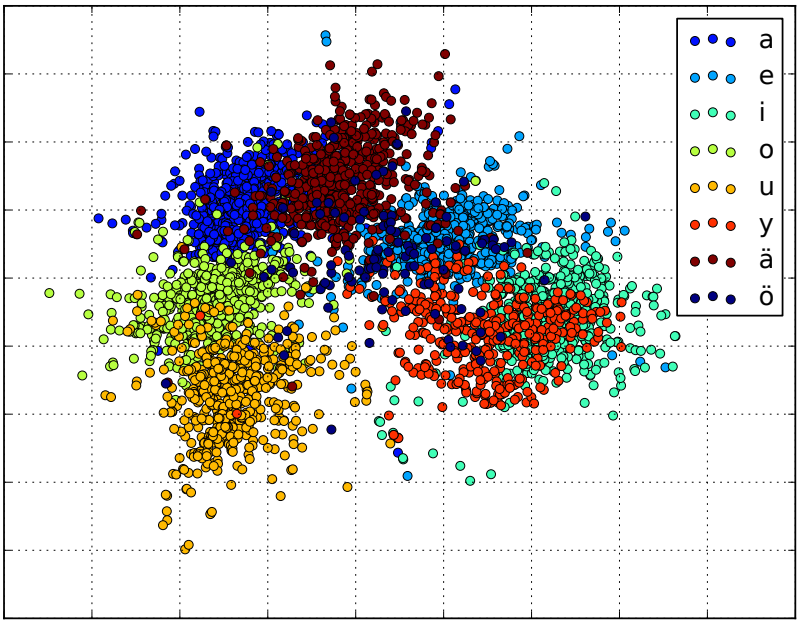



Kuva 30: Käyrälineaarinen komponenttianalyysi (CCA) vokaaleille

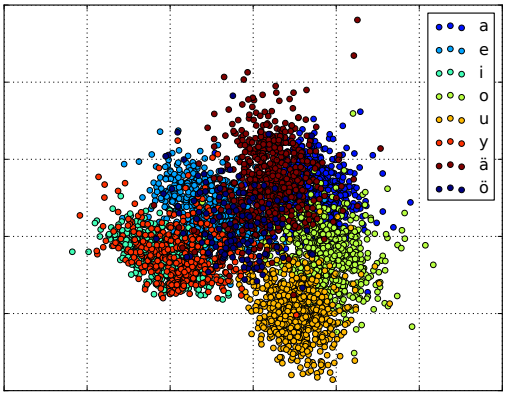
/ö/ on muiden vokaaleiden keskellä. Menetelmän heikkoutena on se, että vokaalit eivät muodostaneet yhtenäistä graafia naapuruston koolla $k = 12$, vaan projektiossa on 4192 vokaalia kaikkiaan 4423:sta vokaalista. Naapuruston koon kasvattaminen ei lisännyt merkittävästi projektioon saatavien vokaalien määrää, mutta algoritmin ajoaika sen sijaan kasvoi merkittävästi.

Kuvassa 32 on esitetty pääkomponenttianalyysin ydinvarianttien tuottamat projektiot vokaaleille. Polynomisen ytimen käyttäminen tuottaa paljon pääkomponenttianalyysia muistuttavan projektion. Sen sijaan gaussisen ytimen käyttäminen antaa tulokseksi erilaisen projektion. Projektiossa vokaalit muodostavat omat ryhmänsä, mutta osa ryhmistä on erittäin tiiviitä (/y/, /o/), osa puolestaan harvoja (/a/,/o/,/u/).

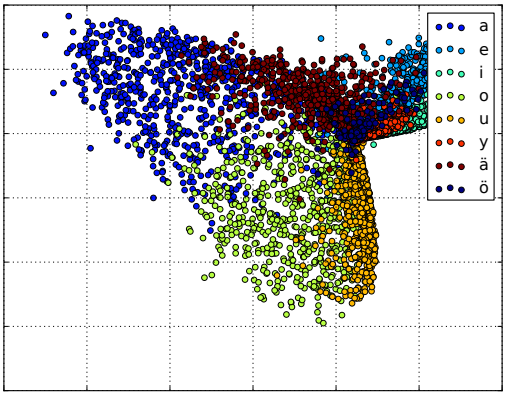
MVU-algoritmi eräänä etuna sanottiin olevan se, että se pystyy automaattisesti löytämään hyvän ytimen, jota projektiossa käytetään. Kuvan 33 perusteella MVU ei kuitenkaan näytä toimivan kovinkaan hyvin vokaalidatan tapauksessa. Tähän asti esitetyistä menetelmistä MVU:n projektion vokaaliryhmät sekoittuvat eniten




Kuva 31: Isomap-algoritmi vokaaleille

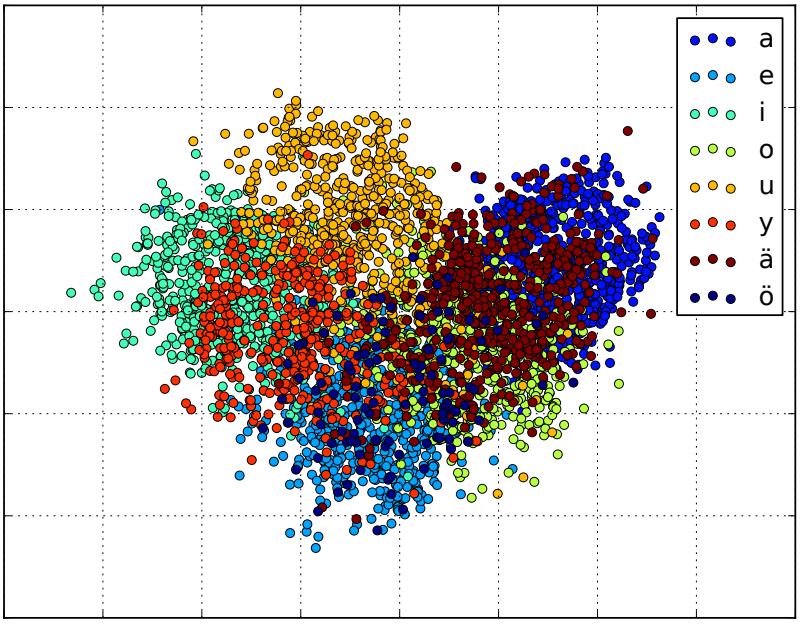



Polynominen ydin




Gaussinen ydin

Kuva 32: Ydinpääkomponenttianalyysi vokaaleille



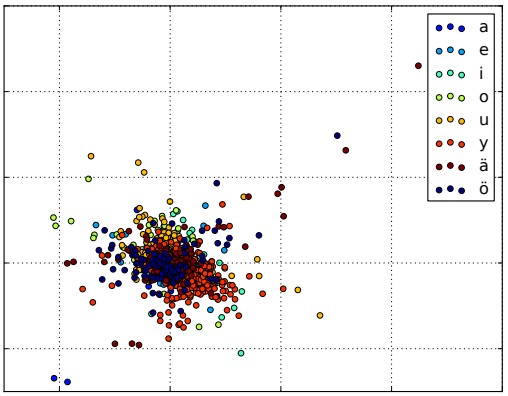

Kuva 33: MVU-algoritmi vokaaleille

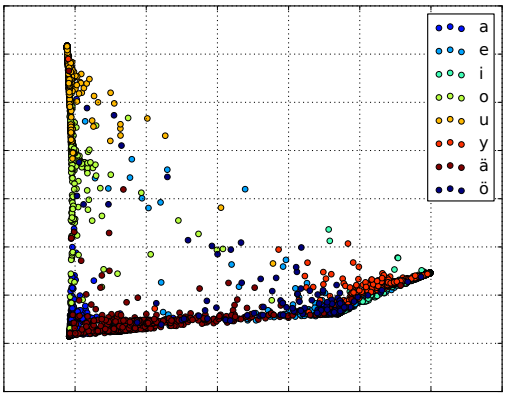
keskenään. Erityisesti projektion keskiosassa vokaaliryhmät /a/, /o/, /u/, /ä/ ja /ö/ ovat päällekäin. Samoin kuin Isomap, MVU:n käyttämä graafi on yhtenäinen vain 4192 vokaalin osalta.

Topologiset menetelmät eivät sovellu vokaalidatan visualisoimiseen. Paikallisesti lineaarinen upotus romauttaa vokaaliryhmät yhdeksi ryhmäksi. Laplace-ominaiskuvaus puolestaan muodostaa l-kirjaimen muotoisen kuvion, johon vokaalit osittain muodostavat omat ryhmänsä (kuva 34). Laplace-ominaiskuvauksessa yhtenäisen graafin koko oli Isomapin ja MVU:n tavoin 4192 vokaalia.

Kuvassa 35 on viimein annettu kaksi satunnaisprojektiota vokaaleille. Yleisesti ottaen satunnaisprojektiot eivät ole aivan yhtä hyviä kuin esimerkiksi pääkomponenttianalyysin antama projektio, mutta osoittavat vokaalien tapauksessa, että osa vokaaliryhmistä on eroteltavissa myös satunnaisprojektion antamassa visualisoinnissa.

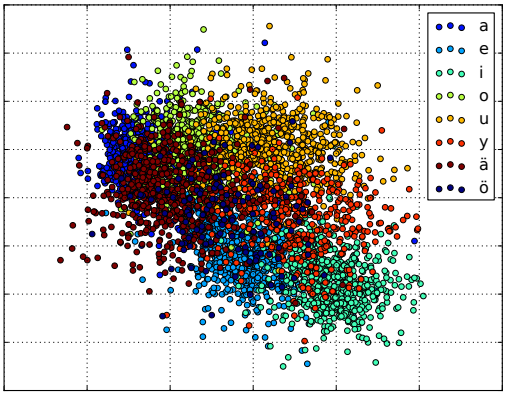
Yleisesti ottaen vokaaleille tehdyt projektiot osoittavat sen, että dimensionaalisuuden vähentämismenetelmiä voidaan soveltaa myös vokaalien visualisoimiseen

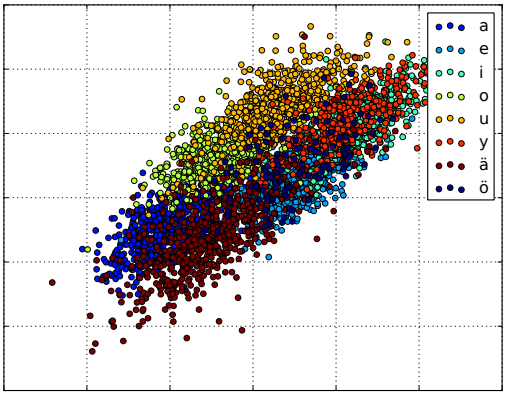


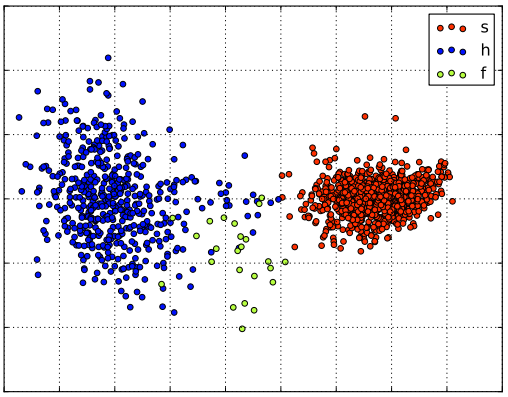

LLE LE

Kuva 34: Paikallisesti lineaarinen upotus (LLE) ja Laplace-ominaiskuvaus (LE) vokaaleille

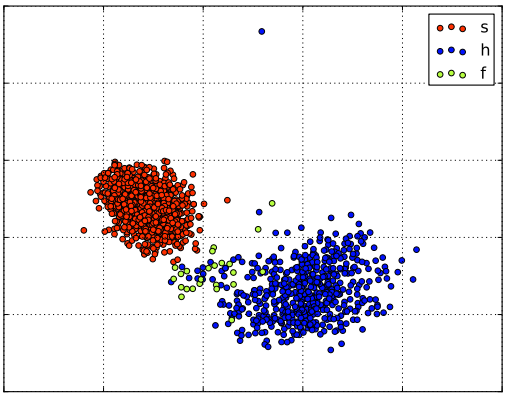



Kuva 35: Satunnaisprojektioita vokaaleille




PCA




MDS

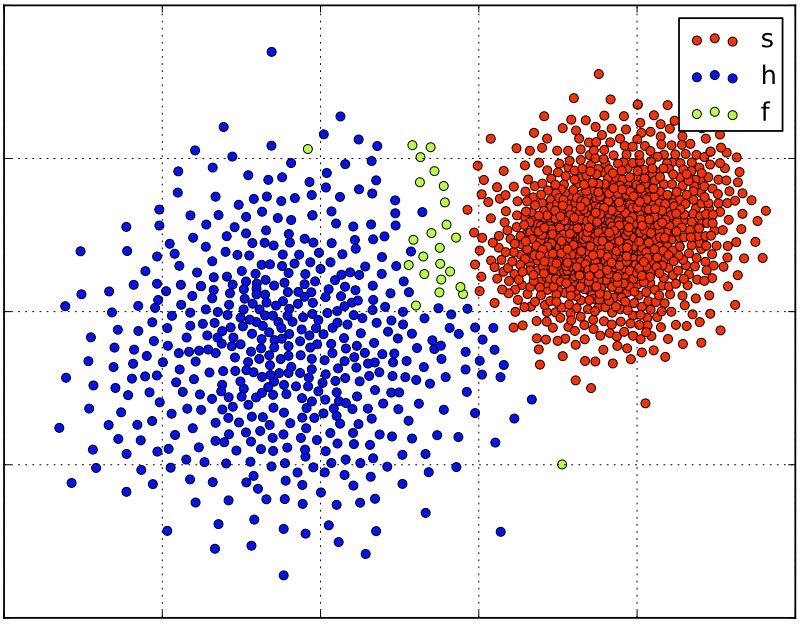
Kuva 36: Pääkomponenttianalyysi ja monidimensioskaalaus frikatiiviäänteille

MFCC-vektoreiden avulla. Niin sanotut perinteiset menetelmät (PCA, MDS ja CCA) näyttävät tässä tapauksessa antavan visuaalisesti hieman parempia tuloksia kuin uudemmat, monimutkaisemmat menetelmät.

### 4.2.2 Frikaatiiviäänteiden projisointi

Vokaalien lisäksi dimensionaalisuuden vähentämismenetelmiä kokeiltiin myös kolmeen muuhun foneemiin, /s/, /h/ ja /f/ -äänteisiin. Nämä kuuluvat foneettisesti frikatiivien luokkaan (/h/-äänne voidaan luokitella myös glottaaliäänteeksi), jotka ovat ns. kohinaäänteitä [60]. Ääntämykseltään ne ovat lähellä toisiaan, mutta eroavat kuunnellessa kuitenkin selvästi toisistaan (/s/ on soinnillinen, /f/ soinniton äänne). Toisin kuin vokaaleille, frikatiiveille /s/, /h/ ja /f/ ei ole olemassa valmista foneettista visualisaatiota.

Dimensionaalisuuden vähentämismenetelmä toimivat yleisesti ottaen hyvin kolmelle äänteelle. Erityisesti /s/- ja /h/-äänteet erottuvat projektoissa toisistaan hyvin. Äänne /f/ sijoittuu projektioissa puolestaan tyypillisesti äänteiden /s/ ja /h/ väliin. Kuvassa 36 on esitetty pääkomponenttianalyysin ja monidimensioskaalauksen (Sammonin jännitefunktio) tulokset kyseisille äänteille. Molemmat menetelmät projoisoivat /s/- ja /h/-äänteet omiksi ryhmikseen, joista /s/ on yleisesti ottaen tiiviimpi. Äänne /f/ on omana hajanaisena ryhmänään toisten äänteiden välissä. Mikäli aineistossa olisi enemmän äänteen /f/ edustajia, erottuisi ryhmä



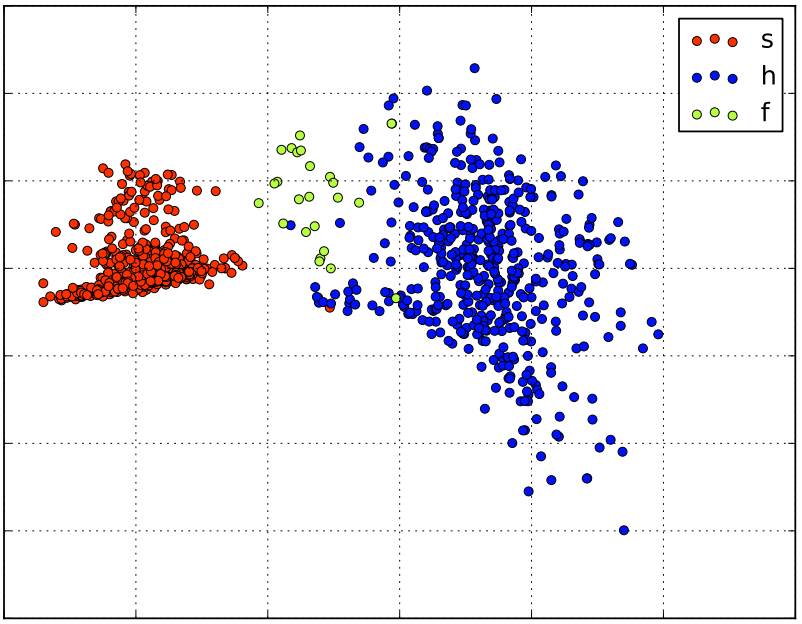

Kuva 37: CCA-algoritmi frikatiiviäänteille

todennäköisesti paremmin omaksi joukokseen.

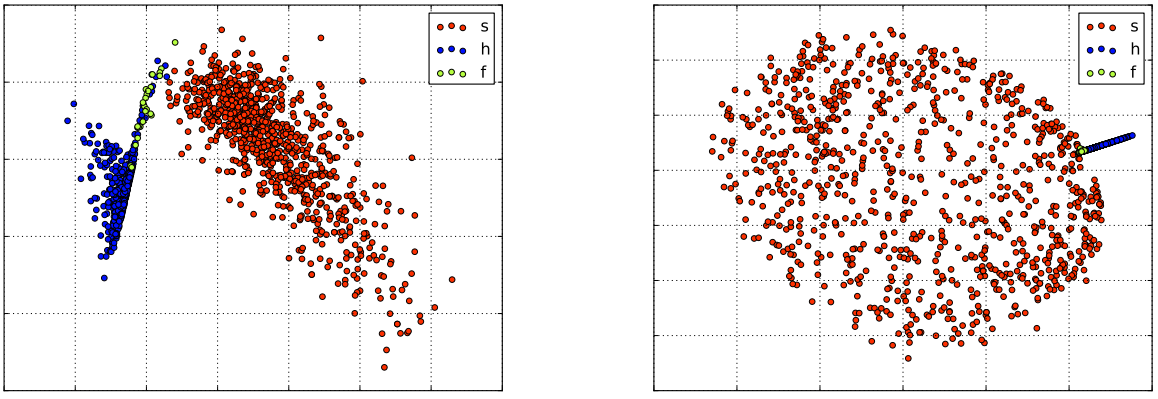
Kuten vokaalien tapauksessa, CCA tuottaa myös hyvän projektion frikatiiviäänteille. Kuvassa 37 voi myös havaita CCA:n taipumuksen tuottaa pyöreitä muotoja. Tämä johtuu käytössä olevasta jännitefunktiosta. CCA:n heikkoutena on se, että /f/-äänne projisoituu lähelle muita ryhmiä. Tämä puolestaan johtuu siitä, että CCA pyrkii minimoimaan projektioetäisyydet.

Kuvassa 38 on esitetty Isomap-algoritmin antama projektio frikatiiveille. Kuten vokaalien tapauksessa, Isomap muodosti yhtenäisen graafin 1384 äänteestä, kun äänteitä oli kaikkiaan 1437. Eri äänneluokat erottuvat projektiossa hyvin.

Ydinpääkomponenttianalyysi toimii frikatiiviäänteiden tapauksessa miltei samalla tavoin kuin vokaaleilla. Polynomisella ytimellä suoritettu projektio tuottaa äänteille selvästi erimuotoiset ja -kokoiset ryhmät, äänne /f/ on myös osin päällekäin äänteen /h/ kanssa. Gaussisen ytimen tuottamassa projektiossa nämä piirteet korostuvat, kun /f/-äänne romahtaa pienelle alueelle. Vaikka gaussisen ytimen tuottama projektio ei ole hyvä visuaalisesti, se kuitenkin erottaa eri

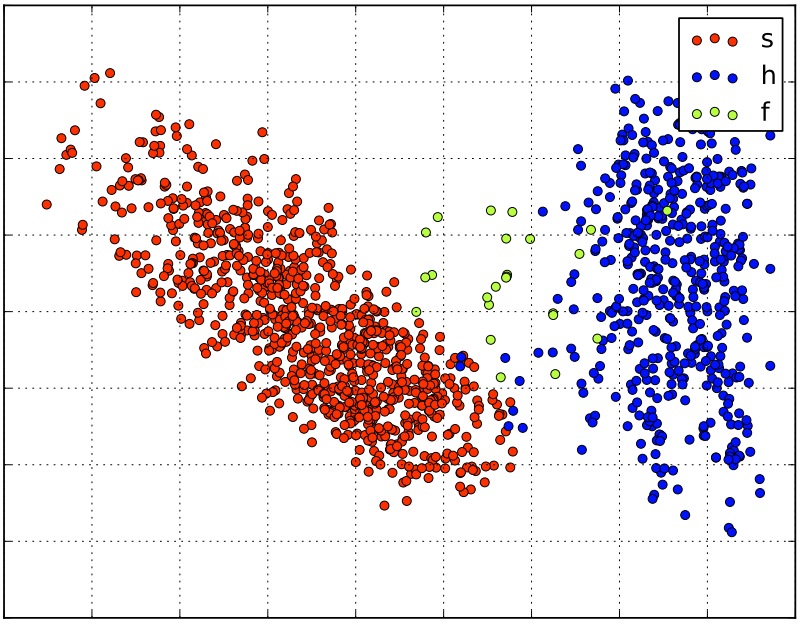



Kuva 38: Isomap-algoritmi frikatiiviäänteille




Polynominen ydin Gaussinen ydin

Kuva 39: Ydinpääkomponenttianalyysi frikatiiviäänteille



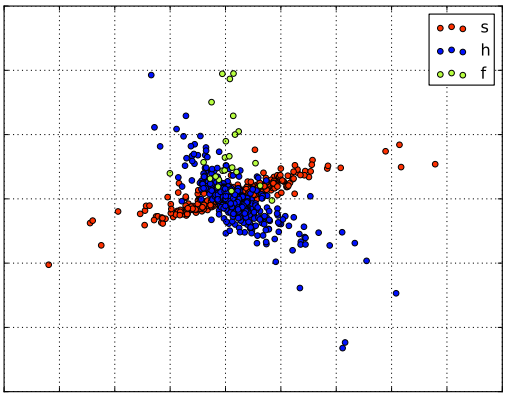

Kuva 40: MVU-algoritmi frikatiiviäänteille

äänteet hyvin.

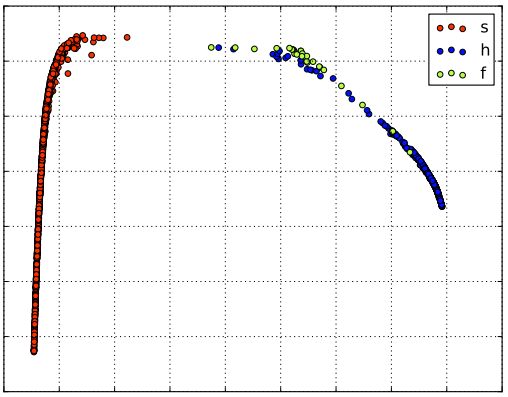
MVU-algoritmi toimii frikatiivien tapauksessa paremmin kuin vokaaleille. Kuvassa 40 eri äänteet erottuvat hyvin omiksi ryhmikseen lukuunottama muutamia päällekkäisyyksiä. Samoin kuten Isomap, MVU:n tekemässä projektiossa on mukana vain 1384 äännettä.

Topologiset menetelmät toimivat edelleen visualisoinnin kannalta huonoiten (kuva 41. Paikallisesti lineaarinen upotus (LLE) kerää äänteet keskelle yhteen ryhmään, mutta toisaalta pitää eri äänteet melko hyvin erossa toisistaan. Laplace-ominaiskuvaus puolestaan näyttää projisoivan äänteet U:n muotoiselle käyrälle. Visuaalisesti kuvaus ei näytä hyvältä, mutta voi mahdollisesti kuvata jotain parametria, joka äänteiden muodostuksessa esiintyy.

Kuvassa 42 on viimein esitetty kaksi satunnaisprojektiota frikatiiveille. Frikatiivien tapauksessa satunnaiskuvaukset olivat yleisesti ottaen hieman huonompia kuin vokaalien satunnaisprojektiot, eivätkä ryhmät erottuneet yhtä selkeästi kuin muita menetelmiä käyttämällä. Kuvassa vasemmalla on kuvaus, jossa ryhmät

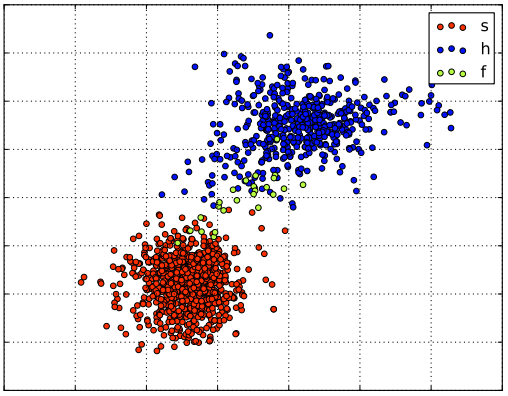


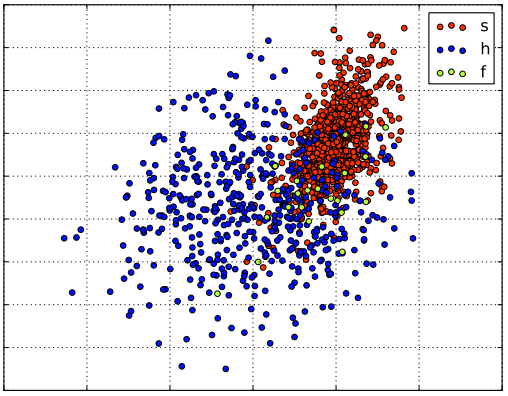

LLE




LE

Kuva 41: Paikallisesti lineaarinen upotus ja Laplace-ominaiskuvaus frikatiiviäänteille




Kuva 42: Satunnaisprojektioita frikatiiviäänteille

erottuvat parhaiten, oikeanpuolimmaisessa kuvassa puolestaan tyypillinen satunnaisprojektio, jossa ryhmät kuvautuvat päällekäin.

## 4.3 Dimensionaalisuuden vähentämisen vaikutus luokitteluun

Edellä esitellyt vokaali- ja frikatiiviäänteiden kaksiulotteisten projektioiden hyvyyttä voitiin tarkastella vain arvostelemalla visualisoinnin tulosta. Jotta menetelmiä pystyttäisiin arvioimaan objektiivisilla kriteereillä, testattiin dimensionaalisuuden vähentämisen vaikutuksia äänteiden luokittelussa.

Molemmat äänneryhmät, vokaalit ja frikatiivit projisoitiin jälleen kahteen ulottuvuuteen. Projisoidusta datasta kymmenesosa käytettiin tukivektorikoneen (*Support Vector Machine*, SVM) koulutusdatana, joka valittiin siten, että kukin äänneluokan edustajia oli sama 10 prosentin osuus mukana koulutuksessa. Loppuja 90 prosenttia äänteistä käytettiin testausdatana. Käytettävä tukivektorikonetoteutus oli LibSVM[11], versio 2.90.

Luokittelua testattiin aluksi alkuperäisellä datalla, jolloin frikatiivien osalta luokittelutarkkuudeksi saatiin 99.46%, vokaaleille 84.36%. Testausta varten käytettiin samoja datan kaksiulotteisia projektioita, jotka esiteltiin edellä visualisoinnin yhteydessä.

Taulukoissa 6 ja 7 on kunkin menetelmän käytön jälkeinen luokittelutarkkuus. Tuloksista ilmenee, että frikatiiviäänteiden osalta käytetyt dimensionaalisuuden vähentämismenetelmät eivät vaikuta suuresti luokittelutarkkuuteen, mikä ilmeni jo visualisoinnin yhteydessä. Paras menetelmä, CCA, ylsi jopa 98.68 prosentin luokittelutarkkuuteen.

Laplace-ominaiskuvaus ei tuottanut visuaalisesti hyvää projektiota, mutta luokittelutuloksen osalta LE on miltei yhtä hyvä kuin kuin CCA. Tämä osoittaa myös sen, että dimensionaalisuuden vähentämismenetelmien hyvyys riippuu paljon kulloisestakin sovelluksesta. Muuten perinteiset menetelmät, pääkomponenttianalyysi ja monidimensioskaalaus, antoivat seuraavaksi parhaat luokittelutulokset.

[11] http://www.csie.ntu.edu.tw/ cjlin/libsvm/

Taulukko 6: Luokittelun tuloksia frikatiiviäänteille

| **Menetelmä** | **Luokittelutarkkuus** | **Korjattu luokittelutarkkuus** |
|---|---|---|
| **Alkuperäinen data** | **99.46** | |
| CCA | 98.68 | 98.68 |
| LE | 98.61 | 98.61 |
| MDS (Sammon) | 98.30 | 98.30 |
| PCA | 98.22 | 98.22 |
| KPCA (polynominen ydin) | 97.91 | 97.91 |
| KPCA (gaussinen ydin) | 97.29 | 97.29 |
| Isomap | 95.82 | **99.52** |
| Satunnaisprojektio | 94.34 | 94.34 |
| MVU | 93.80 | **97.43** |
| LLE | 82.59 | **84.88** |

Osa menetelmistä (Isomap, MVU, LLE, frikatiiveille myös LE) ei projisoinut kaikkia datapisteitä, koska niistä ei muodostunut käytetyllä naapuruston koolla yhtenäistä graafia. Näiden menetelmien osalta taulukkoon on luokittelutuloksen lisäksi lisätty korjattu luokittelutulos. Tämä tulos ottaa huomioon vain ne datapisteet, jotka ovat mukana projektiossa. Isomapin osalta korjattu luokittelutarkkuus on erittäin hyvä, jopa parempi kuin alkuperäisen datan luokittelutarkkuus.

Menetelmistä huonoiten suorituvat maksimaalisen varianssin upotus (MVU) ja paikallisesti lineaarinen upotus (LLE), jotka antavat frikatiiviäänteiden tapauksessa huonomman tuloksen kuin satunnaisprojektio.

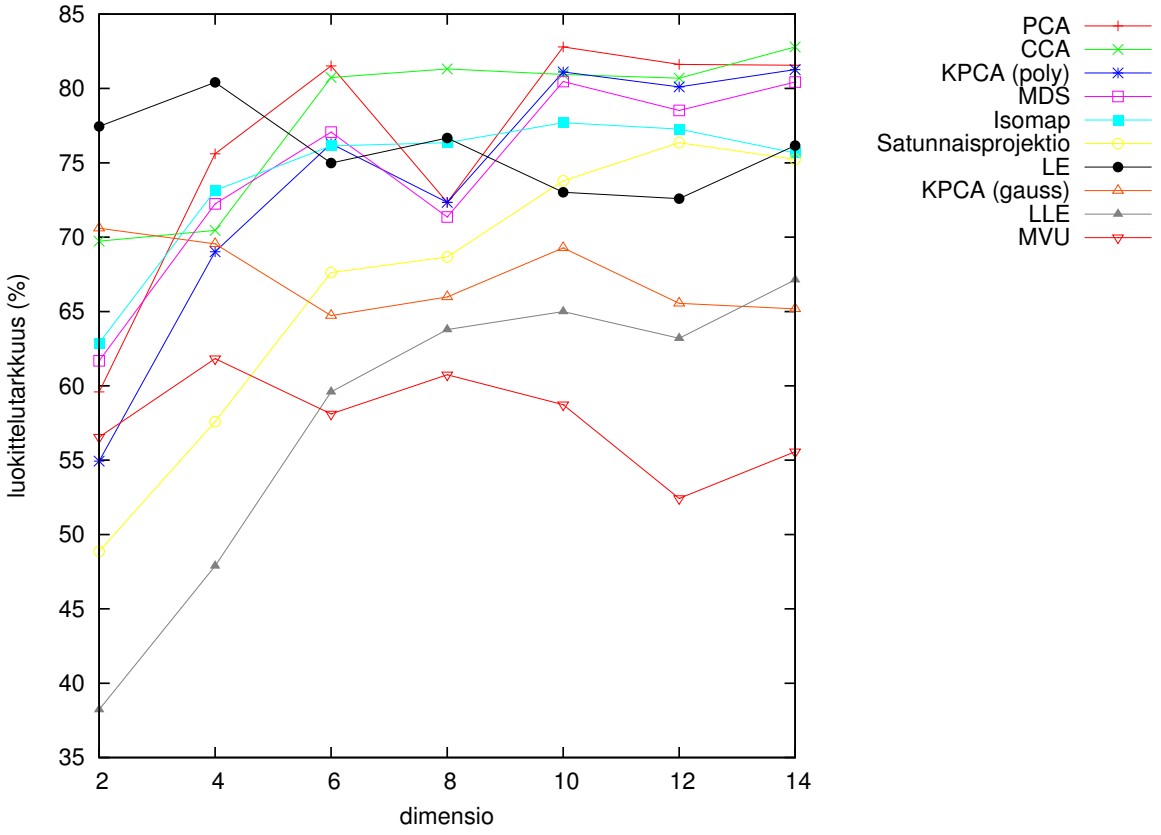
Vokaaliäänteet ovat luokittelun suhteen vaikeampi datajoukko kuin frikatiivit. Vokaaleissa luokkia on enemmän ja ne ovat myös osittain päällekkäisiä. Tästä huolimatta Laplace-ominaiskuvaus ei juurikaan vähennä luokittelutarkkuutta. Tässä vaikeammassa datajoukossa nähdään myös se miten uudemmat, epälineaariset menetelmät menestyvät yleisesti ottaen PCA:ta ja MDS:a paremmin, vaikka niidenkin luokittelutarkkuus laskee selvästi. Tämä viittaa siihen, että vokaaliavaruus muodostaa jatkuvan hyperpinnan, jonka graafi- ja topologiapohjaiset menetelmät pystyvät osittain mallintamaan. Menetelmistä heikoiten menestyy jälleen

Taulukko 7: Luokittelun tuloksia vokaaliäänteille

| **Menetelmä** | **Luokittelutarkkuus** | **Korjattu luokittelutarkkuus** |
|---|---|---|
| **Alkuperäinen data** | **84.36** | |
| LE | 77.45 | **81.70** |
| KPCA (gaussinen ydin) | 70.61 | 70.61 |
| CCA | 69.73 | 69.73 |
| Isomap | 62.89 | **66.34** |
| MDS (Sammon) | 61.68 | 61.68 |
| PCA | 59.59 | 59.59 |
| MVU | 56.55 | **59.66** |
| KPCA (polynominen ydin) | 54.94 | 54.94 |
| Satunnaisprojektio | 48.87 | 48.87 |
| LLE | 38.22 | **39.94** |

LLE, tälläkin kertaa selvästi satunnaisprojektiota huonommin.

Kaksiulotteisen luokittelun lisäksi testattiin vielä sitä, miten luokittelutulos muuttuu kohdedimension muuttuessa. Testidatana käytettiin vain vokaaliäännedataa, koska frikatiividatan luokittelutarkkuus oli kaksiulotteisessa projektiossa lähellä alkuperäistä tarkkuutta. Kuvassa 43 on esitetty, miten dimensionaalisuuden vähentämismenetelmien luokittelutulos muuttuu kohdedimension muuttuessa. Kuvasta voi havaita luokittelutarkkuuden laskevan vain lievästi, kun dimensionaalisuus pienenee. Laskeva trendi on selvä satunnaisprojektiolla ja LLE:llä. Myös muilla menetelmillä luokittelutarkkuus yleisesti ottaen laski dimensionaalisuuden laskiessa, poikkeuksina kuitenkin MVU, KPCA gaussisella ytimellä ja LE, joiden tuottamien projektioiden tarkkuus oli parempi kahdessa ulottuvuudessa kuin 14-ulotteisena. Toisessa tutkimuksessa, jossa dimensionaalisuuden vaikutusta äänidatan luokitteluun tutkittiin, saatiin hieman samansuuntaisia tuloksia [26].




Kuva 43: Vokaalidatan luokittelutuloksen muutos dimension funktiona.

# 5 Yhteenveto

Tässä työssä on esitelty joitakin dimensionaalisuuden analysoimis- ja vähentämistekniikoita. Erityisesti erilaisia dimensionaalisuuden vähentämismenetelmiä on runsaasti muitakin, mutta edellä esitellyt menetelmät on valittu siten, että mahdollisimmat moni toimintaperiaatteeltaan erilainen menetelmä olisi edustettuna.

Pääkomponenttianalyysi ja monidimensioskaalaus ovat menetelminä vanhempia, tunnettuja ja yleisesti käytettyjä. Menetelmien suurin heikkous on siinä, etteivät ne välttämättä pysty säilyttämään datassa olevia epälineaarisia riippuvuuksia. Tämän vuoksi menetelmien rinnalle on kehitetty kasvava joukko epälineaariselle datalle soveltuvia menetelmiä. Usein menetelmien taustaoletuksena on, että data sijaitsee yhtenäisellä hyperpinnalla. Hyperpinta pyritään avaamaan pienempään ulottuvuuteen siten, että joko pinnan muoto, pisteiden väliset etäisyydet, kulmat tai jokin muu piirre säilyy.

Monet menetelmät (Isomap, MVU, LLE, LE) käyttävät pinnan mallintamiseen graafia, mikä asettaa rajoituksia käytettävälle datalle. Mikäli kaikki data halutaan projisoida pienempään ulottuvuuteen, täytyy muodostuvan graafin olla yhtenäinen. Toinen graafin asettama haaste on sopivan naapuruston koon löytäminen. Naapuruston täytyy olla tarpeeksi suuri, että se voi mallintaa hyperpintaa, mutta se ei saa olla liian suuri, koska tällöin graafietäisyydet vastaavat euklidisia etäisyyksiä. Tässä työssä ei käsitelty naapuruston koon määrittämisen problematiikkaa, mikä on yksi mahdollinen tutkimuksen jatkoaihe.

Dimensionaalisuuden vähentämismenetelmiä on julkaisuissa usein havainnollistettu erilaisten keinotekoisten datajoukkojen projektioina kolmesta ulottuvuudesta kahteen ulottuvuuteen. Tässä työssä tehtiin myös näin, mutta tämän lisäksi menetelmien testaamisessa käytetiin todellista dataa, jonka muodosti joukko vokaaleja frikatiiviäänteitä MFCC-vektoreiden muodossa. Esitellyt menetelmät soveltuivat melko hyvin näiden kaksiulotteisiin projektioihin. Suurin osa menetelmistä pystyi muodostamaan sellaisen projektion, että erilaiset äänteet olivat erillään projektiossa. Vokaaliäänteiden osalta, jotkin menetelmät pystyivät heijastamaan äänteiden foneettisia piirteitä – saaduissa projektioissa oli yhtymäkohtia foneettiseen vokaalikarttaan.

Myös graafipohjaiset menetelmät toimivat äänteiden dimensionaalisuuden vähentämisessä. Muodostetut graafit eivät kattaneet koko dataa, mutta projektiot kattoivat frikatiivien tapauksessa vähintään 96% ja vokaalien kohdalla 94% koko datasta.

Visuaalisten projektioiden lisäksi tutkittiin sitä, miten dimensionaalisuuden vähentäminen vaikuttaa luokittelutarkkuuteen. Odotetusti luokittelutarkkuus yleensä heikkeni, kun dimensionaalisuus väheni, mutta heikkenemistä ei kaikissa tapauksissa tapahtunut, tai se ei ollut suurta. Tämä kertoo siitä, että dimensionaalisuuden vähentämismenetelmät toimivat testidatan tapauksessa. Ne pystyvät vähentämään datan dimensionaalisuutta siten, että olennaiset piirteet datasta säilyvät.

# Viitteet


[1] TIMIT. DARPA TIMIT acoustic phonetic continuous speech database. CD-ROM, 1988.

[2] D. Achlioptas. Database-friendly random projections. In *20th Annual Symposium on Principles of Database Systems, Santa Barbara, CA*, pages 274–281, 2001.

[3] N. Ailon and B. Chazelle. Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform. In *Annual ACM Symposium on Theory of Computing*, volume 38, pages 557–563, 2006.

[4] A. Aizerman, E. M. Braverman, and L. I. Rozoner. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

[5] S.-H. Bae. Parallel multidimensional scaling performance on multicore systems. pages 695–702.

[6] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems (NIPS 2001)*, volume 14, 2002.

[7] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neuran Computation*, 15(6), 2003.

[8] R.E. Bellman. *Adaptive control processes: a guided tour*. Princeton University, 1961.

[9] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? *Lecture Notes in Computer Science*, 1540:217–235, 1999.

[10] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250. ACM, 2001.

[11] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[12] I. Borg and P.J.F. Groenen. *Modern Multidimensional Scaling*. Springer, 2005.

[13] F. Camastra. Data dimensionality estimation methods: a survey. *Pattern Recognition*, 36:2945–2954, 2003.

[14] R. Clarke, H.W. Ressom, A. Wang, J. Xuan, M.C. Liu, E. A. Gehan, and Y. Wang. The properties of high-dimensional data spaces: implications for exploring gene and proteing expression data. *Nature Reviews Cancer*, 8:37–49, 2008.

[15] J. Costa and A.O. Hero. Manifold learning with geodesic minimal spanning trees. *CoRR*, cs.CV/0307038, 2003.

[16] S. Dasgupta and A. Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical report, Intl. Comput. Sci. Inst., Berkeley, CA, 1999.

[17] S.B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[18] J. de Leeuw. Applications of convex analysis to multidimensional scaling. In *Department of Statistics Papers*. Department of Statistics, UCLA, 2005.

[19] V. de Silva and J.B. Tenenbaum. Global versus local methods in nonlinear dimensionality reduction. In *Advances in Neural Information Processing Systems*, volume 15, pages 705–712, 2003.

[20] P. Demartines and J. Hérault. Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks*, (8):148–154, 1997.

[21] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, (1):269–271, 1959.

[22] D.L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 100, pages 5591–5596, 2003.

[23] J.-P. Eckmann and D. Ruelle. Ergodic theory of chaos and strange attractors. *Reviews of Modern. Physics*, 57(3):617–656, Jul 1985.

[24] J.-P. Eckmann and D. Ruelle. Fundamental limitations for estimating dimensions and Lyapunov exponents in dynamical systems. *Physica D*, 56:185–187, 1992.

[25] G.A. Edgar. *Measure, Topology, and Fractal Geometry.* Springer-Verlag, 1990.

[26] A. Errity and J. McKenna. An investigation of manifold learning for speech analysis. In *Proceedings of the International Conference on Spoken Language Processing (Interspeech 2006 - ICSLP)*, pages 2506–2509, 2006.

[27] X.Z. Fern and C.E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003)*, volume 20, pages 186–193, Washington DC, 2003.

[28] R.W. Floyd. Algorithm 97: Shortest path. *Communications of ACM*, 5(6), 1962.

[29] D. Fradkin and D. Madigan. Experiments with random projections for machine learning. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 517–522, New York, NY, USA, 2003. ACM Press.

[30] K. Fukunaga and D. Olsen. An algorithm for finding intrinsic dimensionality of data. *IEEE Transactions on Computers*, 20(2):176–183, 1971.

[31] G.H. Golub and C. F. Van Loan. *Matrix Computations.* John Hopkins Univerisity, 3 edition, 1996.

[32] P. Grassberger. An optimized box-assisted algorithm for fractal dimension. *Physics Letters A*, 148(1,2):63–68, 1990.

[33] P. Grassberger and I. Procaccia. Measuring the strangeness of strange attractors. *Physica D: Nonlinear Phenomena*, 9(1-2):189–208, 1983.

[34] F. Hausdorff. Dimension und äusseres mass. *Mathematische Annalen*, 79(1-2):157–179, 1918.

[35] D. Hilbert. Über stetige abbildung einer linie auf ein flachenstuck. *Mathematische Annalen*, (38):459–460, 1891.

[36] H. Hotelling. Analysis of a complex statistical variables into components. *Journal of Educational Psychology*, (24):417–441, 1933.

[37] V. Isham. *Statistical aspects of chaos: a review*, chapter Networks and Chaos-Statistical and Probabilistic Aspects, pages 124–200. Chapman & Hall, London, 1993.

[38] W. Johnson and J. Lindenstrauss. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, (26):189–206, 1984.

[39] J.W. Sammon Jr. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, (C-18):401–409, 1969.

[40] K. Karhunen. Über lineare methoden in der wahrscheinlichkeitrechnung. In *Annales Academiae Scientorum Fennicae*, number 37 in Ser. A. I. Math.-Phys., pages 1–79. 1947.

[41] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 3rd edition edition, 1989.

[42] J.B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrica*, (29):1–26, 1964.

[43] B. Kégl. Intrinsic dimension estimation using packing numbers. In *Advances in Neural Information Processing*, volume 15, 2003.

[44] J. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, 2007.

[45] J.A. Lee, A. Lendassa, N. Donckers, and M. Verleysen. A robust nonlinear projection method. In *Proceedings of ESANN 2000, 8th European Symposium on Artificial Neural Networks*, pages 13–20.

[46] J.A. Lee, A. Lendasse, and M. Verleysen. Curvilinear distance analysis versus isomap. In *Proceedings of ESANN 2002, 10th European Symposium on Artificial Neural Networks*, pages 185–192.

[47] E. Levina and P.J. Bickel. Maximum likelihood estimation of intrinsic dimension. In Lawrence K. Saul, Yair Weiss, and léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 777–784. MIT Press, Cambridge, MA, 2005.

[48] M.-S. Oh and A.E. Raftery. Bayesian multidimensional scaling and choice of dimension. *Journal of the American Statistical Association*, (96), 2001.

[49] E. Ott. *Chaos in dynamical systems*. Cambridge Univeristy, 1993.

[50] G. Peano. Sur une courbe, qui remplit toute une aire plane (on a curve which completely fills a planar region. *Mathematische Annalen*, (36):157–160, 1890.

[51] K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 6(2):559–572, 1901.

[52] K.W. Pettis, T.A. Bailey, A.K. Jain, and R.C. Dubes. An intrinsic dimensionality estimator from near-neighbor information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(1):25 –37, 1979.

[53] J.O. Ramsay. Maximum likelihood estimation in multidimensional scaling. *Psychometrica*, (42):241–266, 1977.

[54] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, (290):2323–2326, 2000.

[55] A. Rényi. On the dimension and entropy of probability distributions. *Acta Mathematica Hungarica*, 10(1–2):193–215, 1959.

[56] L. K. Saul and S. T. Roweis. Think globally, fit locally - unsupervised learning of low dimensional manifolds. *Journal of Machine Learning Research*, (4):119–155.

[57] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computing*, (10):1299–1319, 1998.

[58] J. Shawe-Taylor and N. Cristiani. *Kernel Methods for Pattern Analysis*. Cambridge University, 2004.

[59] P. Somervuo. Speech dimensionality analysis on hypercubical self-organizing maps. In *Maps," Neural Processing Letters*, pages 125–136, 2001.

[60] K. Suomi, J. Toivanen, and R. Ylitalo. *Fonetiikan ja suomen äänneopin perusteet*. Gaudeamus, 2006.

[61] J.B. Tenenbaum. Mapping a manifold of perceptual observations. volume 10, pages 682–688, 1997.

[62] J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December 2000.

[63] W.S. Torgerson. Multidimensional scaling: I. theory and method. *Psychometrika*, (17):401–419, 1952.

[64] G.V. Trunk. Statistical estimation of the instrinsic dimensionality of a noisy signal collection. *IEEE Transactions on Computers*, 25(2):165–171, 1976.

[65] L. van der Maaten, E. Postma, and J. van den Herik. Dimensionality reduction: A comparative review. *Journal of Machine Learning Research 10*, (10):1–41, 2009.

[66] L. Vandenberge and S. Boyd. Semidefinite programming. *SIAM Review*, (38):49–95, 1996.

[67] J. Venna. *Dimensionality Reduction for Visual Exploration of Similarity Structures*. PhD thesis, Helsinki University of Technology, 2007.

[68] P.J. Verveer and R.P.W. Duin. An evaluation of intrinsic dimensionality estimators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:81–86, 1995.

[69] K. Q. Weinberger, F. Sha, Q. Zhu, and L.K. Saul. Graph laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems 19: Proceedings of the 2006 Conference*, pages 1489–1496.

[70] K.Q. Weinberger, B.D. Packer, and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005.

[71] K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the IEEE Conference on*

*Computer Vision and Pattern Recognition (CVPR-04)*, volume 2, pages 988–995, 2004.

[72] K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*, 70(1):77–90, 2006.

[73] K.Q. Weinberger, F. Sha, and L.K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML-04)*, pages 839–846, 2004.