521160P Johdatus Tekoälyyn

Harjoitus #4

Ohjaamaton oppiminen

Kevät 2019

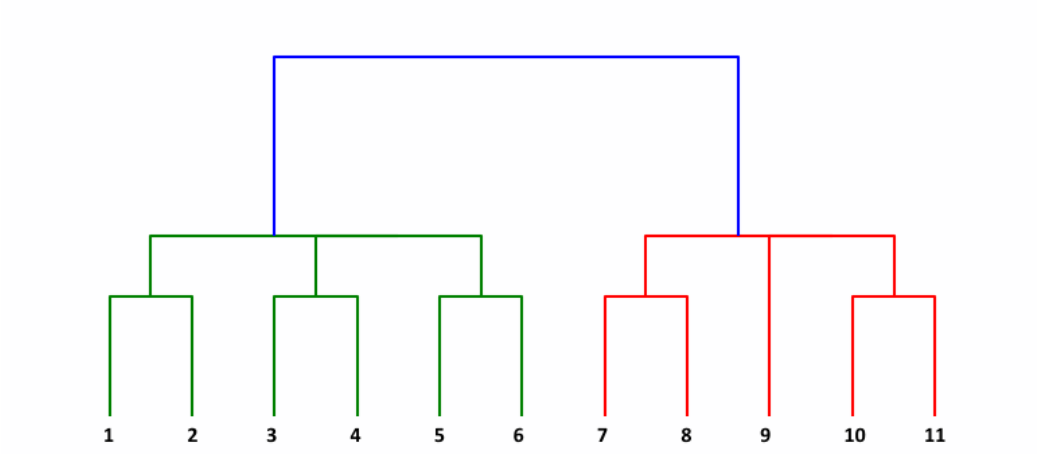
Kun koneoppimisen algoritmi käyttää ilman luokkatietoa sisältävää dataa, on kyseessä ohjaamaton oppiminen. Koska datan näytteille ei ole määritelty etukäteen luokkia, täytyy datan ryhmittely tehdä täysin perustuen datan rakenteeseen. Usein ei myöskään ole etukäteen mitään tietoa mahdollisten ryhmien lukumäärästä tai muista jakamiseen liittyvistä kriteereistä, mikä tekee ongelmasta haastavan. Ohjaamaton oppiminen jaetaan yleisesti klusterointiin ja dimensionaalisuuden vähentämiseen.

Klusterointi

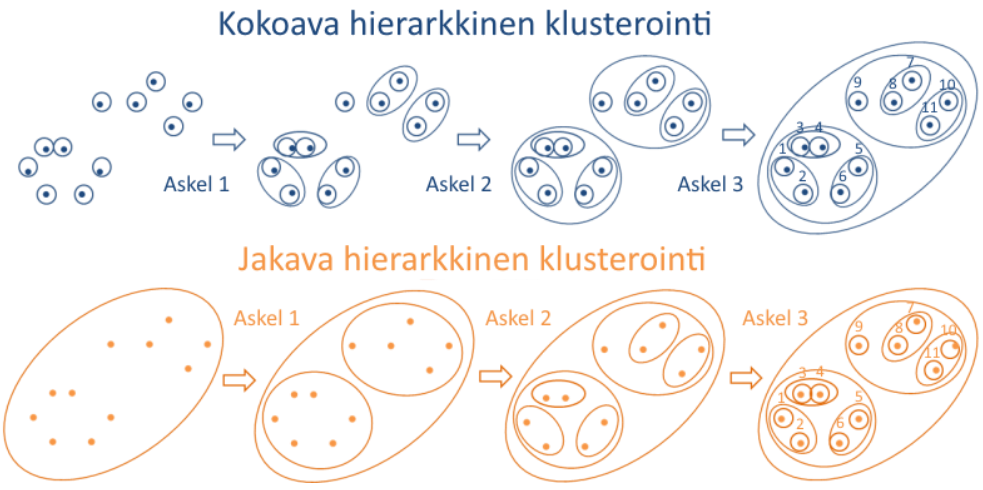
Klusterointi on ohjaamattoman oppimisen tekniikka, jota käytetään datan analysoimiseen ja ryhmittelyyn. Klusteroinnin päämääränä on jakaa data pienempiin ryhmiin eli klustereihin, joiden näytteet ovat mahdollisimman samankaltaisia keskenään, mutta eri klustereiden väliset näytteet ovat mahdollisimman erilaisia toisiinsa nähden. Datan näytteiden samankaltaisuutta voidaan mitata esimerkiksi euklidisen etäisyyden tai todennäköisyystiheyksien perusteella.

Jokaisella eri klusterointiongelmalla ovat omat tunnuspiirteensä, minkä takia ei ole olemassa yleisesti käytettyä menetelmää, joka toimii hyvin jokaisessa tilanteessa. Eri klusterointimenetelmät voidaankin jakaa karkeasti sen mukaan, mitä ominaisuuksia käytetty menetelmä hyödyntää.

Hierarkkiset klusterointimenetelmät ovat joko jakavia tai kokoavia [1]. Jakavissa menetelmissä lähdetään liikkeelle joukosta, johon kaikki näytteet kuuluvat ja ne jaetaan samankaltaisuuksien perusteella aina pienempiin joukkoihin askel kerrallaan, kunnes kaikki näytteet on saatu klusteroitua. Kokoavissa menetelmissä puolestaan lähdetään liikkeelle yksittäisistä näytteistä, jotka pyritään yhdistämään aina suurempiin samankaltaisiin kokonaisuuksiin askel askeleelta. Tämän perusteella pystytään piirtämään yksinkertainen puukaavio eli dendrogrammi, joka kuvaa hierarkkisesti näytteiden välisiä samankaltaisuuksia. Kuvassa 1 on esitetty erään klusterointiongelman dendrogrammi. Kuvassa 2 on muodostettu samalle klusterointiongelmalle Venn-diagrammit kokoavalla ja jakavalla hierarkkisella klusterointimenetelmällä.



Kuva 1. Erään klusterointiongelman dendrogrammi.



Kuva 2. Venn-diagrammit toteutettuna kokoavalla ja jakavalla hierarkkisella klusterointimenetelmällä.

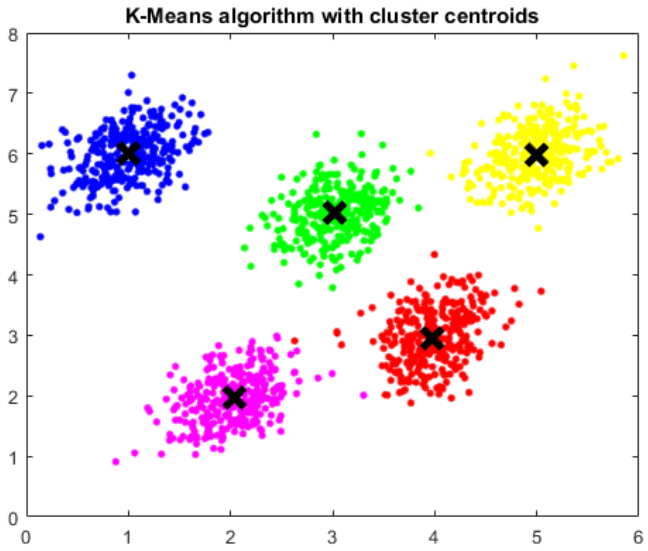
Tiheysperusteisissa klusterointimenetelmissä nimensä mukaisesti datan näytteet jaetaan klustereihin niiden tiheystiedon perusteella. Eräs erittäin tunnettu tiheysperusteinen klusterointimenetelmä on DBSCAN (engl. density-based spatial clustering of applications with noise). Kyseinen algoritmi ottaa parametreinaan näytteiden välisen minimietäisyyden (*epsilon*) sekä pienimmän määrän näytteitä, joista vielä voidaan muodostaa klusteri (*min_samples*). Yhdessä nämä kaksi parametria määräävät tiheyskriteerin, jonka perusteella kyseinen menetelmä yhdistää lähellä toisiaan olevat näytteet yhdeksi klusteriksi. Mikäli tiheyden perusteella jokin näytteistä on kaukana mistään potentiaalisesta klusterista (engl. outlier), jätetään se kokonaan klusteroimatta. Menetelmän hyvä puoli on, että etukäteen ei tarvitse määrittää klustereiden lukumäärää, kuten monessa muussa klusterointimenetelmässä. Lisäksi algoritmi pystyy löytämään tiheystiedon perusteella minkä tahansa muotoisia klustereita. DBSCAN on kuitenkin erittäin sensitiivinen sen määritettyihin parametreihin ja mikäli tiheyskriteeri on valittu väärin, menetelmä tuottaa ei-toivotun näköisen lopputuloksen. DBSCAN ei myöskään onnistu klusteroimaan oikein tapauksia, joissa näytteiden välinen tiheys klusterin sisällä vaihtelee huomattavasti.

Keskipisteperusteisessa klusterointimenetelmässä tai toisin sanoen K-means-menetelmässä mitataan näytteiden samankaltaisuutta vertaamalla niiden etäisyyksiä klusterikeskipisteisiin. Tässä menetelmässä klustereiden lukumäärä tulee tuntea etukäteen. Aluksi K-means-algoritmi sijoittaa sattumanvaraisesti klusterikeskipisteet data-avaruuteen. Tämän jälkeen algoritmi iteroi kahden seuraavan vaiheen välillä:

1. Jokainen näyte datassa on klusteroitu sen mukaan, mikä on näytteen lähin klusterikeskipiste euklidisen etäisyyden perusteella.

2. Klusterikeskipisteet lasketaan uudestaan ottamalla vaiheen 1 mukaisesti klusteroiduista näytteistä keskiarvot ja asettamalla lasketut keskiarvot uusiksi klusterikeskipisteiksi.

Algoritmi iteroi näiden kahden vaiheen välillä kunnes ennalta määrätty pysähtymiskriteeri toteutuu tai etukäteen annettu iteraatioiden määrä saavutetaan. Pysähtymiskriteeri voi olla esimerkiksi tilanne, kun klusterikeskipisteiden sijainnit data-avaruudessa eivät päivity enää iteraatioiden välillä. K-means-algoritmilla suppeneminen lopputulokseen on taattu vaikkakin joskus klusteroinnin lopputulos saattaa olla ei-toivotun näköinen. Lisäksi klusterikeskipisteiden alustaminen on K-means-algoritmin toimivuuden kannalta tärkeässä roolissa. Sattumanvaraisen sijoittelun sijaan yksi hyvä strategia on alustaa klusterikeskipisteet yhtä etäälle mutta mahdollisimman kauas toisistaan. Kuvassa 3 on esitetty klusteroinnin lopputulos K-means-algoritmilla. Kukin näyte datassa kuuluu lähimmän klusterikeskipisteen määräämään klusteriin.



Kuva 3. K-means-algoritmin tuottama lopputulos eräälle datalle, kun klustereiden lukumäärä on viisi.

Koska klustereiden lukumäärä täytyy tuntea ennen monen eri klusterointimenetelmän käyttämistä, klustereiden lukumäärän arvioimiseen on kehitetty erilaisia menetelmiä ja mittoja. Yksi käytetty mitta optimaalisen klustereiden lukumäärän selvittämiseksi on siluettikerroin (engl. silhouette score). Se antaa arviona klusteroinnin lopputulokselle, kuinka hyvin keskimäärin näytteet sopivat niiden omiin klustereihinsa muihin klustereihin verrattuna. Siluettikerroin saa arvoja -1 ja 1 väliltä, missä arvo 1 tarkoittaa, että näytteet omissa klustereissaan ovat täysin samanlaisia keskenään ja arvo -1 tarkoittaa, että näytteet omissa klustereissaan ovat täysin erilaisia toisiinsa verrattuna.

Malliperusteisissa menetelmissä data kuvataan mahdollisimman tarkasti jonkin tunnetun matemaattisen mallin avulla. Matemaattisena mallina voidaan käyttää esimerkiksi todennäköisyystiheysjakaumaa tai se voi pohjautua esimerkiksi neuroverkkoon.

Dimensionaalisuuden vähentäminen

Dimensionaalisuuden vähentämisessä pyritään pienentämään korkeadimensioisen datajoukon piirteiden lukumäärää olennaista informaatiota menettämättä. Tämän seurauksena datan tärkeimmät ominaisuudet voidaan selittää pienemmällä määrällä alkuperäisistä piirteistä valituilla tai muunnetuilla piirteillä.

Dimensionaalisuuden vähentämisessä on kaksi erilaista lähestymistapaa: piirteiden valitseminen (engl. feature selection) ja piirteiden pakkaaminen (engl. feature extraction). Piirteiden valitsemisessa keskitytään löytämään alkuperäisistä piirteistä koostuva osajoukko, joka mallintaa mahdollisimman hyvin koko datan ominaisuuksia. Piirteiden pakkaamisessa puolestaan muunnetaan korkeadimensioinen data matalampidimensioiseksi siten, että menetetään mahdollisimman vähän merkittävää informaatiota. Tässä harjoituksessa käsitellään piirteiden pakkaamista.

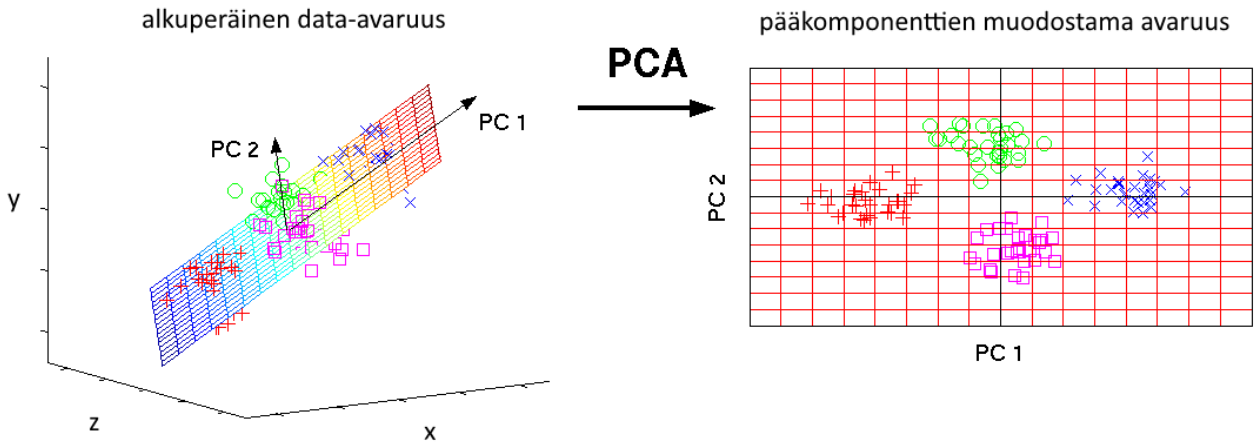
Yleisesti ottaen piirteiden pakkaamisella on kaksi erilaista käyttötarkoitusta. Sitä joko käytetään datan visualisointiin, kun halutaan selvittää ennestään tuntemattoman datan rakenne projisoimalla korkeadimensioinen data kahteen tai kolmeen dimensioon. Toinen yleinen käyttötarkoitus piirteiden pakkaamiselle on pienentää jopa tuhansista piirteistä koostuvan alkuperäisen datan piirteiden lukumäärä muutamaan kymmeneen piirteeseen, jonka jälkeen esimerkiksi ohjatun oppimisen luokittelijan opettamisprosessiin kuluvaa aikaa saadaan pienennettyä huomattavasti. Tämän lisäksi päästään mahdollisesti samalla eroon kohinaisista näytteistä sekä pystytään yhdistämään toistensa kanssa korreloivia piirteitä aivan uusiksi piirteiksi.

Piirteiden pakkaamiseen voidaan käyttää useita erilaisia dimensionaalisuuden vähentämismenetelmiä. Aivan kuten ohjatun oppimisen luokittelussa tai ohjaamattoman oppimisen klusteroinnissa, mikään yksittäinen dimensionaalisuuden vähentämismenetelmä ei ole muita menetelmiä selvästi parempi, vaan ongelmasta riippuen ne toimivat eri tavalla. Näin ollen on lähes mahdoton sanoa etukäteen testaamatta, mitä menetelmää tulisi käyttää vähentämisongelmaan.

Eri dimensionaalisuuden vähentämismenetelmät pyrkivät säilyttämään datan eri ominaisuuksia, sillä datan kaikkia ominaisuuksia ei voi säilyttää. Ne voivat esimerkiksi pyrkiä minimoimaan keskineliövirhettä, säilyttämään datan pisteiden välisiä etäisyyksiä tai pisteiden välisiä naapuruussuhteita. Seuraavaksi esitellään toimintaperiaatteet yleisellä tasolla seuraaville dimensionaalisuuden vähentämismenetelmille: pääkomponenttianalyysi (engl. principal component analysis, PCA), multidimensioskaalaus (engl. multi dimensional scaling, MDS) ja t-SNE (engl. t-distributed stochastic neighbor embedding).

Pääkomponenttianalyysi on lineaarinen dimensionaalisuuden vähentämismenetelmä. Siinä korkeaulotteiseen data-avaruuteen määritellään uudet muuttujat eli pääkomponentit, jotka ovat riippumattomia lineaarikombinaatioita kaikista ilmiön kuvaamiseen käytetyistä muuttujista. Pääkomponentit valitaan siten, että ne kuvaavat suurimman osan datan vaihtelusta ja ne ovat ortogonaalisia toisiinsa nähden. Ensimmäisen pääkomponentin tulee selittää mahdollisimman suuren osan koko datan vaihtelusta, toisen pääkomponentin tulee selittää mahdollisimman suuren osan jäljelle jäävän datan vaihtelusta ja niin edelleen.

Pääkomponenttianalyysi toimii hyvin, mikäli muuttujien välillä on vahvoja korrelaatioita ja datan riippuvuus on lineaarisesti selitettävissä. Muussa tapauksessa on käytettävä jotain epälineaarista dimensionaalisuuden vähentämismenetelmää. Kuvassa 4 kolmiulotteiseen data-avaruuteen on laskettu pääkomponentit PC1 ja PC2, jonka jälkeen pääkomponenteista tulee kaksiulotteisen kuvaajan akselit ja dimensioita saadaan pudotettua yhdellä.



Kuva 4. Pääkomponenttianalyysin toimintaperiaate kolmidimensioisen datan muunnoksesta kaksidimensioiseksi.

Multidimensioskaalaus on puolestaan epälineaarinen dimensionaalisuuden vähentämismenetelmä. Multidimensioskaalaus-algoritmi sijoittaa jokaisen näytteen korkeaulotteisesta avaruudesta matalampiulotteiseen avaruuteen niin, että näytteiden väliset etäisyydet säilyvät muunnoksen jälkeen mahdollisimman ennallaan. Etäisyysmittana voidaan käyttää esimerkiksi euklidista etäisyyttä.

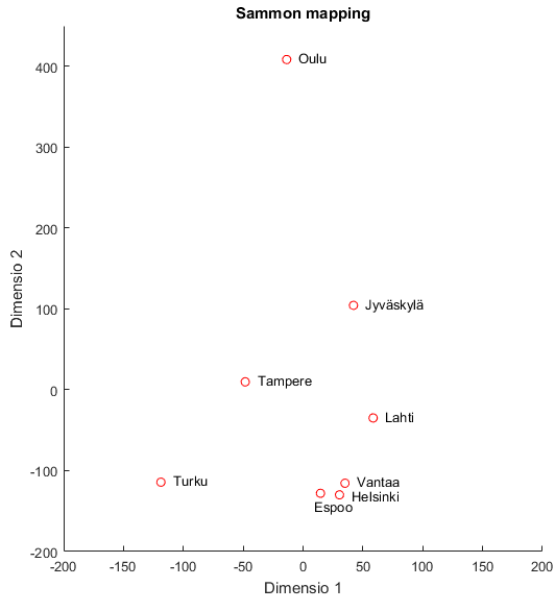
Siinä missä pääkomponenttianalyysi analysoi dataa kovarianssimatriisin avulla saaden selville vektorit datan suurimmalle vaihtelulle, käyttää multidimensioskaalaus näytteiden etäisyystietoa ja häviöfunktiota (engl. loss function) dimensionaalisuuden vähentämisessä.

Täydellistä sovitusta dimensioiden välillä ei pystytä saavuttamaan dimensionaalisuuden kirouksesta johtuen (engl. curse of dimensionality), mutta minimoitavan häviöfunktion avulla voidaan saavuttaa hyviä tuloksia. Multidimensioskaalaus-algoritmi johtaa Sammon-mapping-algoritmiin, kun alkuperäisen multidimensioskaalaus-algoritmin käyttämä minimoitava häviöfunktio normalisoidaan korkeadimensioisen avaruuden etäisyyksillä.

Kuvassa 5 on esitetty Sammon-mapping-algoritmin tuottama lopputulos taulukon 1 symmetriselle etäisyysmatriisille. Algoritmi saa datajoukkona symmetrisessä etäisyysmatriisissa Suomen kahdeksan suurimman kaupungin maantieteelliset etäisyydet toisistaan. Tämän jälkeen algoritmi pudottaa kahdeksandimensioisen datan kahteen dimensioon. Lopputuloksesta huomataan, että kaupunkien paikat kuvaajassa vastaavat melko tarkasti kaupunkien oikeita sijainteja Suomen kartalla. Tämä selittyy sillä, että multidimensioskaalaus pyrkii säilyttämään mahdollisimman tarkasti alkuperäisten näytteiden etäisyydet.

Taulukko 1. Suomen kahdeksan suurimman kaupungin maantieteelliset etäisyydet toisistaan esitettynä etäisyysmatriisissa.

	Helsinki	**Espoo**	**Tampere**	**Vantaa**	**Oulu**	**Turku**	**Jyväskylä**	**Lahti**
Helsinki	0	16	161	15	540	150	235	99
Espoo	16	0	151	24	537	134	234	103
Tampere	161	151	0	150	400	143	131	116
Vantaa	15	24	150	0	526	154	220	84
Oulu	540	537	400	526	0	533	309	449
Turku	150	134	143	154	533	0	271	194
Jyväskylä	235	234	131	220	309	271	0	140
Lahti	99	103	116	84	449	194	140	0



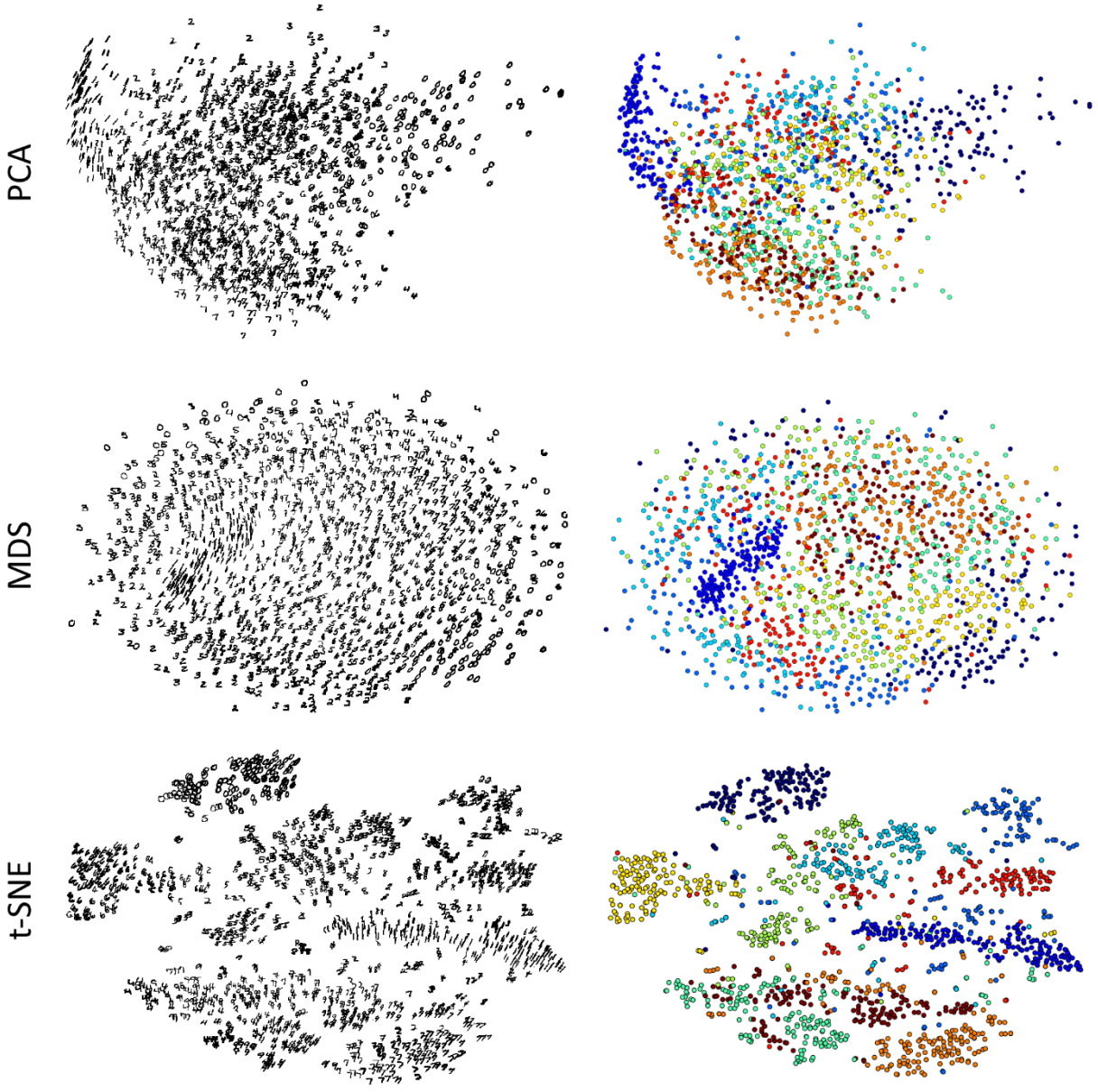
Kuva 5. Sammon-mapping-algoritmin tuottama kaksiulotteinen lopputulos Suomen kaupunkien sijainneista.

t-SNE on vuonna 2008 kehitetty epälineaarinen datan visualisointi ja dimensionaalisuuden vähentämismenetelmä. Se pyrkii säilyttämään sekä datan lokaalin rakenteen eli samanlaiset näytteet alkuperäisessä data-avaruudessa ovat muunnoksen jälkeen mahdollisimman lähellä toisiaan, että datan globaalin rakenteen eli erilaiset näytteet alkuperäisessä data-avaruudessa ovat muunnoksen jälkeen mahdollisimman erillään toisistaan. Epälineaarisilla dimensionaalisuuden vähentämismenetelmillä, kuten t-SNE:llä tai multidimensioskaalauksella on kyky löytää datan mahdollisia epälineaarisia riippuvuussuhteita, toisin kuin lineaarisilla dimensionaalisuuden vähentämismenetelmillä, kuten pääkomponenttianalyysilla.

t-SNE-algoritmi muodostaa aluksi todennäköisyystiheysjakauman jokaiselle näytteelle korkeassa ulottuvuudessa siten, että samanlaiset näytteet saavat mahdollisimman suuren arvon ja erilaiset näytteet mahdollisimman pienen arvon. Seuraavaksi t-SNE-algoritmi määrittää samankaltaisen todennäköisyystiheysjakauman matalammassa ulottuvuudessa sattumanvaraisesti sijoitetuille näytteille. Lopuksi algoritmi minimoi iteratiivisesti Kullback-Leibler divergenssin, jonka seurauksena todennäköisyystiheysjakaumat vastaavat toisiaan ulottuvuuksien välillä.

t-SNE-algoritmia voidaan käyttää käytännössä minkä tahansa korkeadimensioisen datan analysoimiseen mutta erityisesti sitä käytetään lääketieteelliseen kuvantamiseen, kasvonpiirteiden tunnistukseen sekä puheen tunnistukseen.

Kuvassa 6 on esitetty PCA, MDS ja t-SNE algoritmien tuottamat lopputulokset MNIST-datajoukolle. Kuvasta huomataan, kuinka epälineaariset dimensionaalisuuden vähentämismenetelmät t-SNE ja multidimensioskaalaus onnistuvat erottelemaan ryhmiä paremmin kuin lineaarinen dimensionaalisuuden vähentämismenetelmä pääkomponenttianalyysi.



Kuva 6. Dimensionaalisuuden vähentäminen PCA, MDS ja t-SNE algoritmeilla MNIST-datajoukolle.

Esitetään vielä lopuksi taulukossa 2 yleisimpien dimensionaalisuuden vähentämismenetelmien tunnuspiirteet [2].

Taulukko 2. Yleisimpien dimensionaalisuuden vähentämismenetelmien tunnuspiirteet.

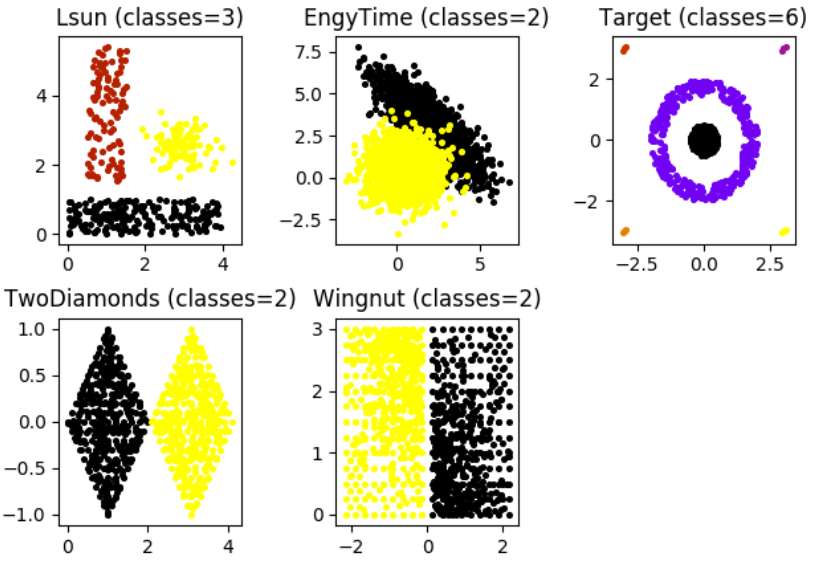
Algoritmi	*Lineaarinen menetelmä?*	*Kompleksisuus*	*Dataa käsitellään*	*Käytetäänkö kustannus-funktiota?*	*Algoritmin luonne*
PCA	kyllä	O(N)	vektoreina	kyllä	varianssin säilytys
LDA	kyllä	O(N)	vektoreina	kyllä	luokittelutuloksen maksimointi
MDS	ei	$O(N^2)$	etäisyyksinä	kyllä	etäisyyksien säilytys
Isomap	ei	$O(N^3)$	etäisyyksinä	kyllä	monikerran muunnos
LLE	ei	$O(N^2)$	vektoreina	kyllä	monikerran muunnos
t-SNE	ei	$O(N^2)$	etäisyyksinä	kyllä	naapuritiedon muunnos
umap	ei	$O(N^2)$	etäisyyksinä	kyllä	ristientropian minimointi
SOM	ei	O(N)	vektoreina	ei	topologian muunnos

Tehdessäsi harjoitusta omalla tietokoneella, asenna tarvittavat python-kirjastot tietokoneellesi seuraavasti:

1. Asenna miniconda (https://conda.io/en/latest/miniconda.html).
2. Mene komentoriville ja aja komento: conda install -c conda-forge umap-learn
   (Huom. Jos et lisää minicondan asennuskansiota ja /Scripts kansiota ympäristömuuttujiin, aja kyseinen komento sekä tehtävä 2 asentuneelta Anaconda prompt komentoriviltä.)
3. Asenna loput tarvittavat kirjastot komennolla: pip install numpy matplotlib scikit-learn scikit-image

Tehtävä 1 (2.5p)

Tehtäväsi on klusteroida viittä erilaista kaksiulotteista datajoukkoa kolmella eri klusterointimenetelmällä ***K-means***, ***DBSCAN*** ja ***kokoava hierarkkinen klusterointi*** ja tutkia, miten algoritmit toimivat eri datajoukoille. Kuvassa 7 on esitelty tehtävässä käytettävät datajoukot ja taulukossa 3 on listattu niiden tärkeimmät ominaisuudet [3].



Kuva 7. Tehtävässä käytettävät alkuperäiset datajoukot.

Taulukko 3. Tehtävässä käytettävien datajoukkojen tärkeimmät ominaisuudet.

Nimi	Datajoukon kuvaus	Klustereiden lukumäärä	Näytteiden lukumäärä
Lsun	Kolme erimuotoista klusteria, joiden näytteiden tiheys klustereiden sisällä vaihtelee jonkin verran.	3	400
EngyTime	Kahdella eri normaalijakaumalla toteutettu data. Eri klustereiden näytteet ovat osittain päällekkäin.	2	4096
Target	Neljä klustereista on selvästi erillään muista klustereista. Kahdesta jäljelle jäävästä klusterista toinen on toisen sisässä.	6	770
TwoDiamonds	Kaksi vierekkäin olevaa timantinmuotoista klusteria, joiden näytteiden tiheys klustereiden sisällä on vakio.	2	800
Wingnut	Molempien klustereiden näytteiden tiheys vaihtelee reilusti niiden sisällä ja klustereiden välinen raja on lähes huomaamaton.	2	1070

Muokkaa tiedostoa nimeltä **clustering.py**. Datajoukot löytyvät tehtäväkansiosta excel-tiedostoina, joista ne muunnetaan python-tiedostossa listoiksi. Tässä tehtävässä *main()* funktio on sinulle valmiiksi annettuna ja sinun tulee toteuttaa seuraavien klusteroinnin suorittavien apufunktioiden sisältö:

Kmeans(X, num_clusters):

1. Toteuta K-means klusterointi datajoukolle X.

```
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=xxx).fit(X)
```

missä antamalla parametri init='k-means++' sijoitetaan klusterikeskipisteet aluksi yhtä etäälle ja mahdollisimman kauas toisistaan, n_clusters viittaa klustereiden lukumäärään ja n_init toteuttaa klusteroinnin xxx kertaa ja valitsee lopuksi parhaan lopputuloksen. Anna n_clusters arvoksi funktion Kmeans() argumenttina käytetty muuttuja num_clusters ja n_init arvoksi esimerkiksi 20.

2. Tallenna klusteroinnin lopputulos datajoukolle X muuttujaan *labels*.

```
labels = kmeans.labels_
```

3. Funktion tulee palauttaa klusteroinnin lopputulos datajoukolle X eli muuttuja *labels*.

Dbscan(X, epsilon, minimum_samples):

1. Toteuta DBSCAN klusterointi datajoukolle X.

```
dbscan = DBSCAN(eps=epsilon, min_samples=minimum_samples).fit(X)
```

missä parametri eps kertoo näytteiden välisen minimietäisyyden, jolla kyseinen näyte vielä klusteroidaan tiettyyn klusteriin ja min_samples kertoo, montako näytettä minimissään tulee olla yhdessä klusterissa. Anna molempien parametrien arvoksi funktion Dbscan() argumentteina käytetyt muuttujat epsilon ja minimum_samples.

2. Tallenna klusteroinnin lopputulos datajoukolle X muuttujaan *labels*.

```
labels = dbscan.labels_
```

3. Sklearn-kirjaston DBSCAN funktio antaa klustereiden ulkopuolisille näytteille (outliers) automaattisesti ennustetuksi klusterin arvoksi -1, mutta haluamme tallentaa ulkopuolisten näytteiden klusterin positiivisena kokonaislukuna, joten -1 arvo tulee korvata seuraavalla vapaana olevalla positiivisella kokonaisluvulla.

```
labels = [len(set(labels)) if i==-1 else i for i in labels]
```

4. Funktion tulee palauttaa klusteroinnin lopputulos datajoukolle X eli muuttuja *labels*.

Agglomerative_clustering(X, num_clusters):

1. Toteuta kokoava hierakkinen klusterointi datajoukolle X.

```
model = AgglomerativeClustering(linkage='ward', n_clusters=num_clusters).fit(X)
```

missä antamalla parametri linkage='ward' yhdistetään näytteet minimoiden niiden välinen varianssi ja n_clusters viittaa klustereiden lukumäärään. Anna n_clusters arvoksi funktion Agglomerative_clustering() argumenttina käytetty muuttuja num_clusters.

2. Tallenna klusteroinnin lopputulos datajoukolle X muuttujaan *labels*.

```
labels = model.labels_
```

3. Funktion tulee palauttaa klusteroinnin lopputulos datajoukolle X eli muuttuja *labels*.

Kun saat toteutettua apufunktiot, ohjelma luo 4 ikkunaa, jotka sisältävät alkuperäiset datajoukot sekä klusteroinnin lopputulokset algoritmeilla K-means, DBSCAN ja kokoava hierarkkinen klusterointi. Kuvaajien x-akselin alapuolelle printataan myös klusterointien luokittelutarkkuudet, jotka kertovat, moniko näytteistä klusteroitiin oikein verrattuna alkuperäisten datajoukkojen luokkatietoihin. Kuvaajat tallennetaan myös pdf-tiedostoon nimeltä **clustering_result.pdf**.

Tämän jälkeen vastaa seuraaviin kysymyksiin:

KYSYMYS1: Mikä tai mitkä algoritmeista toimivat parhaiten kullekin datajoukolle kuvaajien ja luokittelutarkkuuksien perusteella?

KYSYMYS2: Minkä takia K-means algoritmi ei onnistu klusteroimaan oikein datajoukkoa Lsun sekä DBSCAN ei onnistu klusteroimaan oikein datajoukkoa Wingnut?

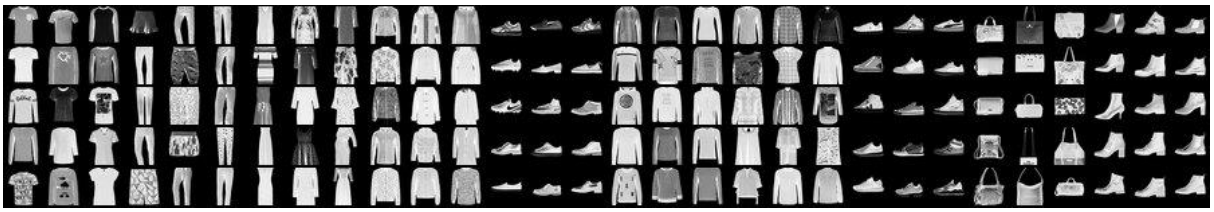
Kirjoita tehtävästä 1 lyhyt raportti, jossa vastaat kysymyksiin KYSYMYS1 ja KYSYMYS2. Palauta raportin lisäksi luotu pdf-tiedosto clustering_result.pdf sekä muokkaamasi python-tiedosto clustering.py.

Tehtävä 2 (2.5p)

Tässä tehtävässä sinun tulee pienentää kolmen eri datajoukon dimensionaalisuutta PCA, MDS, LLE, Isomap, t-SNE ja umap algoritmeilla niin, että datat voidaan esittää normalisoiduissa kaksiulotteisissa koordinaatistoissa. Lopuksi pystytään arvioimaan dimensionaalisuuden vähentämisen lopputulosta ja algoritmien suorituskykyä.

Tehtävässä käytetään seuraavia datajoukkoja:

FashionMNIST sisältää MNIST-datajoukon tapaan yhteensä 70 000 näytettä ja 10 luokkaa [4]. Näytteet ovat 28x28 pikselin kokoisia mustavalkoisia kuvia vaatekappaleista, kuten t-paidoista, housuista, laukuista, sandaaleista, puseroista, takeista jne. Alkuperäisestä fashionMNIST-datajoukosta on valittu balansoitu 700 näytteen joukko, ettei algoritmien laskentaan kulu liian kauan aikaa. Kuvassa 8 on esitetty esimerkkikuvia datajoukon fashionMNIST sisällöstä.



Kuva 8. Esimerkkikuvia datajoukon fashionMNIST sisällöstä.

COIL20-datajoukko sisältää yhteensä 1440 mustavalkoista kuvaa ja yksittäisen kuvan koko on 128x128 pikseliä [5]. Kuvat on otettu 20 esineestä eri kuvakulmista mustaa taustaa vasten. Myös COIL20-datajoukon kohdalla algoritmien laskenta-aikaa on pyritty pienentämään skaalaamalla alkuperäisen datajoukon kuvat kokoon 32x32 pikseliä. Kuvassa 9 on esitetty esimerkkikuvia datajoukon COIL20 sisällöstä.

Kuva 9. Esimerkkikuvia datajoukon COIL20 sisällöstä.

Wines-datajoukko koostuu kolmesta italialaisesta viinilajikkeesta ja sisältää yhteensä 178 näytettä. Piirteinä datajoukolle käytetään 13 viinin ominaisuutta. Tämä datajoukko on jo ennestään tuttu harjoituksesta 2 (Regressio).

Muokkaa tiedostoa nimeltä **dimensionalreduction.py**. Tässäkin tehtävässä *main()* funktio on sinulle annettuna valmiiksi ja sinun tulee toteuttaa seuraavien dimensionaalisuuden vähentämiseen käytettyjen apufunktioiden sisältö:

MinMax_normalization(data):

Tämä funktio normalisoi datan näytteet x-akselin ja y-akselin välille 0-1.

Käydään aluksi läpi, miten kyseinen normalisointi toteutetaan. Kaavassa 1 on esitetty yhtälö näytteiden skaalaukselle välille 0-1.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \quad , \tag{1}$$

missä x_{min} on alkuperäisen datan pienin arvo, x_{max} on alkuperäisen datan suurin arvo ja x_{scaled} on lopputuloksena saatu normalisoitu arvo.

Otetaan esimerkkinä listan A = [115,140,215] normalisointi välille 0-1. Sijoittamalla listan arvot kaavaan 1 (x_{max}=215 ja x_{min}=115) saadaan normalisoiduksi listaksi A_{scaled}= [0, 0.25, 1].

1. Selvitetään aluksi numpylla dimensionaalisuudeltaan vähennettyjen näytteiden minimi- ja maksimiarvot molemmille akseleille.

```
X_min, X_max = np.min(data,0), np.max(data,0)
```

2. Lasketaan tämän jälkeen normalisoidut arvot kaavan 1 mukaisesti.

```
X_scaled = (data - X_min)/(X_max - X_min)
```

3. Funktion tulee palauttaa normalisoidut arvot eli lista *X_scaled*.

Train_model(X_data)

Tässä funktiossa sinulle on jo annettu kuuden eri dimensionaalisuuden vähentämistekniikan toteutukset sekä alkuajanhetki *t0*. Funktion tulee palauttaa aika, joka kului käytetyn algoritmin laskentaan sekä dimensionaalisuudeltaan vähennetyt näytteet normalisoituna x-akselin ja y-akselin välille 0-1.

1. Ota heti käytetyn algoritmin suorittamisen jälkeen loppuajanhetki *t1* samalla tavalla kuin ajanhetki *t0* on otettu.

2. Laske tämän jälkeen algoritmin suorittamiseen kulunut aika, joka on sama kuin *t1 – t0*.

3. Tämän jälkeen normalisoi dimensionaalisuudeltaan vähennetyt näytteet x-akselin ja y-akselin välille 0-1 funktion **MinMax_normalization()** avulla.

```
X_scaled = MinMax_normalization(X_reduced_data)
```

4. Funktion tulee palauttaa normalisoidut arvot sekä dimensionaalisuuden vähentämiseen kuluneen ajan.

```
return X_scaled, deltatime
```

Kun saat toteutettua apufunktiot, ohjelma luo datajoukkojen visualisoinneista pdf-tiedostot **reduction_wines.pdf**, **reduction_coil20.pdf** ja **reduction_fashionmnist.pdf**. Pdf-tiedostoista löytyy tehtävässä käytettyjen kolmen datajoukon visualisoinnit toteutettuna kuudella eri dimensionaalisuuden vähentämismenetelmällä. Lisäksi datajoukoille COIL20 ja fashionMNIST on luotu yksittäiset kuvaajat eri algoritmien lopputuloksista, joihin on lisätty kuvabokseja havainnollistamaan, minkä näköiset näytteet sijoittuvat eri kohtiin kaksiulotteista avaruutta. Datajoukon wines näytteitä ei ole mahdollista visualisoida kuvin. Kuvaajien oikeasta reunasta löytyy selite näytteiden luokista ja x-akselin alapuolelta dimensionaalisuuden vähentämiseen käytetty aika sekä lopputuloksen arvioitu luokittelutarkkuus yksinkertaisella knn-luokittelijalla. Dimensionaalisuuden vähentämisprosessi kestää tietokoneesta riippuen noin 5-10 minuuttia.

Huomaa, että kuvaajissa luokkatiedot visualisoituna eri värein on lisätty vasta vähentämisen jälkeen havainnollistamaan, mihin luokkiin näytteet oikeasti kuuluisivat. Luokkatietoa ei siis ole vähentämismenetelmien käytössä.

Tämän jälkeen vastaa seuraaviin kysymyksiin:

KYSYMYS1: Tarkastellaan aluksi wines-datajoukkoa. Millä dimensionaalisuuden vähentämismenetelmällä saavutetaan tälle datajoukolle paras lopputulos mahdollisimman nopeassa ajassa?

KYSYMYS2: Tarkastellaan seuraavaksi COIL20 datajoukkoa. Mikä tai mitkä algoritmeista onnistuvat mielestäsi parhaiten säilyttämään sekä datan lokaalit piirteet (samannäköiset näytteet ovat mahdollisimman lähellä toisiaan) mutta toisaalta myös datan globaalit piirteet (erinäköiset näytteet omissa ryhmissään ovat erillään toisista ryhmistä)?

KYSYMYS3: Onko datajoukon fashionMNIST muuttujien väliset riippuvuussuhteet täysin lineaarisesti selitettävissä suoritettujen dimensionaalisuuden vähentämismenetelmien tuottamien lopputulosten perusteella?

Kirjoita tehtävästä 2 lyhyt raportti, jossa vastaat kysymyksiin KYSYMYS1, KYSYMYS2 ja KYSYMYS3. Palauta raportin lisäksi pdf-tiedostot reduction_wines.pdf, reduction_coil20.pdf ja reduction_fashionmnist.pdf sekä muokkaamasi python-tiedosto dimensionalreduction.py.

Lähteet

[1] Duda R., Hart P. & Stork D. (2012) Pattern classification. John Wiley & Sons.

[2] Gisbrecht A. & Hammer B. (2015) Data visualization by nonlinear dimensionality reduction. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 5(2), pp. 51-73.

[3] Ultsch A. (2005) Clustering with SOM: U*C, In Proc. Workshop on Self-Organizing Maps, Paris, France, pp. 75-82.

[4] Xiao H., Rasul K. & Vollgraf R. (2017) Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

[5] Nene S., Nayar S. & Murase H. (1996) Columbia Object Image Library.

Harjoituksen 4 oppimistavoitteet

- ymmärtää eri klusterointimenetelmien toimintaperiaatteet ja hahmottaa, mistä niiden tuottamat lopputulokset mahdollisesti johtuvat.
- osaa vertailla klusterointimenetelmien suorituskykyä eri datajoukoille.
- osaa vertailla dimensionaalisuuden vähentämismenetelmien hyvyyttä eri datajoukoille suorituskyvyn, visuaalisen lopputuloksen sekä algoritmin käyttämän ajan perusteella.

Palauta

Palauta pyydetyt tiedostot ja raportit pakattuna tiedostona (.zip tai .rar), jossa on kaksi erillistä kansiota molemmille tehtäville (esim. tehtävä1 ja tehtävä2) Optiman palautuslaatikkoon Harjoitus 4 **22.4.2019 klo 23:59** mennessä. Tästä harjoituksesta on mahdollisuus tienata 5 pistettä (2.5p + 2.5p). Voit antaa palautetta tästä harjoituksesta liittämällä raporttiin erillisen osion palautetta varten. Harjoituksia tullaan kehittämään palautteen pohjalta tuleville vuosille vastaamaan paremmin oppimistavoitteita.