TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND
Arvutiteaduse instituut
Informaatika õppekava

Priit Danelson

Klasteranalüüsi meetodite uurimine visuaalsete andmete abil

Bakalaureusetöö (9 EAP)

Juhendaja: J. Vilo, PhD

Tartu 2015

Klasteranalüüsi meetodite uurimine visuaalsete andmete abil

Lühikokkuvõte:

Klasteranalüüs on laia kasutusvaldkonnaga andmeanalüüsi tehnika, mille rakendamiseks on olemas mitu erinevat algoritmi. Käesoleva töö eesmärk on anda ülevaade kolme levinuma klasteranalüüsi meetodi tööpõhimõtetest ja eripäradest, rakendades hierarhilise klasterdamise, k-keskmiste klasterdamise ja Kohoneni võrgu algoritme näidisandmestiku peal. Lisaks algoritmide tööpõhimõtetele on kirjeldatud ka põhjus, miks näidisandmestikuks on valitud visuaalsed andmed ehk pildid ning kuidas on implementeeritud klasteranalüüsi meetodite rakendamiseks kasutatav skript. Töö sisaldab ka skripti rakendamisel saadud klasterduste analüüsi.

Võtmesõnad:

klasteranalüüs, hierarhiline klasterdamine, k-keskmiste klasterdamine, Kohoneni võrk, kaugusmõõt

A Study of Clustering Methods Using Visual Data

Abstract:

Cluster analysis is a widely used data analysis technique that can be applied by using several different algorithms. This thesis aims to give an overview of the working principles and specifics of the three most commonly used cluster analysis methods by applying hierarchical clustering, k-means clustering and self-organizing map algorithms on sample data. In addition to the description of working principles of the clustering algorithms, there is also a description of how the script used for applying the clustering methods is implemented and an explanation for why visual data or pictures are chosen as the sample data. The thesis also includes an analysis of the clustering results produced by the script.

Keywords:

cluster analysis, hierarchical clustering, k-means clustering, self-organizing map, distance metric

Sisukord

1. Sissejuhatus

Klasteranalüüs ehk objektide teatud tunnuste alusel grupeerimine on laialt kasutatav tehnika, mis leiab rakendust nii kommerts- kui ka teadusvaldkonnas. Kommertsvaldkonnas kasutavad klasteranalüüsi näiteks kindlustusfirmad, kes teatud omaduste põhjal inimesi riskigruppidesse jaotavad, või turundusfirmad, kes klassifitseerivad inimesi eesmärgiga neile kõige sobivamat reklaami pakkuda. Teadusvaldkonnas on klasteranalüüs tähtsal kohal näiteks bioinformaatikas, kus seda kasutatakse geeniekspressioonide uurimisel, ning sotsiaalteadustes, kus see on levinud sotsiaalvõrgustike uurimise abivahend.

Klasteranalüüs on aga suhteliselt abstraktne üldnimetus, mitte konkreetne algoritm, ning seda on võimalik rakendada mitmel erineval viisil. Käesolev töö tutvustabki nendest levinumate meetodite ehk k-keskmiste klasterdamise, hierarhilise klasterdamise ja Kohoneni võrgu tööpõhimõtteid ning eripärasid. Samuti käsitleb käesolev töö andmete kujutamist klasteranalüüsiks sobival kujul, arvutuslikult kahe objekti vahelise sarnasuse leidmist ning klasteranalüüsi tulemuste paremat visualiseerimist olukorras, kus lisaks objektide grupeerimisele on oluline leida ka korrektne järjestus nii gruppidesse kuuluvate objektide kui ka gruppide endi vahel.

Parim viis mingist meetodist ülevaate andmiseks on selle rakendamine ning klasteranalüüsi meetodite rakendamiseks on vaja andmeid, mida klasterdada. Käesolevas töös on nendeks andmeteks valitud pildifaili segamini aetud järjestusega piksliread. Kuna tavaliselt on pildi puhul naaberread üksteisega visuaalselt kõige sarnasemad, siis peaks veatu klasterduse korral naaberread sattuma alati samadesse klastritesse, mis võimaldab tulemust järjestamisalgoritmidega töödeldes taasluua esialgse pildi. Esialgse pildi, segamini aetud piksliridadega pildi ning klasterdamise tulemuseks saadud pildi omavaheline võrdlemine peaks seega hästi illustreerima klasterdamismeetodite mõju.

Töö teine peatükk sisaldab olulisemaid mõisteid, kirjeldades piksliridade klasteranalüüsiks sobivas formaadis kujutamist, arvutuslikult kahe pikslirea vahelise sarnasuse leidmist ning töös kasutatavate klasterdamise algoritmide ja klasterduse visualisatsiooni parandamiseks kasutatavate järjestamisalgoritmide tööpõhimõtteid. Samuti on teises peatükis ülevaade varasematest töödest, mis on tegelenud käesolevas töös kasutatavate klasterdamismeetodite uurimise või omavahel võrdlemisega. Kolmas peatükk sisaldab infot klasterdamiseks valitud andmete ning töös rakendatud algoritmide implementatsiooni kohta. Neljas peatükk sisaldab klasterdamise tulemusi erinevate meetodite puhul ning tulemuste analüüsi. Viimane ehk viies peatükk sisaldab kokkuvõtet tehtud tööst.

2. Mõisted ja taustainfo

2.1 Andmete kujutamine ja kaugusmõõt

Enne, kui on võimalik rääkida andmete sarnasuse järgi klasterdamisest või järjestamisest, on vaja otsustada, kuidas kasutatavaid andmeid kujutada ning kuidas defineerida kahe andmeobjekti vaheline sarnasus.

Enamasti kujutatakse selleks otstarbeks andmeid vektoritena vektorruumis. Selleks moodustatakse igast andmeobjektist, millel on *n* vaadeldavat omadust, *n*-liikmeline vektor, kus vektori iga element esindab ühte vaadeldavat omadust. Näiteks käesoleva töö puhul, kus uuritavateks andmeobjektideks on piksliread ja vaadeldavateks omadusteks piksliritta kuuluvate pikslite RGB väärtused, tähendab see kuju, kus iga rida on esindatud vektorina, mis koosneb 3*n* elemendist, kus *n* on pikslite arv reas.

Andmete kujutamine sellisel viisil võimaldab andmeobjektide vahelise sarnasuse defineerimiseks võtta kasutusele mõiste kaugusmõõt[LCL+04]. Olgu X kõigi vektorkujul olevate andmeobjektide hulk. Kaugusmõõduks võib sellisel juhul nimetada selliseid funktsioone $d : X \times X \rightarrow \mathbb{R}$, mis vastavad neljale järgnevale tingimusele:

1. $\forall x \in X : d(x,x) = 0$
2. $\forall x,y,z \in X : d(x,y) + d(y,z) \geq d(x,z)$
3. $\forall x,y \in X : d(x,y) = d(y,x)$
4. $\forall x,y \in X : x \neq y \rightarrow d(x,y) > 0$

Kahele vektorkujul olevale andmeobjektile rakendatud kaugusmõõdu tulemuse ehk kauguse abil saabki määrata andmeobjektide sarnasust. Mida suurem on kaugus, seda erinevamad on objektid, ning mida lähemal on saadud väärtus nullile, seda sarnasemad.

Eelnevas lõigus oli näha, et kaugusmõõdu definitsioon lubab kaugusmõõduks valida hulgaliselt erinevaid funktsioone. Ilmne on aga ka see, et erinevad kaugusmõõdud ei ole erinevate andmetüüpide puhul võrdse efektiivsusega. Arvestades seda, et objektide vahelise sarnasuse leidmine on andmete klasterdamisel või järjestamisel kesksel kohal, on õige kaugusmõõdu valik väga oluline.

Käesoleva töö puhul on vaja kaugusmõõtu, mis aitaks leida piksliridade vahelist visuaalset sarnasust. Üks võimalik kaugusmõõt on selleks Hamming'i kaugus [Hos12], mis on defineeritud kui kahe sama pikkusega vektori samadel positsioonidel asuvate, kuid üksteisest erinevate, elementide arv. Esmapilgul võib see tunduda sobiv, kuna mida rohkem on kahe pikslirea puhul erinevate värviväärtustega piksleid, seda erinevamad peaksid need read olema ka visuaalselt. Probleem on aga selles, et piltide puhul on enamasti piksliridade vahel sujuvad üleminekud, mis piksli RGB väärtustes kajastuvad, kuid mida silm ei pruugi tajuda. See tähendab, et ka visuaalselt väga sarnased piksliread võivad tegelikult üksteisest iga piksli RGB väärtuste poolest pisut erineda ning Hamming'i kaugus klassifitseerib need read vääralt üksteisest visuaalselt väga kaugeteks.

Hamming'i kaugusest sobilikum kaugusmõõt on eukleidiline kaugus, mis võrdleb samuti kahe vektori samadel positsioonidel asuvaid elemente. Erinevalt Hamming'i kaugusest on tulemuseks aga mitte üksteisest erinevate elementide arv, vaid nende erinevuste ruutude summa ruutjuur, ehk:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$, n – elementide arv vektoris

See tähendab, et eukleidiline kaugus ei arvesta ainult seda, kas vektorites samadel positsioonidel asuvad elemendid on erinevad, vaid ka seda kui palju nad erinevad, mis ongi enamasti

see, mida visuaalse sarnasuse leidmiseks on tarvis. Seetõttu on selles töös põhiliselt kasutusel just eukleidiline kaugusmõõt.

Leidub siiski ka erijuhte, kus eukleidiline kaugus hästi ei toimi. Eelkõige on sellisteks juhtudeks pildid, kus piksliread on nihkes (vt Joonis 1). Jooniselt on näha, et kui pildil kõrvuti asuvate piksliridade pikslite positsioone mõne piksli võrra nihutada, on tegemist sisuliselt identsete ridadega. Eukleidiline kaugus aga võrdleb ainult kohakuti olevaid piksleid, saades antud juhul väära tulemuse, et read on visuaalselt väga erinevad.

Joonis 1. Pilt nihkes piksliridadega.

Sellistel juhtudel võiks toimida paremini enamasti aegridade klasterdamiseks kasutatav kaugusmõõt nimega *Dynamic Time Warping (DTW)* [Sen08]. Tähistagu x esimest ja y teist vektorit, m ja n vastavate vektorite elementide arvu ning $d(x, y)$ kahe elemendi vahelist kaugust, milleks käesolevas töös on elementide vahe absoluutväärtus. *DTW* algoritm koostab kõigepealt $m \times n$ elemendiga kauguste maatriksi A, mille elementideks on mõlema vektori iga elemendi kaugused teise vektori kõigist elementidest, ehk $A_{ij} = d(x_i, y_j),\ i = 1 \dots m,\ j = 1 \dots n$. Seejärel otsib algoritm minimaalse kaugusega tee läbi koostatud maatriksi A, alustades maatriksi elemendist A_{11} ning lõpetades maatriksi elemendi A_{mn} juures. Minimaalse kaugusega teeks loetakse seejuures elementide läbimise järjestust, milles läbitud elementide summa oli minimaalne ning seda summat nimetataksegi kahe vektori vaheliseks kauguseks.

Näiteks, olgu kaks juhuslikku nn nihkes elementidega vektorit $x = (8,12,10,16,24)$ ja $y = (12,10,16,24,8)$. Sellisel juhul on koostatud kauguste maatriksi (vt Joonis 2) põhjal vektorite x ja y vaheline *DTW* kaugus $4 + 0 + 0 + 0 + 0 + 16 = 20$.

	y_1	y_2	y_3	y_4	y_5
x_1	4	2	8	16	0
x_2	0	2	4	12	4
x_3	2	0	6	14	2
x_4	4	6	0	8	8
x_5	12	14	8	0	16

Joonis 2.Vektorite x ja y elementide kauguste maatriks ning seda läbiv lühim tee.

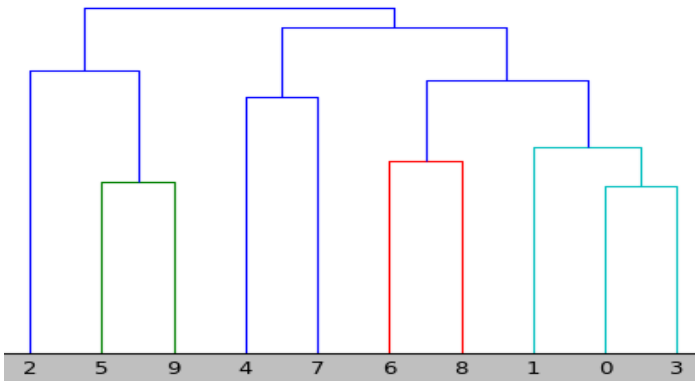
Selline lähenemine tagab selle, et sarnaseks peetakse ka nihkes olevaid piksliridu, kuid on samas arvutuslikult märksa kallim kui eukleidiline keskmine. Seega leiab *DTW* käesolevas töös kasutust ainult nendel spetsiifilistel juhtudel, kus eukleidiline kaugusmõõt ei sobi.

2.2 Klasteranalüüs

Klasteranalüüsi võib defineerida kui andmeobjektide sarnasuse alusel klastritesse ehk rühmadesse jagamist. Seejuures üritatakse saavutada olukorda, kus samadesse klastritesse kuuluvate objektide sarnasus on võimalikult suur ning erinevatesse klastritesse kuuluvate objektide sarnasus omakorda võimalikult väike. See kuulub nn järelevalveta õppimismeetodite hulka, otsides andmestikus peituvaid seoseid ja struktuure otsitavate struktuuride kohta eelinfot omamata. Klasteranalüüs on aga üldnimetus, mitte konkreetne algoritm, ning selle rakendamiseks on mitmeid erinevaid viise, mis sõltuvalt kasutatavatest andmetest omavad erinevaid eeliseid ja puudujääke. Selles töös uuritakse piksliridade klasterdamist ja saadud tulemuste visualiseerimist kolme enim tuntud klasterdamismeetodi puhul.

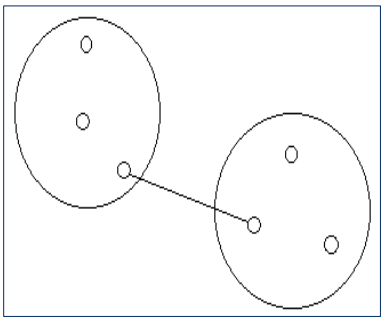
2.2.1 Hierarhiline klasterdamine

Esimene uuritavatest meetoditest on hierarhiline klasterdamine [HTF09], mis jaguneb omakorda veel kaheks alamliigiks: nn ülevalt alla meetod (*divisive hierarchical clustering*) ja nn alt üles meetod (*agglomerative hierarchical clustering*). Ülevalt alla lähenemise puhul moodustatakse kõigepealt kasutatavatest andmetest üks suur klaster, mida igal sammul vastavalt jagunemise kriteeriumitele aina väiksemateks klastriteks jagatakse. Algoritm jätkab tööd seni, kuni kõik andmeobjektid on sattunud eraldi klastritesse. Alt üles lähenemise puhul käib asi vastupidi. Iga andmeobjekt määratakse alguses eraldi klastrisse ning seejärel ühendatakse igal sammul vastavalt ühendamise kriteeriumitele kaks kõige sobivamat klastrit. Algoritmi töö loetakse lõppenuks hetkel, mil kõik andmeobjektid on kogunenud samasse klastrisse. Mõlemal juhul on tulemuseks hierarhiline struktuur nimega dendrogramm (vt Joonis 3), mis kujutab olenevalt meetodist kas igal sammul toimunud klastrite liitumisi või jagunemisi. Seejuures on selge, et hierarhilise klasterdamise puhul on olenevalt meetodist väga suur roll kas klastrite ühendamise või jagamise kriteeriumitel. Kuna käesolev töö rakendab ainult hierarhilise klasterdamise alt üles meetodit, siis tuuakse välja ainult kolm levinuimat klastrite ühendamise kriteeriumi.

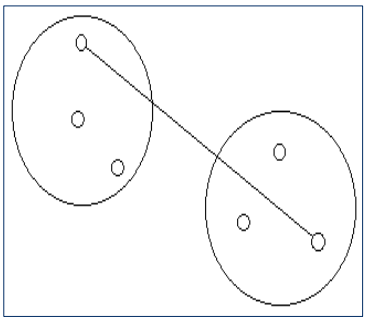


Joonis 3. Kümne vektori klasterdamisel tekkinud dendrogramm.

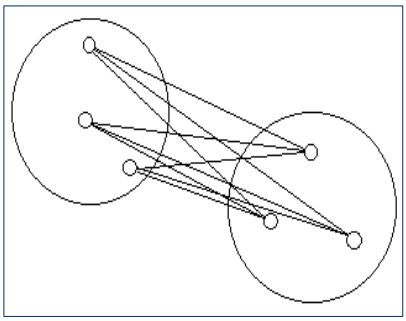
Kõigi järgnevate kriteeriumite järgi ühendatakse klastrid, mille vaheline kaugus on väikseim, erinevus on ainult selles, kuidas kriteeriumid defineerivad klastrite vahelist kaugust. Esimene kriteerium on tuntud kui minimaalse kauguse kriteerium [HTF09]. Selle järgi on kahe klastri vaheline kaugus nende klastrite elementide vaheline minimaalne kaugus (vt Joonis 4.1). Teine on maksimaalse kauguse kriteerium [HTF09], mis on esimesega analoogiline, ainult klastrite elementide vahel otsitakse väikseima kauguse asemel suurimat kaugust (vt Joonis 4.2). Kolmas on tuntud kui keskmise kauguse kriteerium [HTF09], kus klastrite vaheliseks kauguseks võetakse keskmine väärtus klastrite elementide paarikaupa kaugustest (vt Joonis 4.3).



Joonis 4.1.
Minimaalne kaugus.



Joonis 4.2.
Maksimaalne kaugus.



Joonis 4.3.
Keskmine kaugus.

Eelpool nimetatud kriteeriumitest on piksliridade hierarhilisel klasterdamisel intuitiivselt võttes efektiivseim minimaalne kaugus. Põhjuseks see, et visuaalselt korrektse pildi saamiseks on vaja, et tekkinud puustruktuuris lehtedena kujutatud piksliread asuksid õiges järjekorras ehk naaberread oleksid kõrvuti. See eeldab aga, et ühendatakse alati klastreid, mis sisaldavad kindlasti naaberridu, mis on eeldatavalt üksteisest ka väikseima kaugusega, ehk rahuldavad täpselt minimaalse kauguse kriteeriumi. Siin ilmneb aga ka hierarhilise klasterdamise nõrkus antud ülesande puhul. Nimelt ei taga isegi alati ainult naaberridasid sisaldavate klastrite ühendamine õiget järjestust. Põhjuseks see, et klastrite ühendamisel ei ole määratud kumb klaster peaks jääma puustruktuuri vasakuks ja kumb paremaks haruks. Kui klasterdatavaid liikmeid on n tükki, siis on aga selliseid ühinemiskohti alati $n-1$ ja võimalikke erinevaid lehtede järjestusi 2^{n-1}. Seepärast on käesolevas töös visuaalselt parema tulemuse saavutamiseks lisaks tavalisele hierarhilisele klasterdamisele kasutusel ka hierarhiline klasterdamine koos alampuude ümberjärjestamisega, milles kasutatavat järjestusalgoritmi kirjeldatakse täpsemalt peatükis 2.3.2.

2.2.2 K-keskmiste klasterdamine

Teine uuritav klasterdamismeetod on k-keskmiste klasterdamine[RS10], mille nimes olev *k* tuleb meetodile etteantavast tekkivate klastrite arvust. Algoritm on väga laialt levinud eelkõige oma lihtsuse tõttu. Kõigepealt valitakse *k* juhuslikku erinevate väärtustega objekti, millest saavad klastrite keskmed ehk tsentroidid. Seejärel määratakse iga andmeobjekt klastrisse, mille tsendroidi kaugus objektist on kõige väiksem. Kui kõik objektid on klastritesse jagatud, arvutatakse tsentroidid vastavalt klastrite uutele liikmetele ümber, määrates tsentroidide uuteks väärtusteks klastrite liikmete keskmised. Algoritm lõpetab töö, kui tsendroidide väärtused ei muutunud, vastasel juhul alustab algoritm uuesti andmeobjektide klastritesse ümberjagamist.

Selle algoritmiga kvaliteetse klasterduse saamiseks on kõige olulisem valida korrektne klastrite arv *k*. Klastrite arvu leidmine võib aga andmestiku kohta eelinfot omamata olla võrdlemisi keeruline. Üks võimalustest on proovida algoritmi erinevate *k* väärtustega ning vastavalt klasterduste statistilistele kvaliteedinäitajatele valida neist välja parim. Selleks otstarbeks sobivad statistilised näitajad on näiteks Dunn'i indeks, Duda ja Hart'i indeks, Davis-Bouldin'i indeks ja Silhouette'i indeks[MMM13]. Piksliridade klasterdamise omapära on aga just selles, et klasterduse kvaliteedi hindamiseks statistilisi näitajaid vaja ei ole ning piisab ainult visuaalsest vaatlusest ehk parim klasterdus on tulemus, mis meenutab välimuselt enim originaalset pilti.

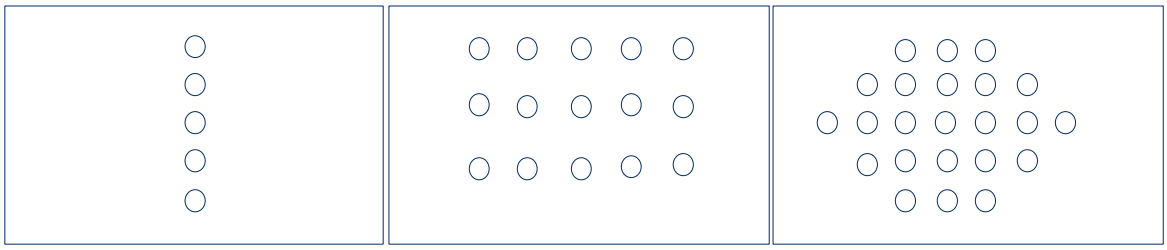
K-keskmiste algoritm on piksliridade klasterdamiseks uuritavatest meetoditest oma olemuselt kõige vähem sobiv. Seda põhjusel, et piksliridadest pildi loomine nõuab lisaks sarnaste ridade grupeerimisele ka nende järjestamist ning tavaline k-keskmiste algoritm ei järjesta

omavahel ei klastritesse kuuluvaid piksliridu ega ka klastreid ennast. Selle puuduse likvideerimiseks on käesolevas töös tavapärase k-keskmiste algoritmi kõrval kasutusel ka täiendatud k-keskmiste algoritm, mis rakendab ahne järjestamise algoritmi, millest kirjutatakse täpsemalt peatükis 2.3.1, kasutades seda nii samadesse klastritesse kuuluvate elementide kui ka klastrite endi omavaheliseks järjestamiseks.

2.2.3 Kohoneni võrk

Kolmas uuritav klasterdamismeetod on Kohoneni võrk[Koh90], mis on inglise keeles tuntud ka kui *Self-Organizing Map* või lihtsalt *SOM*. Algoritmi idee seisneb iseõppiva nn neuronite võrgu koostamises, mis kohandub vastavalt sellele antud sisenditele, andes tulemuseks ruumiliselt organiseeritud klasterduse, kus üksteisega ei sarnane mitte ainult samas klastris olevad objektid, vaid ka tekkinud võrgustikus lähestikku asuvad klastrid.

Selleks koostatakse kõigepealt n neuroniga võrk ehk paigutatakse neuronid ühe- või kahedimensioonilises ruumis soovitud kujundina, millest levinuimad on ristkülik ja kuusnurk, kuid piiranguid otseselt ei ole (vt Joonis 5). Lisaks seotakse iga neuroniga sisendandmetega võrdsete dimensioonidega vektor, mille algväärtus võib olla juhuslik, oluline on ainult see, et erinevate neuronitega seotud vektorid oleksid erinevad. Järgneb võrgu treenimisfaas, kus sisendiks on klasterdatavad andmeobjektid.



Joonis 5. Kohoneni võrguks sobivad neuronite asetused.

Treenimisfaasis leitakse iga sisendi puhul sellele võrgust nii sarnaseima vektoriga neuron kui ka naabrusfunktsiooni järgi selle naabrusesse kuuluvad neuronid. Naabrusfunktsioon on seejuures optimaalse globaalse järjestuse leidmiseks soovituslikult alguses väga lai ning ajas kahanev ehk hõlmab igal järgneval iteratsioonil aina vähem neuroneid. Järgnevalt rakendatakse kõigile leitud neuronite vektoritele kohandumisfunktsiooni, mis lähendab leitud neuronite vektoreid sisendile. Analoogiliselt naabrusfunktsioonile on ka kohandumisfunktsiooni mõju ajas kahanev. Iteratsioonide arv treenimisfaasis on määratud algoritmi parameetrina ning on soovituslikult vähemalt 500 korda suurem kui neuronite arv võrgus. Peale treenimisfaasi lõppu toimub sisendvektorite sidumisfaas, kus iga klasterdatav andmeobjekt määratakse neuroni juurde, mille vektor on sellega kõige sarnasem, moodustades lõpliku klasterduse.

Analoogiliselt k-keskmiste klasterdamisega on ka Kohoneni võrgu puhul tekkivate klastrite arv neuronite arvuga ette määratud ning optimaalse arvu leidmiseks kasutatakse antud töös samuti erinevate tulemuste visuaalset võrdlust. K-keskmiste meetodiga võrreldes on piksliridade pildiks klasterdamisel Kohoneni võrgul aga üks suur eelis, nimelt leitakse sellega ka pildi taastamiseks vajalik klastrite vaheline järjestus. Seega on erinevalt k-keskmiste klasterdamisest järjestusalgoritmi vaja ainult klastrite sisu järjestamiseks ning selleks on kasutusel samuti ahne järjestusalgoritm, millest kirjutatakse täpsemalt peatükis 2.3.1.

2.3 Klasterduse järjestamine

Piksliridadest klasterdamise teel korrektse pildi moodustamine ei ole puhas klasterdamisülesanne, vaid vajab lisaks piksliridade sarnasuse alusel grupeerimisele ka saadud gruppide

ning nendesse kuuluvate piksliridade järjestamist. Kuigi uuritavad klasterdamismeetodid on edukad grupeerimise osas, siis ainuke meetod, mis tegeleb ka järjestamisega, on Kohoneni võrk, ning sedagi ainult klastrite, mitte klastrite elementide tasemel. Järjestuse puudumine aga tähendab, et isegi kvaliteetne klasterdus ei pruugi välimuselt meenutada korrektset pilti. Seetõttu on klasterdamise tulemuse paremaks visualiseerimiseks käesolevas töös kasutusel ka kaks järjestusalgoritmi, mis klasterdamismeetodeid täiendavad.

2.3.1 Ahne järjestamine

Esimene kahest järjestamisalgoritmist on nn ahne järjestamine, mille abil piksliridade järjestamine toimub järgnevalt. Kõigepealt valitakse järjestatavate piksliridade seast välja juhuslik rida, millest saab järjekorra esimene liige. Seejärel otsitakse sellele kõigi ülejäänud piksliridade seast kõige sarnasem rida ning lisatakse see järjekorras kas esimeseks või teiseks. Tulenevalt sellest sammust on pilt lõpus kas õigetpidi või tagurpidi. Seejärel otsitakse kõigi ülejäänud piksliridade seast kõige sarnasem rida nii järjekorra esimesele kui ka viimasele reale. Kui sarnasus järjekorras esimese pikslirea ja sellele kõige sarnasema rea vahel oli suurem kui sarnasus järjekorra viimase pikslirea ja sellele kõige sarnasema rea vahel, siis lisatakse järjekorra algusesse esimesele pikslireale kõige sarnasem rida. Vastasel juhul lisatakse järjekorra lõppu pikslirida, mis oli kõige sarnasem viimasele reale. Antud sammu korratakse kuni kõik read on lisatud järjekorda. Sellisel viisil järjestatakse käesolevas töös klastrite sisu nii Kohoneni võrgu kui ka k-keskmiste klasterdamismeetodi tulemuste parandamiseks.

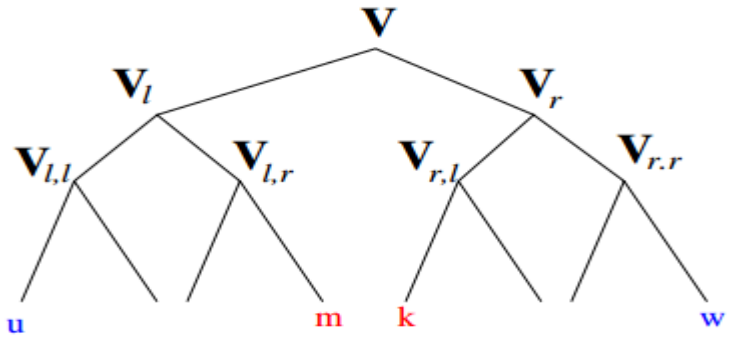
Näiteks k-keskmiste algoritmi puhul on aga lisaks klastrite sisule vaja järjestada ka klastreid. Klastrite ahne järjestamise algoritm on ridade järjestamisega üsna sarnane ning eeldab, et klastrite sisu on juba järjestatud. Järjestuse saavutamiseks valitakse järjekorra esimeseks liikmeks juhuslik klaster ning korratakse järgnevaid samme, kuni kõik klastrid on sattunud järjekorda. Otsitakse järjekorras mitteolevatest klastritest sarnaseim klaster nii järjekorra esimesele kui ka viimasele klastrile. Selleks, et leida sarnaseim klaster järjekorras esimesele klastrile, on vaja leida klaster, mille esimene või viimane pikslirida on sarnaseim esimese klastri esimesele pikslireale. Selleks, et leida sarnaseim klaster järjekorras viimasele klastrile, on vaja leida klaster, mille esimene või viimane pikslirida on sarnaseim viimase klastri viimasele pikslireale. Kui sarnasus järjekorras esimese klastri ja sellele kõige sarnasema klastri vahel oli suurem kui sarnasus järjekorra viimase klastri ja sellele kõige sarnasema klastri vahel, siis lisatakse järjekorra algusesse esimesele klastrile kõige sarnasem klaster. Vastasel juhul lisatakse järjekorra lõppu viimasele klastrile kõige sarnasem klaster. Seejuures toimub klastri järjekorda lisamisel vajadusel ka klastris olevate piksliridade järjestuse ümberpööramine nii, et klastri ja järjekorra liitumiskohta jäävate piksliridade sarnasus oleks maksimaalne.

2.3.2 Optimaalne lehtede järjestamine

Teiseks järjestusalgoritmiks on optimaalne lehtede järjestamine[BGJ01]. Algoritmi nimi tuleneb selle eesmärgist leida puustruktuuris selline alampuude järjestus, et puu lehtede omavahevaheliste kauguste summa oleks minimaalne. Käesolevas töös kasutatakse seda algoritmi hierarhilisel klasterdamisel tekkiva dendrogrammi lehtede järjestamiseks.

Optimaalne lehtede järjestamine on rekursiivne algoritm. Tähistagu v juurtippu ning sellest algavat puustruktuuri ja v_l ning v_r vastavalt tipu v vasakpoolset ning parempoolset alamharu. Selleks, et leida tipu v optimaalne lehtede järjestus, on vaja selle alamharusid v_l ja v_r pöörata nii, et v_l kõige parempoolsemaks leheks m ja v_r kõige vasakpoolsemaks leheks k jäävate elementide omavaheline kaugus $d(m, k)$ oleks minimaalne (vt Joonis 6). Seejuures

peavad aga v_l ja v_r olema enne juba optimaalselt järjestatud ehk sama algoritmi tuleb kõigepealt rakendada v_l ja v_r ning omakorda veel ka nende alamharude peal kuni lehtedeni välja. Niiviisi tekibki rekursioon, mille tulemuseks on optimaalselt järjestatud lehtedega puu.



Joonis 6. Optimaalselt järjestatud harud v_l ja v_r[BGJ01].

2.4 Seotud kirjandus

Klasteranalüüsi meetodeid on erinevates kontekstides uuritud üsna laialdaselt ning järgnevalt on välja toodud mõned huvitavamad käesoleva teksti temaatikaga seotud varasemad tööd.

Kohoneni võrku on põhjalikult käsitlenud T. Kohonen [Koh90], kes kirjeldab selle algoritmi erinevaid variatsioone ning demonstreerib kahedimensioonilise võrgu tööd, rakendades seda kõnetuvastusülesande lahendamiseks. Erinevalt Kohoneni artiklist käsitleb käesolev töö aga ühedimensioonilise võrgu rakendamist ning kasutab võrgu treenimiseks helisignaalide asemel pildifaili piksliridade RGB väärtusi.

Hierarhilisele klasterdamisele, täpsemalt selle aglomeratiivsele variandile, mida käsitleb ka käesolev töö, on keskendunud artikkel [Yag00]. Artiklis uuritakse erinevaid klastrite ühendamise kriteeriume ning nende mõju klasterdamise lõpptulemusele. Autor näitab, et kui klastrite omavahelise kauguse mõõduks valitakse nende liikmete minimaalne kaugus, siis kipuvad ühinema suured klastrid. Samas, kui klastrite omavahelise kauguse mõõduks valitakse nende liikmete maksimaalne kaugus, siis ühinevad pigem väikesed klastrid. Piksliridade klasterdamisel aga need karakteristikud suurt rolli ei mängi.

Kolmanda selles töös rakendatava klasterdamismeetodi ehk k-keskmiste algoritmi kohta on kirjutatud artiklis [MMM13]. Selles kirjeldati k-keskmiste algoritmi olemust ning toodi välja, kui oluline on k-keskmiste algoritmi puhul õige k ehk klastrite arvu valimine. Samuti pakuti välja uudne statistiline meetod optimaalse k määramiseks, kuid käesoleva töö puhul ei ole selle rakendamine ilmselt otstarbekas, kuna pildiandmete klasterdamise suur eelis on fakt, et korrektne klasterdus on enamasti visuaalselt hõlpsasti nähtav.

Klasteranalüüsi meetodite omavahelist võrdlust on varem läbi viinud J. Goddard *et al.* [GMMA00], kuid valdkonnaks oli nende puhul kõnetuvastus ning klasterdatavateks andmeteks kahe erineva häälikutekomplekti helinäidised. Klasterdamise efektiivsust hinnati selle põhjal, kui suure osa helinäidiste häälikuteks klassifitseerimisel algoritm eksis. Võrreldavate meetodite seas olid ka Kohoneni võrk ja k-keskmiste klasterdamine, kuid selget edu kumbki üksteise ees ei saavutanud, kuna ühe häälikutekomplekti puhul oli mõnevõrra täpsem Kohoneni võrk ja teisel juhul vastupidi. Siit järeldub, et meetodite efektiivsus on sõltuv

mitte ainult ülesande tüübist, vaid ka konkreetsetest sisendandmetest ning ka piksliridade klasterdamisel on vaja uurida mitmeid erinevaid pilditüüpe.

Klasteranalüüsi meetodeid on võrreldud ka artiklis [BMG11]. Seal kasutati klasteranalüüsi eesmärgiga klassifitseerida elektritarbijaid nende voolu tarbimise harjumuste alusel. Klasterdatavate andmete saamiseks jagati ööpäev 15-minutilisteks intervallideks ning iga intervalli jooksul tarbija poolt kasutatud voolu hulk oli üks andmepunkt vektoril, mille liikmete arv oli kokku 96. Meetodite efektiivsuse võrdlusesse olid kaasatud ka kõik käesolevas töös kasutatavad klasterdamise algoritmid. Meetodite rakendamisel saadud klastrite kvaliteedi mõõdikutena kasutati klastrite kompaktsust ehk keskmist klastri liikmete kaugust klastri keskpunktist ning klastrite vahelist erinevust ehk klastrite kaugust üksteisest. Antud juhul osutus edukaimaks hierarhiline klasterdamine ning kõige kehvemaks Kohoneni võrk. Väga põhjalikke järeldusi sellest siiski teha ei saa, sest klasterdatavate andmete hulk oli üsna väike.

Eelneva tööga analoogiline oli uurimus, mis võrdles samuti erinevate klasterdamise meetodite efektiivsust elektritarbijate klassifitseerimisel [CNP06]. Ka selles töös kasutati muude algoritmide seas kolme käesolevas töös uuritavat klasterdamise meetodit. Samuti oli samalaadne, kuid mahukam, kasutatav andmestik. Artikli tulemused kinnitasid eelneva töö järeldusi, leides, et efektiivseim oli neist kolmest meetodist hierarhiline klasterdamine. Seega võib nende kahe töö põhjal järeldada, et seda tüüpi ülesannete jaoks on parim lahendus hierarhiline klasterdamine. See järeldus ei pruugi aga kehtida teist tüüpi ülesannete puhul, eriti kuna elektritarbijate klassifitseerimisel on spetsiifilisi kitsendusi. Näiteks on ärilistel põhjustel vaadeldud töödes keskendutud lahendustele, mis tekitavad kuni 15 klastrit, samas kui käesolevas töös on klastrite arv sõltuvuses piltide resolutsioonist ning võib olla märksa suurem.

3. Metoodika

Töö eesmärgiks on anda ülevaade levinumatest klasteranalüüsi meetoditest ning illustreerida võimalikult intuitiivselt vastavate meetodite ja nende erinevate parameetrite mõju klasterdusele, rakendades neid meetodeid näidisandmestikul. Käesolevas peatükis põhjendatakse, miks on selle jaoks klasterdatavaks andmestikuks valitud just piltide segamini aetud järjestusega piksliread ning millised on piirangud kasutatavatele piltidele. Samuti kirjeldatakse, kuidas on implementeeritud erinevate meetodite illustreerimiseks kasutatav skript (vt Lisa I), mis võtab sisendiks pildifailid, paiskab segamini nende piksliridade järjestuse ning genereerib nende põhjal tulemusfaili, mis sisaldab iga sisendiks olnud pildi kohta iga klasterdamismeetodi rakendamisel tekkinud klasterdust.

3.1 Klasterdatavad andmed

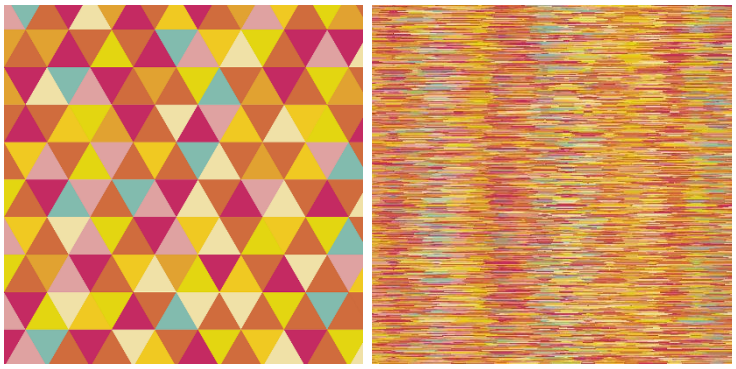
Klasteranalüüsi demonstreerimiseks on vaja andmeid, mida klasterdada. Lähtudes töö eesmärgist, peaksid need andmed olema sellised, mille klasterdamine illustreeriks arusaadavalt ja intuitiivselt klasteranalüüsi mõju. Selliseks andmestikuks on käesolevas töös valitud pildid, mille piksliridade järjestus on segi paisatud. Segi paisatud piksliridadega pilti vaadeldes on näha ainult müra ning sellest kasulikku infot ehk andmestikus peituvaid struktuure või seoseid on üsna keeruline välja lugeda (vt Joonis 7). Andmestikus peituva info leidmiseks saab aga kasutada klasteranalüüsi. Kuna enamasti on pildi piksliridade üleminekud visuaalselt sujuvad, siis on esialgsel pildil kõrvuti asuvad piksliread üksteisega visuaalselt kõige sarnasemad. See tähendab, et rakendades segatud piksliridadega pildi peal klasteranalüüsi ning kasutades võrreldava tunnusena piksliridade visuaalset sarnasust, klasterduvad kokku esialgse pildi naaberread ehk avalduma hakkavad piksliridade vahelised seosed ning ilmnevad esialgse pildi tunnused. Saadud klasterdust veel ka järjestusalgoritmiga töödeldes peaks aga juba enamasti olema võimalik taasluua esialgne pilt. Seega illustreerib valitud andmestik hästi klasteranalüüsi omadust leida sarnaste andmeobjektide grupeerimise kaudu andmestikust varjatud infot.

Joonis 7. Originaalpilt ja segamini paisatud piksliridadega pilt.

Eelnevas lõigus mainiti, et pildi piksliridade üleminekud on enamasti sujuvad. Eksisteerib aga ka pilte, milles leidub järske üleminekuid ehk kõrvuti asetsevate piksliridade puhul ei ole alati sarnasusseost. Selle alusel jagataksegi käesolevas töös kasutatavad pildid kahte klassi: järskude üleminekutega pildid ja sujuvate üleminekutega pildid. Esimesse klassi kuuluvad enamasti nn tehispildid, kus piltide erisektsioonid on selgelt eristuvad ning sektsioonide vahelised seosed puuduvad, näiteks malelaua või kandilise mosaiigi kujutised. Teise klassi kuuluvad näiteks mitmesugused töötlemata fotod, kuna looduses üldjuhul veatute sirgjoontega piiritletud kujundeid ei leidu ning isegi silma jaoks ühtlase tausta, näiteks helesinise taeva puhul, tajub fotoaparaat veidi heledamaid või tumedamaid alasid, mis pikslite

värviväärtustes kajastuvad ning kõrvuti asetsevaid piksliridu seovad. Klasterdamise ja klasterduse visualisatsiooni parandamiseks kasutatavate meetodite näitlikustamiseks on valitud pilt mõlemast pildiklassist.

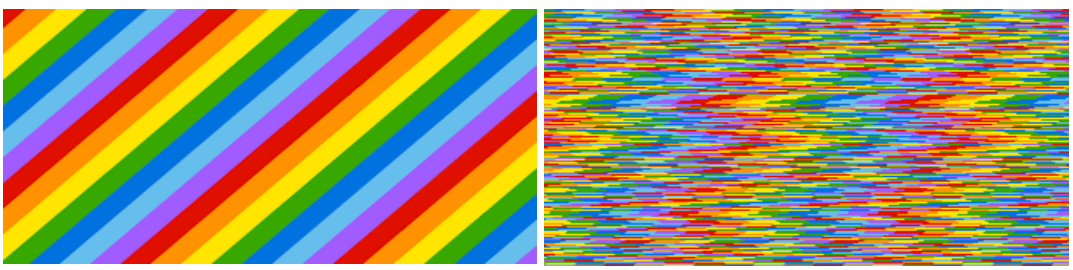
Esimest pildiklassi ehk järskude üleminekutega pilte on valitud esindama 1280x1280 resolutsiooniga pilt kolmnurkade mustrist (vt Joonis 8). Pildil on kümme eristuvat horisontaalset sektsiooni ning nende sektsioonide üleminekukohtades asuvate ridade vahel puuduvad sarnasusseosed. See tähendab, et segatud piksliridadest peaks olema võimalik klasterdamise ning selle tulemuseks oleva klasterduse järjestamise abil taastada vastavad sektsioonid, kuid mitte sektsioonide omavahelist järjestust ehk tulemuseks olev pilt ei oleks identne esialgsega. Seda tüüpi pilt sobib hästi illustreerimaks klasteranalüüsi omadust grupeerida sarnaseid objekte ning samadesse klastritesse kuuluvate elementide järjestamise mõju klasterduse visualisatsioonile. Samas ei võimalda seda tüüpi pilt võrrelda erinevate klasteranalüüsi meetodite efektiivsust selle põhjal, et kui sarnane on tulemuseks saadud pilt esialgsele pildile, kuna need ei saagi identsed olla.



Joonis 8. Kolmnurkade mustri pilt.

Teist pildiklassi ehk sujuvate üleminekutega pilte esindab 2064x1548 resolutsiooniga *Casa Mila* foto (vt Joonis 7). Erinevalt kolmnurkade mustri pildist, ei ole sellel pildil selgelt eristuvaid sektsioone ning piksliridade üleminekud on sujuvad ehk iga pikslirida on väga sarnane oma naaberridadega. Kuna puuduvad selgelt eristuvad sektsioonid, siis ei ole seda tüüpi pilt klasterdamiseks nii loomulik andmestik, kui oli kolmnurkade mustri pilt. Küll aga on selle pildi abil hea näidata, mil viisil võib klasterduse visualisatsiooni parandada lisaks samadesse klastritesse kuuluvate elementide ümber järjestamisele ka klastrite endi ümber järjestamisega. Samuti saab selle pildi puhul esialgset pilti ja tulemuseks olevat pilti vaadeldes visuaalselt võrrelda erinevate klasteranalüüsi meetodite efektiivsust ning nende erinevate parameetrite mõju klasterdusele.

Lisaks nendele kahele pildile klasterdatakse veel ka ühte nn nihkes piksliridadega 300x168 resolutsiooniga pilti (vt Joonis 9), et võrrelda peatükis 2.1 kirjeldatud eukleidilise kauguse ja *DTW* kaugusmõõte vastaval erijuhul. Seda pilti klasterdatakse aga ainult hierarhiliselt, kuna teiste algoritmide implementatsioonid *DTW* kaugusmõõdu kasutamist ei võimalda.



Joonis 9. Nihkes piksliridadega pilt.

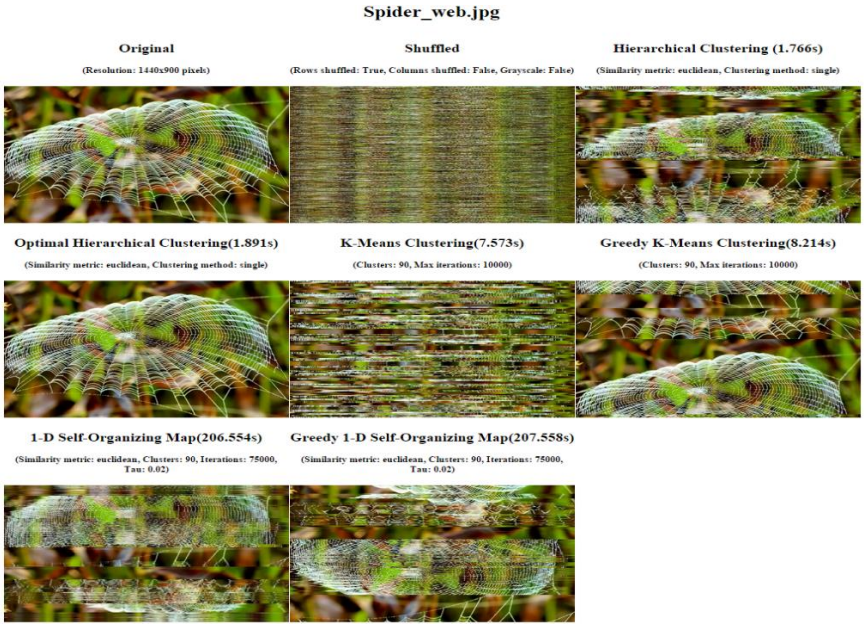
3.2 Kasutatav skript

Et klasterdamismeetodite katsetamine erinevate piltide peal võimalikult lihtsaks teha, on käesoleva töö raames loodud vastav skript (vt Lisa I). Skript on kirjutatud, kasutades Python 3.4.2 ja selle erinevaid vabavaralisi lisapakette, kuid võimaldab Rpy2 liidese kaudu rakendada ka R 3.1.2 ja selle lisapakettide funktsioone, täiendades oluliselt skripti võimalusi.

Skript otsib sisendandmeid ehk kasutaja poolt valitud pildifaile skriptiga samas kataloogis asuvast kaustast nimega *„Images“*. Seejuures sobivad andmeteks kõik JPEG, GIF ja PNG vormingus olevad pildifailid ning nende arv ei ole piiratud, kuigi suurema arvu piltidega kaasneb mõistagi skripti pikem tööaeg. Sisendiks olevate piltide puhul on oluline piltide resolutsioon. Kuna pildiandmeid hoitakse maatriksina, mille iga rida koosneb vastava pikslirea pikslite RGB väärtustest, siis on m x n resolutsiooniga pilti esindavas maatriksis $m * n * 3$ elementi. Seega võtab väga kõrge resolutsiooniga piltide sellisel kujul mälus hoidmine palju ruumi ning arvutuste tegemine palju aega ja mõistlik on piirduda näiteks 1920 x 1080 resolutsiooniga piltidega. Mälu- ja ajakasutuse vähendamiseks on skriptil võimalik väärtustada ka parameeter, mille mõjul kujutatakse piksleid RGB väärtuste asemel halltoonide väärtustena, mis vähendab maatriksis elementide arvu $m * n$ elemendini, kuid mõjutab mõningal määral ka tulemuste täpsust.

Pildifaile võetakse sisendandmete seast ühe kaupa. Esimese asjana peale pildi maatrikskujule viimist segatakse vastavalt skriptis määratud parameetritele maatriks kas veergude või ridade või nii veergude kui ka ridade kaupa. Seejärel rakendatakse ükshaaval segatud andmestiku peal erinevaid uuritavaid algoritme ning luuakse igast tulemusest uus pildifail, mis salvestatakse koos originaalpildi ja segatud andmestikuga pildiga kausta *„Output“*, mis tekitatakse skriptiga samasse kataloogi. Seejuures rakendatakse erinevaid algoritme vastavalt segamise parameetritele ehk kui segati näiteks nii read kui ka veerud, siis ka algoritme rakendatakse nii maatriksi ridadele kui ka veergudele. Erinevate algoritmide implementatsioonide ja võimalike muudetavate parameetrite kohta on täpsem ülevaade alampeatükis 3.2.1.

Lõpuks genereeritakse „*Output*“ kausta tekitatud pildifailide põhjal HTML fail (vt Joonis 10), mis sisaldab infot ka meetodite rakendamiseks kulunud aja ning parameetrite kohta.



Joonis 10. Klasterdamise skripti väljund ühe pildi korral.

3.2.1 Algoritmide implementatsioonid

Algoritme, mida sisendandmete peal rakendatakse, on kokku 6: k-keskmiste klasterdamine, hierarhiline klasterdamine, Kohoneni võrk ning vastavate algoritmide järjestamisalgoritmidega täiendatud variandid. Kuna klasteranalüüs on populaarne tehnika, siis on nende algoritmide rakendamiseks võimalik kasutada mitmeid juba olemasolevaid lahendusi.

Kõik vastavaid algoritme rakendavad skripti funktsioonid võtavad sisendiks neli üldist argumenti, mis on kõigil ühised, ning varieeruva arvu konkreetse algoritmi spetsiifilisi argumente. Üldiste argumentide hulka kuuluvad kolm *boolean* tüüpi väärtust, mis määravad, kas algoritmi tuleks rakendada andmestikuks oleva maatriksi ridadel, kas algoritmi tuleks rakendada andmestikuks oleva maatriksi veergudel ning kas andmestikuks olevas maatriksis kujutatakse piksleid halltoonides või RGB väärtustena ehk kas maatriks on kahe- või kolmemõõtmeline. Neljas üldine agument on andmestikuks olev maatriks ise.

Ühiseid liikmeid on ka algoritme rakendavate funktsioonide väljundiks oleval korteežil. Nendeks ühisteks liikmeteks on vastava algoritmi rakendamisel tekkinud ümberjärjestatud maatriks ning algoritmi tööks kulunud aeg. Ülejäänud väljundiks oleva korteeži liikmed sisaldavad infot algoritmi spetsiifiliste argumentide kohta ning nende arv sõltub juba konkreetsest algoritmist.

Hierarhilise klasterdamise funktsiooni puhul tuleb lisaks neljale üldisele argumendile sisendiks anda veel ka kaks sõne tüüpi argumenti, milleks on piksliridade võrdlemiseks kasutatav kaugusmõõt ning klastrite ühendamise kriteerium, mis kajastuvad mõlemad ka funktsiooni väljundis. Võimalike kaugusmõõdu väärtuste hulka kuuluvad kõik kaugusmõõdud, mis on välja toodud algoritmi rakendamisel kauguste maatriksi koostamiseks kasutatava Pythoni *SciPy* paketi funktsiooni *pdist* dokumentatsioonis[1]. Lisaks neile kaugusmõõtudele on R-i *dtw*[2] paketi kaudu toetatud veel ka kaugusmõõdu väärtus „dtw“. Klastrite ühendamise kriteeriumite võimalikud väärtused võib leida hierarhiliseks klasterdamiseks kasutatava R-i *stats* paketi funktsiooni *hclust* dokumentatsioonist[3].

K-keskmiste klasterdamise funktsioon nõuab samuti kahte algoritmi spetsiifilist argumenti, milleks on klastrite arv ning maksimaalne iteratsioonide arv. Kui klastrite arvu argument pikemat seletamist ei vaja, siis maksimaalne iteratsioonide arv määrab mitu korda tsentroide maksimaalselt ümber arvutatakse enne kui algoritm töö lõpetab. See võimaldab seada iteratsioonide ülempiiri haruldasteks juhtudeks, kus algoritm konvergeerub väga aeglaselt või ei konvergeerugi. Enamasti võiks aga iteratsioonide arv olla piisavalt suur, et algoritm lõpetaks töö enne limiidini jõudmist. K-keskmiste klasterdamiseks kasutatakse antud funktsioonis R-i *stats* paketi funktsiooni *kmeans*[4], mille tsentroidide arvutamise algoritmiks on valitud kõige enam tuntud MacQueen'i meetod.

Kohoneni võrgu funktsioonil on algoritmi spetsiifilisi argumente kolm: võrgus olevate neuronite ehk sisuliselt klastrite arv, treenimisfaasi iteratsioonide arv ning kohandumisfunktsiooni algparameeter, mis iteratsioonide käigus lineaarselt kahaneb. Kohoneni võrgu algoritmi rakendamiseks kasutatakse siin Pythoni paketi *Pycluster*[5] funktsiooni *somcluster*, mis on mõeldud küll ristküliku kujulise neuronite asetuse jaoks, kuid määrates y-telje pikkuseks ühe neuroni ning x-telje pikkuseks sisendiks oleva neuronite arvu, saab seda kasutada ka

[1] http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.spatial.distance.pdist.html
[2] http://cran.r-project.org/web/packages/dtw/dtw.pdf
[3] https://stat.ethz.ch/R-manual/R-patched/library/stats/html/hclust.html
[4] https://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html
[5] http://bonsai.hgc.jp/~mdehoon/software/cluster/cluster.pdf

ühedimensioonilise võrguna. Kaugusmõõduna on selles funktsioonis vaikimisi kasutusel eukleidiline kaugusmõõt.

Klasterdamismeetodite järjestusalgoritmidega täiendatud variantide funktsioonid on suures osas analoogilised vastavate täiendamata klasterdamismeetodite funktsioonidega ning on identse sisendi ja väljundiga. Ainuke erinevus seisneb selles, et peale klasterdamise lõppu ja enne väljundi tagastamist, rakendatakse klasterduse peal veel ka erinevaid järjestamisalgoritme. Hierarhilise klasterdamise puhul rakendatakse optimaalse lehtede järjestuse saamiseks R-i paketi *cba*[6] funktsiooni *order.optimal.* Nii k-keskmiste kui ka Kohoneni võrgu puhul rakendatakse klastritesse kuuluvate elementide järjestamiseks selle sama paketi funktsiooni *order.greedy.* K-keskmiste täiendatud algoritmi puhul järjestatakse veel aga lisaks klastrite sisule ka klastreid ennast ning selle jaoks on kasutusel autori enda lahendus, mis põhineb peatükis 2.3.1 kirjeldatud klastrite ahne järjestamise algoritmil.

[6] http://cran.r-project.org/web/packages/cba/cba.pdf

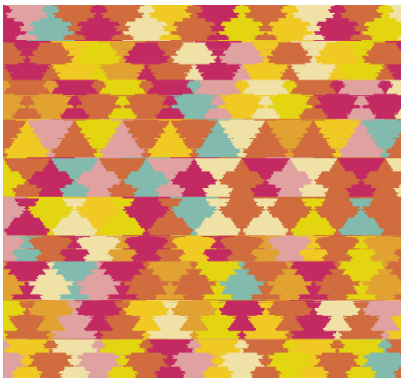
4. Klasterdamise tulemused

Selles peatükis analüüsitakse peatükis 3.2 kirjeldatud klasterdamise skripti väljundit peatükis 3.1 kirjeldatud piltide korral, juhul, kus klasterdatakse ainult piksliridu. Skripti väljundit klasterdamisalgoritmide rakendamisel piksliveergudele või nii piksliridadele kui ka –veergudele analüüsitud ei ole. Küll on aga nende klasterdamiste tulemused ning ka ainult piksliridade klasterdamiste tulemused suurema hulga erinevate piltidega saadaval lisamaterjalidena (vt Lisa II).

4.1 Hierarhiline klasterdamine

Kolmnurkade mustri ja *Casa Mila* piltide hierarhiliseks klasterdamiseks valiti klastrite ühendamise kriteeriumiks minimaalne kaugus ning kaugusmõõduks eukleidiline kaugus. Nihkes piksliridadega pilti klasterdati sama klastrite ühendamise kriteeriumiga, kuid nii eukleidilise kui ka *DTW* kaugusmõõduga, et võrrelda nende erinevust antud erijuhul.

Kolmnurkade mustri pildi hierarhilise klasterdamise tulemuseks oleval pildil (vt Joonis 11) võib aimata küll esialgse pildi kolmnurkset mustrit, kuid tugevalt moonutatud kujul. Moonutus iseloomustab hästi seda, et hierarhilisel klasterdamisel klastrite ühinemisi kujutavas puustruktuuris ei ole kahe klastri ühinemiskohas üheselt määratud kumb peaks jääma puustruktuuri vasakuks ja kumb paremaks pooleks. Seega on sama klasterduse puhul võimalik suur hulk erinevaid puu lehtede järjestusi, mis käesoleva andmestiku puhul kõik erineva pildi annavad. Aega kulus klasterdamiseks 3.64 sekundit.

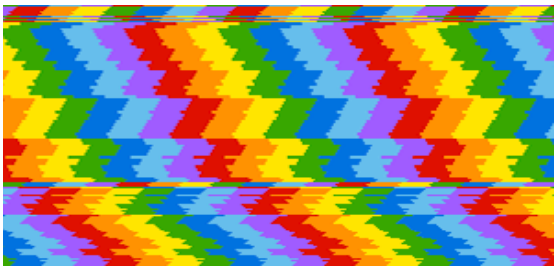


Joonis 11. Kolmnurkade mustri pildi hierarhiline klasterdus.

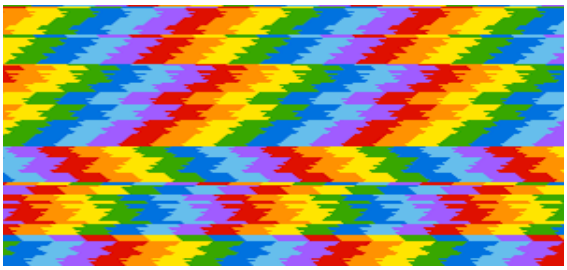
Casa Mila foto hierarhiline klasterdus (vt Joonis 12) on oma olemuselt üsna sarnane kolmnurkade mustri klasterdusega. Tulemuseks oleval pildil on samuti näha tugevalt moonutatud kujul esialgse pildi tunnuseid ning ka moonutuse põhjus on sama. Aega võttis selle foto klasterdamine malelaua pildist ligi 2 korda rohkem ehk 7.47 sekundit. Suurem ajakulu on tingitud piltide resolutsioonide erinevusest, kuna klasterdatavaid piksliridu on rohkem ning nende dimensionaalsus on suurem.

Joonis 12. *Casa Mila* foto hierarhiline klasterdus.

Nihkes piksliridadega pildi hierarhiline klasterdus, kus kaugusmõõduna on kasutatud eukleidilist keskmist (vt Joonis 13.1) ja klasterdus, kus kasutatakse *DTW* kaugusmõõtu (vt Joonis 13.2) on sarnaselt eelnevatele piltidele samuti esialgse pildi tunnustega, kuid moonutatud välimusega, mis muudab keeruliseks ka kaugusmõõtude efektiivsuse visuaalse hindamise. Hoolikalt vaadates on siiski näha, et *DTW* kaugusmõõduga klasterdatud pildi puhul on aimatavate diagonaalsete joonte kaldenurk korrektsem kui eukleidilise kaugusmõõduga klasterdatud pildi puhul. Märksa objektiivsemalt on võimalik hinnata tööks kulunud aega, mis eukleidilise kaugusmõõdu puhul on pildi madala resolutsiooni tõttu ainult 0.02 sekundit. *DTW* kaugusmõõdu puhul töötab algoritm aga lausa 3 287.45 sekundit ehk umbes 160 000 korda kauem. Nii suur erinevus võib olla tingitud osaliselt ka implementatsiooni erinevustest. Nimelt *DTW* kaugusmõõdu puhul on hierarhiliseks klasterdamiseks vajaliku kauguste maatriksi koostamiseks kasutusel R ja eukleidilise kaugusmõõdu puhul Python.



Joonis 13.1. Nihkes piksliridadega pildi hierarhiline klasterdus eukleidilise mõõduga.



Joonis 13.2. Nihkes piksliridadega pildi hierarhiline klasterdus *DTW* mõõduga.

4.2 Hierarhiline klasterdamine optimaalse lehtede järjestamisega

Optimaalse lehtede järjestamisega hierarhiliseks klasterdamiseks kasutati samu pilte ning klasterdamise parameetreid, mis tavalisel hierarhilisel klasterdamisel. Ainuke erinevus seisnes selles, et tulemuseks saadud klasterdust töödeldi ka järjestamisalgoritmiga, mis järjestab hierarhilise klasterduse dendrogrammis alampuid, otsides minimaalse kauguste summaga lehtede järjestust, säilitades samas klastrite vahelised ühendused.

Kolmnurkade mustri pildi puhul on optimaalse lehtede järjestamisega hierarhilise klasterdamise tulemuseks olev pilt (vt Joonis 14) väga sarnane esialgse pildiga ehk pilt on jagunenud korrektselt kümneks eristuvaks sektsiooniks. Erinevuseks on ainult see, et sektsioonide vaheline järjestus on vale, mis on ka oodatud tulemus, kuna sektsioonide üleminekukohtades asuvate ridade vahel ei olnud esialgsel pildi sarnasusseoseid ja seega ei ole sektsioone võimalik ka millegi alusel järjestada. Aega kulus algoritmil 4.23 sekundit ehk tavalise hierarhilise klasterdamisega võrreldes ainult 0.59 sekundit rohkem, mis on võimalik tänu sellele, et järjestamiseks saab kasutada sama kauguste maatriksit, mida kasutati ka klasterdamisel, ja uut kauguste maatriksit ei ole vaja arvutada.

Joonis 14. Kolmnurkade mustri järjestatud hierarhiline klasterdus.

Casa Mila foto puhul on hierarhilise klasterduse järjestusalgoritmiga töötlemisel veel suurem kasutegur kui kolmnurkade mustri pildi puhul. Tulemuseks olev pilt (vt Joonis 15) on peaaegu identne originaalpildiga ning ainuke erinevus seisneb selles, et piksliread on järjestatud tagurpidi. See erinevus tuleneb sellest, et algoritmi eesmärk on klasterdada kokku originaalsel pildil kõrvuti asuvad piksliread, kuid see eesmärk on täidetud ka pildi puhul, mille piksliread on originaaliga vastupidises järjekorras. Seega algoritmi seisukohast on korrektne nii originaalpilt kui ka tagurpidi olevate piksliridadega pilt. Analoogiliselt malelaua pildile on ka selle foto puhul järjestamisega hierarhilise klasterdamise ajakulu tavalise hierarhilise klasterdamisega väga sarnane, võttes aega 7.78 sekundit ehk ainult 0.31 sekundit rohkem.

Joonis 15. *Casa Mila* foto järjestatud hierarhiline klasterdus.

Hierarhilise klasterduse järjestusalgoritmiga töötlemine parandab tulemust märkimisväärselt ka nihkes piksliridadega pildi puhul, seda nii eukleidilise (vt Joonis 16.1) kui ka *DTW* (vt Joonis 16.2) kaugusmõõduga klasterduse korral. Erinevalt tavalisest hierarhilisest klasterdamisest on järjestatud hierarhilise klasterdamise puhul selgelt näha, et *DTW* kaugusmõõt annab seda tüüpi pildi puhul parema tulemuse kui eukleidiline kaugusmõõt. Kui eukleidilise kaugusmõõduga klasterdamisel on tulemuseks oleval pildil originaaliga võrreldes mõningaid erinevusi, siis *DTW* kaugusmõõduga klasterdamise tulemuseks olev pilt on originaaliga sisuliselt identne, olles ainult tagurpidise piksliridade järjestusega. Võrreldes tavalise hierarhilise klasterdamisega, jääb ka selle pildi puhul järjestatud hierarhilise klasterdamise ajakulu samasse suurusjärku, olles eukleidilise kaugusmõõdu puhul 0.05 sekundit ja *DTW* kaugusmõõdu puhul 3414.97 sekundit.

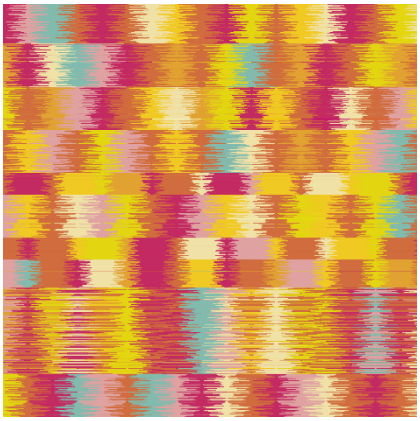
Joonis 16.1. Nihkes piksliridadega pildi järjestatud hierarhiline klasterdus eukleidilise kaugusmõõduga.

Joonis 16.2. Nihkes piksliridadega pildi järjestatud hierarhiline klasterdus *DTW* kaugusmõõduga.

4.3 K-keskmiste klasterdamine

K-keskmiste klasterdamise algoritmi puhul on oluliseks parameetriks klastrite arv. Levinud meetod selle arvu leidmiseks on erineva klastrite arvuga saadud klasterduste omavaheline võrdlemine ja parima välja valimine ning seda kasutati *Casa Mila* foto puhul. Kolmnurkade mustri pildi puhul oli aga ilmseks klastrite arvuks kümme. K-keskmiste algoritmi eripära seisneb veel ka selles, et isegi sama andmestiku korral võivad sõltuvalt juhuslikult valitud esialgsetest tsentroididest nii algoritmi tööks kuluv aeg kui ka algoritmi tulemus olla erinevatel katsetel erinevad. Seega rakendati iga parameetrite komplektiga algoritmi viis korda ning analüüsimiseks valiti välja iga komplekti parim tulemus.

Kolmurkade mustri pildi puhul muudab õige klastrite arvu valimise triviaalseks asjaolu, et esialgsel pildil on kümme selgelt eristuvat horisontaalset sektsiooni ehk kümme klastrit. Kui tavalise hierarhilise klasterdamise puhul oli tulemuseks olev pilt küll moonutatud, kuid siiski äratuntavate tunnustega, siis k-keskmiste klasterdamise tulemuseks oleva pildi (vt Joonis 17) ja esialgse pildi vahel sarnasusi sisuliselt ei olegi. See kinnitab, et täiendamata k-keskmiste algoritm on valitud klasteranalüüsi algoritmidest järjestamise osas kõige nõrgem. Ajakulu jäi kolmnurkade mustri pildi puhul erinevatel katsetel vahemikku 4.41 kuni 6.37 sekundit, mis muudab algoritmi antud pildi puhul aeglasemaks kui oli hierarhiline klasterdamine.



Joonis 17. Kolmnurkade mustri pildi k-keskmiste klasterdus.

Casa Mila foto puhul ei ole klastrite arv ilmne, seega katsetatati klasterdamist kolme erineva klastrite arvuga, milleks olid 15 (vt Joonis 17.1), 30 (vt Joonis 17.2) ja 154 (vt Joonis 17.3) ehk piksliridade arvu täisarvuline jagatis vastavalt saja, viiekümne ja kümnega. Kuna sarnaselt kolmnurkade mustri pildile ei meenutanud ühegi klastrite arvuga saadud tulemuseks olev pilt esialgset pilti, siis ei ole võimalik k-keskmiste algoritmi puhul ilma täiendava järjestamiseta visuaalse vaatlemise teel optimaalset klastrite arvu leida.

Joonis 18.1. *Casa Mila* foto k-keskmiste klasterdus 15 klastriga.

Joonis 18.2. *Casa Mila* foto k-keskmiste klasterdus 30 klastriga.

Joonis 18.3. *Casa Mila* foto k-keskmiste klasterdus 154 klastriga.

Lisaks juhuslikult valitud tsendroidide algväärtustele mõjutas k-keskmiste klasterdamise ajakulu ka klastrite arv. Olenevalt tsendroidide algväärtustest varieerus ajakulu 15 klastri puhul 13.12 kuni 24.98 sekundini, 30 klastri puhul 26.09 kuni 62.15 sekundini ning 154 klastri puhul 151.57 kuni 220.99 sekundini. Seega oli k-keskmiste algoritm hierarhilisest klasterdamisest aeglasem juba 15 klastri puhul ning klastrite arvu kasvades see vahe ainult suurenes. K-keskmiste klasterdamise suurem ajakulu võrreldes hierarhilise klasterdamisega on käesolevas töös tõenäoliselt tingitud algoritmide implementatsioonist, nimelt kasutatakse hierarhilisel klasterdamisel kauguste maatriksi koostamisel R-i vahendite asemel märksa kiiremaid Python-i vahendeid. Üldjuhul peaks k-keskmiste klasterdamine andmestiku kasvades hierarhilise klasterdamisega võrreldes paremini skaleeruma.

4.4 K-keskmiste klasterdamine ahne järjestamisega

K-keskmiste klasterdamise järjestamisega täiendatud algoritmi rakendati samade parameetritega nagu tavalist k-keskmiste klasterdamist. Juurde lisandus ainult tulemuseks oleva klasterduse järjestamine ahne järjestamisalgoritmiga.

Kolmnurkade mustri pildi puhul on järjestamisega täiendatud k-keskmiste algoritmi tulemus (vt Joonis 19) väga sarnane esialgse pildiga, erinedes ainult eristuvate sektsioonide järjestuse poolest, täpselt nagu täiendatud hierarhilise klasterdamisegi puhul. Kuigi kasutatavad järjestamisalgoritmid on mõlemal juhul erinevad, siis põhjus, miks need antud pildi puhul sektsioone korrektselt järjestada ei suuda, jääb samaks ehk kuna esialgsel pildil puuduvad sektsioonide üleminekukohtades asuvate ridade vahel sarnasusseosed, siis ei ole lihtsalt sektsioone võimalik millegi alusel järjestada. Algoritmi tööaeg jäi erinevatel katsetel vahemikku 4.67 kuni 9.95 sekundit.

Joonis 19. Kolmnurkade mustri pildi järjestatud k-keskmiste klasterdus.

Casa Mila foto erinevate klastrite arvudega k-keskmiste klasterduste järjestusalgoritmiga töötlemine parandab tulemuseks olevate piltide välimust märkimisväärselt ning võimaldab ka võrrelda visuaalselt, millise klastrite arvuga on tulemus parim. Selgelt parima tulemuse andis 154 klastriga klasterdus (vt Joonis 20.1), mis omab originaalpildiga võrreldes ainult üksikuid vaevumärgatavaid ebatäpsusi pildi kesk- ja ülaosas. Sellele järgnes 30 klastriga klasterdus (vt Joonis 20.2), mille puhul on pildi ülemine pool üsna tugevalt moonutatud, kuid üldiselt on pilt siiski äratuntav. Kõige kehvema tulemuse andis 15 klastriga klasterdus (vt Joonis 20.3), millest tekkinud pilt on kõige enam moonutatud. Ka *Casa Mila* foto puhul on k-keskmiste klasterduse järjestamisele kuluv aeg üsna lühike, lisades algoritmi tööajale ainult mõne sekundi ehk mõjutades kogu tööks kuluvat aega vähem kui seda mõjutavad tsendroidide esialgsed väärtused.

Joonis 20.1. *Casa Mila* foto järjestatud k-keskmiste klasterdus 154 klastriga.

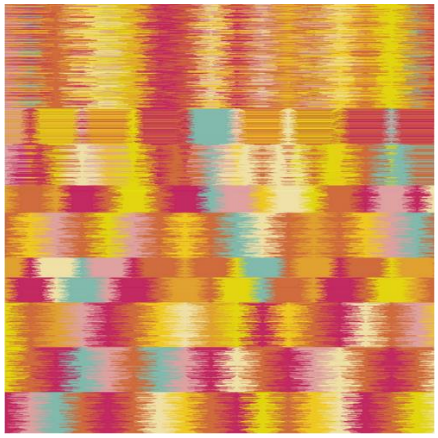
Joonis 20.2. *Casa Mila* foto järjestatud k-keskmiste klasterdus 30 klastriga.

Joonis 20.3. *Casa Mila* foto järjestatud k-keskmiste klasterdus 15 klastriga.

4.5 Kohoneni võrk

Kohoneni võrgu algoritmi rakendamine vajab kõige suurema hulga parameetrite väärtustamist. Üldiste soovituste [Koh90] kohaselt valiti kohandumisfunktsiooni algparameetriks 0.9 ning treenimisfaasi iteratsioonide arvuks viiesaja kordne neuronite arv. Neuronite ehk klastrite arvud olid seejuures võrdlemise eesmärgil samad, mis k-keskmiste klasterdamise korral ehk *Casa Mila* foto puhul 15, 30 ja 154 ning kolmnurkade mustri pildi puhul 10. Sarnaselt k-keskmiste klasterdamisele sõltub ka Kohoneni võrgu puhul klasterdus algseisu juhuslikust väärtustamisest ning seetõttu rakendati ka Kohoneni võrgu algoritmi iga parameetrite komplekti puhul viis korda ning analüüsiks valiti välja iga komplekti parim tulemus.

Kolmnurkade mustri pildi korral on Kohoneni võrgu algoritmi rakendamise tulemusel tekkiv pilt (vt Joonis 21) väga sarnane k-keskmiste algoritmi tulemusega, mis on üsna ootuspärane. Seda põhjusel, et kumbki algoritm ei järjesta klastrite sisu ning Kohoneni võrgu klastrite endi järjestamise omadus on seda tüüpi pildi puhul kasutu. Kohoneni võrk on aga oma 1.69 sekundise ajakuluga teistest algoritmidest antud juhul märksa kiirem. Väiksem ajakulu on tingitud väikesest klastrite arvust, mis ei vähenda mitte ainult igal treenimisfaasi iteratsioonil tehtavate arvutuste arvu, vaid ka iteratsioonide arvu, kuna see sõltub klastrite arvust.



Joonis 21. Kolmnurkade mustri pildi Kohoneni võrgu klasterdus

Erinevalt k-keskmiste klasterdamisest on Kohoneni võrgu algoritmi puhul *Casa Mila* foto korral võimalik optimaalse klastrite arvu kohta järeldusi teha ka juba järjestamisalgoritmiga täiendamata klasterduse puhul. Klasterdused 15 (vt Joonis 22.1), 30 (vt Joonis 22.2) ja 154 klastriga (vt Joonis 22.3) näitavad, et klastrite arvu kasvades suureneb ka tulemuseks oleva pildi sarnasus originaalpildiga. See on ka oodatud tulemus, kuna väiksema klastrite arvu

korral kuulub igasse klastrisse rohkem piksliridu, mis omavahelise korrektse järjestuse puudumise tõttu pildi nö „uduseks“ muudavad. Seejuures ei ole siiski mõistlik valida liiga suurt klastrite arvu, kuna iga järgnev klastrite arvu suurendamine parandab pildikvaliteeti eelmisest vähem ning klastrite arvu ja tööks kuluva aja vahel on ruutsõltuvus. 15, 30 ja 154 klastriga kulus algoritmi tööks aega vastavalt 6.34, 24.01 ja 588.92 sekundit.

Joonis 22.1. Kohoneni võrgu klasterdus 15 klastriga.

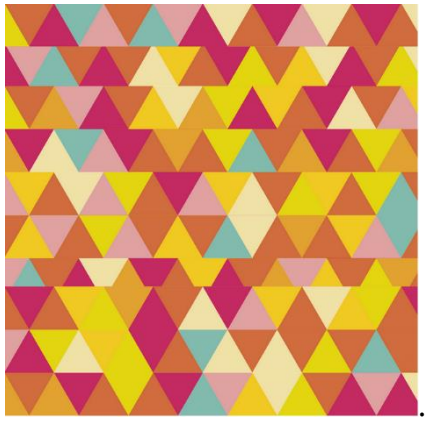
Joonis 22.2. Kohoneni võrgu klasterdus 30 klastriga.

Joonis 22.3 Kohoneni võrgu klasterdus 154 klastriga.

4.6 Kohoneni võrk ahne järjestamisega

Kohoneni võrgu ahne järjestamisega täiendatud algoritmi parameetrid olid analoogiliselt hierarhilise ja k-keskmiste klasterdamisega samad, mis täiendamata algoritmil. Võrreldes tavalise Kohoneni võrgu algoritmiga lisandus ainult klasterduse järjestamine, mis kasutab küll sama järjestusalgoritmi, mis k-keskmiste klasterdamine, kuid seda ainult klastrite sisu, mitte klastrite endi järjestamiseks.

Analoogiliselt eelnevate klasterdamisalgoritmidega, annab ka Kohoneni võrgu algoritmiga saadud kolmnurkade mustri pildi klasterduse järjestamine tulemuseks pildi, mis erineb esialgsest ainult eristuvate sektsioonide järjekorra poolest (vt Joonis 23). Seejuures ei kasvata järjestamise lisamine märkimisväärselt algoritmi tööaega, võttes kokku aega 2.09 sekundit.

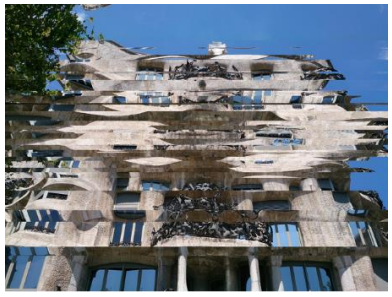


Joonis 23. Kolmnurkade mustri pildi Kohoneni võrgu järjestatud klasterdus

Kohoneni võrguga saadud klasterduse järjestamine muudab *Casa Mila* pildi puhul tulemuseks oleva pildi küll vähem nö „uduseks“, kuid jätab pildi sellegipoolest tugevalt moonutatuks, nii 15 (vt Joonis 24.1), 30 (vt Joonis 24.2) kui ka 154 (vt Joonis 24.3) klastri korral. Selle tingib asjaolu, et järjestamisalgoritm järjestab ainult klastrite sisu, kuna klastrid peaksid eeldatavalt saama järjestatud juba Kohoneni võrgu algoritmi rakendades. Paraku ei suuda algoritm sarnaseid klastreid aga piisava täpsusega järjestada ning seega on lõpptule-

mus moonutatud isegi siis, kui klastrite sisu on korrektselt järjestatud. Kohoneni võrgu algoritm koos klastrite sisu järjestamisega võtab *Casa Mila* foto puhul 15, 30 ja 154 klastri korral aega vastavalt 7.12, 24.69 ja 599.96 sekundit.

Joonis 24.1. Kohoneni võrgu järjestatud klasterdus 15 klastriga.



Joonis 24.2. Kohoneni võrgu järjestatud klasterdus 30 klastriga.

Joonis 24.3. Kohoneni võrgu järjestatud klasterdus 154 klastriga.

4.7 Järeldused

Kolmnurkade mustri pildi klasterdamise tulemuseks olevad pildid olid kõigi järjestamisalgoritmidega täiendatud klasterdamismeetodite puhul väga sarnased. Kuigi kõige kiiremaks osutus neist 2.09 sekundiga Kohoneni võrgu algoritm, siis tuleb arvestada ka seda, et see algoritm võib erinevatel katsetel anda erineva tulemuse ning täpse tulemuse tagastas ainult üks katse viiest. Seega võib parimaks pidada ajakulu poolest järgmist ehk hierarhilise klasterdamise algoritmi, mis võtab küll aega ligi kaks korda kauem ehk 4.23 sekundit, kuid tagastab alati sama tulemuse ning ei vaja mitut katset.

Casa Mila foto puhul on optimaalseima algoritmi valimine lihtsam. Ainuke algoritm, mis suutis foto täielikult taastada oli ahne järjestamisega hierarhiline klasterdamine, kuigi väga täpse pildi andis ka ahne järjestamisega k-keskmiste klasterdamine 154 klastriga. Hierarhilise klasterdamise kasuks räägib aga lisaks väikesele täpsuse eelisele veel ka tööks kulunud aeg, mis hierarhilise klasterdamise puhul oli 7.78 sekundit ning k-keskmiste klasterdamise puhul mitu minutit. Samuti oli hierarhilise klasterdamise puhul kõige lihtsam määrata klasterdamise parameetreid, kuna ei olnud vaja eksperimenteerida erinevate klastrite arvudega.

Kaugusmõõtude võrdlemiseks kasutatava nihkes piksliridadega pildi klasterdamisel andis eukleidilisest kaugusmõõdust täpsema tulemuse *DTW* kaugusmõõt. Samas oli see väga aeglane, võttes aega ligi tunni, kuigi klasterdatava pildi resolutsioon oli ainult 300x168. Seega ei sobi kõrgema resolutsiooniga nihkes piksliridadega piltide jaoks tegelikult kumbki kaugusmõõt ning uurida tuleks ka teisi kaugusmõõte.

5. Kokkuvõte

Klasteranalüüs on laia kasutusvaldkonnaga andmeanalüüsi tehnika. See ei ole aga üks konkreetne algoritm, vaid üldnimetus, ning seda on võimalik rakendada mitmel viisil. Selle töö eesmärgiks oli anda ülevaade kolmest enam levinud klasteranalüüsi meetodist ning demonstreerida nende tööd ja nende erinevate parameetrite mõju klasterdusele, rakendades meetodeid näidisandmestikul, milleks olid erinevate piltide segatud piksliread. Samuti taheti näidata kuidas on järjestamisalgoritme kasutades võimalik parandada klasterduse visualisatsiooni.

Selle jaoks defineeriti esmalt viis kuidas leida klasteranalüüsi jaoks vajalikku andmeobjektide sarnasust ning kirjeldati hierarhilise klasterdamise, k-keskmiste klasterdamise, Kohoneni võrgu ja vastavate meetodite täiendamiseks kasutatavate järjestamisalgoritmide tööpõhimõtteid ning eripärasid.

Järgnevalt kirjeldati täpsemalt kasutatavat näidisandmestikku ning defineeriti kolm erinevat pildiklassi, milleks olid sujuvate üleminekutega, järskude üleminekutega ja nihkes piksliridadega pildid. Samuti kirjeldati kuidas on implementeeritud ning kuidas kasutada skripti, mis muudab lihtsaks kasutaja poolt valitud piltide peal erinevate klasterdamismeetodite rakendamise ning tulemuste võrdlemise.

Viimaks rakendati seda skripti näidisandmeteks valitud piltide peal ning analüüsiti erinevate klasterdamismeetodite tulemusi. Nii sujuvate üleminekutega piltide kui ka järskude üleminekutega piltide puhul osutus efektiivseimaks järjestamisalgoritmiga täiendatud hierarhiline klasterdamine.. Kaugusmõõtude võrdlemiseks kasutatava nihkes piksliridadega pildi puhul andis eukleidilisest kaugusmõõdust täpsema tulemuse *DTW* kaugusmõõt, kuid kuna selle ajakulu oli ebamõistlikult suur, siis osutusid mõlemad ebasobivaks ning vaja oleks mingit kolmandat kaugusmõõtu.

6. Tsiteeritud teosed

[BGJ01] Z. Bar-Joseph, D. K. Gifford and T. S. Jaakkola, „Fast Optimal Leaf Ordering for Hierarchical Clustering," *Bioinformatics*, vol. 17, pp. 22-29, Mar. 2001.

[BMG11] S. M. Bidoki, N. Mahmoudi-Kohan and S. Gerami, „Comparison of Several Clustering Methods in the Case of Electrical Load Curves Classification," in *Elect. Power Distribution Conf.*, 2011, pp. 1-7.

[CNP06] G. Chicco, R. Napoli and F. Piglione, „Comparisons Among Clustering Techniques for Electricity Customer Classification," *IEEE Trans. Power Syst.*, vol. 21, no. 2, pp. 933-940, May 2006.

[GMMA00] J. Goddard, A. E. Martinez, F. M. Martinez and T. Aljama, „A comparison of Different Clustering Algorithms for Speech Recognition," in *Proc. of the 43rd IEEE Midwest Symp. on Circuits and Syst.*, 2000, vol. 3, pp. 1222-1225.

[Hos12] S. Hosangadi, „Distance Measures for Sequences," *CoRR*, vol. abs/1208.5713, 2012.

[HTF09] T. Hastie, R. Tibshirani and J. Friedman, „14.3.12 Hierarchical Clustering," in *The Elements of Statistical Learning*, 2nd ed., New York, Springer, 2009, pp. 520-528.

[Koh90] T. Kohonen, „The Self-Organizing Map," *Proc. of the IEEE*, vol. 78, no. 9, pp. 1464-1480, Sep. 1990.

[LCL+04] M. Li, X. Chen, X. Li, B. Ma and P. M. B. Vitanyi, „The Similarity Metric" *IEEE Trans. Inf. Theory*, vol. 50, no. 12, pp. 3250-3264, Dec 2004.

[MMM13] A. M. Mehar, K. Matawie and A. Maeder, „Determining an Optimal Value of K in K-Means Clustering," in *Bioinformatics and Biomedicine IEEE Int. Conf.*, 2013, pp. 51-55.

[RS10] P. Rai and S. Singh „A Survery of Clustering Techniques", in *Int. J. of Comput. Applicat.*, vol. 7, no. 12, Oct. 2010.

[Sen08] P. Senin, „Dynamic Time Warping Algorithm Review," Information and Computer Science Department, University of Hawaii at Manoa, Honolulu, Dec. 2008.

[Yag00] R. R. Yager, „Intelligent Control of the Hierarchical Agglomerative Clustering Process," *IEEE Trans. Syst. Man Cybern.*, vol. 30, no. 6, pp 835-845, Dec. 2000.

Lisad

I. Piksliridade klasterdamise skript

Skripti lähtekoodi repositoorium – viimati uuendatud 06.05.2015.

https://github.com/priitd/image-clustering

II. Täiendavad klasterdamise tulemused

Piksliridade klasterdused – viimati uuendatud 13.05.2015.

http://kodu.ut.ee/~priitd/clustering/clustered-rows/

Piksliveergude klasterdused – viimati uuendatud 13.05.2015.

http://kodu.ut.ee/~priitd/clustering/clustered-columns/

Piksliridade ja –veergude klasterdused – viimati uuendatud 13.05.2015.

http://kodu.ut.ee/~priitd/clustering/clustered-rows-and-columns/

III. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina **Priit Danelson** (sünnikuupäev: 22.05.1992)
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Klasteranalüüsi meetodite uurimine visuaalsete andmete abil**,
(*lõputöö pealkiri*)

mille juhendaja on Jaak Vilo,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **13.05.2015**