Eötvös Loránd Tudományegyetem

Informatikai Kar

Programozáselmélet és Szoftvertechnológiai Tanszék

DRÓNNAL RÖGZÍTETT LÉGIFELVÉTELEK KLASZTEREZŐ ALGORITMUSAINAK ELEMZÉSE

CSERÉP MÁTÉ
egyetemi tanársegéd
témavezető

TARCZALI TAMÁS
programtervező informatikus MSc
szerző

Budapest, 2022

TARTALOMJEGYZÉK

1. FEJEZET

BEVEZETÉS

Napjaink egyik fő, a számítástudomány szakterületét felbolygató, de a hétköznapi ember életébe is begyűrűző kihívását jelenti a létrehozott, rögzített, másolt, és fogyasztott adatok mennyiségének rendkívüli gyorsaságú növekedése. Erre az évek óta a szakzsargon részét képező adatrobbanás kifejezés gyakori előbukkanása is utal, bár ettől talán nagyobb hangsúllyal bír a tény, hogy a világszerte keringő adat mennyiségét manapság már zettabyte-okban mérhetjük [1].

A természettudományhoz kapcsolódó ismereteink bővülésével (illetve a bővítésre vonatkozó igény növekedésével), valamint a felvételek rögzítését szolgáló eszközök és technológiák elérhetőbbé válásával a távérzékelés területének digitális lábnyoma is hasonló tendenciákat mutat. A távérzékelés feladatköre alatt leggyakrabban a földfelszín entitásairól, bizonyos, a távoli adatfelvételt elősegítő eszközökre (például drónokra) felszerelt érzékelők által rögzített légi- és űrfotók elemzésére gondolhatunk. A kis méretű, felszínhez közeli útvonalakon működtetett drónok fontos tulajdonsága – szemben például a szintén gyakran alkalmazott műholdakkal – hogy ezen eszközök, távoli objektumok által visszavert vagy kibocsátott sugárzás adott spektrális sávjait feltérképező szenzorai kisebb felszíni területek, részletekben gazdag szemlélését teszik lehetővé. A földfelszín, illetve a rajta végbemenő változások és folyamatok ilyen módú vizsgálata ebből adódóan magával vonja a feldolgozáshoz szükséges számítások mennyiségének és összetettségének nagyságát is.

Általánosságban, és a távérzékelés konkrét felhasználási területén is előtérbe kerül tehát az igény a keletkező adatok algoritmikus feldolgozására – aminek kimeneteként a felhasználó kezébe már az adott szempontok alapján szűrt, releváns információ juthat. Az idevágó eljárások a számítástudomány (és így a szoftvertechnológia) fontos és érdekes

problémáira keresnek megoldásokat; megalkotásukat átszövik a számítási bonyolultság csökkentéséhez, avagy a hatékonyság növeléséhez kapcsolódó kihívások.

Az adathalmazok tanulmányozásához kötődő, sűrűn megmutatkozó feladat a sokaság egyes pontjainak bizonyos szempontok szerint történő rendszerezése, vagyis a klaszteranalízis. A problémakör gyakran megjelenik a hétköznapi életünkben, emellett behálózza a tudomány számos területét is. Így például az orvosi képalkotásban, piackutatásban vagy az ökológiában, de természetesen a számítástudomány különböző területein is (gondolva itt akár az ajánlórendszerekre, vagy anomáliadetektálásra) felhasználják a rá születő megoldásokat. Az ilyen technikák ugyancsak fellelhetők a távérzékelésben, ahol a rögzített felvételek képpontjainak, az általuk reprezentált célkategóriához, vagyis tematikus osztályhoz sorolására alkalmazhatók.

A klaszterezés problémájának egyik fő nehézsége abban rejlik, hogy a fogalom nehezen általánosítható, hiszen minden felhasználási területhez kapcsolódóan más és más definíció adható meg az adatok egy megfelelő felosztására. Részben ennek köszönhető, hogy a probléma megoldására készülő módszerek számos különböző megközelítés alapján keresik a várt eredmény, mindemellett épp ezért is bizonyul érdekes feladatnak a létező (gyakran a generikus fogalmakra alapozva alkotott) megoldások egy konkrét felhasználási környezet tekintetéből történő szemügyre vételezése.

Az eddigiekben körülírt témákhoz köthető a *Giwer* (*Geoimage Workflow Editing Resources*) programcsomagjának projektje [2, 3], melynek hivatása – más létező képfeldolgozó rendszerekhez (mint akár a QGIS) hasonlíthatóan – a felhasználói körének ellátása számos térinformatikában és távérzékelésben gyakran használatos funkcionalitással; nagy hangsúlyt fektetve a drónfelvételekre, de nem kizárva az egyéb hordozók segítségével rögzített, különböző formátumokban hozzáférhető felvételeket sem. A Giwer különlegességét ezeken felül az adja, hogy (az úgynevezett *WorkflowBuilder* komponens segítségével) lehetőséget nyújt az általa rendelkezésre bocsátott metódusok és módszerek tetszőleges munkafolyamattá kombinálására. Természetesen mindezeket, ahogy azt bármely magas mércéknek megfelelni kívánó szoftver esetén elvárhatjuk, a felhasználóbarát, egyszerű és gyors működés biztosítása mellett kívánja elérhetővé tenni a publikum számára – a későbbiekben a teljes forráskóddal karöltve.

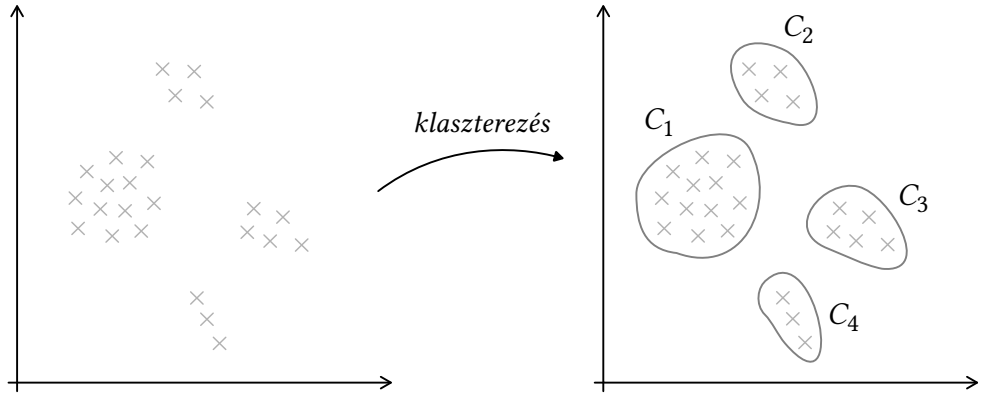
A bemutatott Giwer projekt klaszterezésért felelős algoritmusokkal történő bővítésének feladatához kapcsolódóan válhatott ezen diplomadolgozat céljává a klaszteranalízis

témakörének, valamint a releváns eljárások tulajdonságainak alaposabb tanulmányozása, főként a távérzékelés és a drónfelvételek nézőpontjából. A célkitűzés részeként tehát felsorakoztatható az ilyen módszerek realizációja során felmerülő kérdések és nehézségek megismerése, az optimalizációs lehetőségek felderítése, de a fejlesztéshez tartozóan emellett még néhány jelentősebb létező megoldás feltárása is, hiszen ezek mindegyike meglehetősen értékes információt hordozhat magával. A dolgozat rendeltetése ezek felett elsődlegesen a megalkotott eljárások erősségeinek és hátrányainak feltérképezése, gyakorlati körülmények melletti használhatóságuk és az erre vonatkozóan esetlegesen előtérbe kerülő problémák vizsgálata; nem utolsó sorban pedig a tőlük elvárható kimenetek minőségének felmérése.

2. FEJEZET

IRODALMI ÁTTEKINTÉS

Ahogy azt a bevezetésben is olvashattuk, a klaszterezés – illetve analóg módon az osztályozás – feladata alatt egy adathalmaz pontjainak, a megoldandó problémakör által megszabott szempontok (avagy a pontok bizonyos jellemzői) szerinti, megfelelő csoportosításának megtalálására gondolhatunk. Ezt vázolja fel a 2.1. ábra. Vajon miként tekinthető ez pontosan a távérzékelés szemszögéből nézve, illetve milyen metódusok használhatók a keresett klaszterek kialakítására?



2.1. ábra. A klaszterezés folyamatának illusztrációja. A kezdeti adatpontokat valamilyen szempont figyelembevételével a megfelelő csoportokba soroljuk.

2.1. A KLASZTEREZÉS MODELLJE A TÁVÉRZÉKELÉSBEN

Esetünkben a klaszterezés a multispektrális légi- és űrfotók minden egyes képpontjához az azt legjobban jellemző osztály címkéjének rendelését jelenti, a sávokban rejlő információt felhasználva. Tágabb értelemben ezen képpontok ábrázolhatók (a klaszterezés általános modelljéhez illeszkedve) a következőképp: minden ilyen pont egy $x_{(i,j)} = \left(x_{(i,j)_1}, x_{(i,j)_2}, \ldots, x_{(i,j)_d}\right)$ d-dimenziós vektor, ahol d az ezen pixelt tartalmazó

felvételhez tartozó spektrális sávok számának feleltethető meg. Megjegyzendő, hogy az $x_{(i,j)}$ képpont minden koordinátája a megfelelő sáv ponthoz tartozó intenzitásértékét tartalmazza, így ezen atomi építőelemek értékkészlete diszkrét értékekből áll elő: egy 8 bites színmélységű felvétel esetén például ez $[0\,..\,255]$, ami egyetlen byte-on ábrázolható. Általánosságban is, az egyes képpontokra az $\mathbb{N}_0$ feletti vektortér elemeiként gondolhatunk. A teljes, n pixelsorból és m pixeloszlopból felépülő raszteres felvétel tehát az ily módon megadott pontokból álló

$$X = \begin{pmatrix} x_{(1,1)} & \cdots & x_{(1,m)} \\ \vdots & \ddots & \vdots \\ x_{(n,1)} & \cdots & x_{(n,m)} \end{pmatrix}$$

mátrixszal definiálható. Ez a mátrix (esetenként ennek a sorfolytonos bejárással előálló vektoros megfelelője) modellezi a klaszterezés alapjául szolgáló adathalmazt.*

A megoldandó probléma szempontjából érdekes klaszterek kialakítása ezek alapján lényegében azt jelentheti, hogy a fent definiált pontok közül az egymáshoz jobban hasonlítókat azonos osztályba soroljuk, míg az egymástól távolabbi – tehát nagyobb különbségekkel rendelkező (nem összetévesztendő a képpontok pixelrácson való elhelyezkedései közti távolsággal) – pontokat eltérőbe. A fent definiált X felvétel egy klaszterezését egy $C = \{C_1, C_2, \ldots, C_k\}$ multihalmazok halmazaként formalizálhatjuk, ahol az $\bigcup_{i=1}^{k} C_i$ multihalmaz egyenlő az X-ből alkotott multihalmazzal.

A pontok közti eltérések mértékének kifejezésére különféle távolságfogalmak használhatók, melyek céltól és megközelítéstől függő megválasztásával a pontok (vagyis az azok korábban emlegetett jellemzőinek) egyértelmű összehasonlítására kapunk lehetőséget. Általánosságban egy $\delta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_0^+$ függvény jelölheti a keresett távolságot, ha $\mathcal{X}$ az összes lehetséges, korábbi módon definiált képpontot tartalmazó halmaz. A klaszterező eljárásoknál használt ilyen függvények nem minden esetben – bár gyakran – bírnak a metrikákra jellemző tulajdonságokkal. Talán természetes elvárásnak mondható, ha azt mégis megköveteljük, hogy $x, y \in \mathcal{X}$ esetén a $\delta(x, y) = 0$ tulajdonság pontosan akkor teljesüljön, ha x és y minden koordinátájában megegyezik.

* Érdemes megjegyezni, hogy itt nem a hagyományos matematikai értelemben vett halmazról van szó, hiszen ugyanazon intenzitásértékekkel rendelkező képpontok többszöri előfordulása is lehetséges. A feladat számos megközelítésében szükségtelen a pontok sorrendiségének (pontosabban az egymáshoz való térbeli viszonyaik) figyelembevétele, ilyenkor persze közelebb eshet a mátrix helyet multihalmazzal történő modellezés az adathalmaz kifejezéshez.

A klaszterezések során az egyik leggyakrabban használatos ilyen mérték a szokványos euklideszi távolság – részben a hétköznapi élethez való közelsége miatt is. Ez esetben a függvényünk a következő formájú:

$$\delta(x,\, y) = \sqrt{\sum_{i=1}^{d} |x_i - y_i|^2},$$

bármely $x = (x_1,\, x_2,\, \ldots, x_d)\,,\; y = (y_1,\, y_2,\, \ldots, y_d) \in \mathcal{X}$ képpontokra. A továbbiakban δ-val ezt az értelmezést jelöljük. Egyes algoritmusok expliciten a definiált távolságmértékkel dolgoznak a kívánt eredmény eléréséért, míg mások gyakran implicit módon próbálják közelíteni azok értékeinek optimumát – tehát azt, hogy az azonos klaszterbe sorolt pontok között mért ilyen távolságok minimálisak, míg a különbözőbe soroltak közt maximálisak legyenek.

2.2. A klaszterek kialakításának módszerei

A létező klaszterező algoritmusok számos szemszögből megkülönböztethetők egymástól. Talán az egyik legfontosabb ilyen megkülönböztetés a manuális és automatikus klaszterezések szétválasztása (előbbiben az eljárásunk paraméterezéséhez előzetes ismeretre van szükségünk az eredményként létrejövő csoportok számát illetően, utóbbinál ezzel szemben ennek kiszámítása is a metódusunk feladata), ugyanis a gyakorlatban igen fontos szempontot jelent az egyes bemeneti paraméterek megválasztásának bonyolultsága. Éppen ezért jellemző, hogy az egyes manuális módszerek utólag több különböző automatizált változattal kerülnek általánosításra; ily módon tekinthetünk akár például a későbbiekben részletezett *k-means* és az ISODATA viszonyára is [4]. Hasonlóan megemlíthető például a szigorú és az átlapoló klaszterezések családjainak megkülönböztetése is. A köztük lévő eltérés abban rejlik, hogy lehetővé teszik-e ugyanazon pont több különböző csoportba történő besorolását. A szigorú esetben ez tehát azt jelenti, hogy bármely $i \neq j$ esetén $C_i \cap C_j = \emptyset$ teljesül.

A továbbiakban ettől kicsit távolabbról, a használt alaptechnikák és a létrejövő klaszterek minősége szempontjából közelítjük meg (nem kimerítő módon) a szóban forgó algoritmusok besorolását [5].

2.2.1. *Centroidalapú megközelítések*

A következőkben a *centroidokra* (térbeli alakzatok geometriai középpontjaira) építő, iteratív módszerek kerülnek prezentálásra. Közös ismertetőjük, hogy a csoportosítási folyamat során iteratív módon, adott számú (k darab) klaszter megformálásához k darab klaszterközéppontot (vagy egyszerűen csak középpontot) is felhasználnak. Ezen pontok reprezentálják – egyes időpillanatokban* – a hozzájuk rendelt klaszterek centroidjának értékét, vagyis lényegében a klaszterek pontjainak átlagvektorát; tehát ekkor, a C_i klaszterhez tartozó z_i középpontra

$$z_i = \frac{\sum_{x \in C_i} x}{|C_i|}$$

teljesül. Kezdeti értékük általában sztochasztikus módon kerül kiválasztásra az első iterációt megelőzően. A legtöbb ilyen módszer célja, a szóban forgó megközelítés részeként a klaszterenkénti, középpontokhoz kapcsolódó négyzetes hiba összegének (*sum of squared errors*, avagy SSE), tehát a

$$\sum_{i=1}^{k} \sum_{x \in C_i} \delta(x, z_i)^2$$

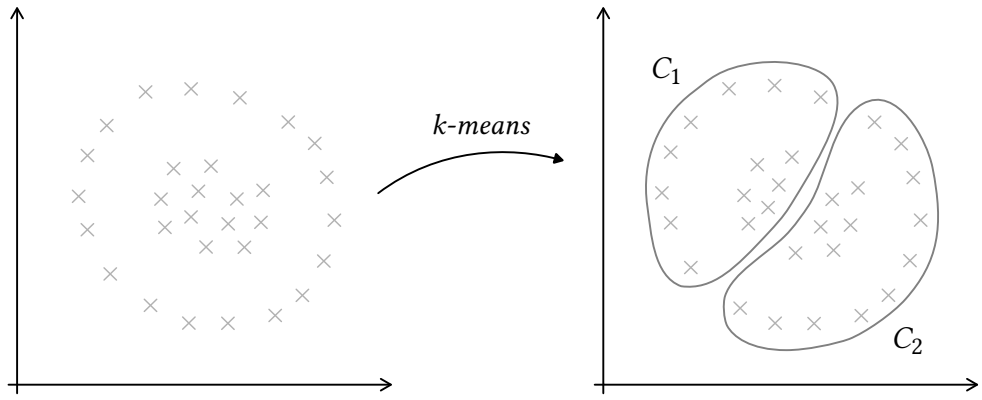
értékének minimalizálása [6].

A *k-means* (k-közép) metódus az egyik, talán legszélesebb körben használt, reprezentatív eljárás a klaszteranalízis területén. Gyakran szerepel a kaszterezés feladatát illusztráló eljárások között, hiszen könnyen átlátható működése egyszerűen megfoghatóvá teszi a legtöbb releváns alapfogalmat. A módszer alapértelmezetten manuális működést tesz lehetővé, azonban (főleg a használhatóság szemszögéből) jelentős gyakorlati haszon társítható az eljárás klaszterek számát megállapító automatizmussal történő kiegészítéséhez. Rengeteg ilyen megoldás létezik: egyes megközelítések a fő iteráció több fix k értékre történő futtatásával, majd a kapott eredmények összehasonlításával adják meg a klaszterek optimális számát (és egyben magukat a klasztereket is) [7]; míg mások alapjaiban megváltoztatják az algoritmus felépítését.

* Természetesen nem mindig igaz, hogy a középpont a hozzá tartozó klaszter centroidja. Ilyen értelemben a centroid-klaszter kapcsolat adott lépések után helytelenné válik és csak egy későbbi frissítő lépéssel alakul ki újra. Példaképp, mint látni fogjuk, a k-means esetén a klaszter pontjainak újraosztása utáni időpontban a középpontban a régi centroid található.

Ilyen például az ISODATA (*Iterative Self-organizing Data Analysis Technique*) algoritmus [8] is. Bár a módszer nagyban hasonlít a *k*-means esetéhez, különbségük, hogy az ISODATA eljárásnak a bemenő adatok legjobb csoportosítása mellett – különböző heurisztikák segítségével – a klaszterek optimális számának automatikus megállapítására törekvés is szerves része; így tehát a felhasználó részéről nem szükséges a kapcsolódó *a priori* ismeretek rendelkezésre állása. Mindezt az algoritmus erre vonatkozó, „integrált" dinamizmusa teszi lehetővé: az iteráció során megszüntetésre, kettéosztásra, vagy akár egyesítésre kerülhetnek a leendő klaszterek, bármikor, amikor ez indokoltnak bizonyul.

Az *X-means* [9] az ISODATA-hoz hasonló céllal módosít a *k*-means eljárásnál látottakon, azzal a különbséggel, hogy itt a *k*-means futtatása és az e során kialakított klaszterstruktúrát javító lépés elkülönülve, felváltva követik egymást a vélt optimum megtalálásáig. A módszer az utóbbi lépés beékelése által tudja bejárni a tetszőlegesen definiált potenciális klaszterszámok intervallumát, mely után a végeredményt a kipróbált értékek legjobbja adja meg. A technika eszköztára ez esetben limitáltabb: csak az arra leginkább alkalmasnak vélt klaszterek megfelelő kettéosztásával módosítja a keresett csoportokat az egyes *k*-means futások közt.



2.2. ábra. A *k*-means hibás működésének illusztrációja. Az egyik klaszter adatpontjai nem az eljárás által „preferált" szférikus alakban helyezkednek el a térben (jelen esetben kör helyett gyűrű), így nem a természetesnek vélhető csoportok alakulnak ki az eljárás eredményeképp.

Érdemes megemlíteni a tárgyalt megközelítés alapjaiból eredő fő negatívumokat, mint például azt, hogy ezen eljárások érzékenyek az iteráció kezdetén helytelenül választott kiindulópontokra, így könnyen lokális minimumba juthatnak, ahonnan később már nem tudnak továbblépni. Adott esetben ez az eljárás többszöri futtatásával egyszerűen elkerülhető. A módszerek alapjait adó SSE és távolságfogalom mennyiségeire történő

építés azt is magával vonja, hogy a térben nem szférikus alakot felvevő eloszlású, vagy a különböző mennyiségű pontot tartalmazó klaszterekkel rosszul birkóznak meg, így egyes esetekben alapjaiban elkerülhetetlen a nem várt eredményre jutás. Erre láthatunk egy példát a 2.2. ábrán. Ezen felül nehezen kezelik még az *outliereket* (tehát a kiugró, többitől távol eső adatpontokat) is, hiszen ezek könnyen elferdítik a középpontok értékeiként számított átlagokat [10, 11, 12].

2.2.2. *Medoidokat használó eljárások*

Egy másik, hasonló megközelítést követnek a *medoidalapú* módszerek (a medoid azon adatpont, amelynek minimális az átlagos eltérése a klaszter tagjaitól). Egy ilyen medoid a H halmaz esetén az

$$\arg\min_{x \in H} \sum_{y \in H} \delta(x,\ y)$$

egyenlettel írható fel. Tehát a geometriai középpontokkal szemben itt ténylegesen a bemenet adatpontjai közül kikerülő középpontok reprezentálják a klasztereket.

A legfőbb ilyen eljárás a PAM (*Partitioning Around Medoids*), mely két fő lépést különít el: a k darab kezdeti medoid értékének mohó megkeresését és a többi adatpont hozzájuk sorolását (BUILD); valamint az ezt követő, iteratív, optimum megtalálásáért felelős javító lépést (SWAP) [13]. Az utóbbi során a módszer az összes meglévő medoidhoz megkeresi a nem-medoid adatpontok közül a legjobb jelöltet a cserére, tehát azt, amelyre leváltva az adott (már meglévő) medoidot, jobb klasztereket kapunk. Ez az iteráció addig halad, amíg ily módon javulást tudunk elérni az eredményben, melynek mértékét általában az egyes pontok és medoidok közti átlagos eltérés mennyisége adja meg [14].

A CLARA (*Clustering Large Applications*) metódusa a PAM alapjaira építve, sztochasztikus módon próbál javítani annak futási idején (kis k érték esetén) [15]. A PAM algoritmusának többszöri, a bemeneti adathalmaz egy véletlen részhalmazára történő, (akár párhuzamos) végrehajtását – és a kimaradó adatpontok legközelebbi klaszterbe sorolását – követően a kapott potenciális csoportosítások közül a legmegfelelőbb kiválasztásával adódik az eljárás végeredménye.

Az irodalomban gyakran említett medoidalapú eljárás a CLARANS (*Clustering Large*

Applications Based on Randomized Search) is. Ez a módszer a medoidok keresését egy gráfkeresésként absztrahálja. A gráf egy-egy csúcsa reprezentálhatja a medoidok egy potenciális halmazát. Azon csúcsok lehetnek összeköttetésben, amelyekhez tartozó ilyen halmazok egyetlen elemükben térnek el; minden csúcs felcímkézhető a korábbiakban látottakhoz hasonló, a medoidok által definiált klaszterezés minőségét mérő értékkel. Látható, hogy ez az absztrakció a PAM és a CLARA esetére is illeszthető: az előbbinél mohó módon keressük az optimális csúcsot a szomszédokon történő végighaladással, utóbbinál pedig egy részgráfra limitáljuk ezt a folyamatot. A CLARANS újdonsága ezekkel szemben abban rejlik, hogy a teljes keresési tér fontolóra vétele helyett, illetve a keresés erőteljes kezdeti korlátozása nélkül, az adott lépésekben – dinamikus módon – a végigjárandó szomszédok közül csak azok egy véletlen részét ellenőrzi [14].

A fent bemutatott eljárások elvetik a centroidalapú módszerek alapját képező átlagokat, ezáltal robusztusabb viselkedést elérve az outlierekkel és zajjal szemben. A távolságok minimalizálása miatt itt is szférikus térbeli elhelyezkedésű adatpontokkal rendelkező bemenetre várható az igazán jó eredmények elérése, viszont a heurisztikus inicializációnak köszönhetően gyakran nagyobb eséllyel kerülik ki a lokális minimumba ragadás veszélyét. Láthattuk, hogy a legtöbb variáns a futási idő csökkentését helyezte fókuszává, aminek oka, hogy az alap megoldás viszonylag nagy számítási bonyolultsággal rendelkezik, beleértve ebbe a memóriaigényt is. Ezeken azonban a bemutatottak mellett számos további megvalósítás tudott sikeresen javítani [13].

2.2.3. *Sűrűségalapú metódusok*

Az eddig ismertetett módszerek egyik legfőbb hátulütője, hogy bizonyos struktúrájú inputot helytelenül kezelnek. Ennek okozója az, hogy első sorban a kialakított klaszterek minőségét jellemző kompaktsági (*compactness*) kritériumnak próbáltak megfelelni az adatpont-középpont távolságok minimalizálásával – az összekapcsoltsági (*connectivity*) kritérium viszont háttérbe szorult. Ezen változtatnak a pontok sűrűségét középpontba helyező módszerek.

A sűrűségalapú klaszterezések családjának reprezentatív tagja a DBSCAN (*Density-based Spatial Clustering of Applications with Noise*), alapját az alábbiakban vázolt fogalmak képzik. Az egyes adatpontok magpontnak számítanak, ha (az ε paraméterrel) adott sugarú

$N_\varepsilon(x) = \{ y \in X \mid \delta(x, y) \leq \varepsilon \}$ környezetük egy bizonyos p_{min} küszöbértéknél nagyobb számú pontot tartalmaz; határpontoknak nevezhetők azok a pontok, amelyekre ez a tulajdonság nem teljesül. Azon x pontok, melyek egy z magpont ε-környezetében helyezkednek el, tehát $x \in N_\varepsilon(z)$ és $|N_\varepsilon(z)| \geq p_{min}$ teljesülnek, közvetlenül sűrűség-elérhetők ebből a magpontból. A reláció szerint összekapcsolt pontokból képezhető sorozatok végpontjait nevezzük a kezdőpontból sűrűség-elérhetőnek. Az így kapott relációt szimmetrikussá téve definiálható a sűrűség-összekapcsoltság, mely tehát akkor teljesül két pontra, ha létezik olyan magpont, amelyből adott környezet szerint mindkét pont sűrűség-elérhető. A DBSCAN eljárása az utóbbi fogalom szerint összekapcsolható pontokból kialakított csoportok létrehozásával törekszik az adathalmaz klasztereinek megformálására [16].

A széles körben ismert DBSCAN egyik fő hibáját kívánja orvosolni az OPTICS (*Ordering Points to Identify the Clustering Structure*), mely nevezetesen abból ered, hogy a legtöbb adathalmaz pontjainak sűrűsége nem jellemezhető jól fix paraméterértékkel. Az egy klaszterbe tartozó pontok sűrűsége például klaszterenként nagy mértékben eltérhet, ami a DBSCAN esetében hibás eredményhez vezet, függetlenül a paraméterek megválasztásától. Ezt elkerülendő, az OPTICS lényegében egy adott korlát alatti lehetséges ε értékek melletti ilyen klaszterezések kiszámítását közelíti [17]. Az eljárás fontos tulajdonsága, hogy a végleges osztályok kialakítása nem képezi részét, ez tehát további utómunkát igényel – a módszer az ehhez kellő információt biztosítja a feldolgozott pontok sorrendbe tételével, és a hozzájuk kapcsolható távolságértékek kiszámításával. Az utóbbi szerint egymáshoz közel eső pontok egy klaszterbe osztandók, így tehát a sorrendben egymást követő alacsony távolságértékek adják meg a klasztereket (ahol is a kisebb értékek nagyobb sűrűségre utalnak). Ezeket pedig a sorban szereplő kiugró értékek különítik el egymástól.

A sűrűségalapú eljárások közös tulajdonsága, hogy kevésbé korlátozza a kimeneteik minőségét az adathalmaz pontjainak elhelyezkedése; emellett a zajos, outlierekkel tarkított bemenettel is jól megbirkóznak. Természetesen ezek a módszerek sem maradtak hátulütő nélkül: jellemzőjük a relatíve magas számítási bonyolultság, illetve gyakran a felhasználást megnehezítő paraméterek jelenléte. A DBSCAN ε-jának megfelelő megadása például általában nem intuitív – a k-means k-jához hasonlóan, előzetes ismeret birtoklását igényli [18]. Ennek orvoslására számos újítás lelhető fel a szakirodalomban, így például az ε paramétert eldobó (és egyben jelentőségteljes gyorsulást is hozó) HDBSCAN [19], vagy a teljesen paramétermentes DEBARA [20] eljárások.

2.2.4. *Hierarchikus módszerek*

A hierarchikus módozatok lényege, hogy az adatpontok csoportjait klaszterek hierarchiájának – alulról felfelé, vagy épp fordított irányba haladó – felépítésével alakítják ki. Az előbbi, *agglomeratív* típus esetén kezdetben minden pont külön klasztert ad, a hierarchiában felfelé haladva, lépésenként történik csoportokba vonásuk; az utóbbi, *divizív* esetben pedig egyetlen klaszterből kiindulva, rekurzívan osztjuk fel a pontokat. Az eljárás során kialakuló hierarchiát egy *dendogram* fastruktúrájával szokás ábrázolni.

Az egyik alapvető agglomeratív módszer a *single-linkage* (vagy *legközelebbi szomszéd*) klaszterezés, melyben azon csoportok összevonásával épül fel a hierarchia, melyek a legkisebb távolságú, különböző klaszterekben szereplő pontokból képzett pontpárt tartalmazzák. Az algoritmus számítási bonyolultsága meglehetősen nagy, de számtalan közelítő eljárás született, amelyek gyorsabb működést biztosítanak: mint például az SLINK [21].

A BIRCH (*Balanced Iterative Reducing and Clustering Using Hierarchies*) megpróbál javítani az előtte szereplő klaszterező eljárások memóriaigényén, azáltal, hogy dinamikus módon (az adatpontok folyamatos beillesztésével) épít egy *clustering feature* fát. Ez egy specifikus tulajdonságokat teljesítő adatstruktúra, melynek levelei bizonyos alklasztereket reprezentálnak [22]. Az eljárás az összes adatpont helyett ezeket a leveleket felhasználva, egy hierarchikus klaszterezést futtat le (ezt követheti még egy opcionális finomító lépés az esetleges pontatlanságok javítására); tehát lényegében az összes adatpont helyett egy tömörített struktúrát kell a memóriába tölteni. Megjegyzendő, hogy a módszer nem feltétlenül sorolandó a hierarchikus eljárások közé, hiszen a megfelelő módosításokkal a klaszterező lépés más metódusokra cserélhető – így akár a *k*-means eljárására is.

2.2.5. *Felügyelt klaszterezés*

Egy másik szemszögből tekintve, a klaszterezés feladatának megoldásait elválaszthatjuk annak függvényében is, hogy *felügyelet nélküli* (mint az eddig felsorakoztatott eljárások mindegyike), vagy pedig *felügyelt* elvű működést mutatnak. Az utóbbi lényege, hogy a felhasználó rendelkezésére áll valamilyen ismeret, ami alapján egy általánosabb probléma megoldását keresi. Esetünkben leggyakrabban ez egy extra, úgynevezett tanító lépés beékelődését jelenti, a keresett adatcsoportok kialakításának folyamatát megelőzően.

Az említett tanító lépés feladata, egy, a kitűzött cél elérésére alkalmas modell megépítése, a birtokunkban lévő tanító adathalmazra* alapozva, a benne rejlő összefüggések feltérképezésével. A kapott modell segítségével később más adatból is ki tudjuk nyerni a megfelelő információt. Egy ilyen modellt felhasználva kaphatunk predikciót „tetszőleges" bemenetre a keresett klaszterek alakulásáról.

A *random forest* egy felügyelt – klaszterek kialakítására alkalmas – módszer, amely a „döntési fákból felépülő *ensemble*" megnevezéssel foglalható össze [23]: az önálló döntési fák hibáit javítja, bizonyos mértékű sztochasztikus viselkedés hozzáadásával. Ennek lényege, hogy több, véletlenszerűen variált fából álló erdő erősségeinek összesített kiaknázására törekszik. A modell erdejeit alkotó döntési fák fogalma leggyakrabban a gépi tanulás témakörénél lelhető fel, de széles körben felbukkan sok egyéb felhasználási területen is. Ezek egy-egy (általában bináris) fastruktúrát alkotnak, melyben az egyes csúcsok – a gyökérből kiindulva – bizonyos kérdéseket tesznek fel az inputtal kapcsolatosan, csak hogy az erre kapott válasz szerint kettéágazva újabb csúcsok következhessenek, újabb kérdéssel; míg végül el nem jutunk az inputra vonatkozó konklúzióhoz. A kapott végleges megállapítások alkotják a fa leveleit. A random forest eljárása a döntési fák erdejének kialakítása után az általánosabb, vagyis a tanításhoz felhasználttól eltérő (ismeretlen klaszterekkel rendelkező) input adathalmaz adatpontjait a fák által adott besorolási predikciók alapján tudja csoportosítani.

2.2.6. *Egyéb technikák*

Az eddig tárgyalt megközelítések mellet természetesen számos más klaszterezési módszer látott már napvilágot, a matematika és a számítástudomány különböző területeiről származó ötletekre alapozva. Léteznek például eloszlásalapú, gráfalapú, neurális hálók modelljére építő, de még biológiai folyamatok által inspirált megoldások is [24].

2.3. A kapott eredmények kiértékelése

A klaszterezés problémájának egyik legkevésbé triviális részfeladata a jó klaszterek koncepciójának definiálása, tekintve, hogy ez felhasználási területenként nagyban eltérhet,

* Ez alatt például egy olyan adathalmazra gondolhatunk, amely a bemeneti adatpontok mellett tartalmazza az ezekhez tartozó, ismert kimenetet is.

sőt, akár inputonként más és más tulajdonságokkal bíró eredmény minőségét tekinthetjük az elvárthoz közel esőnek. A kapott klaszterek színvonalának felmérését, illetve a különböző eredmények összehasonlítását több irányból közelíthetjük meg [25]: *internális* módon, bizonyos mérőszámok alapján; *externálisan*, egy létező, általánosan elfogadott eredménnyel történő összevetés által; vagy akár manuálisan, tehát egy, a problémakörben jártas személy inspekciója által. Mindegyik technikának megvan a maga fő hátulütője, névlegesen elsősorban az alábbiak fontolandók meg.

- Az internális esetben a mérőszám definiálása önmagában is felfogható egy minimalizálandó klaszterezési célértéknek, hasonlóan tehát ahhoz, amit a konkrét eljárás is alkalmaz. Így például egy távolságalapú kritériumot minimalizáló klaszterező eljárás eredményéhez (szemben egy másik metodológiával megszerzett eredménnyel) elfogult módon magas értéket adhat egy szintén ilyen, távolságokra építő internális mérőszám. Emellett ezek a mértékek általában nagyban függenek az *a priori* paraméterektől, mint a létrehozandó klaszterek száma [25].
- Az externális minősítés problémája, hogy természetszerűen gyakran nem áll rendelkezésre egy ismerten helyes eredmény, amihez hasonlítani tudnánk a saját kimenetünket. Egy ilyen eredmény származhat akár például egy másik elfogadott eljárás végrehajtásából, de ekkor ismét felmerülhet az klaszterek validálásának problémája.
- Végül, az emberi felülvizsgálat során az ezzel megszokott módon járó problémák merülnek fel, mint a szubjektivitás, vagy a figyelmetlenségből adódó hibák lehetősége. Emellett ez a folyamat nagy mennyiségű adat esetén igen nehézkessé válhat.

Ezen akadályok elkerülésének legjobb módja lehet tehát, ha több megközelítést együttesen használva próbálunk megfelelő minősítést keresni.

2.3.1. *Internális indexek*

A legtöbb internális mérőszám, vagy CVI (*Clustering Validity Index*) a kialakult klaszterek kompaktságának és szeparáltságának (vagy gyakran az ezzel ellentétes összekapcsoltságának) mértékére alapozva épül fel. Egy ideális eredményben minden klaszter magas kompaktságú, tehát a tartalmazott adatpontjai közel állnak a klaszter reprezentáns középpontjához; egyben minden klaszterpár magas szeparáltságú, vagyis a különböző csoportok pontjai távol helyezkednek el egymástól.

A CVI-k közül kiemelendő a gyakorlatban (beleértve a távérzékelés területét is) sokszor használt *Davies–Bouldin index*, vagyis az

$$\frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\frac{\mu_i+\mu_j}{\delta(z_i,\,z_j)}$$

érték, ahol k a klaszterek száma, z_i a C_i klaszterhez tartozó centroid, μ_i pedig a C_i klaszter pontjainak z_i-től vett, δ szerinti átlagos távolsága. A képlettel kapott, nullához minél közelebb eső értékek indikálják a jobbnak vélhető klaszterezést.

Hasonló mérőszám a *Caliński–Harabasz index* is [26], $n \cdot m$ bemeneti képpont esetén:

$$\frac{B}{k-1}\cdot\frac{n\cdot m-k}{W}.$$

A képletben B a klaszterek centroidjai és a globális centroid közti szeparáltságot, W pedig a klaszteren belüli pont-centroid távolságnégyzetek összegeit jelöli, avagy

$$B=\sum_{i=1}^{k}|C_i|\;\delta(z,\,z_i)^2 \text{ valamint } W=\sum_{i=1}^{k}\sum_{x\in C_i}\delta(x,\,z_i)^2,$$

a korábbi jelölésekkel – a z összes pontra vonatkozó (globális) centroid mellett. Ezen mennyiség akkor lesz nagyobb, ha a kapott klaszterek sűrűsége magas, és a klaszterek egymástól jól szeparáltak.

Mindezek mellett említésre érdemes mérték még a *Xie–Beni index*, vagy a távérzékelés felhasználási területén Li és tsai. által legideálisabbnak talált WSJI [27] internális validitási indexei is.

2.3.2. *Externális mértékek*

Az externális minősítés esetéhez kötődően is érdemes megemlíteni néhány, a kapott eredményt a birtokunkban lévő viszonyítási alappal történő összevetésére használt mérőszámot [28]. Az egyik legegyszerűbb, könnyen átlátható ilyen mérték az eredményre vonatkozó *purity* (tisztaság) mennyisége:

$$\frac{1}{|X|}\sum_{i=1}^{k}\max_{j}|C_i\cap T_j|,$$

ahol T_j az összehasonlításhoz használt célosztályok egyike. Láthatjuk tehát, hogy az eredmény klaszterei és a célosztályok közti megfeleltetést ez esetben úgy kapjuk meg, hogy minden, az eredmény részét képező klaszter esetén megkeressük azt a célosztályt, amellyel a legtöbb közös tartalmazott képpontja van. Ezt a megfeleltetést felhasználva a purity (0 és 1 közti) mennyisége az említett közös pontok száma alapján értékeli az eredményünk minőségét: az elvárthoz közeli output tisztasága közvetlenül 1 alatt mozog. A mérték egyik fő hibája, hogy sok, kevés pontot tartalmazó klaszter esetén (ha a célosztályaink számossága alacsonyabb) könnyen kapunk 1-hez közeli értéket: ebből adódóan egyes esetekben ez a módszer kevésbé bizonyul alkalmasnak a különböző számú klaszterrel rendelkező eredmény és viszonyítási alap összevetésére. Hasonló a helyzet a nem egyenletesen felosztott klaszterek esetén is – amikor is egyes célcsoportok jóval több pontot tartalmaznak a többitől: ekkor a purity értéke akár úgy is magasan maradhat, ha a kis célosztálynak megfelelő klaszter létre sem jött a kapott outputunkban.

Egy másik lehetőség két – akár különböző k értékekkel elért – besorolás, ez esetben információelméleti fogalmakra alapozott összevetésére az úgynevezett *Normalized Mutual Information* mennyisége. Ez egy másik mértékre, a *Mutual Information*-re alapozva, annak (a klaszterek és a célértékek halmazainak entrópiáját felhasználva történő) normalizálásával éri el, hogy a különböző klaszterszámok mellett kapott outputok elfogultság nélkül hasonlíthatók legyenek. Az NMI, a purity esetével egyező módon 0 és 1 közötti értéket vehet fel, ahol a magasabb értékek jelölik a viszonyítási alaphoz közelebb eső klaszterezést.

A korábbiakhoz kapcsolódóan, a további számos létező externális mérőszám közül nem utolsó sorban megjegyzendő a szintén nagy gyakorisággal alkalmazott *F-measure*, valamint a *rand index* is.

3. FEJEZET

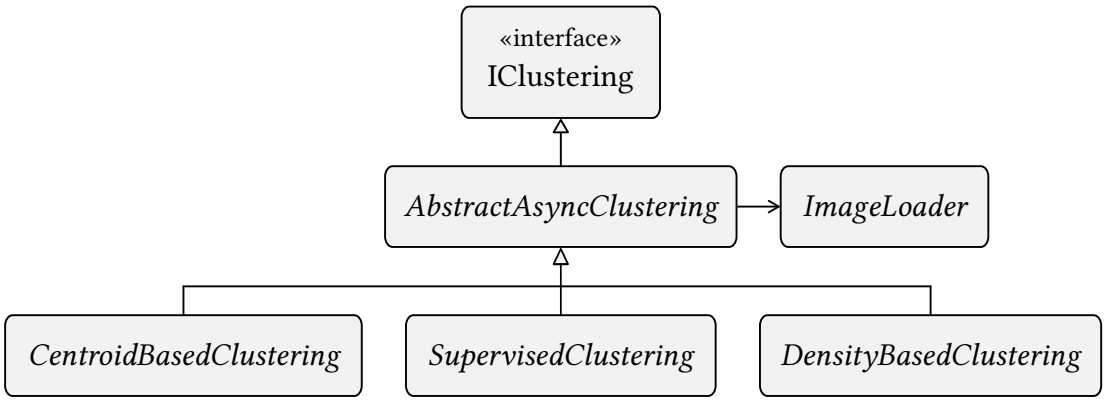
METODOLÓGIA ÉS IMPLEMENTÁCIÓ

Általános eljárások gyakorlati vizsgálatának – így azok ilyen jellegű összehasonlításának is – elengedhetetlen eleme a metódusok valamely konkrét megvalósításának rendelkezésre állása. A korábbiakban bemutatottak alapján természetesen a klaszterezés általános modellje könnyen átültethető a távérzékelés, és így a drónfelvételek esetén is szóba kerülő fogalmakra. Ettől függetlenül számos optimalizációs lehetőség és kiaknázásra váró heurisztika rejlik a feladat konkrét felhasználási területének ismeretében. A továbbiakban áttekintjük a későbbi vizsgálat alapját képező módszerek felépítését a drónfelvételekre történő alkalmazás szemszögéből, illetve az ezek realizációja során felmerülő főbb kérdéseket és problémákat.

3.1. A megvalósítás áttekintése

A vizsgálat célpontjaiként az előző fejezetben bemutatott, széles körben használt algoritmusok sorából esett választás több, különböző megközelítést alkalmazóra – ezáltal reprezentálva több különféle, a feladatkörhöz kapcsolódó lehetséges megoldási stratégiát. Természetesen az egyes algoritmusok számos különböző variánsa került már bemutatásra a szakirodalomban. Esetünkben ezek közül a leginkább átfogó, általános célú változatok adták a kiindulási alapot (melyek eredményeinek pontossága mindemellett várhatóan a távérzékelésben megkövetelhető mértékekhez közeli). Ez a következőket jelentette: a *k*-means esetén a Forgy/Lloyd [10], az ISODATA-nál a Tou és Gonzalez féle [29], míg a random forest algoritmusánál a Breiman által bemutatott [30] verziót. Végül, de nem utolsó sorban a DBSCAN módszerénél a megvalósítás Ester és tsai. eredeti megközelítésre [16] építkezett.

Az eljárások megvalósításai C# programnyelven, .NET környezetben, a Giwer programcsomag fejlesztésének részét képezik. A metódusoknak készen kell állniuk a drónnal rögzített, potenciálisan nagy méreteket öltő felvételekre is – még ha a műholdas felvételek gyakori hiperspektrális voltával ellentétben itt csak multispektralitásról beszélhetünk (aminek lényege, hogy a drónokra felszerelt érzékelők nem összefüggő frekvenciasávokat térképeznek fel, hanem csak ezek bizonyos diszkrét tartományait); a keletkező képek egyes pontjainak dimenzionalitása így jóval kisebbnek tekinthető. Szükséges tehát, a gyakran egymást ellentétes irányba mozgató memóriaigény és futási idő egyensúlyának megfelelő finomhangolása. Különösen igaz ez a centroidalapú metódusokra, hiszen alaphelyzetben ezeknek feltétele az összes adatpont rendelkezésre állása az egyes iterációkban. Tehát itt a kellő sebesség biztosításához érdemes memóriában tartani a teljes felvételt, az összes spektrális sávjában fellelhető minden intenzitásértékkel, ami már egy relatíve magas alsó korlátot jelenthet a felhasznált memória mennyiségére. Nem hagyható persze figyelmen kívül ezek mellett az egyéb általános tervezési alapelvek (mint például az újrafelhasználhatóság) kellő mértékű teljesülése sem.



3.1. ábra. A klaszterezési logikához tartozó osztálydiagram. Az interfész közvetlen őse egy absztrakt osztály, amely definiál néhány alapvető, minden konkrét klaszterezéshez szükséges mozzanatot, így például a képpontok adott formátumú betöltését (amihez a szintén absztrakt *ImageLoader* leszármazottai nyújtanak segítséget).

A fejlesztési környezet megválasztása önmagában szűkíti az alkalmazható tervezési és megvalósítási stílusok halmazát, így tehát ebből és a korábban említett szempontokból adódóan adja magát az objektumorientált megközelítés alkalmazása. A fejezet tárgyát képező módszerek megvalósításaként így első sorban egy, az újrafelhasználást segítő osztályhierarchia felépítése történhetett meg. Ennek fő eleme a klaszterezés folyamatához tartozó interfész, mely biztosítja a felhasználó számára az egyes eljárások futtatása mellett

azok megfelelő paraméterezését, mind az általános esetre, mind a konkrét eljárásra vonatkozóan. Erre az interfészre építve tetszőleges logika alapján tudjuk a bemeneti felvételünk kívánt klasztereit előállítani. Az így definiált hierarchia váza a 3.1. ábrán tekinthető át.

Mivel az egyes megvalósítások – alkalmazkodva a fejlesztés környezetéhez – 8 bites felvételek fogadására lettek felkészítve, így lényegében a legtöbb folyamatra egyszerűen (a multispektrális bemenet esetén többdimenziós) byte-tömbökkel folytatott munkaként gondolhatunk. Ezen megfontolásból fogalmazza meg az interfész szintén byte-ok (bár ez esetben már egyetlen dimenziójú) tömbjeként a képpontokhoz rendelt klasztereket reprezentáló eredményt is, mely tehát így 256 különböző csoport megkülönböztetésének lehetőségét biztosítja: ez gyakorlati szempontból a légi- és műholdfelvételek vizsgálata esetén bőségesnek bizonyul.

3.2. A CENTROIDALAPÚ ELJÁRÁSOK REALIZÁCIÓJA

A *k*-means és az ISODATA közös tulajdonsága, hogy centroidalapú megközelítés szerint, iteratív módon keresik a megfelelő csoportokat. Ezt kihasználandó, adja magát egy (az imént bemutatott interfészt implementáló) közös absztrakt ősosztály definiálásának lehetősége. Ezen ős tartalmazhat számos olyan eljárást, amely mindkét metódusban felhasználásra kerül majd, mint például az egyes képpontok klaszterközépponthoz rendelése, vagy ezen középpontok értékének a pillanatnyi centroidra való frissítése. Hasonlóan érdemes itt definiálni a későbbiekben gyakran alkalmazott, összes klaszterhez tartozó SSE értékének kiszámításának módját.

Közösnek tekinthetők akár a két klaszterező módszer kezdeti középpontjainak értékeit inicializáló módszerek is, ha ezek nem is egyeznek meg az egyes implementációk esetén. Érdemes megemlíteni, hogy, még ha az ilyen eljárások megválasztása nem is feltétlenül rendelkezik megrendítő erejű hatással a klaszterezés végkimenetelére [31], mindenképp érdemes szemügyre venni több lehetőséget, a megfelelő választással potenciálisan csökkentve a lokális minimumba jutás lehetőségének esélyét. Esetünkben a megvalósítás a *k*-means eljárásánál a leendő középpontok távolságát maximalizálni törekvő módszert követi, mely ehhez a bemeneti képpontok közül jelöli ki a megfelelő számú, egymástól (a klaszterezés által használt távolságfüggvény szerint) lehető legtávolabb fekvőt. Az

ISODATA algoritmusánál feltételezhető, hogy a választás a módszer klaszterek formálására és megszüntetésére vonatkozó dinamizmusából fakadóan kisebb jelentőséggel bír – így itt a folyamat egyszerűen a képpontok közül történő véletlenszerű választásként került definiálásra, aminek természetes előnye a művelet alacsony futási ideje.

Megjegyzendő, hogy a legjobb eredmény elérése érdekében ezen felül fontos lehet a felügyelt inicializálás megvalósítása is: ennek keretében a felhasználó által manuálisan kijelölt képpontok figyelembevételével választjuk meg a kiindulási középpontokat [32]. Alapesetben az eljárások az inicializációtól függően változó azonosítókat rendelnek az eredményül kapott klaszterekhez, így ezek értelmezése további utómunkát igényel (például az azonosítók felcímkézését a klaszter képpontjai által reprezentált földfelszíni területek tematikus kategóriájával). Ezen probléma enyhítését is szolgálhatja a manuális inicializálás megfelelő implementációja: a kezdeti pontválasztás során lényegében az említett tevékenységet előzetesen végezhetjük el, azzal a járulékos haszonnal, hogy a kialakult klaszterek minőségét is pozitív irányba tolhatjuk el.

Az újra felhasználható dizájn megalkotásának eleme még esetünkben egy, a korábban ecsetelt centroid-klaszter kapcsolatot leíró osztály létrehozása is, ahol definiálhatjuk a specifikus metódusokat. De ehhez hasonlóan, az egyes klaszterközéppontokat és a releváns elemi műveleteket befoglaló tagot is felvehetjük a kialakított osztályhierarchiába. A fentiek megléte mellett pedig a már realizált ősosztályból történő leszármazással, majd lényegében néhány metódus meghívásával lebonyolítható a *k*-means eljárás iterációinak megszervezése; az ISODATA esetén ehhez természetesen szükséges még a korábban részletezett dinamizmus elemeinek beépítése. A továbbiakban a két algoritmus kerül részletesebb ismertetésre.

3.2.1. *A k-means módszer*

A *k*-means algoritmusának legalapvetőbb változata viszonylag kevés, de annál nagyobb jelentőségű paraméter megadásával indítható útjára a felhasználó által. Ahogy azt a metódus elnevezése is sugallja, *k* értékével adhatjuk meg a leendő csoportok számát. Ehhez azonban szükséges előzetes információ birtokában lenni a kialakítandó klaszterekre vonatkozóan, ami természetszerűleg sok esetben nem teljesül.

A klaszterek optimális számának megtalálására számos technika létezik [33]. Egy ilyen megközelítés az *elbow módszerrel* való kiegészítés is, mely a k_{min} és k_{max} értékeivel

definiált intervallumból kikerülő klaszterszámokkal történő, különböző k-means futások által adódó eredmények SSE értékeivel felvázolható görbe könyökpontját keresi meg. Az ily módon kapott pont adja meg a feltételezetten ideális k értéket. A módszer előnye az automatizálás megvalósítására például az, hogy a realizációban a k-means futtatások könnyen párhuzamosíthatók. De az is megemlíthető, hogy a megfelelő paraméterezéssel a manuális k-means-t kapjuk vissza, ezzel nagyobb kontrollt adva a felhasználónak – arra az esetre, ha mégis rendelkezik a leendő klaszterekre vonatkozó információval.

A klaszterszám mellett a k-means fontos paramétere az $iter_{max}$ is, amely felső korlátot ad a végrehajtandó iterációk számára. Végül c_{max} szabja meg, hogy az SSE értékének milyen mértékű változása mellett nem érdemes az iterációt tovább folytatni. Így tehát megállásra bírhatjuk az algoritmust, ha az már elenyészőnek vélt javulást volna csak képes felmutatni.

3.1. algoritmus. A k-means eljárása, kibővítve az X bemeneti felvétel k klaszterszámának automatikus megtalálásáért felelős elbow módszerrel.

```
1.  errors := EmptyArray(k_max − k_min)
2.  FOR k := k_min TO k_max DO
3.      Z := InitializeCenters(k)          // k darab kezdeti középpont inicializálása
4.      iter := 1
5.      WHILE CurrentSSE( ) / PrevSSE( ) < c_max AND iter ≤ iter_max DO
6.          FOREACH x ∈ X DO
7.              x.AssignToClosestCenter( )
8.          END
9.          FOREACH z ∈ Z DO
10.             z.SetToCentroidOfAssignedPoints( )
11.         END
12.         iter := iter + 1
13.     END
14.     errors[k] := CurrentSSE( )
15. END
16. k_opt := ArgElbowOf(errors)  // az SSE-k könyökpontjához tartozó k megkeresése
17. RETURN PointAssignment(k_opt)        // a k_opt-hoz tartozó iteráció eredménye
```

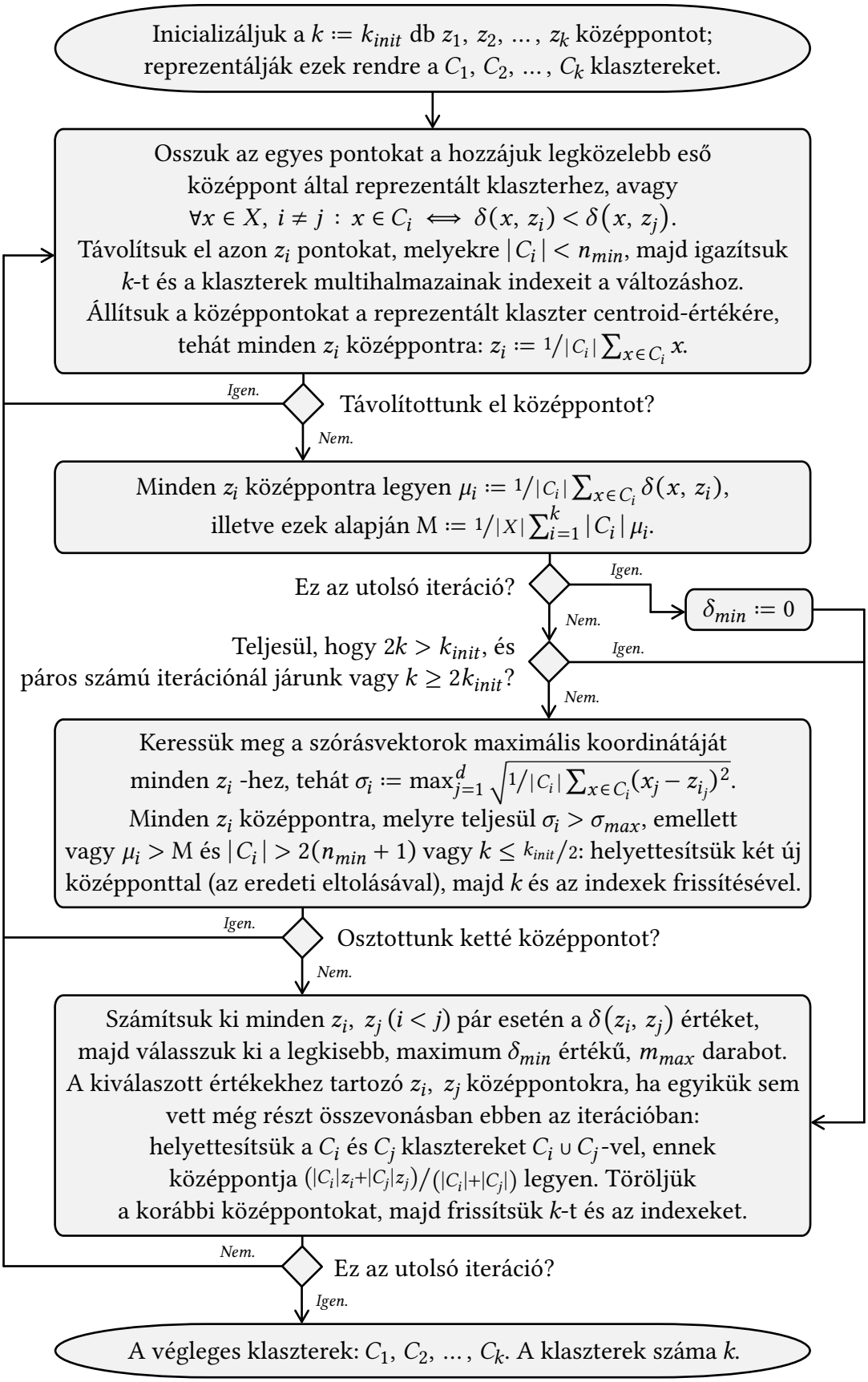
Az eljárás fő lépései (a Forgy/Lloyd által eredetileg bemutatott változatában [10]) az alábbiak szerint vázolhatók fel. Kezdésként inicializálunk k db klaszterközéppontot: értékük általában véletlenszerű módon kerül kiválasztásra az adatpontokat is tartalmazó intenzitástérből. Ezután kezdődhet meg az iteráció, mely során felváltva hozzárendeljük a pontokat a hozzájuk legközelebb eső klaszterhez, majd pedig a korábbiakban már látott módon frissítjük a hozzárendelt klaszterközéppontokat a klaszterek centroidértékékével. A pontok klaszterhez sorolása lényegében a következőképpen formalizálható:

$$\forall x \in X,\ i \neq j:\ \ x \in C_i \iff \delta(x, z_i) < \delta\big(x, z_j\big),$$

ahol C_i az i. klaszter pontjainak multihalmaza, z_i pedig az ehhez rendelt középpont. Az iterációk száma a paraméterek által kontrollált mértéket ölti, így tehát minden iteráció kezdetekor szükséges ellenőriznünk, hogy az iterációk közti SSE változás aránya kisebb-e c_{max}-tól, ha igen, akkor továbbhaladunk. Az, hogy milyen lépésszám után érhetjük el a beállított küszöböt, erősen függ az inicializáló lépés sikerességétől. Utóbbi indokolja egy felső korlát $iter_{max}$ paraméterben történő megszabását. A leírtak alapján adódik tehát a k-means fő eljárása, amit a 3.1. algoritmus pszeudokód formájában foglal össze.

3.2.2. *Az ISODATA algoritmus*

Eddig láthattuk, hogy az ISODATA fő megkülönböztető jegye, hogy a klaszterek számának megállapítására vonatkozó feladatot átveszi a használójától. Ez persze nem jelenti azt, hogy nincs is szükség a metódus megfelelő paraméterezésére. A felhasználónak kell megadnia például a kezdeti klaszterszámot (k_{init}), ahonnan indulva találhatja majd meg végül az algoritmus a k szám optimumát. Egyes implementációkban (így megvalósításunkban is) ezen kívül felső és alsó korlátot is adhatunk előzetesen erre a számra, segítve a várt eredményhez való konvergenciát. Az iteratív jellegből adódóan a legtöbb realizáció arra is lehetőséget ad, hogy az iterációk számát ($iter_{max}$) kontrolláljuk. A klaszterek minimális elemszámát n_{min} adja meg, de szintén a használó feladata, hogy σ_{max} értékét beállítsa a pontok és a hozzájuk tartozó klaszterközéppontok koordinátái között maximálisan megengedett szórás értékére. Ide kapcsolódóan, fontos paraméterek még a klaszterközéppontok közti minimálisan szükséges távolság (δ_{min}), valamint nem utolsó sorban az iterációnként maximálisan összevonható klaszterek számát megadó m_{max} is [29].



3.2. ábra. Az ISODATA működését bemutató folyamatábra.

A módszer alapját a *k*-means eljárásánál is látott centroidalapú működés jelenti (az ezzel kapcsolatos lépések éppen ezért itt már nem kerülnek részletezésre). E köré épül a klaszterek számát optimalizáló heurisztikus műveletek sorozata [31]. A továbbiakban bemutatott lépések iterációnként hajtódnak végre; egyes megvalósításokban (így a Giwer projektben szereplőben is) ezek kizárják egymást, tehát egy iterációban csak egyfajta, a klaszterek számát módosító művelet végrehajtása történhet meg [4].

1. Elsőként, minden iterációban azon klasztereket távolítjuk el*, melyek n_{min}-nél kevesebb ponttal rendelkeznek. Alapesetben ezt a lépést veszi körül a képpontok megfelelő klaszterbe osztásának, illetve a középpontok centroidértékkel történő frissítésének folyamatai.
2. Ezután, bizonyos feltételek teljesülése mellett kettéosztunk minden olyan klasztert, mely középpontjának koordinátáját a tartalmazott pontok megegyező koordinátáival összekötő vektorok szórása közül a legnagyobb ilyen (valamely koordinátához tartozó) szórásérték nagyobb, mint σ_{max}; illetve vagy a klaszer pontjainak középponttól vett átlagos távolsága nagyobb az ugyanezen távolságátlagok összes klaszterre vetített átlagánál és túl nagy a klaszterünk, vagy pedig túl kicsi a különböző klaszterek száma. A kettéosztás annyit jelent, hogy a korábbi klasztert töröljük, majd két újat vezetünk be helyette. Ezek középpontjait az eredeti középpont, az előbbiekben megkapott maximális szórású koordinátája mentén – két különböző irányba – történő eltolásával kapjuk. Ennek mértékét úgy kell eltalálnunk, hogy jelentőségteljes változást segítsünk elő, anélkül, hogy túl messze kerülnénk a korábban sikeresen elért pozíciótól [29].
3. Végül említésre érdemes még a klaszterek összevonásának folyamata. Ehhez vesszük azon (egyedi) klaszterpárokat, amelyek középpontjainak távolsága nem haladja meg a δ_{min} értéket. Kiválasztjuk közülük azt a maximum m_{max} darabot, amelynek említett távolságértéke sorrendben a legkisebb. Ezután végighaladunk ezeken a párokon, és – ha a pár egyik tagja sem vett még részt összevonásban a jelenlegi iterációban – egy újjal helyettesítjük a két klasztert, melynek középpontja a korábbiak (a hozzájuk tartozó klaszter elemszámával) súlyozott átlaga.

A teljes algoritmus működésének részleteit ezek alapján a 3.2. ábra vázolja [4], a főbb lépéseket pedig pszeudokód formájában a 3.2. algoritmus illusztrálja.

* Itt, és a továbbiakban is, gondoljunk a klaszterekre (a tartalmazott pontokra) és a hozzájuk tartozó középpontokra egy entitásként. Így például egy klaszter törlése során a hozzárendelt középpont is törlésre kerül – konkrét implementációban adott esetben a klaszter törlése akár ki is hagyható, elég a középpontot eltávolítanunk. A többi ilyen műveletnél analóg módon járhatunk el.

3.2. algoritmus. Az ISODATA eljárás pszeudokódja az X bemeneti felvétel esetén.

```
1.  Z := InitializeCenters(k)              // k darab kezdeti középpont inicializálása
2.  FOR iter := 1 TO iter_max DO
3.      FOREACH x ∈ X DO
4.          x.AssignToClosestCenter()
5.      END
6.      hasDeleted := RemoveCentersWithFewAssignedPoints(n_min)
7.      FOREACH z ∈ Z DO
8.          z.SetToCentroidOfAssignedPoints()
9.      END
10.     IF NOT hasDeleted THEN
11.         hasSplit := SplitCentersWhereNecessary(n_min, σ_max)
12.         IF NOT hasSplit THEN
13.             MergeCentersWhereNecessary(m_max, δ_min)
14.         END
15.     END
16. END
17. RETURN PointAssignment()
```

3.3. A felügyelt klaszterezés megvalósítása

Ahogy a 2. fejezetben olvashattuk, a felügyelt klaszterezés lényege, hogy a képpont-célérték párokból álló tanító adathalmazból a kellő információt kinyerve egy generikusabb klaszterező függvény (avagy modell) birtokába jussunk. Ehhez illeszkedően az implementáció alapjaként érdemesnek bizonyul egy, az ilyen modelleket leíró interfész létrehozása, amely rendelkezik a predikciót végző metódussal (persze hasznos lehet itt megkövetelni a felépített modell későbbi használathoz szükséges szerializációs eljárások, így tehát a mentési és a betöltési funkcionalitás meglétét is). A hierarchia elemeként definiálható a tanítási folyamathoz kapcsolódó interfész is, mely az előbbiekben bemutatott modell elkészítéséért felelős – kellő szabadságot biztosítva a konkrét eljárástól függő paraméterek megadására. Ezek után akár egy olyan absztrakt osztály is kialakítható,

mely a betanított általános modell interfészét felhasználva bonyolítja le a klaszterek predikciójának folyamatát, tetszőleges bemeneti felvételre.

3.3.1. *Random forest*

A korábban vizsgált eljárásokhoz képest bátran kijelenthető, hogy a működtetéshez (a felügyelt módszereknél általánosságban is) itt van szükség a legtöbb információ *a priori* rendelkezésre állására, gondolva első sorban a tanító adathalmazra, amelynek megválasztása önmagában sem egyszerű feladat. Esetünkben egy kézenfekvő megközelítés lehet erre, ha például a korábban megismert metódusok használata során kapott, ellenőrzött eredményeket használunk fel egy generikusabb modell kialakítására.

Már teljes egészében a random forest sajátossága, hogy a felhasználónak a tanítás működtetéséhez meg szükséges még a létrehozandó modell fáinak számát (t) is adnia, amellyel adott értelemben a véletlenszerűség mértékét tudja megszabni. Hasonlóan fontos ehhez a pontválasztási arány (ρ) és a felhasznált attribútumszám (f) értékeinek megválasztása – ezekre részletesebben később térünk ki. Egyes implementációk mindemellett lehetőséget adnak a fák méretének szabályozására (mint például a fa maximális magasságára vagy a fa egyes csúcsainak maximális méretére vonatkozó korlátok), így határt szabva egyúttal az algoritmus futási idejének is [34].

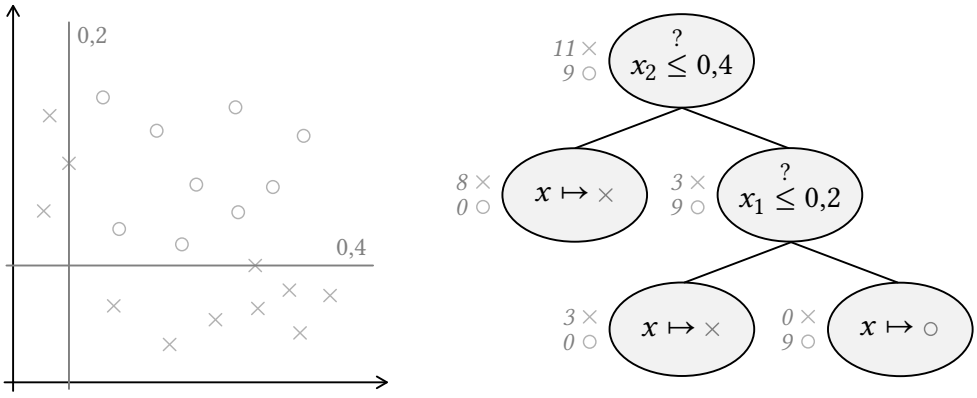
Breiman eredeti algoritmusának [30] építőkövei az említett döntési fák, illetve az ezen fák egyes egyedeinek erősségeit kinyerni szolgáló *bootstrap aggregating* (avagy *bagging*) és attribútum-válogatás (*feature selection*) ötletei.

A *bagging* lényege, hogy a túltanulás (*overfitting*, rosszul általánosító modell építése) esélyének csökkentését segítendő, az egyes fákat nem az eredeti tanító adathalmazt (mely jelenleg egy többdimenziós képpontokból álló bemeneti adathalmazból, illetve az egyes pontokhoz tartozó tényleges klaszter azonosítójából, a célértékből áll) felhasználva építjük fel. Ehelyett minden fa esetén annak egy véletlenszerűen választott részével dolgozunk. Fontos, hogy ez visszatevéses módon történik, tehát egyazon pont akár többször is kiválasztható. A ρ paraméterrel szabályozható a válogatott és az eredeti pontok számosságának aránya; a kiválogatott pontok a megfelelő ρ érték mellett a visszatevésnek köszönhetően akár nagyobb számban is jelen lehetnek, mint a bemenetben.

Az erdőépítés másik fontos folyamata a *feature selection*: a fákban az egyes csúcsok által megfogalmazott kérdéseknél nem a bemeneti adatpontok minden attribútumát

(vagyis koordinátáját) vesszük figyelembe, hanem csak ezeknek egy csúcsonként különböző, véletlenszerű részét. Ennek számosságát a korábbiakhoz hasonlóan a felhasználó szabályozhatja: ez esetben az f paraméter értékének megválasztásával.

Az erdőt megadó minden egyes döntési fa építése során, az előbbiek szerint válogatott pontokból indulva, rekurzívan, csúcsról csúcsra haladva keressük a pontok legjobb felosztását. Ez a felosztás felelhet meg az adatra vonatkozó kérdésnek; jelen esetben egyszerűen koordináták értékének ellenőrzésről van szó, tehát azt vizsgáljuk, hogy milyen v_{best} érték és i_{best} koordináta mentén érdemes az adatpontjaink gyűjteményét kétfelé osztani. Ezen értékek alapján az x pontokat $x_{(i_{best})} \leq v_{best}$ esetén bal gyerekhez, $x_{(i_{best})} > v_{best}$ esetén pedig a jobbhoz sorolva kapjuk meg a csúcs két gyerekét, és így haladhat is tovább a rekurzió. Ez valójában a pontok terében egy lineáris vonal adott tengely mentén történő behúzásaként képzelhető el, amely tehát a pontokat két csoportra osztja.



3.3. ábra. Példa egy betanított döntési fára, és a felhasznált tanító adathalmazra. Az egyes adatpontok klasztereit, avagy az ismert célértékeket az adatpontok alakja jelöli.

A legjobb ilyen v_{best} érték (és koordináta) megválasztásához persze elsőként fontos a felosztás minőségének mérésére alkalmas mennyiséget megfogalmazni. Ehhez szakirodalomban számtalan fellelhető lehetőség (például *Gini impurity* vagy *information gain*) egyikéhez fordulhatunk [24]. Esetünkben az information gain értékénél tapasztalhatóhoz hasonló minőségű eredményt biztosító, viszont attól kisebb számítási bonyolultságot magában hordozó [35] Gini impurity képlete került felhasználásra:

$$\sum_{i=1}^{k} p_i \left(1 - p_i\right),$$

ahol p_i a C_i klaszterhez tartozó pontok számának összes ponthoz viszonyított aránya.

Ha az építés során nincs több felosztás, vagy teljesül valamely, a felhasználó által definiált leállási feltétel, akkor az így kapott levélhez már csak hozzá kell rendelnünk az általa megadott predikció értékét. Ezt a levélhez tartozó felosztás pontjainak célértékei között leggyakrabban előforduló érték adja meg. Egy felépített döntési fára láthatunk példát a 3.3. ábrán, az erdőépítés folyamatát pedig a 3.4. algoritmus vázolja.

A tanítás eredményeképp kapott fából oly módon nyerhetjük ki az (általános, tehát már nem tanító adat esetén vett) inputunk egyes pontjaira vonatkozó klaszterazonosítókat, hogy a gyökérből indulva ellenőrizzük a köztes csúcsok által definiált kérdéseket és a válasznak megfelelő gyerek irányába haladunk tovább mindaddig, amíg levélhez nem érünk. Az elért levél által definiált klaszter lesz az inputhoz rendelhető predikció.

A teljes random forest eljárása (ahogy a 3.3. algoritmus pszeudokódján megtekinthető) a bemutatott módszerek vegyítését próbálja előnyére fordítani, avagy a segítségükkel számos, minél inkább különböző döntési fát alkotni, hogy ezáltal minél pontosabb eredményhez juthassunk az általános bemenetünkre. A végeredményt tehát egy adott bemeneti adatpontra az összes betanított fa eredménye között leggyakrabban előforduló célérték adja meg, ez lesz így hát az adatpont klasztere.

3.3. algoritmus. A random forest predikciós eljárásának pszeudokódja, amely egy korábban betanított modellt (*forest*) használ az X felvétel klaszterezésére.

```
1.  FOREACH x ∈ X DO
2.      predictions := EmptyDictionary()
3.      FOREACH tree ∈ forest.Trees() DO
4.          n := tree.Root()
5.          WHILE NOT n.IsLeaf() DO
6.              n := n.ChildBasedOnSplitFor(x)  // a következő csúcs lekérdezése
7.          END
8.          p := n.Prediction() // az elért levélhez tartozó célérték-predikció tárolása
9.          predictions[p] := predictions[p] + 1
10.     END
11.     x.AssignToMajorityElementOf(predictions)
12. END
13. RETURN PointAssignment()
```

3.4. algoritmus. A random forest tanító metódusa X felvétel és Y célértékhalmaz esetén.

```
1.  trees := InitializeTrees(t)                    // t darab üres fa (gyökér) inicializálása
2.  FOR j := 1 TO t DO
3.      n := trees[j].Root()
4.      data := SampleWithReplacement(X, Y, ρ)
5.      n.SetPointsAndTargets(data)
6.      s := EmptyStack()
7.      s.Push(n)
8.      WHILE NOT s.IsEmpty() DO
9.          n := s.Pop()
10.         IF ShouldTerminateOn(n) THEN           // levélhez értünk
11.             p := n.MajorityElementOfTargets()
12.             n.SetLeafPrediction(p)
13.         ELSE                                   // köztes csúcsnál vagyunk
14.             features := SampleFeatures(f) // f darab koordináta kiválasztása
15.             data := n.PointsAndTargets()
16.             FOREACH i ∈ features DO
17.                 FOREACH v ∈ data.PointsAtFeature(i) DO
18.                     MemorizeGiniImpurityOfSplit(data, i, v)
19.                 END
20.             END
21.             i_best, v_best := SplitWithLowestGiniValue()
22.             data_right, data_left := SplitDataPoints(data, i_best, v_best)
23.             n_right := RightChild(n, data_right, i_best, v_best)
24.             n_left := LeftChild(n, data_left, i_best, v_best)
25.             s.Push(n_right)
26.             s.Push(n_left)
27.         END
28.     END
29. END
30. RETURN ForestOf(trees)
```

3.4. A SŰRŰSÉGALAPÚ MÓDSZEREK KIALAKÍTÁSA

Természetesen a korábban leírtakhoz hasonlóan a sűrűségalapú eljárások megvalósításának alaplépéseként is felmerül egy, a közös fogalmakat és műveleteket összefoglaló ősosztály definiálása. Itt többek közt leírható a magpontok kiszámításához, vagy az egyes képpontok megadott sugarú szomszédságainak összegyűjtéséhez kapcsolódó számítások miként is.

3.4.1. *A DBSCAN metódusa*

A DBSCAN algoritmusa csupán két különböző paraméter megadásának feladatát bízza a felhasználóra, bár a korábban látottakhoz képest ez a legtöbb esetben mégis nagyobb nehézségeket hordoz magával.

3.5. algoritmus. A DBSCAN fő eljárása, X bemeneti adathalmaz esetén.

```
1.  c := InitializeClusterLabel()
2.  FOREACH x ∈ X DO
3.      IF NOT x.IsVisited() THEN
4.          x.MarkAsVisited()
5.          N := NeighborsOf(x, ε)              // az ε sugarú környezet lekérdezése
6.          IF |N| ≥ p_min THEN
7.              ExpandNeighbors(x, N, c, ε, p_min)   // részletek: 3.6. algoritmus
8.              c := NewClusterLabel()
9.          ELSE
10.             x.MarkAsNoise()
11.         END
12.     END
13. END
14. RETURN PointAssignment()
```

Ahogy az már a 2. fejezetben (a továbbiakban használt alapfogalmakkal együtt) említésre került, a komplikációt elsősorban az eljárás során vizsgált pontszomszédságok sugarát

megszabó ε értékének megkötése okozza, hiszen ez nagymértékben befolyásolja a kialakuló klaszterek minőségét, megállapítása azonban gyakran a felhasználási területhez és az inputhoz kapcsolódó előismeretek jelenlétét igényli. Emellett a kiterjesztendő szomszédságok minimális méretét megszabó p_{min} paraméter is jelentős befolyással bír az eljárás működésére vonatkozóan.

A módszer működése a magpontok megtalálásán, és a sűrűségük szerint velük relációban lévő adatpontok kiterjesztésén – tehát a sűrűség-összekapcsolt pontok összegyűjtésén – alapszik. A folyamat első lépéseként így (ahogy ez a 3.5. algoritmus pszeudokódján is látható) minden egyes, még nem vizsgált x adatpontot feljegyzünk a meglátogatott pontok közé, és, ha ezen x egy magpont, kiterjesztjük az ε sugarú szomszédságát felépítő pontokat.

3.6. algoritmus. A DBSCAN részét képező, az x ponthoz tartozó N szomszédságot kiterjesztő eljárás, amely összegyűjti a pillanatnyi klaszterazonosítóhoz (c) tartozó csoport pontjait.

```
1.  x.AssignToCluster(c)
2.  FOREACH x' ∈ N DO
3.      IF NOT x'.IsVisited() THEN
4.          x'.MarkAsVisited()
5.          N' := NeighborsOf(x', ε)        // az ε sugarú környezet lekérdezése
6.          IF |N'| ≥ p_min THEN
7.              N := N ∪ N'
8.          END
9.      END
10.     IF NOT x'.IsAssignedToACluster() THEN
11.         x'.AssignToCluster(c)
12.     END
13. END
```

A kiterjesztés során a következő (a 3.6. algoritmus pszeudokódján szereplő) módon járunk el: x-et egy új klaszterbe tesszük, majd végighaladunk a hozzá tartozó ε sugarú N környezet x' elemein. Abban az esetben, ha a szemügyre vett x' adatpontot korábban még nem ellenőriztük, jelöljük a meglátogatás tényét. Ezután, feltéve, hogy ez az x' magpont

is, az ő szomszédait is hozzáadjuk kiterjesztésre az N-hez. Ezután, ha az éppen vizsgált x' még nincs semmilyen klaszterhez sem beosztva, az x pontéval megegyező klaszterhez sorolhatjuk.

3.4.2. *Problémák a DBSCAN-hez kötődően*

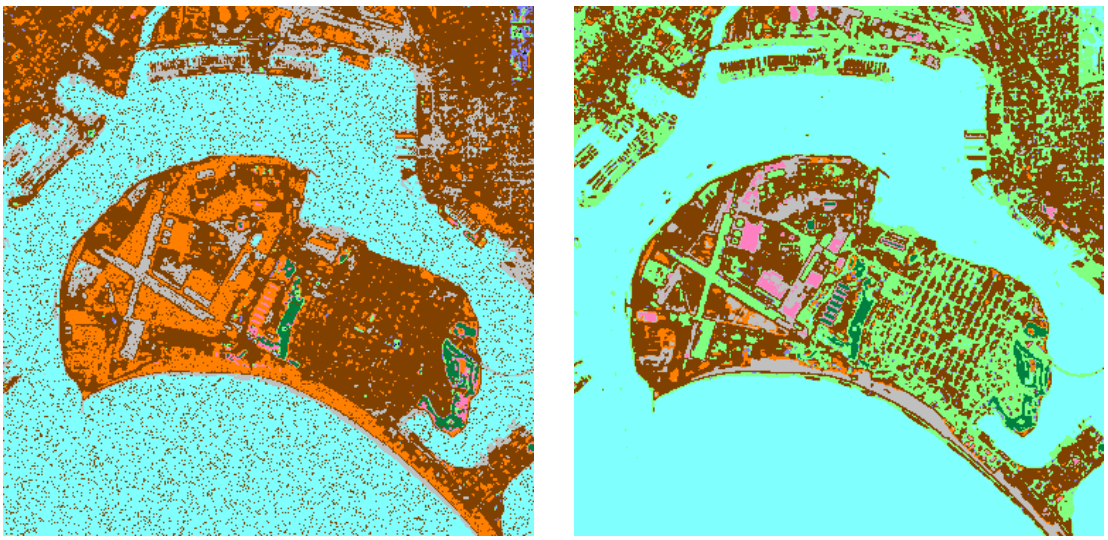
A bemutatásra került algoritmus megvalósítása a távérzékelési adathalmazok környezetében könnyen rávilágít az eljárás elsődleges problémájára: a műveletigény nagyságára. A metódus ugyanis általánosságban $\mathcal{O}\left((n \cdot m)^2 \cdot d\right)$ aszimptotikus bonyolultságúnak tekinthető (ahol $n \cdot m$ a bemeneti felvétel pontjainak száma, d pedig az egyes képpontok dimenzionalitása)* [36, 18]. A problémát erősíti esetünkben a multispektrális felvételek képpontjainak relatíve nagynak tekinthető mennyisége, de az egyes pontok dimenzionalitása is, így élesebben hangsúlyozódhat tehát, hogy a DBSCAN futási ideje nem tudja megközelíteni a k-means-ét [19].

A szakirodalomban számos megközelítés született a módszer orvoslására, egyesek alapjaiban megváltoztatják a működést – így például Gan és Tao a legközelebbi szomszéd fix sugarú keresését a BCP (*bichromatic closest pair*) problémát megoldó algoritmusra cserélével, emellett közelítő működés definiálásával javít a futási időn [36]. Ezzel szemben, a Giwer projekt fejlesztésének folyamata alatt is felmerülő, mintavételezéses módszer (*downsampling*) közelebb áll a DBSCAN eredeti struktúrájához. Itt ugyanis a fő javulás a bemeneti pontok számának csökkentésétől várható, hiszen a magpontszomszédság keresésének teljes folyamata alatt a metódus csak a felhasználó által beállított arányú, véletlenszerűen választott részhalmazával dolgozik [37].

Természetesen ekkor a választásból kimaradó pontok besorolása újabb problémát vet fel. A projekt implementációja során ennek megoldására több lehetőség kipróbálásra került: elsőként a fő eljárás (3.5. algoritmus) alatt kiterjesztett magpontok közül a legközelebb eső csoportjához; majd a válogatott pontokból kialakított klaszterek legközelebbi centroidjával rendelkezőhöz; végül pedig az összes válogatott pont legközelebbikének klaszteréhez osztása. A végigpróbált paraméterek mellett kapott eredmények azonban gyakran zajos, összemosódó természetűnek könyvelhetők el a gyakorlati szempontból érdekes példákon (ahogy a 3.4. ábrán megfigyelhető), feltételezve a mintavételezés ará-

* Ezzel szemben a k-means esetén ez $\mathcal{O}((n \cdot m) \cdot d \cdot iter \cdot k)$, ahol mind $iter$ (tényleges iterációszám) és k (klaszterszám) jóval kisebb $(n \cdot m)$-től, illetve d-től.

nyának olyan mértékű megválasztását, hogy a futási idő összevethető legyen a korábbi eljárásoknál látottakkal. Ez a jelenség a legközelebbi magpontot felhasználó módszernél csekélyebb mértékben állt fenn – hiszen itt a megfelelő paraméterek megtalálásával már akár az elvárthoz közeli eredményt kaphattunk.



3.4. ábra. Példa az említett zajra a mintavételezést alkalmazó DBSCAN eljárásának eredményében, a fennmaradó pontokat a legközelebbi magponthoz soroló változatban (balra). A mintából kimaradó adat besorolásakor egyes pontok nem a várt klaszterbe kerülnek. Jobbra a random forest módszerével vegyített változat – azonos paraméterek (és 10%-os mintavételezés) mellett kapott, ettől a zajtól mentes – eredménye látható.

Mindemellett a megvalósítás műveletigényének mérséklése során fontos szempont volt a legközelebbi szomszéd problémájának hatékonyabb kiszámítása, tekintve, hogy belőle származik a DBSCAN bonyolultságának jelentős része. Így például adódik a lehetőség egy olyan speciális adatszerkezet használatára, mely a szomszédságok felkutatásának gyorsítását szolgálja. Esetünkben ez a *k-d tree* (*k-dimenziós fa*) struktúráját jelentette, mely a bemeneti adathalmaz (a módszernél használt nevezéktanban tehát k dimenziós) adatpontjaiból felépítve $\mathcal{O}\big((n \cdot m) \cdot \log(n \cdot m)\big)$ bonyolultságúra csökkenti az egyes pontok fix sugarú szomszédságainak összegyűjtését. Egy másik alternatíva az optimalizációra a sikeresen klaszterezett adatpontok eljárás folyamata során történő törlése a keresési térből (természetesen ehhez elsődleges lépésként szükséges a magpontok kezdeti összegyűjtése, hiszen a keresés alatt eltávolított pontok miatt később ez már nem ellenőrizhető helyesen). Megemlítendő azonban, hogy ez nem egyeztethető össze a k-d fa használatával, hiszen egy-egy adatpont struktúrából történő eltávolítása $\mathcal{O}\big(\log(n \cdot m)\big)$ műveletigényűnek tekinthető, így ez esetben nem hozná a várt javulást.

Az eddig bemutatott változatok problémáinak javításaként felvetődik még a korábbiakban vizsgált módszerek tárházából a random forest módszerével történő vegyítés is, hiszen a mintavételezett pontokon előállított részeredmény (amely tekinthető egyfajta tanító adatnak: rendelkezésünkre állnak a bemeneti adatok mellett az azokhoz tartozó, a DBSCAN által megállapított célértékek is) alapján épített modellel az kiterjeszthetővé válik a felvétel egészére. A megközelítéssel kapott klaszterezések mentesnek bizonyultak az egyértelmű zajtól, illetve az eljárás futási ideje is a kívánt tartományban maradt, azonban a kapott eredmények itt sem érik el a korábban szemléltetett eljárások által biztosítottakat.

Összességében az is észrevehető, hogy a mintavételezésre alapozott gyorsítás nem csökkent módszer aszimptotikus bonyolultságán, hiszen csak az ezt kitevő $n \cdot m$ tényező nagyságát mérsékli. Ebből persze az is adódik, hogy a megoldástól nem várható a DBSCAN skálázhatóságának javulása. Ettől függetlenül a metódus működtethetőségével kapcsolatban, a célterületen a gyakorlatban is előforduló bemenetek esetén érdemi javulás érhető el. A bemutatott módszerek által kialakított klaszterek minőségére részletesebben a 4. fejezetben is kitérünk.

4. FEJEZET

EREDMÉNYEK

A továbbiakban a Giwer projekt keretében megszülető, korábban ismertetett klaszterező módszereket vizsgáljuk, de a fejezetben szereplő összehasonlítás részét képezi mindemellett egy gyakran használt, nyílt forráskódú, általános célú implementáció is: az ML.NET gépi tanulási könyvtára [38] által tartalmazott (a vizsgálatok környezetébe beültetett) *k*-means metódus.* Említésre érdemes, hogy a random forest fejlesztése során egy másik hasonló, generikus könyvtár, a SharpLearning [39] is tesztelés alá került, azonban a benne felhasznált implementációs megfontolásoknak betudható magas erőforrásigénye miatt ez a megvalósítás alkalmatlannak bizonyult a drónfelvételek méretei melletti, így a konkrét célterületen történő működtetésre. A fejezetben ezáltal csak a Giwer programcsomagjában helyet kapó realizációt szemléljük, mely ehhez képest jelentős javulást hoz magával, mind futási idő, mind pedig memóriaigény tekintetében. Esetünkben ez azt is jelenti, hogy az általános célú realizációval szemben ez az implementáció nagyobb bemenetekre is, túlcsordulás nélkül használhatóvá válik. Megjegyzendő még, hogy random forest módszere és a DBSCAN különböző eljárásai – természetükből, vagy a velük járó problémák következményeképp adódóan – a többi módszertől eltérő szempontok figyelembevételével kerül kiértékelésre.

4.1. A mérések és környezetük

Az elemzés célpontjai elsősorban az algoritmusok főbb, gyakorlati jelentőségű tulajdonságai. Ide sorolandó a felhasználási területen elvártaknak megfelelő, multispektrális

* A mérések során az eljárás paraméterei természetesen a Giwer projektben szereplő *k*-means futtatásaival összeegyeztethető módon kerültek beállításra, így lényegében a klaszterszám és az iterációk száma egyező, míg például az inicializációs metódus az ML.NET környezetében definiált alapértelmezett értéket kapta.

bemeneten mérhető futási idő, memóriaigény, valamint természetesen a kapott klaszterezés minősége is.

- A memóriahasználat szemléléséhez a *dotMemory* profilozóprogram [40] biztosít segítséget. Megjegyzendő, hogy az ilyen jellegű összehasonlítást bonyolítja, hogy az implementáció szemétgyűjtéses környezetben valósult meg. Esetünkben a futás során kapott csúcsértékek jelentik az alapot, így természetesen csak felső becslést kaphatunk a szükséges memória mennyiségére.
- Az eredmény színvonalának számszerűsítéséhez a 2. fejezetben megismert internális mennyiségek közül a Davies–Bouldin (DBI), és a Caliński–Harabasz (CHI) indexek kerülnek felhasználásra. Ezeknek a korábbiakban bemutatottaknak megfelelően rendre az alacsonyabb, valamint a nagyobb értékei jelentik a klaszterezés magasabb minőségét – azonos bemenetet feltételezve.
- Szintén láthattuk már, hogy az externális esetben szükséges egy megfelelő összehasonlítási alap birtokában lennünk: a fejezetben látható eredmények esetén ehhez a QGIS programcsomag [41], és az SCP (*Semi-automatic Classification Plugin* [42]) felhasználásával, a bemeneteken manuálisan kijelölt tanítóterületekre alapozva futtatott felügyelt osztályozás kimeneteit használjuk fel. Az így kapott klaszterezések a purity, valamint a Normalized Mutual Information (NMI) mértékeivel vethetők össze az implementált módszerek eredményeivel: az alappal jobban összeegyeztethető outputot mindkét esetben az 1-hez tartó értékek indikálják.

A leírt szempontok elemzéséhez egy-egy bemeneten és metóduson öt egymástól független, egyező paraméterezésű futtatás során mért eredmény átlaga kerül összevetésre. Ezen mérések mindegyike 8 GB memória, és egy x64-alapú Intel Core i7-4720HQ processzor mellett, Windows 10 operációs rendszer környezetében történik.

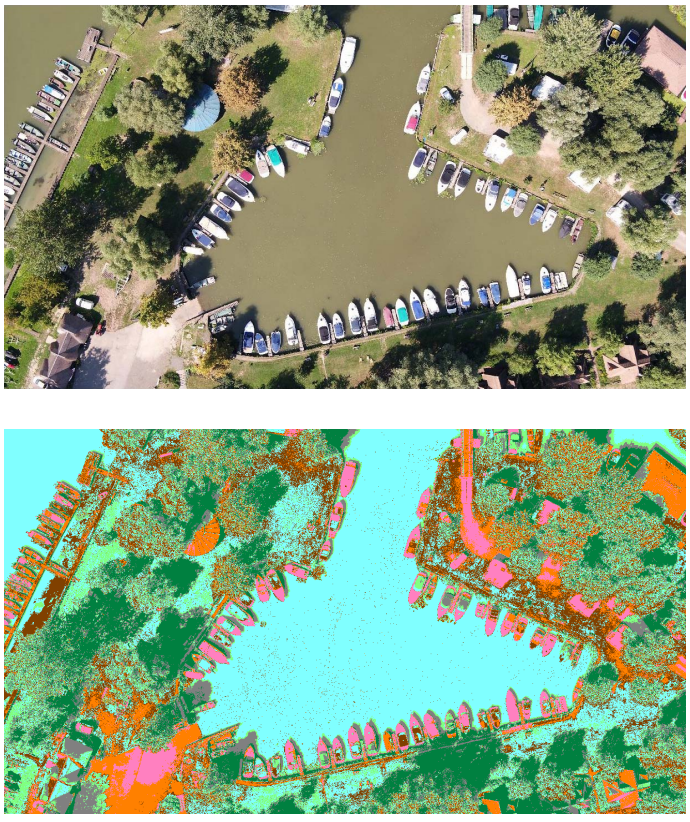
Az eljárások vizsgálatának bázisát négy különböző felvétel adja. Ezek különböző földfelszíni területeket rögzítenek, különböző spektrális sávokból származó intenzitásértékekkel. A bemenetek között szerepel két drónnal, illetve két műholddal rögzített felvétel is. Emellett a négy választott (esetünken egységesen 8 bites színmélységűre konvertált) input méreteiben is széles skálát reprezentál, hiszen az egyes képpontok dimenzionalitásán kívül a fotók felbontása is jelentősen eltér.

A felvételekre vonatkozó tudnivalók a 4.1. táblázatban olvashatók mellett a következők szerint foglalhatók össze. A LANDSAT-ként hivatkozott fotó egy LANDSAT műholdcsempe

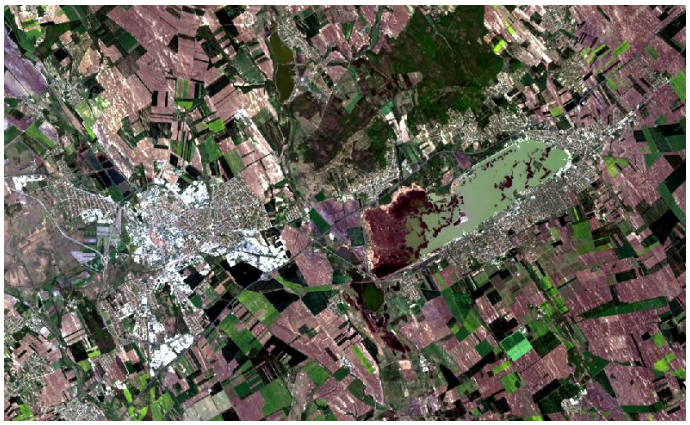
részlete, Székesfehérvár és a Velencei-tó környékéről. A SPOT egy másik, SPOT típusú műholddal rögzített multispektrális, XS módú – a látható vörös és zöld sávokon kívül a közeli infravörös tartományával is rendelkező – űrfelvétel egy kikötőről, és az azt körülvevő városi és ipari területekről. A DRONE_1 csak a látható vörös, zöld és kék sávot tartalmazó, drónnal rögzített légifotó egy vízfelületről, és annak partjáról – az ott fellelhető vegetációval és egyéb entitásokkal. Végül, a DRONE_2 egy gyümölcsösöket ábrázoló drónfelvétel [43], melynek a mérések során öt sávja kerül felhasználásra. A bemeneteket a 4.1., 4.2., 4.4., és a 4.5. ábrán tekinthetjük meg, az eredményül kapott kimenetek példáival kiegészítve. A főbb vizsgált eljárások által adott klaszterek a 4.3. ábrán vethetők össze.

4.1. táblázat. A felhasznált felvételek fontosabb attribútumai.

azonosító	felbontás (MP)	sávok (db)
LANDSAT	0,68	11
SPOT	0,56	3
DRONE_1	15,68	3
DRONE_2	71,03	5



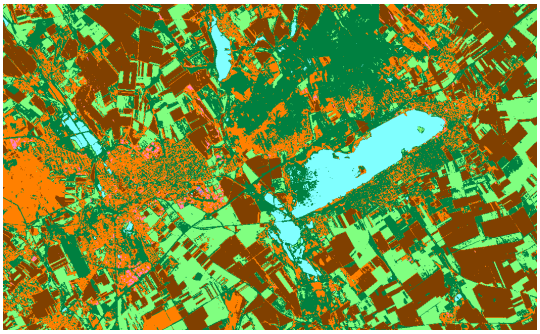
4.1. ábra. A DRONE_1 drónfelvétel, és egy, a manuális *k*-means futtatásával előállított klaszterezés.



4.2. ábra. A LANDSAT űrfelvétel.

i) *k*-means ML.NET

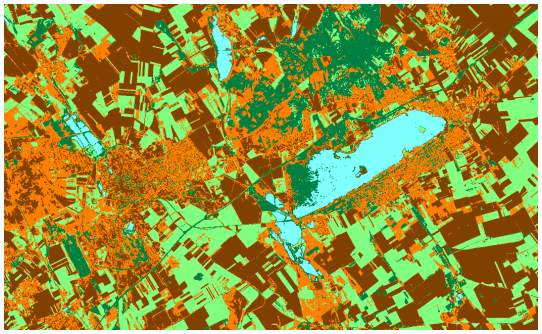
ii) *k*-means



iii) *k*-means elbow

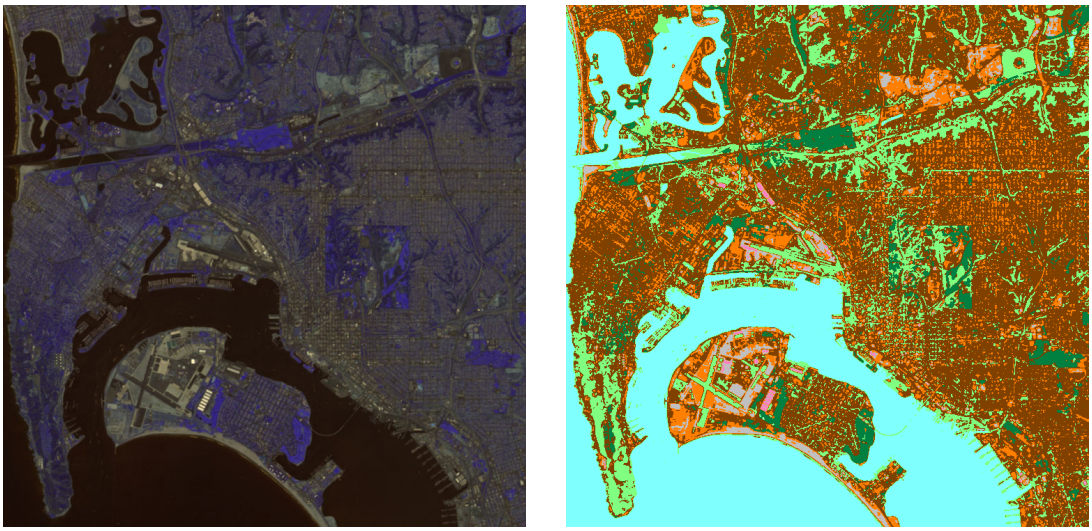
iv) ISODATA

v) random forest

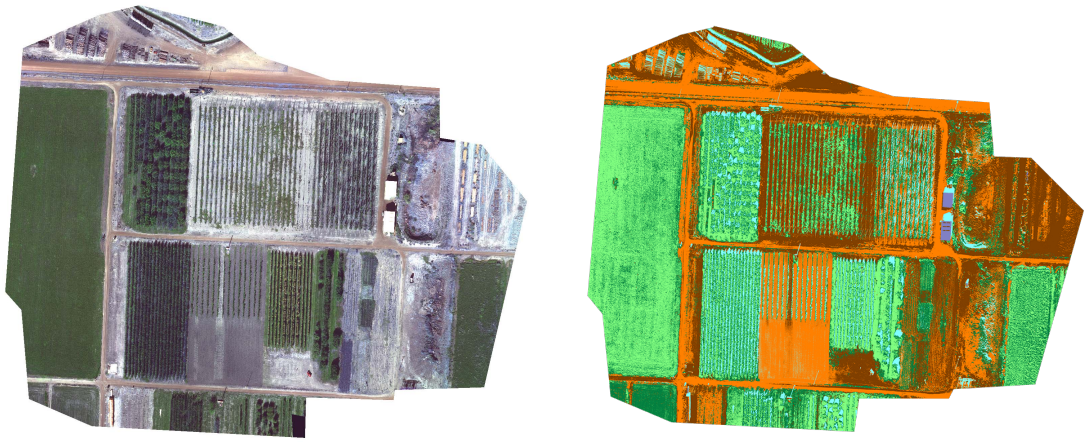


vi) *k*-means és random forest

4.3. ábra. Példák az egyes klaszterező eljárások kimeneteire a LANDSAT felvételén. A QGIS-szel kapott referencia itt nem kerül külön feltüntetésre, hiszen lényegében megegyezik a random forest eredményével, ahogy arra a 4.2. táblázat megfelelő sorának purity értéke is rámutat.



4.4. ábra. A SPOT űrfelvétel és az ISODATA eljárás eredményeképp kapott klaszterezése.



4.5. ábra. A DRONE_2 bemenete, és a random forest (QGIS-szel kapott osztályozás egy részletét felhasználó) eljárásának eredménye.

4.2. A KAPOTT ÉRTÉKEK ÖSSZEGZÉSE

A fontosabb eljárások futtatásai során kapott eredmények átlagának felvételenkénti összesítése látható a 4.2., 4.3., 4.4., valamint a 4.5. táblázaton. Megjegyzendő, hogy ezúttal a random forest eljárása esetén a QGIS segítségével előállított osztályozások részleteit használjuk fel tanító adathalmazként. Ezen részletek az eredeti felvételek olyan szegmensei, amelyek lehetőség szerint a legtöbb, a felvételre általánosan jellemző tulajdonságnak megfelelő képpontot tartalmazzák. Méretüket tekintve a választott tanító bemenetek a SPOT, LANDSAT, DRONE_1, illetve a DRONE_2 input esetén rendre az eredeti felvétel 9, 20, 10 és 7%-os szeletét reprezentálják. Gyakorlati felhasználást tekintve természetesen ilyen

jellegű adat rendelkezésre állása ritkán elképzelhető, így ezt az eljárást a későbbiekben ennek hiányával előálló használati körülmények mellett is vizsgáljuk.

Mindemellett – a módszerhez kötődő, korábban említett problémák (elsősorban az elnyújtott futási idő) miatt – a DBSCAN már előzetesen is alkalmatlannak bizonyult a többi metódussal történő összevethetőségre, így szintén külön mérjük az implementáció során kialakított közelítő megoldásokra vonatkozó tulajdonságokat.

4.2. táblázat. A mérések a LANDSAT bemenetén.

módszer	futási idő (s)	memória (MB)	DBI	CHI	purity	NMI
QGIS SCP	–	–	1,001	439 206	–	–
k-means ML.NET	12,1	144	0,938	569 903	0,81	0,63
k-means	17,29	61	0,947	570 796	0,77	0,62
k-means elbow	65,29	63	0,876	446 766	0,76	0,61
ISODATA	7,66	102	0,964	446 032	0,73	0,61
random forest	0,86	70	1,008	436 096	0,97	0,91

4.3. táblázat. A vizsgált eljárások eredményei a SPOT inputon.

módszer	futási idő (s)	memória (MB)	DBI	CHI	purity	NMI
QGIS SCP	–	–	1,031	419 424	–	–
k-means ML.NET	10,08	153	0,779	984 996	0,79	0,56
k-means	5,53	40	0,814	860 412	0,69	0,5
k-means elbow	22,09	45	0,821	808 529	0,67	0,47
ISODATA	2,94	84	0,843	767 283	0,69	0,51
random forest	0,88	43	2,808	225 618	0,71	0,49

4.4. táblázat. A metódusok eredményei a DRONE_1 drónfelvételén.

módszer	futási idő (s)	memória (GB)	DBI	CHI	purity	NMI
QGIS SCP	–	–	4,247	1 233 328	–	–
k-means ML.NET	234,2	2,3	0,744	39 800 519	0,62	0,33
k-means	108,6	0,85	0,78	42 859 742	0,63	0,32
k-means elbow	174,3	0,95	0,752	41 131 524	0,62	0,33
ISODATA	66,8	1,35	1,024	33 572 757	0,63	0,33
random forest	16,9	0,79	4,006	1 350 165	0,97	0,92

4.5. táblázat. A DRONE$_2$ felvételén kapott értékek.

módszer	futási idő (min)	memória (GB)	DBI	CHI	purity	NMI
QGIS SCP	–	–	0,736	44 498 652	–	–
k-means ML.NET	–	9,0 +	–	–	–	–
k-means	6,69	4,2	0,667	45 134 476	0,88	0,82
k-means elbow	18,59	5,6	0,792	35 984 131	0,86	0,78
ISODATA	5,77	6	0,933	27 139 895	0,8	0,63
random forest	3,39	3,6	0,736	44 536 299	1	0,99

4.2.1. *A DBSCAN mintavételező változatai*

A 3. fejezetben említetteknek megfelelően a Giwer programcsomag klaszterező eljárásainak fejlesztése során több, a DBSCAN eljárásának lassúságán javítani próbáló megoldás készült el. Mivel a módszer alapvetően nagyságrendekkel lassabbnak bizonyult az összehasonlítást képező többi algoritmushoz képest, így a hangsúly elsősorban a teljes felvétel helyett annak csak egy véletlenszerűen mintavételezett részét kiterjesztő közelítő megoldásokra helyeződött. Így két fő (a mintavételből kimaradó pontokat a legközelebbi magponthoz osztó, illetve a random forest felügyelt klaszterezése segítségével besoroló) eljárás, a SPOT felvétel képpontjainak (a szomszédságok kezdeti kiterjesztése során) 10%-át felhasználó futtatása által hozott eredményeinek átlagait foglalja magában – két eltérő paraméterezés mellett – a 4.6., és a 4.7. táblázat.

4.6. táblázat. A DBSCAN eljárás változatainak eredményei a SPOT-on, $p_{min} = 10$ és $\varepsilon = 5,57$ paraméterek mellett.

módszer	futási idő (s)	memória (MB)	DBI	CHI	purity	NMI
legközelebbi magpont	37	23	0,994	140 329	0,62	0,37
random forest	38,7	57	1,378	386 970	0,68	0,47

4.7. táblázat. A DBSCAN két változatának eredményei a SPOT felvételén, $p_{min} = 40$ és $\varepsilon = 4,5$ paraméterekkel.

módszer	futási idő (s)	memória (MB)	DBI	CHI	purity	NMI
legközelebbi magpont	110	23	0,998	693 745	0,68	0,5
random forest	111	58	1,323	394 865	0,63	0,42

4.2.2. *Felügyelet nélküli random forest*

A random forest eljárásának gyakorlatban történő felhasználási lehetőségeként megemlíthető, kézenfekvő munkafolyamat, a szóban forgó felügyelt módszer kombinálása a Giwer projektjében helyet kapó *k*-means metódussal. Ennek során a *k*-means algoritmusát a bemeneti felvételünk egy részletére futtatjuk, a kapott klaszterezéssel pedig felépítjük a random forest egy modelljét, amely segítségével végül predikciót kapunk a teljes felvétel klasztereire.

A konkrét mérések során a LANDSAT műhold-, és a DRONE_2 drónfelvétel, a korábbiakban választottakkal megegyező részleteit használjuk fel a tanító adat előállítására, amiből aztán kinyerjük a bemenet csoportosítását. Az így kapott eredmények olvashatók le a 4.8., valamint a 4.9. táblázatról.

4.8. táblázat. A random forest eredménye a *k*-means (a LANDSAT bemenet részletén történő) futtatásával előállított tanító adat mellett, valamint a korábban közölt, normál *k*-means-re vonatkozó eredmények. A memóriaigény esetéhez zárójelben tartozó adat egy további optimalizációs lehetőség kiaknázása mellett elérhető érték (erről részletesebben az 5. fejezetben olvashatunk).

módszer	futási idő (s)	memória (MB)	DBI	purity	NMI
k-means és random forest	1,9	76 (64)	0,89	0,76	0,66
k-means	17,3	61	0,95	0,77	0,62

4.9. táblázat. A random foresttel vegyített *k*-means eredménye (a DRONE_2 felvétel részletén futtatva), illetve az alap *k*-means korábbi adatai. A memóriaigénynél szereplő zárójeles értékre az 5. fejezetben térünk ki.

módszer	futási idő (min)	memória (GB)	DBI	purity	NMI
k-means és random forest	4	3,4 (2,2)	1,11	0,66	0,59
k-means	6,7	4,2	0,67	0,88	0,82

5. FEJEZET

DISZKUSSZIÓ

Természetesen az adatok puszta közlése számos lényeges, kapcsolódó információ nélkülözését jelentené. Érdemes tehát a kapott eredmények kiértékelésére és a fejlesztés – illetve a mérések – során gyűjtött fontosabb tapasztalatok bemutatására is sort keríteni. A következő bekezdések célja így a 4. fejezetben prezentált adatok áttekintése és a releváns kérdések válaszainak keresése.

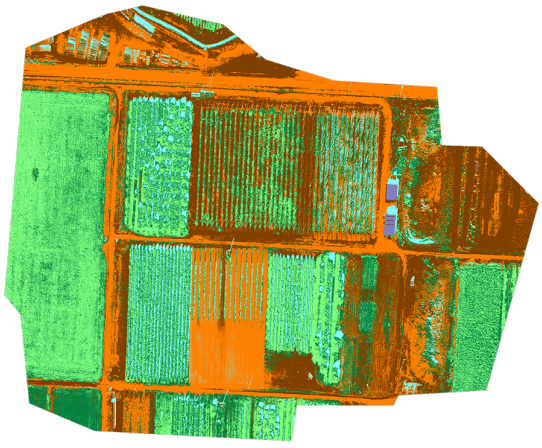
5.1. AZ EREDMÉNYEK TÁRGYALÁSA

A keletkezett adatokat (és a kapcsolódó kimeneteket) szemlélve kitűnik, hogy a megvizsgált módszerek (eltekintve az metódusok esetenként előforduló tévesztéseitől) relatíve jól közelítik a QGIS SCP segítségével előállított összehasonlítási alapokat, sőt, legtöbbjük az internális CVI-k tekintetében még jobban is teljesít. Ez persze betudható lehetne a feltüntetett internális mértékek valódi értékelő képességének gyengeségeként is. Azonban, bár ezek a referenciaként szolgáló klaszterezések felügyelt módon kerülnek kialakításra, fontos kiemelni, hogy egy így előálló eredmény sem hozhatja egészében az elvárt minőséget – tekintve, hogy ez az osztályozás szintén, még a tematikus csoportok célnak megfelelő kiválasztása és a reprezentáns területek helyes kijelölése mellett is, könnyen mellékvágányra juthat. Összességében tehát ezen manuális módszer jobban összeegyeztethetőnek bizonyul az automatizált társaival, mint ahogy akár arra számítanánk.

Érdemes megjegyezni, hogy a DRONE_1 bemeneten – köszönhetően annak, hogy a drónfelvétel kizárólag a látható vörös, zöld és kék spektrális sávokból épül fel – egyik módszer sem tud igazán jó klaszterezést kialakítani, beleértve ebbe a QGIS segítségével létrehozott megoldásokat is. A drónfelvételek esetén általánosságban is észrevehető, hogy

ezen, kis területet nagy részletességgel ábrázoló felvételek esetén könnyen formálódnak „összefutó” klaszterek (ahogy ez a 4.1. ábrán látható, a fák lombozatán, illetve a füves területeken és a vízfelületeken). Természetesen a DRONE$_2$ felvételén a közeli infravörös, és az úgynevezett *red edge* tartomány rendelkezésre állása nagyban javítani tud a vegetáció megfelelő detektálásán.

Az is észrevehető, hogy ettől még generikusabb esetekben is fedezhetők fel hibák a módszerek által létrehozott klaszterekben. A példa kedvéért, ahogy a 4.3. ábra esetén is látható, a legtöbb megoldás nehézségekbe ütközött a LANDSAT felvétel bal szélén (Székesfehérvár nyugati oldalán) található földterületek megfelelő besorolásával, így azok a városias földfelszíni részleteket reprezentáló klaszterbe kerültek. Ehhez hasonlóan, a DRONE$_2$ drónfelvételén megfigyelhető az is, hogy a felvétel felső részén húzódó csatorna vízfelületét az eljárások az egyéb felszíni entitások árnyékaival megegyező klaszterbe osztották (ezt láthatjuk például az 5.1. ábrán). Jelen esetben ezt hibát a bemenet információtartalma idézi elő, hiszen az egyes spektrális sávok intenzitásértékei valóban nagyban hasonlítanak – persze az emberi szemlélő ettől függetlenül könnyen felfedezi (akár már a vizsgált entitás alakjából, vagy a környezetében lévő objektumokból adódó többletinformáció segítségével) a két felszíni terület tematikus osztályának különbségét.



5.1. ábra. A DRONE$_2$ felvételének QGIS SCP segítségével létrehozott osztályozása. Látható, hogy az eredmény nagyban hasonlít a 4.5. ábrán szereplőhöz.

Rátérve a konkrét eljárásokra, a mért adatok rámutatnak, hogy a Giwer-es és az ML.NET-es *k*-means módszerek közti fő különbség a memória tekintetében figyelhető meg, hiszen az utóbbi számottevően nagyobb memóriaigénnyel rendelkezik. Ennek okozója, hogy – mivel egy általános célú megvalósításról beszélünk – elkészítésekor

például nem kerülhettek kiaknázásra a bemeneti adatra vonatkozó, a felhasználási terület ismeretében felmerülő megszorítások. Ennek is lehet a következménye, hogy a nyílt forráskódú könyvtár implementációja a legtöbb esetben egyező iterációszám mellett is lassabb futásúnak bizonyul. A többcélú megvalósítás által hozott fő komplikációra a nagyobb méretekkel rendelkező input tudott csak igazán rávilágítani: esetünkben a DRONE$_2$ felvételén történő futtatás során az eljárás memóriaigénye meghaladta a környezet által biztosított mennyiséget, a túlcsordulás következtében tehát a módszer erre a bemenetre nem tudott eredményt adni. Ehhez kapcsolódó implementációs tapasztalatként megemlíthető, hogy ezeknél az algoritmusoknál a megszokottnál még talán óvatosabb megfontolást igényel az objektumorientált fejlesztési stílus frappáns használata, hiszen könnyen ütközhetünk hasonló problémába, ha a konkrét területtel járó optimalizációs lehetőségek és az újrafelhasználhatóság egyensúlyát nem a kellő mértékűre hangoljuk. A skálázhatóságától eltekintve azonban észrevehető, hogy a többi bemenetnél az ML.NET-es *k*-means változat konzisztensen jobb eredményeket ad – legtöbbször mind az externális, mind az internális minősítések szerint. A különbség oka a két eljárás eltérő inicializációs módszerére vezethető vissza. Alapértelmezetten az ML.NET realizációjában a *k*-means++ továbbfejlesztett, párhuzamos változata, a *k*-means|| [44] kerül felhasználásra, amelynek előnye, hogy segítségével az eredményhez közelebbi kiindulópontból kezdődhet az iteráció, tovább csökkentve a lokális minimumba jutás lehetőségét.

Mindettől függetlenül felismerhetjük, hogy a manuális *k*-means eredményeinek minősége (és memóriaigénye) az élvonalban mozog, még ha ehhez nem is a legjobb futási idő társul. Ezt persze elősegíti, hogy itt a klaszterek számát a felhasználó állítja be; jobban közelítve a felügyelt módozattal előálló eredmények esetét, mind a színvonal, mind pedig akár az üzemeltetés tekintetében is – szembeállítva az elbow metódussal kiegészített változattal és az ISODATA (vagy akár a DBSCAN) algoritmusával.

Az elbow metódussal kiegészített változat kimenetei általánosságban jól követik a manuális módszerét, azonban a kívánt klaszterszám megkeresése memóriaigény növekedését, illetve a futási idők jelentős elnyúlását hozzák magukkal, így tehát gyakran a vizsgált módszerek sorából ez az eljárás bizonyul a leglassabbnak. A megfelelő paraméterek megadása mellett azonban (az ISODATA esetéhez hasonlóan) a legtöbb futtatás a várthoz közeli *k* értéket – és hozzá tartozó klasztereket – hozott.

Látható ezek mellett az is, hogy az összehasonlítottak közül az ISODATA működteté-

séhez szükséges a legtöbb memória rendelkezésre állása (az ML.NET-es *k*-means után), viszont a futási időkben előnyre tudunk szert tenni mindegyik *k*-means változattal szemben, úgy, hogy az eredmények minősége (és az általa megtalált klaszterek száma is) jól közelíti a többi vizsgált társáét.

A random forest erőssége abban rejlik, hogy a jól megválasztott tanító adat jellemvonásainak felismerését viszonylag gyorsan és kevés memóriát felhasználva tudja, magas hűséggel, nagyobb területekre kiterjeszteni. Megfelelő tanító adatok mellett ugyanis gyakran a legalacsonyabb futási idő és memóriaigény mellett kaphatjuk vissza esetünkben a felhasznált klaszterezési eredménnyel lényegében megegyező kimenetet a felvétel egészére. Természetesen ezzel adódik a hozzá kapcsolható fő probléma is: a kellő minőségű adat beszerzése ritkán valósítható meg, hiszen az ideális működéshez fontos, hogy olyan adathalmaz birtokában legyünk, amely nagyban hasonlít a predikcióra szánt bemenethez, a tanító felvétel spektrális sávjai és az ábrázolt felszíni terület tematikus osztályai tekintetében – ezért is került tanulmányozásra kézenfekvően a konkrét bemeneti felvétel egy részletét javára fordító, *k*-means eljárással elegyített megoldás. Példaképp megfigyelhető az eredményekben, hogy a SPOT felvételén a random forest csak gyengébb közelítését adta vissza a teljes felvétel klasztereinek, még az eredeti bemenet nagyobb (20%-os, szemben a többi eset 10%-os, vagy az alatti méretű) részletét felhasználva is. Ez tehát visszavezethető arra, hogy a szóban forgó bemeneten nem lehet ilyen méretű, a másik inputoknál tapasztaltakhoz hasonlóan jól általánosító részletet kijelölni (ahogy ezt a mérések folyamata alatt, a SPOT-on véghezvitt számos különféle próbálkozás is érzékeltette) az eljárás számára.

A *k*-means metódusával vegyített megközelítés eredményei szintén erre mutatnak rá: a DRONE_2 bemeneten használt tanítóterületen a *k*-means nem tudott olyan klasztereket létrehozni, amelyek igazán jól kiterjeszthetők lettek volna a teljes felvételre, azonban a LANDSAT esetén a *k*-means által önmagában biztosítotthoz nagyon közeli outputot kaphattunk, miközben a futási idő a töredékére csökkent. Ezekből következtethetünk arra, hogy a nagyobb területet rögzítő műholdfelvételt jobban képesek reprezentálni annak egyes részletei, szemben a drónfelvételek nagy részletességű, kisebb területeivel – tekintve, hogy az utóbbinál a részterületekre fókuszálva a teljes bemeneten egyébként elenyésző súllyal rendelkező komponensek nagyobb mértékben torzíthatják a kialakított csoportokat. A 4.8., és a 4.9. táblázat arra is rávilágít, hogy nagyobb bemenetek esetén

potenciálisan a futási idő mellett jelentőségteljes csökkenést érhetünk el a memória terén is. Itt további kiaknázandó optimalizáció lehetősége merül fel, hiszen a random forest predikciós eljárása képpontonként halad végig az inputon, így nem szükséges a felvétel egészét memóriában tartani. Ezáltal akár a tanító lépés memóriaigényével egyező mértékűre csökkenthetjük (ahogy az említett táblázatokon feltüntett értékek mellett zárójelben olvashatjuk) a teljes megközelítés ilyen jellegű követelményeit, mellyel adott esetben az ML.NET *k*-means realizációja és a DRONE_2 felvétel párosításánál megfigyeltekhez hasonló problémák is elkerülhetővé válnak. Megjegyzendő továbbá, hogy az utóbbiakban említett mérések a két eljárás munkafolyamat-szerű, egymást követő (elkülönülő) futtatása mellett kerültek rögzítésre; a két algoritmus szorosabb integrációjától további javulás várható el – főleg a futási idő tekintetében.

Végül említésre érdemes még a DBSCAN mintavételezett változatainak esete. Bár az alap eljárás futási ideje akár már a kisebb felvételeken is több órás nagyságrendűnek mutatkozott, a mintavételezést alkalmazó megvalósítások mégis meg tudták közelíteni a korábban vizsgált módszereket, a metódusok eredményeinek minősége ezenfelül „konvergál" a korábban látottakhoz. Ettől függetlenül azonban a paraméterek megválasztása komoly problémát jelent (még ha a mintavételezésnek köszönhetően kevésbé érzékeny is az ε értékére [37]): ahogy a 4.6., és a 4.7. táblázatból kiderül, különböző ilyen értékek még a megvalósítások közt is nagyban eltérő eredményeket hoznak. Amit ezen felül a mért mennyiségek nem érzékeltethetnek, az annak a ténye, hogy az ilyen minőségű értékek megtalálása is bonyolult, hosszú időt igénylő feladat lehet a felhasználó számára (ez halmozottan igaz a mintavételezést nem alkalmazó változat esetére, ahol a helyes paraméterek empirikus módon történő megkeresését lényegében ellehetetleníti az eljárás futási ideje). Mindezeket figyelembe véve tehát kijelenthető, hogy a vizsgált módszerek közül a DBSCAN bizonyul a gyakorlatban legkevésbé hasznosíthatónak – távérzékelési felhasználást feltételezve.

5.2. TOVÁBBI KUTATÁSI LEHETŐSÉGEK

A kapott eredmények számos további kérdésre ráterelik a figyelmet, amelyek a témával történő későbbi vizsgálatok és megvalósítások alapját képezhetik. Így tehát hasznos lehet triviálisan a rengeteg egyéb (akár a 2. fejezetben felsorakoztatott), a klaszteranalízis

feladatát megoldani próbáló algoritmus szemügyre vétele a légi- és űrfelvételeken történő applikáció szemszögéből.

Ahogy azt az ML.NET *k*-means implementációjának eseténél láttuk, érdemes a centroidalapú eljárásoknál még nagyobb hangsúlyt fektetni az inicializációs eljárásokra, hiszen ezek könnyen az eredmény minőségének javulását hozhatják. Ez persze nem csak a *k*-means módszerére érvényes: az ISODATA metódusánál is célravezető lehet több ilyen módszer alaposabb tanulmányozása. Hasonlóan, a felhasználó által manuálisan kiválasztott kezdőpontokból történő indulás lehetősége is jelölt lehet a további elemzésekre.

Felmerül a random forest, tanítóterületek rendelkezésre állásának hiányában történő alkalmazhatóságának problémája is, így tehát a tanító adatként felhasznált részterületek megfelelő (akár automatikus) megválasztása. Erre egy potenciális iránynak mutatkozhat az eredeti felvétel valamilyen reprezentáns (vagy akár a DBSCAN-nél látottakhoz hasonlóan véletlenszerű) mintavételezése és egy kijelölt felügyeletlen módszer, az így kiválasztott pontokra történő futtatása.

Mindezek mellett a DBSCAN megfelelő paraméterezésének könnyítésére célszerű lehet egy, a *k*-means elbow módszerénél látotthoz hasonló megközelítés próbára tétele – persze itt szintén szóba jön az erre vonatkozó egyéb megoldások (mint a HDBSCAN, vagy épp az OPTICS) megvalósításainak elemzése is.

6. FEJEZET

KONKLÚZIÓ

MINDENT EGYBEVÉVE, a diplomadolgozatban definiált fő célkitűzésre – a drónokkal rögzített, nagyfelbontású légifotók (és az űrfelvételek) multispekrális bemeneteire alkalmazható klaszterező módszerek összehasonlító elemzésére – alapozva, a megelőző fejezetek folyamán betekintést nyerhettünk a szakirodalom számos különböző klaszterező algoritmusába, azok működési elveibe, és az eredményeik lehetséges kiértékelési módjaiba. Az ismertetett módszereket felhasználva, a Giwer programcsomag klaszterező metódusainak megvalósításával párhuzamosan alaposabb vizsgálat és összehasonlítás alá került közülük több, különböző megközelítést képviselő (így például centroidokra építő, sűrűségalapú, felügyelt, de manuális és automatikus) megoldás is. Ehhez a fő hangsúlyt a gyakorlati használat mellett előtérbe kerülő attribútumok kapták, mint a memóriaigény és a futási idő nagysága, illetve az eljárások által adott eredmények minősége. Az utóbbi tulajdonságok analizálása során négy, a célterülethez társítható reprezentáns felvétel képezte a mérések bemenetét; illetve a fejlesztés és egyes vizsgálatok folyamata alatt néhány releváns, nyílt forráskódú realizáció is megfontolás alá (és ahol az érdemlegesnek bizonyult, összehasonlításra) került.

Az elemzéseknek köszönhetően megfigyelhettük a vizsgált módszerek erősségeit, és persze a használatukkal esetlegesen járó hátulütőket, de a hozzájuk kapcsolható fő akadályokat is. Arra is fény derült, hogy a *k*-means, az ISODATA, és a random forest eljárások mindegyikének létjogosultsága kétségtelenül megalapozott – de ugyanez jelenthető ki általánosan a DBSCAN esetében is, még ha a konkrét célterületen történő alkalmazás mellett jóval kevésbé mutatkozott versenyképesnek az elemzett társaival szemben. Persze nem juthatott szerephez ezen oldalak keretei között számos egyéb, gyakorlati szempontból szintén jelentős (akár nagyban eltérő alapötletre támaszkodó) klaszterező eljárás: ezek

realizációja és elemzése így elsőként szerepelhet a további kutatási lehetőségek listájában. Mindemellett kijelenthető, hogy – mivel természetszerűleg egyik elemzett megoldás sem tud egyértelműen a többi fölé emelkedni – mindenképpen a felhasználó előnyére válik az eszköztárába venni minél több különböző módszert, hogy a konkrét feladat körülményei által definiált megkötésekhez alkalmazkodva választhasson ezek közül, a várt kimenet lehető leghatékonyabb elérése érdekében.

Hangsúlyozandó tapasztalat, hogy ezen relatíve nagy erőforrásigényű folyamatokat definiáló eljárások megvalósítása során igazán tekintélyes jelentőséggel bír a célterület ismeretében beépíthető apró optimalizációk összessége, illetve az, hogy az általános megoldásokban (vagyis gyakran a szabadon elérhető könyvtárakban is) az ilyen finomítások hiánya könnyen a működés rovására mehet. A Giwer programcsomagjában helyet kapó klaszterező metódusokról így kijelenthető, hogy jó alapot tudnak képezni a drónfelvételeken, és a távérzékelés konkrét szakterületéből vett egyéb hasonló jellegű bemeneteken történő felhasználásra.

Megbizonyosodhattunk ezeken felül a tényről, hogy ezen eljárások sem lehetnek hibátlanok, erősítve a klaszteranalízis területén történő további fejlesztések iránti törekvés igényét. Érdemes megemlíteni utóbbiak irányainak színességét: gyakran jelennek meg olyan eljárások, amelyek más szakterületekből merített ötletekre alapoznak, gondolva itt a például a random forest döntési fáinak esetére. Hasonlóan rátalálhatunk egyéb területek nyomaira – így akár az evolúciós algoritmusokra, vagy a neurális hálózatokra építő módszerek eseteiben. A problémakör megoldásainak alakulása szoros párhuzamban áll tehát a szakma (és a tudomány) többi ágával, avagy nem mellékes az általuk felölelt kompetenciák nélkülözhetetlensége.

Összességében tehát a dolgozatban válaszokat kaptunk a téma megválasztásával kitűzött kérdésekre, az elkészítés (és a felhasználói működtetés) folyamata során előtűnő újabb problémákra, még ha persze mindez nem is jelenti azt, hogy ne került volna terítékre megannyi megoldásra váró feladat és további kutatási lehetőség a témához kapcsolhatóan. Mindezek mellett a dolgozaton történő munka alatt szerzett implementációs; a felhasznált eszközökre, a rendelkezésre álló megvalósításokra és programkönyvtárakra vonatkozó; illetve nem utolsó sorban a konkrét eljárásokhoz és azok tárgyköréhez kapcsolódó személyes tapasztalat és készség sem elhanyagolható eleme a diplomadolgozat készítésének folyamatával adódó értékek sorának.

IRODALOMJEGYZÉK

[1] D. Reinsel, J. Gantz és J. Rydning. *The Digitization of the World from Edge to Core*. IDC, 2018. https://seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (utolsó elérés: 2021. 05. 13.).

[2] I. Elek. *Giwer's Users' Guide*. Eötvös Loránd University, 2021. ápr. http://mapw.elte.hu/giwer/shortDesc.pdf (utolsó elérés: 2021. 05. 01.).

[3] I. Elek és M. Cserep. „Processing Drone Images with the Open Source Giwer Software Package". *Proceedings of the Future Technologies Conference*. Springer. 2021, 201–209. old.

[4] N. Memarsadeghi és tsai. „A Fast Implementation of the ISODATA Clustering Algorithm". *International Journal of Computational Geometry & Applications* 17 (2007. febr.), 71–103. old. DOI: 10.1142/S0218195907002252.

[5] R. Xu és D. Wunsch. „Survey of Clustering Algorithms". *IEEE Transactions on Neural Networks* 16.3 (2005), 645–678. old.

[6] M. R. Anderberg. *Cluster Analysis for Applications*. 19. köt. Probability and Mathematical Statistics. Academic Press, 1973. ISBN: 0120576503.

[7] R. Tibshirani, G. Walther és T. Hastie. „Estimating the Number of Clusters in a Data Set via the Gap Statistic". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), 411–423. old.

[8] G. H. Ball és D. J. Hall. *ISODATA: A Novel Method of Data Analysis and Pattern Classification*. Techn. jel. Menlo Park: Stanford Research Institute, 1965.

[9] D. Pelleg és A. Moore. „*X*-means: Extending *k*-means with Efficient Estimation of the Number of Clusters". *In Proceedings of the 17th International Conf. on Machine Learning*. Morgan Kaufmann, 2000, 727–734. old.

[10] L. Morissette és S. Chartier. „The *k*-means Clustering Technique: General Considerations and Implementation in Mathematica". *Tutorials in Quantitative Methods for Psychology* 9.1 (2013), 15–24. old.

[11] S. Äyrämö és T. Kärkkäinen. „Introduction to Partitioning-based Clustering Methods with a Robust Example". *Reports of the Department of Mathematical Information Technology. Series C, Software engineering and computational intelligence* 1/2006 (2006).

[12] J. B. MacQueen. „Some Methods for Classification and Analysis of Multivariate Observations". *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Szerk. L. M. L. Cam és J. Neyman. 1. köt. University of California Press, 1967, 281–297. old.

[13] E. Schubert és P. J. Rousseeuw. „Faster *k*-medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms". *International Conference on Similarity Search and Applications*. Springer. 2019, 171–187. old.

[14] R. T. Ng és J. Han. „CLARANS: A Method for Clustering Objects for Spatial Data Mining". *IEEE Transactions on Knowledge and Data Engineering* 14.5 (2002), 1003–1016. old.

[15] L. Kaufman. *Finding Groups in Data : An Introduction to Cluster Analysis*. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics. New York: Wiley, 1990. ISBN: 0471878766.

[16] M. Ester és tsai. „A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 96. köt. 34. 1996, 226–231. old.

[17] M. Ankerst és tsai. „OPTICS: Ordering Points to Identify the Clustering Structure". *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, 1999, 49–60. old.

[18] E. Schubert és tsai. „DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". *ACM Transactions on Database Systems* 42.3 (2017), 1–21. old.

[19] L. McInnes és J. Healy. „Accelerated Hierarchical Density Based Clustering". *2017 IEEE International Conference on Data Mining Workshops*. IEEE. 2017, 33–42. old.

[20] J. Schneider és M. Vlachos. „Fast Parameterless Density-based Clustering via Random Projections". *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. 2013, 861–866. old.

[21] R. Sibson. „SLINK: An Optimally Efficient Algorithm for the Single-link Cluster Method". *The Computer Journal* 16.1 (1973), 30–34. old.

[22] T. Zhang, R. Ramakrishnan és M. Livny. „BIRCH: An Efficient Data Clustering Method for Very Large Databases". *SIGMOD Rec.* 25.2 (1996), 103–114. old. ISSN: 0163-5808. DOI: 10.1145/235968.233324.

[23] L. Rokach. „Ensemble-based Classifiers". *Artificial Intelligence Review* 33.1 (2010), 1–39. old.

[24] S. R. Safavian és D. Landgrebe. „A Survey of Decision Tree Classifier Methodology". *IEEE Transactions on Systems, Man, and Cybernetics* 21.3 (1991), 660–674. old.

[25] D. Pfitzner, R. Leibbrandt és D. Powers. „Characterization and Evaluation of Similarity Measures for Pairs of Clusterings". *Knowledge and Information Systems* 19.3 (2009), 361–394. old.

[26] B. Desgraupes. „Clustering Indices". *University of Paris Ouest-Lab Modal'X* 1 (2013).

[27] H. Li és tsai. „Performance Evaluation of Cluster Validity Indices (CVIs) on Multi/Hyperspectral Remote Sensing Datasets". *Remote Sensing* 8.4 (2016), 295. old.

[28] C. D. Manning, P. Raghavan és H. Schütze. „Flat clustering". *Introduction to Information Retrieval*. Cambridge University Press, 2008, 321–345. old. DOI: 10.1017/CBO9780511809071.017.

[29] J. T. Tou és R. C. Gonzalez. *Pattern Recognition Principles*. Applied Mathematics and Computation. Addison-Wesley Publishing Company, 1974. ISBN: 9780201075878.

[30] L. Breiman. „Random Forests". *Machine Learning* 45.1 (2001), 5–32. old.

[31] J. A. Richards. *Remote Sensing Digital Image Analysis: An Introduction*. 5th ed. Springer Berlin Heidelberg, 2013. ISBN: 9783642300622.

[32] A. Juan és E. Vidal. „Comparison of Four Initialization Techniques for the *k*-medians Clustering Algorithm". *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition*. Springer. 2000, 842–852. old.

[33] A. Hardy. „On the Number of Clusters”. *Computational Statistics & Data Analysis* 23.1 (1996), 83–96. old. ISSN: 0167-9473. DOI: 10.1016/S0167-9473(96)00022-9.

[34] G. Biau és E. Scornet. „A Random Forest Guided Tour”. *TEST* 25.2 (2016), 197–227. old.

[35] L. Raileanu és K. Stoffel. „Theoretical Comparison Between the Gini Index and Information Gain Criteria”. *Annals of Mathematics and Artificial Intelligence* 41 (2004. máj.), 77–93. old. DOI: 10.1023/B:AMAI.0000018580.96245.c6.

[36] J. Gan és Y. Tao. „DBSCAN Revisited: Mis-claim, Un-fixability, and Approximation”. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, 519–530. old.

[37] J. Jang és H. Jiang. „DBSCAN++: Towards Fast and Scalable Density Clustering”. *Proceedings of the 36th International Conference on Machine Learning*. Szerk. K. Chaudhuri és R. Salakhutdinov. 97. köt. Proceedings of Machine Learning Research. PMLR, 2019, 3019–3029. old.

[38] *ML.NET GitHub Repository*. https://github.com/dotnet/machinelearning (utolsó elérés: 2021. 04. 26.).

[39] *SharpLearning GitHub Repository*. https://www.github.com/mdabros/sharplearning (utolsó elérés: 2021. 04. 26.).

[40] *dotMemory Product Documentation*. https://jetbrains.com/help/dotmemory (utolsó elérés: 2021. 04. 26.).

[41] QGIS Development Team. *QGIS Geographic Information System*. QGIS Association. 2021. URL: https://www.qgis.org.

[42] *Semi-automatic Classification Plugin GitHub Repository*. https://www.github.com/semiautomaticgit/SemiAutomaticClassificationPlugin (utolsó elérés: 2021. 05. 01.).

[43] *MicaSense Altum Data*. https://www.micasense.com/altum-sample-data (utolsó elérés: 2021. 04. 26.).

[44] B. Bahmani és tsai. „Scalable *k*-means++”. *Proceedings of the VLDB Endowment* 5.7 (2012. márc.), 622–633. old. ISSN: 2150-8097. DOI: 10.14778/2180912.2180915.

KÖSZÖNETNYILVÁNÍTÁS

Ezúton szeretnék köszönetet mondani Cserép Mátének, aki a dolgozat és a kapcsolódó implementációk létrejöttét témavezetői segítségével támogatta; Elek Istvánnak, aki a Giwer programcsomagot megtervezte és alapjait megalkotta, a klaszterező eljárások fejlesztésének folyamatát követte és ötleteivel elősegítette; valamint Borbély Dávidnak, a Giwer projektben szereplő, és így a dolgozatban elemzett *k*-means eljárás (és elbow metódus) megvalósításáért, és a hozzá tartozó osztályhierarchia elkészítésében való segítségéért.

A diplomamunka a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg (EFOP-3.6.3-VEKOP-16-2017-00001: Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási technológiák területén).