Klaszterezés telekommunikációs adatokon

Kolok Balázs Csegő

Témavezető: Hévizi László

November 2020

1. Bevezetés

A modern telekommunikációs hálózatok megállás nélkül, rohamosan fejlődtek az elmúlt évtizedekben. Egyre több frekvencia sávon, egyre több és szofisztikáltabb cellát hoznak működésbe a nagyobb és jobb minőségű adatforgalom érdekében. A növekvő adatforgalom szabályozása és irányítás az egyik vezető fejlesztési területté nőte ki magát. A folyamatos fejlesztésekkel és a rengeteg felgyülemlő információval az analitikus módszerek nehezen tudnak lépést tartani. Egyik reményteli megoldási lehetőség a témában felmerülő problémákra a gépi tanulás.

1.1. Az adatokról

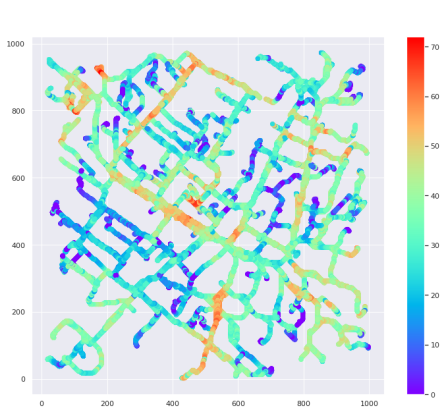
A telekommunikációs adatokat egy szimulációs környezet szolgáltatta számunkra. A szimulátor egy nagyváros belvárosának forgalmát utánozza. A környezet egy 200×200-as rács egy négyzetkilométernyi területen, amin felhasználók mozognak. Tetszőleges sok kiszolgáló cellát telepíthetünk a térképre, amik biztosítják a felhasználók számára az adatforgalmat. A felhasználók szabályos időközönként reportolnak, hogy melyik cellát milyen erősséggel érzékelik egy 0-tól 97-ig terjedő logaritmikus skálán. A szimulációs környezetben így kapunk minden felhasználótól diszkrét időközönként egy a cellák száma hosszú mérés listát, ezt nevezzük RSRP listának.

A projekt során 600011 darab reporttal dolgoztunk, amik RSRP listája 339 mérést tartalmazott. A reportok az RSRP listákon kívül még tartalmazták az aktuális x és y szerinti pozícióját a felhasználóknak, egy felhasználói ID számot és egy időbélyeget a report pillanatáról (1. ábra). Az 2a. ábrán a reportok pozíciója szerint láthatjuk az adatpontokat, a legerősebb cella erőssége szerint színezve.

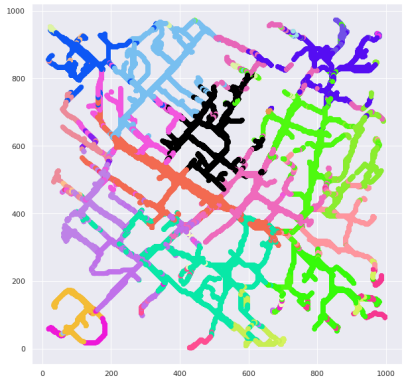
Egyik fontos tulajdonsága az adathalmaznak, hogy a listákban viszonylag kevés nem nulla mérés található, hiszen a felhasználó számára egyszerre csak

	pos_x	pos_y	UID	Timestamp	rsrp_list
0	537.1	595.5	1351	1601996388439000	[42.013000000000005, 21.227999999999994, 10.94...
1	939.0	440.9	1139	1601996388439000	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
2	372.0	465.3	1193	1601996388440000	[19.180999999999997, 5.626999999999995, 43.842...

1. ábra. Az első három felhasználói report



(a) RSRP listák maximális mérései



(b) RSRP listák legerősebb cella szerint csoportosítva

2. ábra. Szimulációs környezet

kevés cella érzékelhető. Ebből kiindulva észszerűnek tűnik dimenziócsökkentéssel kezelhetőbbé tenni az adatokat.

1.2. A Probléma megfogalmazása

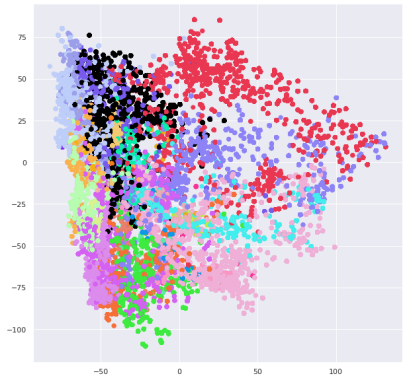
A projektmunka során két főbb problémával foglalkoztunk. Az adatok dimenziócsökkentésével és a megfelelő klaszterező algoritmus megtalálásával. A klaszterezés célja, hogy a rádiós teret partícionáljuk, zónákra bontsuk. Mi lehet ennek a haszna? Hatalmas segítség lenne az adatforgalom irányításának folyamatában, ha előre tudnánk jelezni a felhasználók jövőbeli rádiós helyzetét. Ekkor ugyanis előre fel tudnánk készíteni a felhasználót egy estleges kiszolgáló cella váltásra, ami javítaná az adatszolgáltatás minőségét. De hogy jön ehhez az adatok klazterezése? Tegyük fel, hogy a rádiós teret sikerült felbontanunk $X = X_1, \ldots, X_n$ diszjunkt klaszterekre. Ekkor bármilyen RSRP listára meg tudjuk határozni az őt tartalmazó X_i-t. Tehát nyomon tudjuk követni, hogy a felhasználó melyik klaszterben járt a reportok pillanatában. Az így kapott klaszterszekvenciákra felépíthető egy prediktáló módszer, ami az eddig bejárt klaszterekből előre jelzi, hogy a felhasználó melyik klaszterben milyen valószínűséggel fog tartózkodni a következő report pillanatában.

2. Dimenziócsökkentés

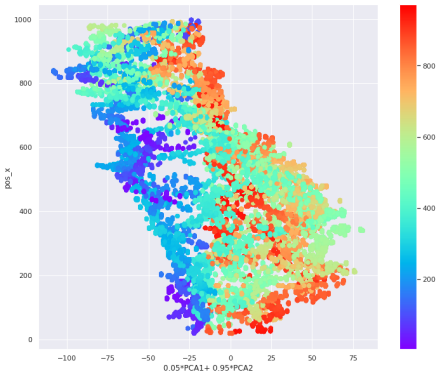
Célunk a dimenziócsökkentéssel, hogy kezelhetőbb adathalmazt nyerjünk, amit fel tudunk használni a klaszterező algoritmusok inputjaként. Két dimenziócsökkentő módszert alkalmaztunk az adathalmazon a főkomponens analízis (PCA) és az Isomap algoritmusokat.

2.1. PCA alkalmazása

Az PCA széleskörben elterjedt dimenziócsökkentésre használt módszer. Egy n dimenziós vektorokból álló adathalmaz főkomponensei n dimenziós ortogonális



(a) Első és második főkomponens szerinti vetítés



(b) főkomponensek és a koordináták összefüggése

3. ábra. PCA

vektorok, amelyek karakterizálják, hogy mely irányokban terjed szét legjobban az adathalmaz. Az első főkomponens vektor maximalizálja az irányára vetített adatpontok szórását. Az i-dik főkomponens vektor ezután úgy határozható meg, hogy ortogonális az első $i-1$ főkomponens vektorra és maximalizálja a irányára levetített adatpontok szórását.

Az n dimenziós adatok k dimenziós reprezentánsát a főkomponensek meghatározás után, úgy nyerjük, hogy az eredeti adatpontokat levetítjük az első k főkomponens által feszített térbe. A 3a) ábrán látható az első és a második főkomponens által feszített térre való levetítés az adatoknak. A pontokat a legerősebb RSRP mérésük szerint színeztük, azaz azok a pontok egyszínűek, amelyek RSRP listájában azonos cellához tartozik a legerősebb mérés. Bizakodásra ad okot, hogy elkülöníthetőek az egyes színek, és nem hasonlít egy random színezésre.

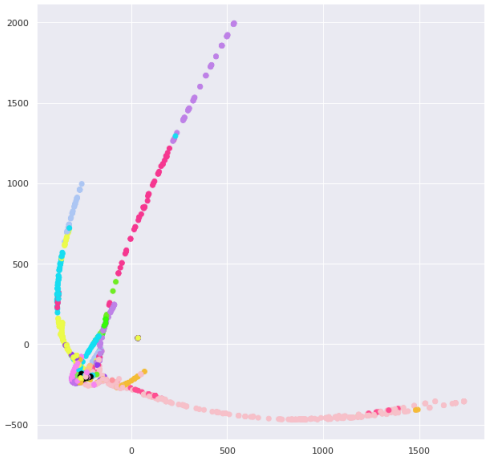
A 3b) ábrán az első és a második főkomponens és a koordináták összefüggését szemléltettük. A grafikon vízszintes tengelye a $0.05\text{PCA}_1 + 0.95\text{PCA}_2$ értéke a pontoknak, ahol PCA_1 értéke egy pontnak az első főkomponens szerinti koordinátáját jelenti, PCA_2-t hasonló módon kapjuk. A grafikon függőleges tengelye az x koordinátája a reportoknak. A pontokat pedig az y koordinátájuk szerint színeztük. Enyhe összefüggés figyelhető meg az értékek között, de kijelenteni nem tudjuk, hogy erős lineáris függés lenne az x koordináta és a $0.05\text{PCA}_1 + 0.95\text{PCA}_2$ összeg között.

2.2. Isomap alkalmazása

Az Isomap algoritmust egyre széleskörben elterjedt nem lineáris dimenziócsökkentő algoritmus. Működése az adatok gráf alapú reprezentációján alapszik. Legyen minden adatpont egy csúcs a gráfban. Ezután húzzuk be az éleket. Minden csúcsból vezessen él a k legközelebbi csúcsba, ahol k az algoritmus inputja. Két csúcs távolsága az eredeti adatpontok távolsága. Egy másik opció az élkonstrukcióra, hogy egy csúcsot azokkal a csúcsokkal kössük össze, amelyek tőle egy adott távolságon belül vannak. Miután meghatároztuk a gráf éleit, súlyozzuk az éleket a csúcsok közötti távolsággal. Ezután minden pontpárra

számoljuk ki a legrövidebb utat a gráfban Dijkstra vagy Floyd-Warshall algoritmussal. Az így kapott távolságok alapján hozzuk létre az alacsonyabb dimenziós beágyazást.

Problémaként merült fel, hogy kevés adatpont feldolgozására alkalmas a módszer. Ugyanis a távolságok kiszámolására n^2 tárra van szükség. Ez esetünkben viszont már 20000 adatpont esetén sem biztosított. Így az adathalmazt véletlenszerűen mintavételezve csökkentettük adatpontok számát 10000-re.



4. ábra. Az adatok 2D-s beágyazása Isomap algoritmussal

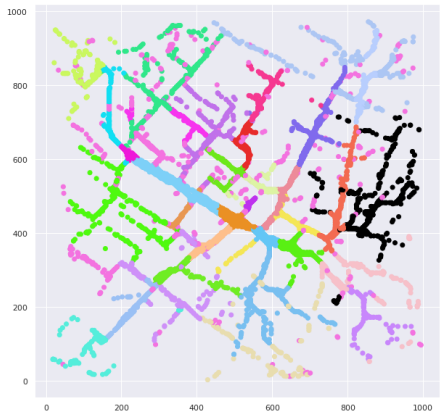
Az 4. ábrán is látszik, hogy esetünkben az Isomap módszer nem adja meg a számunkra elvárt eredményt. Az adatok többségét egyhelyre sűríti, nem tartja meg az egymáshoz viszonyított távolságokat.

3. Klaszterezés

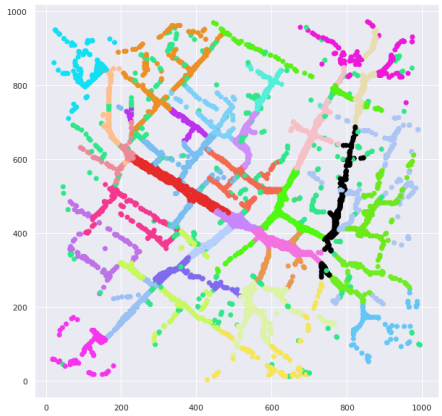
A projektmunka fő célkitűzése az volt, hogy megfelelő klaszterező módszert találjunk, ami jól klaszterezi az RSRP listákat. A feladat tehát n RSRP listát k darab klaszterbe sorolni. Mit jelent az, hogy jó klaszterezés? Fontos, hogy információval szolgáljon az adott felhasználó térbeli és rádiós helyzetéről. Előny lenne, ha két cella határán a klaszterek gyorsabban váltakoznának, mint a cella közepén. Ez azért van mert, ha a felhasználó a cella közepén tartózkodik, nem fog kiszolgáló cellát váltani, mivel az aktuális kiszolgáló cellája domináns. Ellenben a cellák határán valószínűbb a cella váltás. Ezért ott jobb, ha részletesebb képet adnak a klaszterek a felhasználó helyzetéről.

Első nekifutásra az egész adathalmazon klasztereztünk. Majd heurisztikusan feltettük, hogyha két adatpont legerősebb mérése nem megegyező cellához tartozik, akkor azok az adatpontokat csakis különböző klaszterbe sorolhatjuk. Így külön párhuzamosan klaszterezhetjük az adatpontotokat, első körben különválasztva az adatokat legerősebb cella szerint, majd ezeken a csoportokon belül alakítjuk ki a kisebb klasztereket.

A projektmunka során a K-Means, K-Medoids, a Agglomerative Clustering, a Birch és a Self Organizing Map (SOM) algoritmusokat vizsgáltuk meg és alkalmaztuk a feladat megoldására.



(a) Agglomerative Clustering által adott klaszterek, $k = 34$



(b) BIRCH által adott klaszterek, $k = 34$

3.1. K-means és K-medoids alkalmazása

A K-Means algoritmus egyszerű, és viszonylag gyors algoritmus. Célja az n darab adatpontot k klaszterbe sorolni úgy, hogy az adatpontok a saját klaszterük középpontjához legyen a legközelebb az összes középpont közül. Bementként megkapja a klaszterek k számát. Majd tetszőleges k darab középpontból kiindulva minden adatpontot a hozzá legközelebb lévő középpont klaszterébe helyez. Ezután minden klaszterre kiszámolja az új klaszter középpontokat, majd újra hozzárendeli az adatpontokhoz a legközelebbi középpont klaszterét, majd megint középpontot számol, és így tovább. A középpont számítás miatt csak az euklideszi távolság függvénnyel használható.

Az algoritmus továbbfejlesztése a K-Medoids. A K-Medoids futása közben nem a középpontokkal számol, hanem minden klaszterhez egy reprezentáns elemet választ, ami minimalizálja a klaszterbe tartozó pontoktól vett távolságok összegét. Így tetszőleges távolság függvénnyel alkalmazhatjuk az algoritmust.

3.2. Agglomerative Clustering alkalmazása

Az Agglomerative Clustering (AC) algoritmus hierarchikus klaszterekből fa szerkezet épít fel. Kiindulásként a legalsó szinten minden adatpont külön klaszterben van, majd az egymáshoz legközelebb lévő klaszter párt összevonjuk. Az összevont klaszter lesz a két klaszter szülő klasztere, amit épp az aktuális szinttel egyre nagyobb szintre helyezünk a fában. Az így kapott fa minden i. szintre meghatároz egy i darab klaszterből álló felbontást.

A klaszterek közötti távolságot tetszőlegesen választhatjuk. Leggyakrabban használt távolságok A és B klaszter között:

Minimális távolság: $\min d(a, b) | a \in A, b \in B$,

Maximális távolság: $\max d(a, b) | a \in A, b \in B$,

Átlagos távolság: $\frac{1}{|A||B|} \sum_{a \in A, b \in B} d(a, b)$.

Számunkra fontos tulajdonsága az algoritmusnak, hogy tetszőleges távolságfüggvény szerint tud klaszterezni.

3.3. BIRCH alkalmazása

A BIRCH algoritmus [1] hatékony klaszterező módszer, ami az adatok egyszeri végig olvasásával is elfogadható eredményt képes felmutatni. Az algoritmus futása során egy kiegyensúlyozott fát épít fel. Az algoritmus futása négy fázisra osztható, amelyek közül a második és a negyedik opcionális.

1. Az első fázisban az algoritmus felépít egy kiegyensúlyozott CF (clustering feature) fát. Adott A halmaz CF klaszter reprezentánsa alatt egy $(|A|, L, S)$ hármast értünk, ahol $L = \sum_{a \in A} a$, valamint $S = \sum_{a \in A} a^2$. A klaszter reprezentánsok segítségével a CF fa dinamikusan az adatpontok beszúrásával épül fel a következő tulajdonságok megtartásával:
   - Minden levélben legfeljebb L darab CF_i tárolunk, amelyek legfeljebb T átmérőjű alklaszterek reprezentánsai.
   - Minden belső pontban legfeljebb B darab $(CF_i, child_i)$ párt tartalmaz, ahol CF_i a belső pont i leszármazottjához tartozó reprezentáns, $child_i$ pedig egy i-re mutató pointer. A T korlátot a paraméterként kapjuk, a B és L számokat pedig egy P memória limit korlátozza, ami a csúcs méretére vonatkozik. Ezt is a felhasználó szabhatja meg.

   Egy x adatpont beszúrásánál a fa gyökerétől indulva mindig arra a gyerekre lépünk, amelyik reprezentánsa által meghatározott klaszter közelebb van x-hez. Így elért levél alklaszterébe rakjuk x-et. További meggondolható adatkezelésre van szükség, ha az alklaszter mérete túl lépné az L korlátot.

2. Az így előállított CF fát a második fázisban optimálisan javíthatjuk mélységének csökkentésével. A leveleket végig vizsgálva és javítva a levelekben szereplő alklasztereken.

3. A harmadik fázisban tetszőleges klaszterező algoritmussal klaszterekbe rendezzük a levelekben található alklasztereket.

4. A negyedik opcionális fázisban az adatpontokon meg egyszer végig menve javítható a klaszterezés.

A BRICH előnye az eddig említett algoritmusokkal szemben, hogy az adatok egyszeri olvasásával tud klaszterezni.

3.4. Összegzés, eredmények

Egy klaszterező módszerek vizsgálata, jóságuknak eldöntése nem egyértelműen meghatározható. Ugyanis ezek unsupervised tanuló algoritmusok, azaz nincs egy előre meghatározott jó csoportosítása az adatoknak, amivel validálni tudnánk a klaszterezés jóságát. Abban az esetben, ha mégis van egy validáló felosztásunk, akkor az adjusted random index (ARI) megfelelő mérőszám lehet. Az ARI két felosztás hasonlóságát határozza meg egy 0 és 1 közötti számmal a következő képpen. Legyen D az adathalmazunk és rajta két klaszterezés $X = \{X_1, \ldots, X_k\}$ és $Y = \{Y_1, \ldots, Y_l\}$.

a: D azon elempárjainak száma, amelyek X -ben egy halmazban vannak és Y-ban is egy halmazban vannak.

b: D azon elempárjainak száma, amelyek X -ben különböző és Y-ban is különböző halmazban vannak.

c: D azon elempárjainak száma, amelyek X -ben különböző, de Y-ban egy halmazban vannak.

a: D azon elempárjainak száma, amelyek X -ben egy, de Y-ban egy halmazban vannak.

A két felosztás R random indexét a következő képlet adja:

$$R(X,Y) = \frac{a+b}{a+b+c+d}$$

ARI pedig ebből számolható:

$$\mathrm{ARI}(X,Y) = \frac{R - \mathbf{E}(R)}{\max(R) - \mathbf{E}(R)},$$

ahol $\mathbf{E}(R)$ a várható random index. Minél közelebb van 0-hoz az ARI, annál jobban különbözik két klaszterezés. Teljes egyezés esetén (indexeléstől eltekintve) 1 az ARI. Az 1 táblázatban az ARI bemeneti X a klaszterező módszer által adott és az Y pedig az RSRP listák maximális erősségű cellája szerinti felosztás. Azaz Y szerint akkor van két RSRP lista egy klaszterben, ha ugyanazon cellához tartozó a mérésük a legerősebb. A klaszterek száma X és Y esetén is 34 volt.

Ha azt feltételezzük, hogy akkor jó egy klaszterezés, ha az egy klaszterbe kerülő adatpontok között kicsi távolságok vannak, akkor használhatjuk az átlagos középpontól való maximális eltérést:

$$\mathtt{maxdist}_{av}(X) = \frac{\sum_{i=1}^{k} \max_{x_i \in X_i}(d(\overline{x}, x_i))}{k},$$

ahol d valamilyen D-n értelmezett távolság függvény.
Ugyanezen feltevés mellett, használhatjuk a klaszterek átlagos szórását ($\mathtt{std}_{av}$) is. Az utóbbi két mérőszám bemeneténél a reportok pozícióját tekintettük bemenetnek. Egy klaszter szórásának a hozzá tartozó reportok pozíció vektorainak szórásaként határoztuk meg. Hasonlóan definiáltuk a maximális eltérést.

	ARI	$\mathtt{maxdist}_{av}$	$\mathtt{std}_{av}$
K-means	0.2696	196.55	76.24
K-means PCA után	0.2275	198.97	80.02
K-means Isomap után	0.2275	201.57	85.31
K-medoids	0.2121	200.15	82.64
AggClus	0.2811	171.25	71.94
BIRCH	0.3104	194.67	77.19

1. táblázat. A klaszterező módszerek összehasonlítása

Az algoritmusokat scikitlearn [2] környezetben futtattuk. A fenti táblázat adatai szerint megállapítható, hogy a vizsgált algoritmusok hasonlóan teljesítettek az adathalmazon. Azon feltevés mellet, hogy akkor jó a klaszterezés, ha egy klaszteren belül kis távolságok vannak és kicsi a szórás a Agglomerative Clustering mondható a legjobbnak, és a BIRCH is reményteli próbálkozásnak tűnik. Ez azt sejteti, hogy a hierarchikus klaszterezési módszerek célravezetőek lehetnek.

Ami minket különösen érdekelt, hogy egy cellán belül hogyan klaszterez egy algoritmus. Képes-e arra, hogy a cella határokon finomabb klaszterezést állítson

elő mint a belsején? Ennek vizsgálatához tekintsük az algoritmusok azon futásait, ahol olyan adatpontokat adtunk meg inputként, amik legerősebb cellája megegyezik. A 6. ábrán ezen futások végeredménye látható, különböző színnel színezve a klasztereket. Megfigyelhető, hogy mindegyik klaszterező hasonlóan nagyjából egyenletes méretű klasztereket produkál, ha az alapbeállításnak megfelelő euklideszi távolságot használjuk.

Feltehető tehát, hogy a kívánt klaszterezés eléréséhez valamilyen speciális távolságfüggvényre van szükségünk. Olyan távolságfüggvényt keresünk, amelyek a magas és az alacsony mérések közötti különbségeket kis, míg a közepes értékek közötti különbségeket nagy súllyal számolja. Ennek hátterében az a feltevésünk áll, hogy a nagy értékeknél valószínűleg egy cella közepén, a közepes értékeknél pedig a cellák határán lehetünk. A kis értékek pedig nem megbízhatóak rádiós tapasztalatokból kiindulva. A cellák határán finomabb klaszterezést szeretnénk, ezért az ott található adapontok között nagyobb távolságot szeretnénk megkövetelni, mint a belső pontok között.

A projektmunka során több távolságfüggvénnyel kísérleteztünk. Két ilyen jelöltünket következőképpen definiáljuk:

$$d_s(v,u) = \left(\sum_{i=1}^{n} \left(w_s(v_i,u_i)|v_i - u_i| \right)^r \right)^{\frac{1}{r}}$$

$$w_s(a,b) = \sin \frac{(a+b)\pi}{194}$$

$$d_\delta(v,u) = \left(\sum_{i=1}^{n} \left(w_\delta(v_i,u_i)|v_i - u_i| \right)^r \right)^{\frac{1}{r}}$$

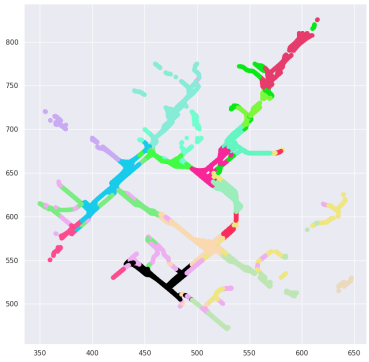
$$w_\delta(a,b) = \frac{1}{|c|\sqrt{\pi}} \exp\left(-\left(\frac{\frac{(a+b)\pi}{194} - \frac{1}{2}}{c} \right)^2 \right),$$

ahol c konstans paraméter. Az $\frac{(a+b)\pi}{194}$ hányados két mérés összegét 0 és 1 közé normálja, mivel a mérések 0 és 97 közé esnek. A $w(.)$ súlyfüggvények pedig a kívánt módon súlyozzák a koordináta különbségeket mértékük szerint (lsd. 8. ábra). A 7. ábrán az AC algoritmus által kialakított klaszterek láthatóak a fenti távolságfüggvényeket használva. Az ábrákon megfigyelhető az általunk kívánt tulajdonsága a klaszterezésnek, a cella belső pontjai egy nagyobb klaszterbe tartoznak, míg a határon kisebb klaszterek találhatók.

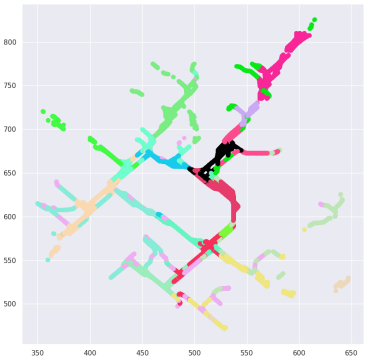
Hivatkozások

[1] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, „Birch: an efficient data clustering method for very large databases," in *SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, New York, NY, USA, 1996, pp. 103–114, ACM.

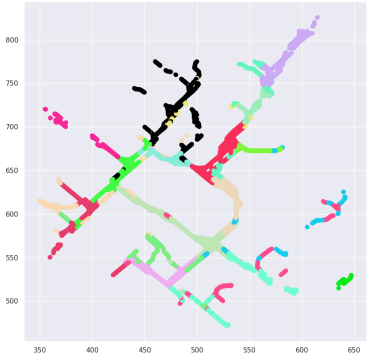
[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, „Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.



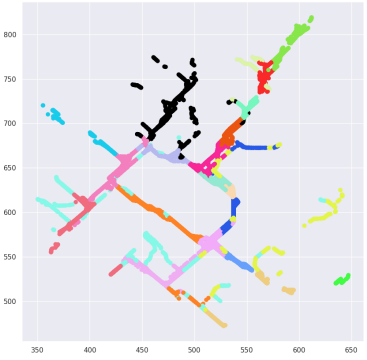
(a) kmeans



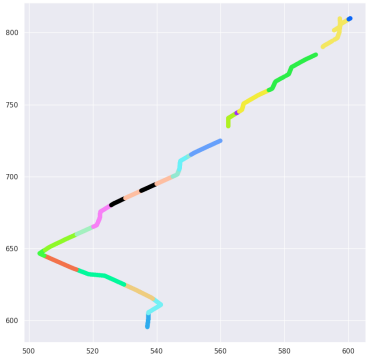
(b) kmeans pca után



(c) BIRCH

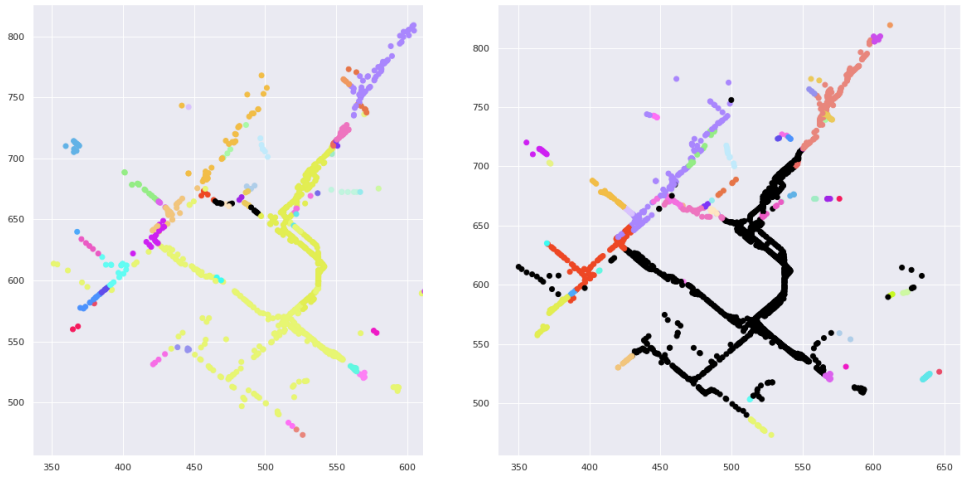


(d) AC



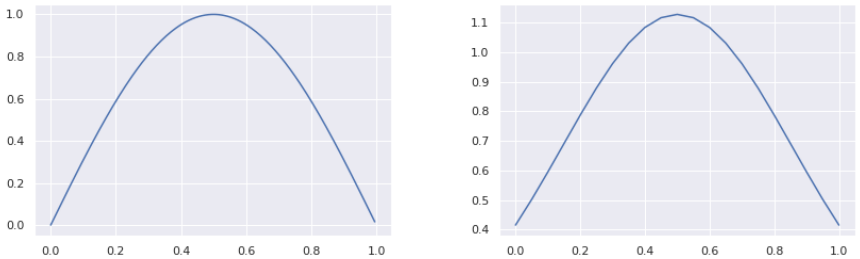
(e) Egy felhasználó klaszter váltásai az első cellán belül

6. ábra. RSRP listák klaszterezése, amelyekre $\mathtt{argmax}(\mathtt{RSRP}) = 1$, azaz azok az RSRP listák amelyeknek az első mérése a maximális



(a) AC d_δ távolságfüggvénnyel $c = 0.5$ (b) AC d_δ távolságfüggvénnyel $c = 0.7$

7. ábra. Első cella RSRP listáinak klaszterezése AC algoritmussal speciális távolság függvénnyel



(a) w_s súlyfüggvény (b) w_δ súlyfüggvény $c = 0.5$

8. ábra. Súlyfüggvények