Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Knoll Zsolt

HIERARCHIKUS STRUKTÚRÁJÚ ADATHALMAZOK SPEKTRÁLIS KLASZTEREZÉSE

TDK DOLGOZAT

KONZULENS

Dr. Szűcs Gábor, Papp Dávid

BUDAPEST, 2019

Tartalomjegyzék

Összefoglaló

A klaszterezés az adatelemzés egyik ismert módszere, ennek több fajtája is van. A választóvonal élessége szerint létezik kemény és puha klaszterezés. A kemény klaszterezés csoportjába azok a módszerek tartoznak, melyeknél egy pont vagy beletartozik egy klaszterbe, vagy nem. Ezzel ellentétben a puha klaszterezésnél minden klaszternek része valamilyen mértékben az adott pont. Ehhez az elem kap egy tagsági súlyt, amelynek az értéke 0 (semmilyen mértékben nem tartozik bele a klaszterbe), valamint 1 (teljes mértékben beletartozik a klaszterbe) közé esik.

A dolgozatomban több olyan – kemény klaszterezéshez tartozó – spektrális klaszterezésen alapuló algoritmust mutatok be, mely nem pontokat, hanem ponthalmazokat rendez csoportokba. Ennek motivációja, hogy a klaszterező algoritmus hierarchikus struktúrájú (azaz ahol a ponthalmazok, még magasabb szinten lévő ponthalmazokba vannak szervezve) adatokon is alkalmazható legyen. Ehhez kidolgoztam egy módszert, így egy általános (generális) megközelítéshez jutottam. Ez a megközelítés gyakorlati szempontból is hasznos. A dolgozatban két különböző adathalmazon mutatom be ezen algoritmusok hasznosságát. Az első adathalmazon albumokat klaszterezek, melyeket az albumban szereplő dalok jellemeznek. A második adathalmazon növényekről készült képeket csoportosítok. Itt a hierarchikus struktúrát úgy kapjuk meg, hogy a pontoknak a növényekről készült képek, míg a csoportoknak a növényi fajok felelnek meg. A kapott klasztereken megvizsgáltam a pontok közötti kapcsolatokat abból a szempontból, hogy mennyire tükrözi az elvárt struktúrát, így lehetővé vált különböző klaszterező algoritmusok összehasonlítása.

Abstract

Clustering is a well-known method of data analysis, which has several types. According to the sharpness of the dividing line, there are hard and soft clustering. Hard clustering means, that a point is either included in a cluster or not. But in soft clustering, each cluster contains the given point. To do this, the element receives a membership weight of between 0 (not included in the cluster at all) and 1 (fully included in the cluster).

In my TDK paper I present several algorithms based on spectral clustering, which clusters groups of points rather than points. This is motivated by that the clustering algorithm can be run on hierarchical datasets (it means, that the point-sets are included in higher level point-sets). For this, I developed a method, thus it leads to a general approach. This approach is also useful in practice. In my paper I present the utility of these algorithms on two different datasets. On the first dataset, I cluster albums, which are characterized by the songs in the album. In the second dataset, I group pictures of plants. Here the images of plants represent the points, and the species are the point-sets in the hierarchical dataset. On the obtained clusters, I examined the relationships between the points from the point of view of how they reflect the expected structure, thus it was possible to compare different clustering algorithms.

1 Bevezetés

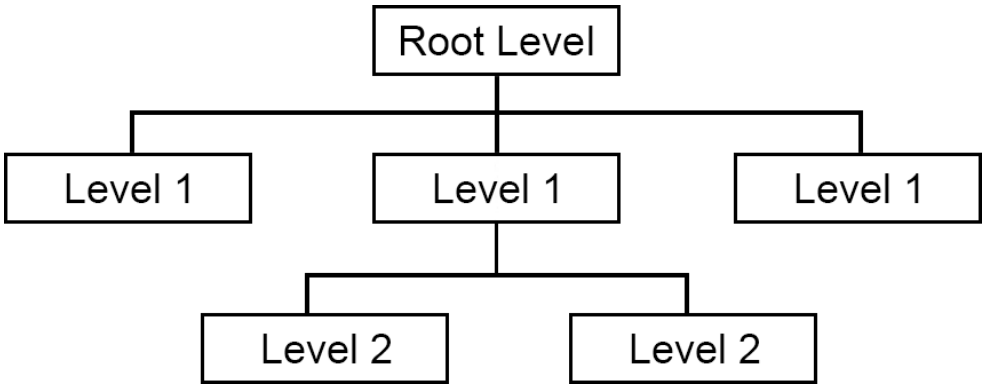
„Az adat az új olaj”. Az idézet sokaknak ismerős lehet, akik már hallottak a BigData-ról vagy az adatelemzésről. Az idézet arra utal, hogy a mai világban elérhető adatokból kinyerhető információkban rengeteg potenciál van, amely nemcsak az üzleti életben, de a tudomány különböző területein is hasznos lehet, emiatt az adatelemzés területeivel egyre többen foglalkoznak.

Ahhoz hogy az adatokban rejlő információkat kinyerjük már nem elég emberi erőforrást alkalmaznunk, hanem a számítógépekre és a gépi tanulásra is egyre nagyobb szükségünk van. Aszerint, hogy a gépi tanuláshoz szükségesünk van-e tanító adathalmazra beszélhetünk felügyelt és felügyelet nélküli gépi tanulásról.

A felügyelt gépi tanulási módszerek lényege, hogy az általunk ismert adatokon építünk egy modellt, amelyet az újonnan érkező adatokra tudunk alkalmazni. Ezen módszerek közé tartozik a klasszifikáció és a regresszió, melyek lényege, hogy az egyedek egy nem ismert tulajdonságát meghatározzuk az általunk ismert adatokból. A klasszifikáció véges számú diszkrét értékeket felvevő tulajdonságok esetén alkalmazható, míg a regresszió a folytonos attribútumok meghatározásánál használható.

A nemfelügyelt gépi tanulási módszereknek nincs szükségük egy tanító adathalmazra, hanem közvetlenül használhatóak az adatokon. Ezek közé tartozik a klaszterezés, amely nem új egyedek egy tulajdonságának meghatározására használható, hanem segítségével a meglévő egyedeinket tudjuk csoportosítani.

A hierarchikus adatmodell a valóságban előforduló hierarchikus szerkezetek leképzésére szolgál. A gyakorlati életben a társadalmi, ipari, de még a természeti rendszerek is hierarchikus felépítést mutatnak. A hierarchikus adatmodellben az adatok fastruktúrába vannak szervezve, ahol a felül lévő egyedeket szülőknek, míg az alul lévő egyedeket gyerekeknek nevezzük. A fa legmagasabb szinten lévő elemét nevezzük gyökérnek, melyből az összes adatpont közvetve, vagy közvetlenül leszármazik. Fontos, hogy minden gyereknek csak egy szülője lehet, azonban egy szülőnek egyszerre több gyereke is lehet, vagyis egy-több kapcsolat van a szülők és a gyerekek között. Hierarchikus adatok struktúráját szemlélteti az 1. ábra.

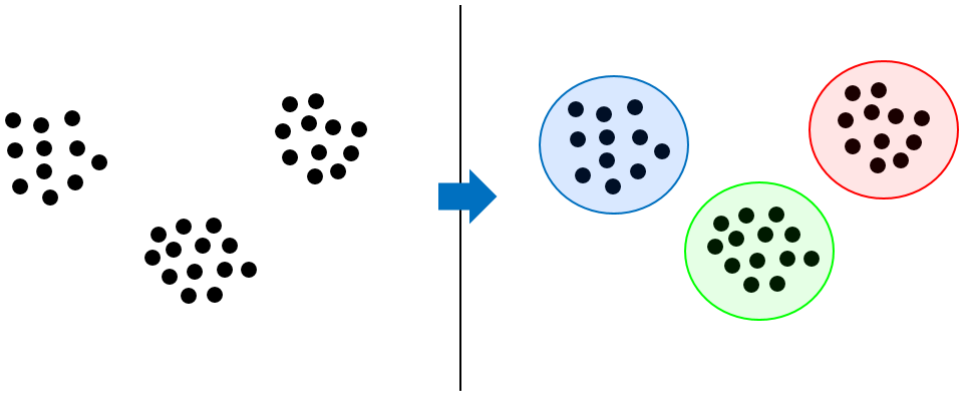


1. ábra Hierarchikus adatok szerkezete

Dolgozatom célja, hogy olyan klaszterező algoritmusokat mutassak be, melyek nem pontokat, hanem hierarchikus struktúrájú adatokat képesek klaszterezni. Ehhez elsőként a klaszterezésről írok, megemlítve a különböző klaszterezési módszereket, külön kiemelve a spektrális klaszterezést és ennek működését. A negyedik fejezetben térek ki a hierarchikus struktúrájú adatok klaszterezésére, elsőként a reprezentációt készítő algoritmusokat felsorolva, majd a pontokat külön klaszterező algoritmusokról írok. Az ezt követő fejezetben mutatom be azt a feltételrendszert melynek segítségével elérhető, hogy a pontokat külön klaszterező algoritmusoknál minden esetben egy klaszterbe kerüljenek a pontok. Ezután bemutatom az algoritmusok tesztelésére használt adathalmazokat, legvégül pedig az algoritmusokat értékelem ki, és hasonlítom össze őket.

2 Klaszterezés áttekintése

A klaszterezés egyike az adatelemzés legnépszerűbb módszereinek, melyet többek között a képelemzésben, statisztikában, biológiában és a pszichológiában is alkalmaznak. Lényege, hogy az adatainkat csoportosítsuk úgy, hogy a hasonló egyedek kerüljenek egy csoportba. A klaszterezés koncepcióját az 2. ábra szemlélteti.



2. ábra Klaszterezés koncepciója

2.1 Távolság metrikák

A klaszterezéshez szükségünk van egy hasonlósági függvényre, ami azt definiálja, hogy két adatpontot mikor mondunk hasonlónak. Több széles körben elterjedt távolság függvény is létezik. Az egyik legismertebb ilyen távolság az euklideszi távolság, ami kétdimenzióban a két pontot összekötő szakasz hosszát adja meg, több dimenzióban pedig a következő képletet használva kapjuk meg a p és q pontok közötti távolságot:

$$d = \sqrt{\sum_i (p_i - q_i)^2} \tag{1}$$

Másik ismert távolságmérték a Manhattan-távolság. Az elnevezés onnan származik, hogy Manhattan szigetén a merőleges útkereszteződések miatt a valóságban ezzel a távolságmértékkel számítható út hosszat kell megtenni két kereszteződés között. A távolságot a következő képlet segítségével tudjuk meghatározni p és q pontok között:

$$d = \sum_i |p_i - q_i| \tag{2}$$

Egy harmadik hasonlósági mérték a Gaussian kernel. Ezt a hasonlósági mértéket a következő képlet segítségével tudjuk számolni két pont között:

$$d = e^{-\frac{|x_i - x_j|^2}{2 * \sigma^2}} \quad (3)$$

ahol σ paraméter szabályozza a környezet szélességét.

2.2 Általános klaszterező módszerek

A klaszterező algoritmusoknak több fajtája is van, melyek közül a legalapvetőbbek a hierarchikus, sűrűség alapú, eloszlás alapú, valamint középpont alapú megközelítést alkalmazó eljárások.

A k-közép algoritmus [9], a legegyszerűbb középpont alapú megközelítést alkalmazó klaszterező algoritmusok közé tartozik, és a spektrális klaszterező algoritmusok is alkalmazzák. Kezdetben az algoritmus k darab centroidot, középpontot választ, melyek a klaszterek középpontjai lesznek, majd a pontokat azokba a klaszterekbe sorolja be, amelyiknek a centroidjához a legközelebb vannak. Ezután újra számolja a centroidokat, és ismételten elvégzi a pontok klaszterbe sorolását. Ez addig ismétlődik, amíg van olyan pont, melyet másik klaszterbe sorol be az algoritmus, mint az előző iterációban.

Hierarchikus klaszterezés [13] lehet összevonó vagy felosztó. Az összevonó klaszterezés kezdetben minden pontot külön klaszterbe sorol, majd minden iterációnál a két legközelebbi klasztert összekapcsolja, míg végül egy minden elemet tartalmazó klasztert kapunk. A felosztó hierarchikus klaszterezés ennek az ellentétét valósítja meg. Kezdetben minden adatpontot egy klaszternek tekint és ezeket egyre kisebb klaszterekre osztja, míg végül minden pont külön klaszterbe kerül.

A sűrűségalapú algoritmusok nem aszerint alkotnak klasztereket, hogy egy középponthoz képest milyen távolságra szerepelnek a pontok, hanem sűrűség alapon. Az egyik legismertebb sűrűség alapú klaszterező algoritmus a DBSCAN [10]. Az algoritmusnak szüksége van egy ε-nal jelölt sugárra, amely azt határozza meg, hogy két pont abban az esetben szomszédos, hogyha a közöttük lévő távolság kisebb mint ε. Egy klaszterbe azok az elemek tartoznak melyek szomszédjai egymásnak, és van legalább egy pont közöttük melynek minpts szomszédja van, amely a DBSCAN algoritmus másik paramétere. Előfordulhatnak olyan pontok, melyek nem tartoznak egyetlen klaszterbe se, ezeket outlier-ként kezeli az algoritmus.

3 Spektrális klaszterezési eljárás bemutatása

Ebben a fejezetben a spektrális klaszterezés működését és az algoritmus lépéseinek megértéséhez szükséges elméleti hátteret mutatom be, külön alfejezetekbe foglalva a nagyobb részeket. A spektrális klaszterezés a középpont alapú algoritmusok közé tartozik, melynek lényege, hogy a magas dimenziójú adatainkat először egy alacsonyabb dimenziójú térbe transzformáljuk, ahol ezt követően elvégezzük a klaszterezést.

3.1 Hasonlósági gráf

A spektrális klaszterezés azon alapul, hogy az adatainkból egy irányítatlan gráfot építünk, ezt nevezzük hasonlósági gráfnak. A gráf pontjai az adatpontok, míg a közöttük lévő éleket hasonlósági függvény segítségével tudjuk súlyozni. A szomszédossági gráf lehet súlyozott vagy súlyozatlan.

Több különböző módszer is létezik a hasonlósági gráfok létrehozására, melyek közül az egyik legegyszerűbb módszer az ε-szomszédossági gráf [17]. Ezen módszert alkalmazva, abban az esetben kötünk össze két pontot, hogyha a közöttük lévő távolság kisebb, mint ε. Általában ennek a módszernek a segítségével egy súlyozatlan gráfot kapunk, azonban az éleket súlyozhatjuk a két pont közötti távolság mértékével.

Másik ismert és széleskörben használt módszer a k-legközelebbi szomszéd gráf [17]. A módszer lényege, hogy egy x_i pontot, akkor kötünk össze egy x_j ponttal, hogyha az x_i pont k-legközelebbi szomszédjai között szerepel az x_j pont. Ez a definíció azonban egy irányított gráfot eredményez, mert a szomszédosság nem szimmetrikus. Két módszer létezik, hogy ezt a gráfot szimmetrikussá tegyük. Az első, hogy eltekintünk az élek irányától, azaz összekötünk egy x_i és x_j pontot, hogy x_i k-legközelebbi szomszédjai között szerepel x_j, vagy, hogyha x_j pont k-legközelebbi szomszédjai között szerepel x_i. Ezt a módszert hívják általában a k-legközelebbi szomszéd gráfnak. A második módszer, hogyha mindkét ponttól elvárjuk, hogy a másik k-legközelebbi szomszédjai között szerepeljen. Ezen módszer használatával létrejövő gráfot szokás kölcsönös k-legközelebbi szomszéd gráfnak hívni. Mindkét esetben az éleket súlyozzuk a végpontok közötti hasonlóság értékével.

A harmadik módszer, amivel a hasonlósági gráfot létrehozhatjuk az összefüggő gráf módszer [17]. Ebben az esetben minden pontot minden ponttal összekötünk és a két pont közötti élt súlyozzuk a végpontok hasonlóságának mértékével.

A hasonlósági gráfot egy A mátrix formában szokás tárolni. Súlyozatlan gráf esetén $A_{ij} = 1$, hogyha két pont össze van kötve, valamint $A_{ij} = 0$, hogyha a két pont nincs összekötve. Súlyozott gráf esetén A_{ij} megegyezik a gráfban az x_i és x_j-edik pontok közötti él súlyával, abban az esetben, hogyha az x_i és az x_j pontok össze vannak kötve, míg ellenkező esetben $A_{ij} = 0$.

3.2 Laplacian mátrix

A hasonlósági mátrix segítségével határozható meg a Laplacian mátrix. Több Laplacian mátrix létezik, melyeket különböző esetekben érdemes használni. A teljesség igénye nélkül a legismertebb Laplacian mátrixok a normalizálatlan Laplacian mátrix, valamint a normalizált Laplacian mátrix, melynek két fajtája a szimmetrikus és a balról normalizált mátrix.

A normalizálatlan Laplacian mátrix meghatározásához szükségünk van egy n*n-es D fok mátrixra, amely azt adja meg, hogy az i-edik pontnak mekkora a fokszáma, és a következőképpen számolható: $D_{ii} = \sum_j A_{ij}$ [5]. Ezután a Laplacian mátrix meghatározható a következő képlet segítségével [12]:

$$L = D - A \tag{4}$$

A normalizált Laplacian mátrixokat egymáshoz hasonlóan kell meghatározni, a következő képletek alapján [1][18]:

$$L_{sym} \coloneqq D^{-1/2} L D^{-1/2} = I - D^{-1/2} A D^{-1/2} \tag{5}$$

$$L_{rw} \coloneqq D^{-1} L = I - D^{-1} A \tag{6}$$

Attól függően, hogy melyik Laplacian mátrixot használjuk különböző algoritmusok léteznek a klaszterek meghatározására.

3.3 Spektrál klaszterező algoritmusok

Ebben az alfejezetben két spektrális klaszterező algoritmust mutatok be. Elsőként a normalizálatlan spektrális klaszerezés algoritmusát [9], majd a normalizált spektrális klaszterezés algoritmusát mutatom be Ng, Jordan és Weiss alapján [14]. Mindkét spektrál

klaszterező módszer hasonlóképpen indul, elsőként egy A hasonlósági gráfot határozunk meg, majd ebből kiszámítjuk a Laplacian mátrixot. A különbség a Laplacian mátrix kiszámításában rejlik.

3.3.1 Normalizálatlan spektrális klaszterezés algoritmusa

A normalizálatlan spektrális klaszterezés lépései a következők [9]:

1. *L* mátrix kiszámítása (4) alapján.
2. *L* mátrix első k sajátvektorának kiszámítása: $u_1, \dots u_k$.
3. Legyen $U \in R^{n \times k}$ mátrix, ami az $u_1, \dots, u_k$ vektorokat, mint oszlopokat tartalmazza.
4. Minden $i=1,\dots,n$, legyen $y_i \in R^k$, az *U* mátrix *i*-edik sorának megfelelő vektor.
5. Klaszterezzük a pontokat $(y_i)_{i=1,\dots,n}$ a k-közép algoritmussal $C_1, \dots, C_k$ klaszterekbe.

Kimenet: $A_1, \dots, A_k$ klaszterek, ahol $A_i = \{j \mid y_j \in C_i\}$

3.3.2 Normalizált spektrális klaszterezés algoritmusa

A normalizált spektrális klaszterezés algoritmusának, Ng, Jordan és Weiss alapján lépései a következők [14]:

1. L_{sym} mátrix kiszámítása (5) alapján.
2. L_{sym} mátrix első k sajátvektorának kiszámítása: $u_1, \dots u_k$.
3. Legyen $U \in R^{n \times k}$ mátrix, ami az $u_1, \dots, u_k$ vektorokat, mint oszlopokat tartalmazza.
4. *T* mátrix meghatározása a következő képlet segítségével:

$$t_{ij} = u_{ij} / \left(\sum_k u_{ik}^2 \right)^{1/2} \tag{7}$$

5. Minden $i=1,\dots,n$, legyen $y_i \in R^k$, az *T* mátrix *i*-edik sorának megfelelő vektor.
6. Klaszterezzük a pontokat $(y_i)_{i=1,\dots,n}$ a k-közép algoritmussal $C_1, \dots, C_k$ klaszterekbe.

Kimenet: $A_1, \dots, A_k$ klaszterek, ahol $A_i = \{j \mid y_j \in C_i\}$

4 Hierarchikus adatok klaszterezése

Kétféle megközelítést dolgoztam ki hierarchikus adatok klaszterezésére [2]. Az egyik, hogyha minden ponthalmazt reprezentálunk egy ponttal, amit bizonyos szempontok szerint a legjellemzőbbnek gondolunk, ebben az esetben a ponthalmazok automatikusan egyben maradnak, hiszen maguk a ponthalmazok lesznek a klaszterezés bemenete. A másik megközelítés, hogy egyedi pontonként klaszterezünk, a hasonlósági gráf létrehozásának módosításával érjük el azt, hogy a ponthalmazok a klaszterezést követően is egyben maradjanak.

Elsőként az algoritmusok által elvárt adatstruktúrát mutatom be, majd a két eltérő megközelítést alkalmazó algoritmusokat, mely során a ponthalmazok reprezentációjának segítségével klaszterező módszereket követően, a pontokat külön klaszterező eljárásokat mutatom be.

4.1 Algoritmusok bemenete

Az algoritmusok bemenete egy x_i pontokat tartalmazó adathalmaz, valamint S_i halmazok, melyek leírják, hogy mely pontok tartoznak ugyanabba a ponthalmazba. Formálisan, ha x_i és x_j egy ponthalmazba tartoznak, akkor és csak akkor $x_i, x_j \in S_k$.

Az algoritmus feltételezi, hogy minden pont pontosan egy darab ponthalmazba tartozik, és mindegyik ponthalmaz tartalmaz legalább egy pontot. Ha egy pont egyszerre több ponthalmazba is beletartozna, úgy ezeket a ponthalmazokat egy nagy halmaznak véve már teljesül a feltételezés. Ezt azért tehetjük meg, mert egy ponthalmaz elemei egy klaszterbe kell, hogy kerüljenek, így könnyen belátható, hogyha egy pont több ponthalmazba is tartozik, akkor ezeket a halmazokat az algoritmusnak egy klaszterbe kell sorolnia. Ha van olyan pont, amelyik egyik ponthalmazba se tartozik bele, akkor fel tudunk venni, egy újabb ponthalmazt, amelynek ez az egy pont lesz az eleme, így ismét teljesül a kiindulási feltételünk.

4.2 Ponthalmaz reprezentációs módszerek

Azokban az esetekben, amikor a hierarchikus adatainkat szeretnénk klaszterezni az egyik megoldás, hogy elkészítjük a ponthalmazok reprezentációját, és ezeken a reprezentációkon futtatjuk le a klaszterező algoritmust. Ennek a módszernek az előnye,

hogy a ponthalmazok mindenképpen azonos klaszterbe kerülnek az algoritmus futása után.

4.2.1 Matematikai középérték alapú reprezentáció

A legegyszerűbb módszer, hogyha az egy ponthalmazba tartozó pontok attribútumainak értékeiből kiszámítjuk az általunk választott matematikai középérték segítségével a ponthalmaz azonos attribútumának értékét, így megkapva a ponthalmaz reprezentációját.

4.2.2 Bag of Words

A Bag of Words modell, röviden BoW, az egyik legnépszerűbb reprezentációs megközelítés, mely szövegben lévő szavak eloszlásának reprezentálására szolgál [6][11]. Az alapötlet, hogy kódszavakat hozzunk létre, amelyekkel az adatainkat reprezentáljuk hisztogram módjára [18].

Ezen modellt alapul véve kétféle algoritmust dolgoztam ki a hierarchikus adatstruktúrában található ponthalmazok reprezentálásához. Elsőként a spektrális klaszterezésre alapuló BoW algoritmust mutatom be, majd utána a Gaussian Mixture Model-en alapuló BoW algoritmust.

4.2.2.1 Spektrális klaszterezéssel

A spektrális klaszterezésen alapuló BoW algoritmus a kódszavak létrehozásához spektrális klaszterezést használ. Az adatokat klaszterezés segítségével K db csoportba sorolja, majd klaszterenként választ egy-egy pontot, melyek a kódszavak lesznek. Célszerű erre a célra a klaszterek centroidját vagy a klaszterenkénti átlagos attribútum értékeket választani.

Miután létrehozta a kódszótárt az algoritmus ennek segítségével készíti el a reprezentációját a ponthalmazoknak. Ezen reprezentációk mindegyike egy K hosszú vektor. Minden ponthoz megkeresi a hozzá legközelebbi kódszót, majd ennek a kódszónak megfelelő indexű helyen eggyel megnöveli a reprezentációul szolgáló vektor értékét.

4.2.2.2 Gaussian Mixture Model-lel

A Gaussian Mixture Model-re épülő BoW algoritmus a kódszavak létrehozásához Gaussian Mixture Model-t, röviden GMM, használ [16]. Az algoritmus a pontokat

klaszterezi *K* darab klaszterbe GMM segítségével, így minden ponthoz kapunk egy *K* hosszú vektort, melynek *i*-edik eleme azt mutatja meg, hogy mekkora eséllyel tartozik bele az adott pont az *i*-edik klaszterbe. Ezek a K hosszú vektorok lesznek a pontok kódszótár szerinti reprezentációi.

A pontból a ponthalmazok reprezentációt úgy készítjük el, hogy az egy ponthalmazba tartozó pontok reprezentációit összegezzük.

4.3 Spektrális klaszterezés súlyozott gráffal

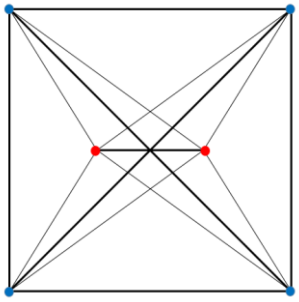
A továbbiakban bemutatásra kerülő algoritmusoknál már nem készítjük el a ponthalmazok reprezentációját, hanem a spektrális klaszterezéshez szükséges hasonlósági gráfot építjük fel oly módon, hogy az azonos ponthalmazba tartozó pontokat egy klaszterbe sorolja az algoritmus.

4.3.1 Teljes gráfot építő algoritmus

A teljes gráfot építő algoritmusnál úgy építjük fel a hasonlósági gráfot, hogy minden pontot összekötünk minden másik ponttal, és az éleket súlyozzuk attól függően, hogy a két pont egy ponthalmazba tartozik-e.

Abban az esetben, hogyha a két pont nem tartozik egy ponthalmazba, akkor a közöttük lévő él súlya a hasonlósági függvényünk által meghatározott érték lesz, míg hogyha egy ponthalmazba tartoznak, akkor az él súlyának *n*-t állítunk be.

Az 3. ábra szemlélteti két ponthalmaz között felépülő teljes hasonlósági gráfot, ahol a vastagabb él jelenti az erősebb kapcsolatot.



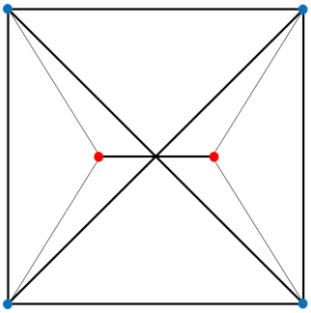
3. ábra Két ponthalmaz között felépülő hasonlósági gráf a teljes gráf algoritmussal

4.3.2 Hiányos gráfot építő algoritmus

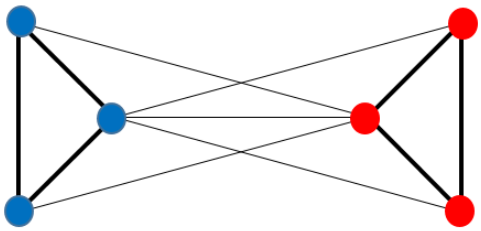
A hiányos gráfot építő algoritmusnál a ponthalmazokon belül minden pontot minden ponttal összekötünk, azonban más ponthalmazoknál csak az adott ponthalmazból a hozzá legközelebbi ponttal kötjük össze, hogyha több ilyen pont is van, akkor tetszőlegesen választunk ezek közül egy pontot.

Ebben az esetben is súlyozzuk a gráf éleit, ponthalmazon kívül a hasonlósági függvény által számított értékkel, míg ponthalmazon belül az élek súlya *n*-nel lesz egyenlő.

A 4. és 5. ábra szemlélteti két ponthalmaz között felépülő hiányos hasonlósági gráfot, ahol a vastagabb él jelenti az erősebb hasonlóságot.



4. ábra Két ponthalmaz között felépülő hasonlósági gráf a hiányos gráf algoritmussal



5. ábra Két ponthalmaz között felépülő hasonlósági gráf a hiányos gráf algoritmussal

Formálisan leírva a (8) és (9) képletek segítségével határozható meg, hogy melyik pontot melyik ponttal kötjük össze:

$$npp(x_i, x_j) = \left\{ \begin{array}{c|c} Igaz & sim(x_i, x_j) \geq sim(x_i, x_l), \forall x_l \\ & (ahol\ x_j, x_l \in S_t, x_j \neq x_l) \\ Hamis & egyébként \end{array} \right\} \quad (8)$$

$$A_{ij} = \left\{ \begin{array}{c|c} n & x_i \in S_t, x_j \in S_t \\ sim(x_i, x_j) & x_i \in S_t\ és\ x_j \notin S_t\ és\ npp(x_i, x_j) \\ 0 & egyébként \end{array} \right\} \quad (9)$$

ahol *n* az adatok száma, x_i az *i*-edik adatpont, *sim* az általunk használt hasonlósági függvény és S_t halmaz a *t*-edik ponthalmaz elemeit tartalmazza.

5 Feltételrendszer súlyozott gráf élsúlyainak meghatározására

A pontokat külön klaszterező algoritmusoknál előfordulhat, hogy bár erősebbek a ponthalmazon belüli élek, mint a két ponthalmaz között futók, ám mégis szétszakadnak a ponthalmazok, így nem teljesül, hogy a ponthalmazok egy klaszterbe kerüljenek.

Ebben a fejezetben azt vizsgálom meg, hogy mekkora ponthalmazon belüli hasonlósági értékkel érhető el, hogy a ponthalmazok egy klaszterbe kerüljenek, és ne szakadjanak szét. Ehhez először a gráfvágások nézőpontjából mutatom be a klaszterezést, majd a feltételrendszer kidolgozását mutatom be.

5.1 Klaszterezés a gráfvágások nézőpontjából

A klaszterezés lényege, hogy a pontokat csoportosítsuk úgy, hogy a hasonló pontok kerüljenek egy csoportba. A hasonlósági gráfot vizsgálva ezt úgy is meg tudjuk fogalmazni, hogy a csoportok között futó élek súlya minimális legyen, míg az egy csoporton belül futó élek súlyát maximalizáljuk. Ez a megközelítés vezet el minket a vágásokig.

5.1.1 Cut

Egy *A* szomszédossági mátrixot tekintve a legegyszerűbb megközelítés a gráf partícionálására, hogyha megoldjuk a minimális vágás problémát. Megadott k számú halmazra a minimális vágás probléma megoldása egy olyan $C_1,\dots,C_k$ partíció, amely minimalizálja a következőt [17]:

$$cut(C_1,\dots,C_k) = \frac{1}{2}\sum_{i=1}^{k} W(C_i,\overline{C_i})\,. \tag{10}$$

Ez a módszer azonban sok esetben nem az elvárt eredményt adja, mert egy-egy kiugró értéket külön klaszterbe sorolva kisebb érték érhető el, mint amit a tényleges klaszterekre bontás eredményez.

Ennek a problémának a megoldására alkalmazhatóak a RatioCut, valamint az Ncut vágások, melyek az optimalizációnál számításba veszik a klaszterek méretét is, így elősegítve az egyforma méretű klaszterek kialakulását.

5.1.2 RatioCut

A RatioCut a klaszterek méretét a benne található pontok számával határozza meg, így próbálja elkerülni az alacsony elemszámú klaszterek létrejöttét. A RatioCut a következő képletnek keresi a minimum értékét [17]:

$$RatioCut(C_1, \dots, C_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(C_i,\ \overline{C_i})}{|C_i|} = \sum_{i=1}^{k}\frac{cut(C_i,\ \overline{C_i})}{|C_i|} \tag{11}$$

5.1.3 Ncut

A normalized cut, röviden Ncut, a klaszterek méretét a benne található élek súlyának összegeként méri, és definíciója a következő [17]:

$$vol(C) = \sum_{i \in C}\sum_{j=1}^{n} A_{ij} \tag{12}$$

$$Ncut(C_1, \dots, C_k) = \frac{1}{2}\sum_{i=1}^{k}\frac{W(C_i,\ \overline{C_i})}{vol(C_i)} = \sum_{i=1}^{k}\frac{cut(C_i,\ \overline{C_i})}{vol(C_i)} \tag{13}$$

Láthatjuk, hogy az Ncut a RatioCut-tal ellentétben nem a klaszterek elemszámára optimalizál, hanem egyformán erős éleket tartalmazó klaszterek létrejöttét segíti.

5.2 Feltételrendszer elméleti kidolgozása klaszterezéshez

Az alább olvasható feltételrendszert használva biztosítható, hogy a pontokat külön klaszterező algoritmusoknál az egy ponthalmazba tartozó pontok ugyanabba a klaszterbe kerüljenek.

Elsőnek a használt jelöléseket foglalom össze, majd utána a levezetés következik.

Használt jelölések:

- n: adatok száma
- k: klaszterek száma
- C_i : i-edik klaszter
- $|C_i|$: i-edik klaszter elemszáma
- S_i : i-edik ponthalmaz
- A: hasonlósági mátrix

- A_{ij}: A hasonlósági mátrix i-edik sorának j-edik eleme
- Z: a keresett érték, az egy ponthalmazba tartozó elemek közötti hasonlóság értéke

A normalizált spektrális klaszterezés a normalizált vágásra alapszik [4], ennek a függvénynek keresi a minimum értékét:

$$Ncut(C_1, \dots, C_k) = \sum_{i=1}^{k} \frac{cut(C_i,\ \bar{C_i})}{vol(C_i)} = \frac{1}{2} \sum_{i=1}^{k} \frac{\sum_{j \in C_i} \sum_{l \in \bar{C_i}} A_{jl}}{\sum_{j \in C_i} \sum_{l \in \bar{C_i}} A_{jl} + \sum_{j \in C_i} \sum_{l \in C_i} A_{jl}} \quad (14)$$

A fent meghatározott definíciót fejtem ki alább két esetben, a korábban ismertetett hasonlósági gráf építési módszerrel kapott hasonlósági gráfot alapul véve. A keresett Z értéket Ncut becslésével próbálom meghatározni két esetet vizsgálva:

1. Ha ∀ i,j-re igaz, hogyha i ∈ S_t és j ∈ S_t ⇒ i, j ∈ C_m

Azaz abban az esetben, hogyha minden ponthalmaz egy klaszterbe kerül.

$InterCluster_1$ legyen a klaszterből kifelé menő élek összege, abban az esetben, hogyha minden a klaszteren belül szereplő ponthalmaz összes pontja ebbe a klaszterbe tartozik, míg $WithinCluster_1$ legyen ugyanebben az esetben a klaszteren belüli hasonlóságok összege.

$$InterCluster_1(C_i) = \sum_{j \in C_i} \sum_{l \in \bar{C_i}} A_{jl} \quad (15)$$

$$WithinCluster_1(C_i) = \sum_{j \mid S_j \in C_i} \left(\sum_{l \in S_j} \sum_{m \in S_j} Z + \sum_{l \in S_j} \sum_{m \in C_i \setminus S_j} A_{lm} \right. \quad (16)$$

(15) és (16) jelöléseket felhasználva $Ncut_1$-et a következőképpen írhatjuk le:

$$Ncut_1(C_1, \dots, C_k) = \frac{1}{2} \sum_{i=1}^{k} \frac{InterCluster_1(C_i)}{InterCluster_1(C_i) + WithinCluster_1(C_i)} \quad (17)$$

2. Hogyha ∃ pontosan egy olyan S_t, és ∃ pontosan egy olyan i ∈ S_t, melyre ∀ j≠i ∈ S_t igaz, hogy j ∈ C_m, míg i ∈ C_o.

Legyen m := k-1, o := k, és u := i.

Tehát létezik pontosan egy olyan ponthalmaz, melynek létezik pontosan egy olyan pontja, mely más klaszterbe kerül, mint a ponthalmaz többi pontja.

InterCluster$_{2.1}$ jelölje a klaszterből kifelé menő élek összegét, abban az esetben, hogyha a klaszteren belül található pontosan egy olyan ponthalmaz melynek van egy pontja ami nem ebbe a klaszterbe tartozik, míg WithinCluster$_{2.1}$ legyen ugyanebben az esetben a klaszteren belüli hasonlóságok összege.

$$InterCluster_{2.1}(C_i, S_t, u) = \sum_{j \in C_{k-1}} \sum_{l \in \overline{C_{k-1} \cup S_t}} A_{jl} + \sum_{j \in S_t \setminus u} \mathrm{Z} + \sum_{j \in \overline{C_{k-1} \cup S_t \setminus u}} A_{uj} \quad (18)$$

$$WithinCluster_{2.1}(C_i, S_t, u) = \sum_{j \neq t \mid S_j \in C_i} \left[\sum_{l \in S_j} \sum_{m \in S_j} \mathrm{Z} + \sum_{l \in S_j} \sum_{m \in C_i \setminus S_j} A_{lm} \right] + \sum_{l \in C_i} A_{ul} + Z \quad (19)$$

InterCluster$_{2.2}$ jelölje a klaszterből kifelé menő élek összegét, abban az esetben, hogyha a klaszteren belül található pontosan egy olyan pont, amelyhez tartozó ponthalmaznak semelyik másik pontja nem eleme ennek a klaszternek, míg WithinCluster$_{2.2}$ legyen ugyanebben az esetben a klaszteren belüli hasonlóságok összege.

$$InterCluster_{2.2}(C_i, S_t, u) = \sum_{j \in C_i} \sum_{l \in \overline{C_i \cup u}} A_{jl} + \sum_{j \in S_t \setminus u} \mathrm{Z} + \sum_{j \in \overline{S_t \setminus u}} A_{uj} \quad (20)$$

$$WithinCluster_{2.2}(C_i, S_t, u) = \sum_{j \neq t \mid S_j \in C_k} \left[\sum_{l \in S_j} \sum_{m \in S_j} Z + \sum_{l \in S_j} \sum_{m \in C_k \setminus S_j} A_{lm} \right] + \sum_{j \in C_k \setminus u} A_{uj} \quad (21)$$

(15), (16), (18), (19), (20), és (21) jelöléseket felhasználna Ncut$_2$ a következőképpen írható le:

$$\begin{aligned} Ncut_2(C_1, \dots, C_k) = & \frac{1}{2} \frac{InterCluster_1(C_i)}{InterCluster_1(C_i) + WithinCluster_1(C_i)} + \\ & + \frac{1}{2} \frac{InterCluster_{2.1}(C_{k-1}, S_t, u)}{InterCluster_{2.1}(C_{k-1}, S_t, u) + WithinCluster_{2.1}(C_{k-1}, S_t, u)} + \\ & + \frac{1}{2} \frac{InterCluster_{2.2}(C_i, S_t, u)}{InterCluster_{2.2}(C_i, S_t, u) + WithinCluster_{2.2}(C_i, S_t, u)} \end{aligned} \quad (22)$$

Olyan Z értéket keresünk, melyre minden esetben $Ncut_1(C_1, \dots, C_k) \leq Ncut_2(C_1, \dots, C_k)$, ezért Ncut$_1$-et felül-, míg Ncut$_2$-őt alábecsülöm. Mivel az ½-es szorzó mindkét oldalon szerepel, ezért $Ncut_1(C_1, \dots, C_k)$-t és $Ncut_2(C_1, \dots, C_k)$-t is megszorzom 2-vel.

NCut$_1$ felülbecslésénél InterCluster$_1$-et felül-, míg WithinCluster$_1$-et alábecslem. Ehhez InterCluster1-en belül a hasonlósági értékeket mindenhol 1-nek veszem és a

ponthalmazok számát maximalizálom, míg WithinCluster1-en belül a hasonlósági értékeket 0-nak veszem és a ponthalmazok számát minimalizálom.

$$InterCluster_1(C_i) \leq n * n * 1 = n^2 \tag{23}$$

$$WithinCluster_1(C_i) \geq \sum_{j \mid S_j \in C_i} \left(1^2 * \mathrm{Z} + |S_j|(|C_i| - |S_j|) * 0\right) = n * Z \tag{24}$$

$$Ncut_1(C_1, \dots, C_k) \leq \sum_{i=1}^{k} \frac{n^2}{n^2 + n * \mathrm{Z}} = \frac{k * n^2}{n^2 + \mathrm{Z}} = \frac{k * n}{n + \mathrm{Z}} \tag{25}$$

NCut_2 alábecslésénél az InterCluster értékeket alul-, míg WithinCluster értékeit felülbecslem. Ehhez InterCluster-en belül a hasonlósági értékeket mindenhol 0-nak veszem és a ponthalmazok számát minimalizálom, míg WithinCluster-en belül a hasonlósági értékeket 1-nak veszem és a ponthalmazok számát maximalizálom.

$$InterCluster_1(C_i) \geq \sum_{j \in C_i} \sum_{l \in \overline{C_i}} 0 = 0 \tag{26}$$

$$InterCluster_{2.1}(C_i, S_t, u) \geq \sum_{j \in C_{k-1}} \sum_{l \in \overline{C_{k-1} \cup S_t}} 0 + 1 * Z + \sum_{j \in \overline{C_{k-1} \cup S_t \setminus u}} 0 = Z \tag{27}$$

$$InterCluster_{2.2}(C_i, S_t, u) \geq \sum_{j \in C_i} \sum_{l \in \overline{C_i \cup u}} 0 + \sum_{j \in S_t \setminus u} \mathrm{Z} + \sum_{j \in \overline{S_t \setminus u}} 0 = Z \tag{28}$$

$$\begin{aligned} WithinCluster_{2.1}(C_i, S_t, u) &\leq \sum_{j \neq t \mid S_j \in C_{k-1}} [n^2\mathrm{Z} + n * n * 1] + (n-1) * 1 + Z \leq \\ &\leq n * n^2\mathrm{Z} + n * n * n * 1 + (n-1) + \mathrm{Z} \leq n^3 * Z + n^3 + n + Z \end{aligned} \tag{29}$$

$$\begin{aligned} WithinCluster_{2.2}(C_i, S_t, u) &\leq \sum_{j \neq t \mid S_j \in C_k} [n^2\mathrm{Z} + n * n * 1] + (n-1) * 1 \leq \\ &\leq n * n^2\mathrm{Z} + n * n * n * 1 + (n-1) * 1 \leq n^3 * \mathrm{Z} + n^3 + n \end{aligned} \tag{30}$$

(26), (27), (28), (29) és (30) alapján NCut_2-re elmondhatjuk:

$$\begin{aligned} Ncut_2(C_1, \dots, C_k) &\geq 0 + \frac{\mathrm{Z}}{Z + n^3 * \mathrm{Z} + n^3 + n + \mathrm{Z}} + \frac{\mathrm{Z}}{\mathrm{Z} + n^3 * \mathrm{Z} + n^3 + n} \geq \\ &\geq \frac{2 * \mathrm{Z}}{Z + n^3 * \mathrm{Z} + n^3 + n + \mathrm{Z}} = \frac{2 * \mathrm{Z}}{(n^3 + 2) * \mathrm{Z} + n^3 + n} \end{aligned} \tag{31}$$

Olyan Z értéket keresek, melyeknél minden esetben igaz, hogy Ncut_1 kisebb, mint Ncut_2, tehát:

$$NCut_1(C_1, \dots, C_k) < NCut_2(C_1, \dots, C_k)$$

$$\frac{k * n}{n + \mathrm{Z}} < \frac{2 * \mathrm{Z}}{(n^3 + 2) * \mathrm{Z} + n^3 + n}$$

$$0 < 2\mathrm{Z}^2 + (2n - kn^4 - 2kn)\mathrm{Z} - (kn^4 + kn^2)$$

$$\mathrm{Z} < \frac{kn^4 + 2kn - 2n - \sqrt{k^2n^8 + 4k^2n^5 + 4k^2n^2 + 8kn^4 - 4kn^5 + 4n^2}}{4}$$

vagy

$$\mathrm{Z} > \frac{kn^4 + 2kn - 2n + \sqrt{k^2n^8 + 4k^2n^5 + 4k^2n^2 + 8kn^4 - 4kn^5 + 4n^2}}{4}$$

Így a ponthalmazok közötti hasonlóságnak nagyobbnak kell lennie, mint:

$$\frac{kn^4 + 2kn - 2n + \sqrt{k^2n^8 + 4k^2n^5 + 4k^2n^2 + 8kn^4 - 4kn^5 + 4n^2}}{4} \quad (32)$$

Fontos megjegyezni, hogy ez csak abban az esetben igaz, hogyha az általunk használt hasonlósági függvény értékkészletének maximuma 1 és minimuma 0.

6 Algoritmusok kiértékelése

Ahhoz hogy az algoritmusokat ellenőrizni és összehasonlítani tudjam több adathalmazon is le kellett őket futtatnom, amelyhez két adathalmazt használtam. Először ezekről az adathalmazokról nyújtok egy áttekintést, majd pedig a kiértékeléshez használt módszereket mutatom be. Legvégül pedig az eredmények ismertetése és az algoritmusok összehasonlítása következik.

6.1 Használt adathalmazok

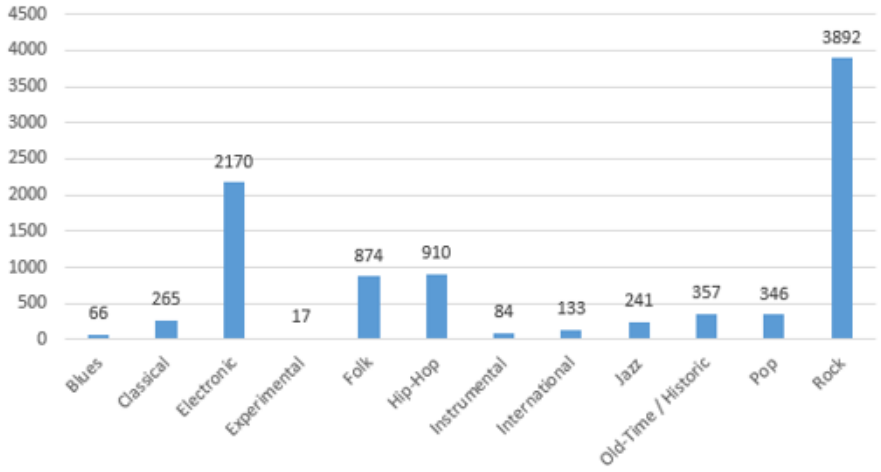
A teszteléshez és összehasonlításhoz két olyan adathalmazon is lefuttattam az algoritmusokat, melyek hierachikus adatokat tároltak. Ezen adathalmazok közül az első a Free Music Analysis (FMA) audio adathalmaz volt [3], amely zeneszámok adatait tartalmazza, míg a második adathalmaz a LifeCLEF's PlantCLEF 2015 verseny adathalmaza volt, mely növényekről készült megfigyelések és képek adatait tartalmazza.

Először a zenei adathalmazt mutatom be, majd ezt követően a növényi adathalmazt ismertetem.

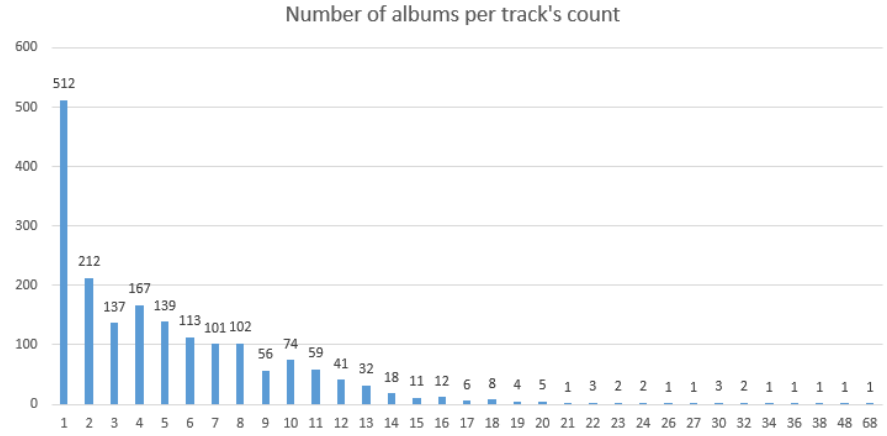
6.1.1 Zenei adathalmaz

A Free Music Analysis (FMA) adathalmaz [3] 106.574 zeneszám adatait tartalmazza 16.341 előadótól és 14.854 albumhoz tartozik, melyek 161 hierarchikusan szervezett műfajhoz tartoznak. A teszteléshez két kisebb adathalmazt hoztam létre ezekből az adatokból. Az FMA adathalmaz 518 db számított audiójellemzővel írja le a zeneszámokat, melyeket az algoritmusok bemeneteként használtunk.

Az első műfaj-adathalmazhoz (FMA-Műfaj) 9.355 zeneszámot választottam ki 1.829 albumból, melyek a műfajhierarchián belüli 12 legmagasabb kategóriához tartoznak. Az egy albumba tartozó zeneszámok száma nagy szórást mutat, 512 albumba csak egy darab zeneszám tartozik, míg van olyan album is, melybe 68 zeneszám tartozik bele, ezt szemlélteti az 6. ábra. Műfajok szerint az látható, hogy a zeneszámok közel kétharmada az elektronikus vagy a rock műfajhoz tartozik, ez látható a 7. ábrán. A hierarchikus struktúrát úgy kapjuk meg ebben az esetben, hogy a ponthalmazok az egyes albumok, melyek magukba foglalják a zeneszámokat, amik a pontoknak felelnek meg.

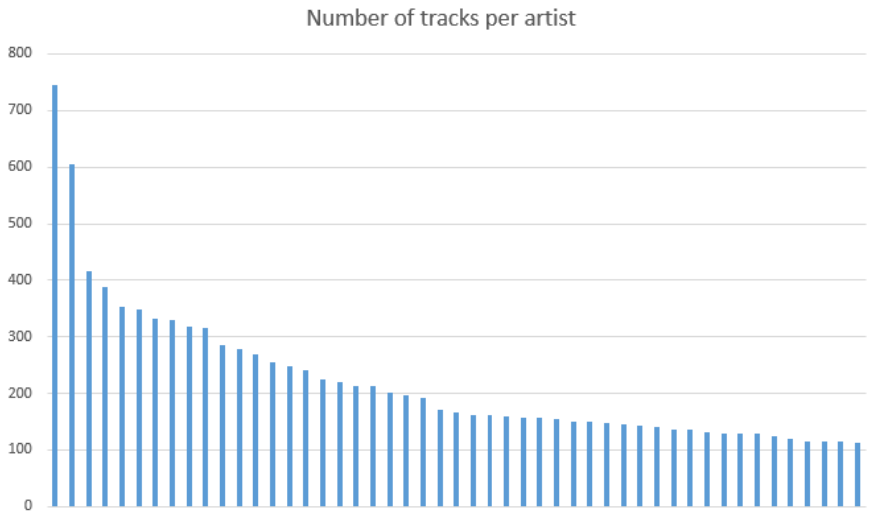


6. ábra Az első zenei adathalmazba tartozó zeneszámok eloszlása műfaj szerint



7. ábra Egy albumba tartozó zeneszámok száma az első zenei adathalmazban

A második előadó-adathalmazhoz (FMA-Előadó) azt az 50 előadót választottam ki, akiknek a legtöbb zeneszáma szerepel az adathalmazban. Így ez az adathalmaz 10.848 zeneszámból áll, melyek 1.171 albumhoz tartoznak. Az egy zeneszerzőhöz tartozó zeneszámok számát mutatja a 8. ábra zeneszám szerint csökkenő sorrendben. Látható, hogy a legtevékenyebb előadó több mint 700 zeneszámmal szerepel az adatbázisban, míg az első 50 között mindenkinek több mint 100 zeneszáma van. Ebben az esetben pontoknak ugyanúgy, mint a FMA-Műfaj adathalmaznál, a zeneszámok felelnek meg, azonban ezeket nem az albumok, hanem az előadójuk szerint rendezzük hierarchikus struktúrába.



8. ábra Egy zeneszerzőhöz tartozó zeneszámok száma

Az első adathalmazt a 12 műfaj miatt 12, míg a másodikat az 50 előadó miatt 50 klaszterbe csoportosítottam. Célom az volt a két esetben, hogy az FMA-Műfaj adathalmazon műfajoknak megfelelő klaszterek alakuljanak ki, míg az FMA-Előadó adathalmazon zeneszerző szerinti csoportokat kapjunk.

6.1.2 Növényi adathalmaz

A második adathalmaz a 2015-ös PlantCLEF verseny képeit tartalmazza [7]. Ez az adathalmaz 27.907 megfigyelésen készült 91.759 növényről készült képet tartalmaz, melyekhez meta-adatok is tartoznak úgy, mint faj, hely, szavazat. Egy „megfigyelés”-en ugyanarról a növényről készítünk képeket, melyeken az egyed különböző részei szerepelnek, például levél, törzs, termés stb.

A szavazat a felhasználóktól kapott értékelések átlaga egy 1-től 5-ig terjedő skálán. Az adathalmazt leszűkítettem, hogy csak a 4 és 5 értékű szavazattal rendelkező képek szerepeljenek benne, ezáltal 26.093 képet használtam fel 9.989 megfigyelésből, mely 988 fajról készült.

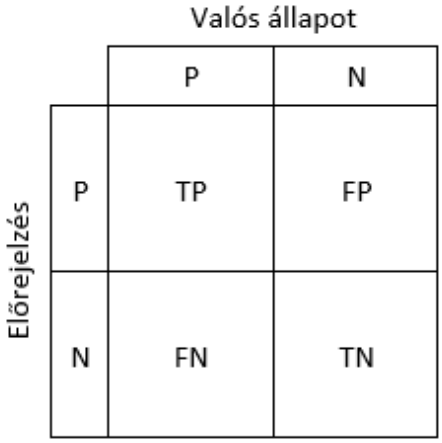
Ebben az adathalmazban a hierarchikus struktúrát a megfigyelésekkel érjük el, mely összefoglalja az azonos növényről készült képeket (azaz a különböző növényrészeket tartalmazó „megfigyelés” felel meg egy ponthalmaznak). A célom az volt, hogy az azonos fajú növényekről készült képek kerüljenek egy klaszterbe.

Az PlanCLEF adathalmaz nem tartalmaz reprezentációt a képekhez, ezért ezeket saját magamnak készítettem el. A klaszterezéshez használt adatok létrehozásához 128

dimenziós SIFT (Scale Invariant Feature Transform [8]) leírókat számítottam ki, melyeket ezután Fisher-vektorokba kódoltam [15]. Ez az eljárás nagyon nagy dimenziós 65.536 hosszú reprezentációs vektorokat eredményezett.

6.2 Kiértékelési módszerek

A klaszterek kiértékeléséhez két metrikát használtam, a True Positive Rate-t és az F1 mértéket. Mindkét mérték meghatározásához szükség van a True Positive (TP), False Positive (FP), True Negativ (TN), valamint a False Negative (FN) értékekre, melyeket a konfúziós mátrix kiszámításával kaphatunk meg, ahogy az a 9. ábrán látható.



9. ábra Konfúziós mátrix

A TP azokat a kapcsolatokat jelenti, melyek azonos klaszterbe kerültek, és azonos klaszterbe is kellett kerülniük. Ezzel szemben a FP azokat jelöli, melyek annak ellenére kerültek azonos klaszterbe, hogy nem ez lett volna az elvárt működés. TN azokat jelöli, amelyek nem kerültek egy klaszterbe, és nem is kellett nekik, míg a FN azokat, melyek nem kerültek egy klaszterbe, pedig azonos klaszterbe kellett volna kerülniük.

A True Positive Rate (TPR) másnéven érzékenység a valósan pozitívnak ítéltek és az összes pozitív érték hányadosát jelenti. Tehát azoknak a kapcsolatoknak a számát, melyek helyesen két azonos klaszterbe tartozó pont között futnak, osztjuk el azoknak a kapcsolatoknak a számával, melyek két olyan pont között futnak, melyeknek egy klaszterbe kellett volna kerülniük. A konfúziós mátrix elemei alapján ez a következőképpen határozható meg:

$$TPR = \frac{TP}{TP + FN}$$

Az F_1 mérték egyaránt számításba veszi az előbb megemlített TPR-t, valamint a precíziót (másnéven Positive Predictive Value), és ezeknek a harmonikus közepét jelenti, ahogy az alábbi képlet is mutatja:

$$F_1 = \frac{2TP}{2TP + FP + FN}$$

Azért döntöttem az F_1 mérték kiszámítása mellett, mert önmagában a TPR vagy a precízió félrevezető lehet, mivel ezek a konfúziós mátrix különböző elemeit mérik. Emellett ezek gyakran ellentétesek, ezért az egyik növelése gyakran a másik csökkenését eredményezi, míg az F_1 mérték általánosságban mutatja meg a klaszterezés hatékonyságát. TPR-t azért vontam be a kiértékelésbe, mert ez méri az arányát a helyesen egy klaszterbe csoportosított pontoknak.

6.3 Algoritmusok összehasonlítása

A teszteléshez a korábban említett 3 adathalmazt használtam fel, melyek mindegyike hierarchikus adatokat tartalmaz, ezáltal felhasználhatók az algoritmusok tesztelésére. A következő 5 algoritmust teszteltem és hasonlítottam össze:

- spektrális klaszterezés, melynél a ponthalmazreprezentációkat a pontok átlagával határozzuk meg (Átlag)
- Bag of Words reprezentáció, mely spektrális klaszterezést használ a kódszavak meghatározására (BoW-SC)
- Bag of Words reprezentáció, mely Gaussian Mixture Model-t használ a kódszavak meghatározására (BoW-GMM)
- Teljes gráfot használó spektrális klaszterezés, ponthalmazon belül *n* értékkel (TGA-N)
- Hiányos súlyozott gráfot használó spektrális klaszterezés, ponthalmazon belül *n* értékkel (HGA-N)

A Bag of Words megközelítést alkalmazó algoritmusokhoz a különböző adathalmazokra különböző kódszóhosszakat állítottam be. Mindkét FMA adathalmazhoz 200 hosszú kódszótárat használtam, míg a PlantCLEF adathalmazon 256 hosszút.

Az 1. táblázat tartalmazza az algoritmusok által elért TPR értékeket az adathalmazokra, míg a 2. táblázat tartalmazza a F_1 mértéket.

	FMA-Műfaj	FMA-Előadó	PlantCLEF
Átlag	0,132	0,147	0,008
BoW-SC	0,198	0,242	0,107
BoW-GMM	0,312	0,194	0,114
TGA-N	0,908	0,369	0,371
HGA-N	**0,927**	**0,494**	**0,403**

1. táblázat Az algoritmusok TPR értéke a különböző adathalmazokra

	FMA-Műfaj	FMA-Előadó	PlantCLEF
Átlag	0,181	0,162	0,012
BoW-SC	0,223	0,178	0,085
BoW-GMM	0,301	**0,191**	0,110
TGA-N	**0,375**	0,090	0,077
HGA-N	0,374	0,081	**0,146**

2. táblázat Az algoritmusok F_1 mértéke a különböző adathalmazokra

Az 1. táblázatot tekintve a HGA-N és TGA-N algoritmusok eredményezték a legmagasabb értékeket - különösen magas 0,9 feletti eredményekkel az FMA-Műfaj adathalmazon - míg a legrosszabb értékeket mindhárom esetben az Átlag érte el.

A 2. táblázaton azt látjuk, hogy a HGA-N csak a PlanCLEF adathalmazon eredményezte a legmagasabb értéket, míg az FMA-Műfaj adathalmazon a TGA-N-nel közel azonos eredményt értek el, addig az FMA-Előadó adathalmazon már a BoW-GMM eredményezte a legmagasabb F_1 mértéket.

A két táblázatot egybevetve elmondhatjuk, hogy a HGA-N és a TGA-N algoritmusok volt ahol egész magas (0,9 feletti) TPR értékeket vettek fel, azonban az F_1 értékek maximuma ennél jóval kevesebb 0,375 volt. Ez azzal magyarázható, hogy az algoritmusok kevés FP párt eredményeztek, azonban a FN párok száma sokkal magasabb volt. Összességében az is elmondható, hogy a PlantCLEF adathalmazon az algoritmusok kisebb pontokat értek el, mint az FMA adathalmazokon.

A gráfos módszereknél *n* érték nem volt elég nagy ahhoz, hogy minden esetben egyben tartsa a ponthalmazokat. Több esetben is előfordult, hogy ezek szétszakadtak, és egy ponthalmazba tartozó pontok több klaszterbe kerültek. Emiatt módosítottam a ponthalmazokon belüli súlyokat a (32)-ben meghatározott *Z* értékkel:

- Teljes gráfot használó spektrális klaszterezés, ponthalmazon belül *Z* értékkel (TGA-Z)
- Hiányos súlyozott gráfot használó spektrális klaszterezés, ponthalmazon belül *Z* értékkel (HGA-Z)

A 3. táblázat mutatja, hogy a megvizsgált két adathalmazon a ponthalmazon belüli erősebb élekkel sikerült elérni, hogy ne szakadjanak szét a ponthalmazok.

		Szétszakadt ponthalmazok száma			
	Összes ponthalmaz	TGA-N	HGA-N	TGA-Z	HGA-Z
FMA-Műfaj	1829	2	**0**	**0**	**0**
FMA-Előadó	1171	43	34	**0**	**0**

3. táblázat A pontokat külön klaszterező algoritmusok esetében a szétszakadt ponthalmazok száma

7 Összefoglalás

Sikerült kidolgoznom több spektrális klaszterezésen alapuló algoritmust, melyek hierarchikus adatokon futtathatók. Az egyik algoritmuscsoport a ponthalmaz reprezentációk adatait a ponthalmazban található pontokkal határozza meg, míg a másik megközelítés a hasonlósági gráf éleinek súlyozásával a pontokon futtatja le a spektrális klaszterező algoritmust.

Mindkét módszer segítségével elérhető, hogy a klaszterezés megőrizze a hierarchikus struktúrát, azaz minden ponthalmaz egy klaszterbe kerüljön. A reprezentációs módszerek ezt a kritériumot definíció szerint teljesítik, azonban a súlyozott gráfot alkalmazó módszereknél is teljesíthető ez a kritérium az 5. fejezetben kidolgozott feltételrendszer segítéségével.

Az algoritmusokat három adathalmazon is lefuttattam. A súlyozott gráf megközelítéseket alkalmazó algoritmusok közül azok, melyeknél a ponthalmazon belül *n* értéket állítottunk be a hasonlósági mátrixnál nem tartották egyben a ponthalmazokat, azonban *z* értékkel már megtartották a hierarchikus struktúrát.

Az eredményeket tekintve a TGA-N és HGA-N algoritmusok érték el a legmagasabb pontszámokat, azonban ezek nem tartották meg a ponthalmaz hierarchikus struktúráját. A reprezentációs módszerek közül a legjobban a BoW-GMM algoritmus teljesített, amely egy esetben a legjobb eredményt adó algoritmusnak is bizonyult, míg a BoW-SC egy kicsit gyengébb eredményeket ért el, az Átlag pedig szinte minden esetben a legrosszabb TPR és F_1 pontszámot el az algoritmusok közül.

Köszönetnyilvánítás

A TDK dolgozatban ismertetett eredmények a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Kar Balatonfüredi Hallgatói Kutatócsoport szakmai közössége keretében jöttek létre a régió gazdasági fejlődésének elősegítése érdekében. Az eredmények létrehozása során figyelembe vettük a balatonfüredi központú Rendszertudományi Innovációs Klaszter által megfogalmazott célkitűzéseket, valamint a párhuzamosan megvalósuló EFOP 4.2.1-16-2017-00021 pályázat támogatásával elnyert „BME Balatonfüredi Tudáscentrum" térségfejlesztési terveit.

A kutatás az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósult meg (EFOP-3.6.2-16-2017-00013, Innovatív Informatikai és Infokommunikációs Megoldásokat Megalapozó Tematikus Kutatási Együttműködések).

Irodalomjegyzék

[1] Chung, F. R., & Graham, F. C. (1997). Spectral graph theory (No. 92). American Mathematical Soc.

[2] Dávid Papp, Gábor Szűcs, Zsolt Knoll: Machine preparation for human labelling of hierarchical train sets by spectral clustering, 10th IEEE International, Conference on Cognitive Infocommunications, Naples, Italy, 2019, Oct 23-25., pp. 157-162

[3] Defferrard, M., Benzi, K., Vandergheynst, P., & Bresson, X. (2016). Fma: A dataset for music analysis. arXiv preprint arXiv:1612.01840.

[4] Hofmeyr, David. (2018). Connecting Spectral Clustering to Maximum Margins and Level Sets.

[5] HU, P. (2012). Spectral Clustering Survey.

[6] Joachims, T. (1998, April). Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer, Berlin, Heidelberg.

[7] Joly, A., Müller, H., Goeau, H., Glotin, H, Spampinato, C., Rauber, A., Bonnet, P., Vellinga, W. P., Fisher, B.: LifeCLEF 2015: multimedia life species identification challenges, Proceedings of CLEF 2015 (2015).

[8] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110.

[9] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. (1996). A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96), Evangelos Simoudis, Jiawei Han, and Usama Fayyad (Eds.). AAAI Press 226-231.

[11] McCallum, A., & Nigam, K. (1998, July). A comparison of event models for naive bayes text classification. In AAAI-98 workshop on learning for text categorization (Vol. 752, No. 1, pp. 41-48).

[12] Mohar, B. (1997). Some applications of Laplace eigenvalues of graphs. In Graph symmetry (pp. 225-275). Springer, Dordrecht.

[13] Murtagh, Fionn & Contreras, Pedro. (2011). Methods of Hierarchical Clustering. Computing Research Repository - CORR. 10.1007/978-3-642-04898-2_288.

[14] Ng, A.Y. & Jordan, Michael & Weiss, Y. (2001). On Spectral Clustering: Analysis and an Algorithm. Adv. Neural Inf. Process. Syst.. 2.

[15] Perronnin, F., & Dance, C. (2007, June). Fisher kernels on visual vocabularies for image categorization. In 2007 IEEE conference on computer vision and pattern recognition (pp. 1-8). IEEE.

[16] Reynolds, D. (2015). Gaussian mixture models. Encyclopedia of biometrics, 827-832.

[17] von Luxburg, U. Stat Comput (2007) 17: 395. https://doi.org/10.1007/s11222-007-9033-z

[18] Weisstein, E. W. (1999). Laplacian matrix. Online available: http://mathworld.wolfram.com/LaplacianMatrix.html

[19] Zhang, Yin & Jin, Rong & Zhou, Zhi-Hua. (2010). Understanding bag-of-words model: A statistical framework. International Journal of Machine Learning and Cybernetics. 1. 43-52. 10.1007/s13042-010-0001-0.