MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

Korszerű információs technológiák

Klaszteranalízis

Dr. Tompa Tamás
egyetemi adjunktus
Általános Informatikai Intézeti Tanszék

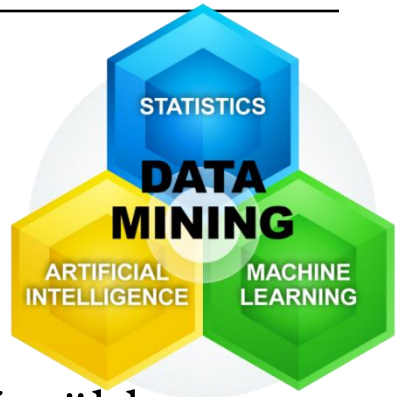
Miskolc, 2023.

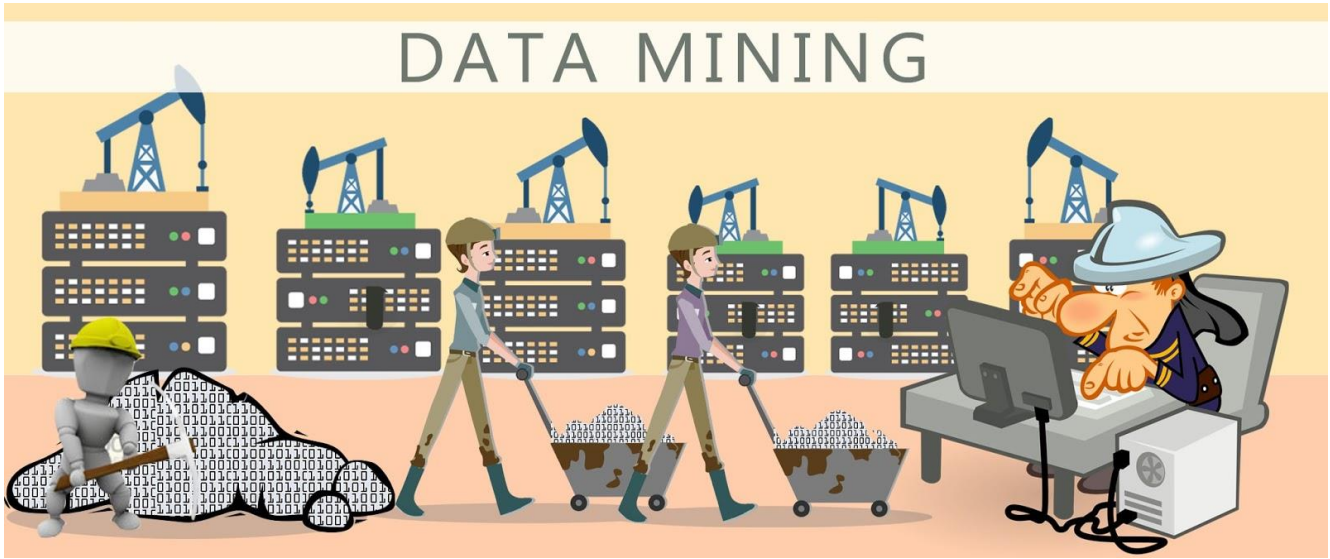
Tartalom

- Klaszterezés fogalma
- Klaszterezés típusai
- Adatok, klaszterek és klaszterező algoritmusok jellemzői
- Partícionáló klaszterezési algoritmusok
- Hierarchikus klaszterezési algoritmusok
- Sűrűség-alapú klaszterezési algoritmusok
- Klaszterezés bemutató Matlab-ban

Adatbányászat

- Adatbánya̋szat (data mining)
  - olyan folyamat, amellyel hasznos információ fedezhető fel automatikus módon nagy adattárakban, algoritmusok alkalmazásával
  - képesek előrejelzést adni: pl. egy új vásárló többet fog-e költeni, mint 100 euro





Adatbányászat

- **Adatbányászat feltételei**
  - **Nagy mennyiségű adat**
    - a kinyert szabályok statisztikai jelentőségét növeli
    - minél több az adat annál kisebb az esélye, hogy a talált összefüggés csak a véletlen műve
  - **Sok attribútum**
    - ha az objektumokat leíró attribútumok száma kicsi akkor hagyományos összefüggésekkel is feltárható a tudás
    - sok attribútum -> hagyományos összefüggések nem elegendőek -> adatbányászati módszerek alkalmazása

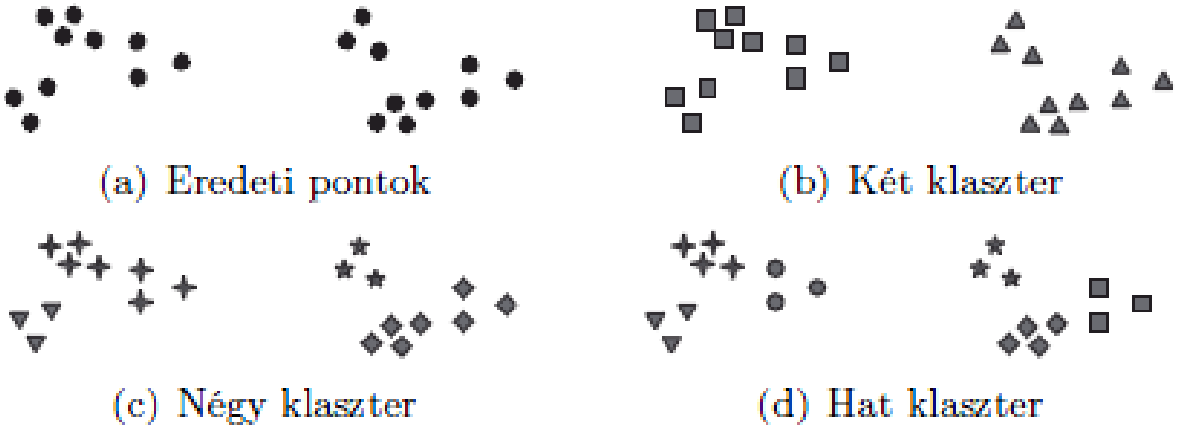
Adatbányászat

- **Adatbányászat feltételei**
  - **Tiszta adat**
    - a zajok, hibás bejegyzések csak nehezítik az adatbányászatot
  - **Torzítatlan adat**
    - adatok megfelelő kiválasztása
  - **Alkalmazási terület akcióképessége**
    - a kinyert tudás felhasználása

Klaszterezés

- **Cél:**
  - **egy adathalmaz pontjainak hasonlóság alapján való csoportosítása**
    - pl. weboldalak csoportosítása tartalmuk szerint; webfelhasználók csoportosítása böngészési szokásaik szerint
  - ugyanazon tulajdonságú elemek ugyanazon csoportba (klaszterbe) helyezése
  - különböző csoportok különböző tulajdonság elemeket tartalmazzanak
  - csoportok kialakítása nem egyértelmű feladat
    - Pl.: francia kártya pakli szín szerint 4 csoport, de figura szerint 13 csoport

Klaszterezés – példa

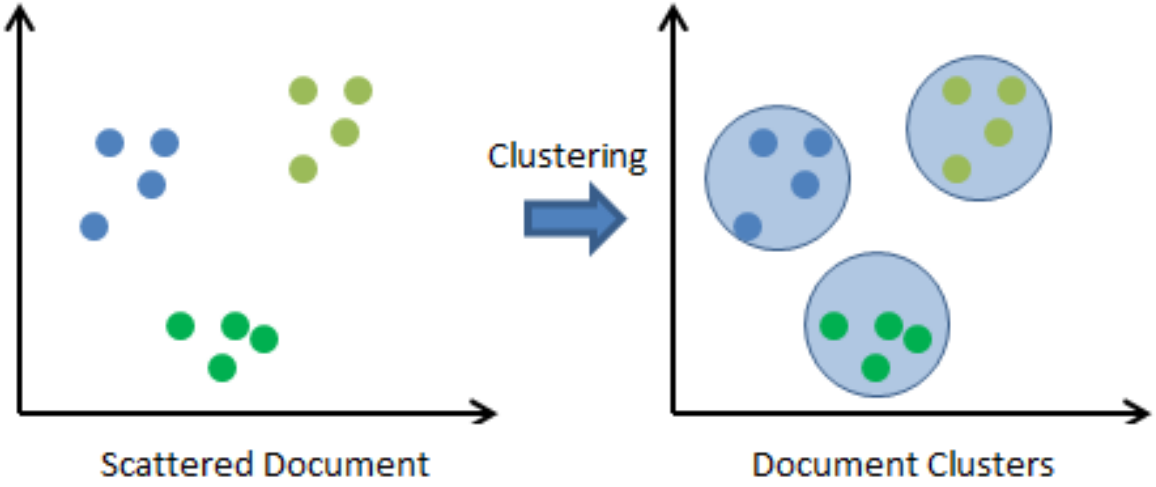


(a) Eredeti pontok

(b) Két klaszter

(c) Négy klaszter

(d) Hat klaszter



Klaszterezés vs. Osztályozás

- Hasonló feladat az osztályozás, de:
- **osztályozás** esetén
  - a választható osztályok valamilyen modellel, előre leírva adottak
  - az adatok több előre meghatározott kategóriák (osztályok) egyikéhez történő hozzárendelése a célja
  - felügyelt (supervised)
- **klaszterezés** esetén
  - nem léteznek előzetesen megadott kategóriák
  - az adatok alakítják ki a klasztereket és azok határait
  - felügyelet nélküli (unsupervised)

Osztályozási módszerek

- **1R osztályozási algoritmus:**
  - célja: döntési szabály tanulása, amely a bemeneti adatokat a helyes kimenettel (osztályozási eredményekkel) párosítja
  - azt a döntési szabályt állítja fel, hogy **minden bemeneti tulajdonságot egyetlen osztályhoz rendel, és azt az osztályt választja ki, amelyhez a legtöbb tulajdonság tartozik**.
  - ez a döntési szabály az 1R nevét is magyarázza, az "1R" az "egy tulajdonság, egy szabály" rövidítése

Osztályozási módszerek

- **Bayes-osztályozási algoritmus:**
  - valószínűségi számításokat alkalmaz a bemeneti adatok osztályozásához, az algoritmus a Bayes-tételre épül

1. Adottak a bemeneti tulajdonságok és a hozzájuk tartozó osztálycímkék
2. Kiszámítjuk a priori valószínűségeket, amelyek az osztályok előfordulási valószínűségeinek becslésére szolgálnak
3. Kiszámítjuk a feltételes valószínűségeket, amelyek azt mutatják, hogy adott bemeneti tulajdonságok mellett milyen valószínűséggel tartozik egy adott osztályhoz a bemenet
4. Alkalmazzuk a Bayes-tételt, amely segítségével a feltételes valószínűségeket szorozzuk össze a priori valószínűségekkel az osztályok poszterior valószínűségeinek kiszámításához
5. Végül az algoritmus kiválasztja a legvalószínűbb osztályt a poszterior valószínűségek alapján

Klaszterező algoritmusok típusai

- Partícionáló
  - mindegyik objektum pontosan egy részhalmazba kerül
  - a pontokat *k* diszjunkt csoportra osztják úgy, hogy minden csoportba legalább egy elem kerüljön
- Hierarchikus
  - egymásba ágyazott klaszterek fába rendezett halmaza
  - hierarchikus adatszerkezet: fa (dendogram)

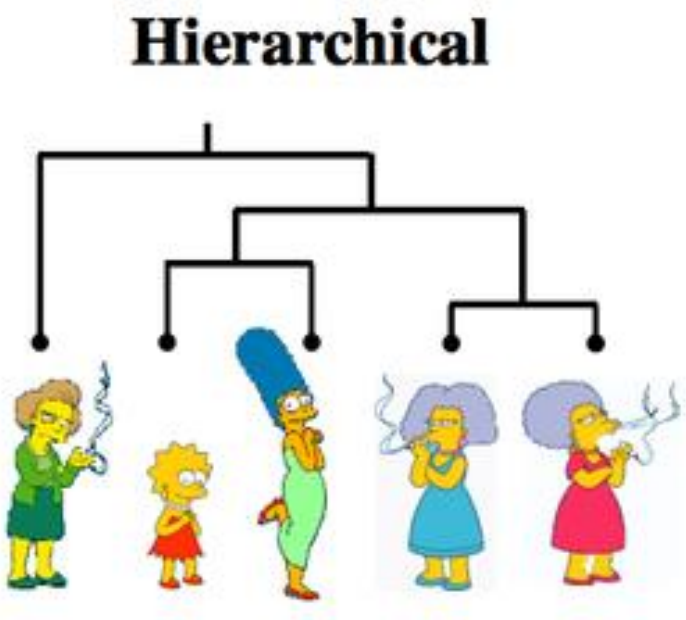
Klaszterező algoritmusok típusai

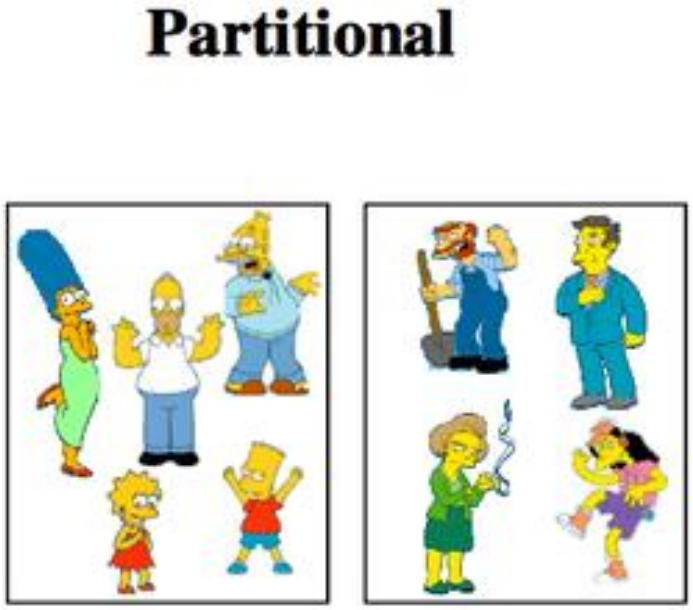
- Kizáró
  - mindegyik objektumot csak egyetlen klaszterhez tartozhat
- Sűrűség-alapú
  - egy klasztert addig növesztenek, amíg a sűrűség a „szomszédságában” meghalad egy bizonyos korlátot
- Grid-alapú
  - elemek rácspontokba való rendezése

Klaszterezés típusai

- Átfedő
  - egy objektum egyszerre több csoporthoz is tartozhat
- Fuzzy
  - minden objektum minden klaszterbe tartozik egy tagsági érték alapján
  - tagsági érték: 0 (egyáltalán nem tartozik bele) és 1 (teljesen beletartozik)
  - pl. fuzzy c-means

Klaszterezés típusai





Klaszterezés jellemzői

- Adatpontok hasonlósági viszonyai az alapja
  - hasonlósági függvény meghatározása
    - távolság- vagy hasonlósági függvények
      - numerikus adatokra: pl. Euklideszi metrika, Manhattan-távolság, Minkowski-metrika
      - bináris, kategórikus adatokra: pl. Rand-hasonlóság, Jaccard-hasonlóság
    - nagy adathalmazok esetén nem oldható meg elég gyorsan -> elfogadható futási időben nem ér véget
      - megoldás: dimenzió-csökkentés
        - módszerek: pl. szinguláris-felbontás, ujjlenyomat-alapú, PCA (*Principal Component Analysis*,), SVD (*Singular Value Decomposition*)

Klaszterezés jellemzői

- Egy C klaszter eleminek száma $|C|$
- Klaszter átmérője: $\mathrm{D}(C)$
  - Elemek közötti átlagos távolság:

$$D_{avg}(C) = \frac{\sum_{p \in C} \sum_{q \in C} d(p,q)}{|C|^2}$$

  - Elemek között maximális távolság:

$$D_{max}(C) = \max_{p,q \in C} d(p,q)$$

- Klaszterezés jósága
  - nincs minden szempontot kielégítő mérték
  - -> függvények minimalizálása

Hasonlóság és távolság tulajdonságai

- Elempárok távolsága/hasonlósága számítható
  - **Def.**: Tetszőleges két **x,y** klaszterezendő elem esetén azok hasonlóságát **s(x,y)** jelöli.
    - $0 <= s(x,y) <= 1$
    - $s(x,y) = s(y,x)$
    - $s(x,x) = 1$

Távolság tulajdonságai

- Hasonlóság helyett legtöbbször **távolság** használata
  - **x,y** pontok esetén **d(x,y)**-al jelölve x,y közötti távolságot

- Egy **d(x,y)** függvény távolság ha:
  - minden (x,y) esetén $0<=d(x,y)<\infty$
  - minden x adatpontra $d(x,x)=0$
  - szimmetrikus, azaz minden x,y esetén $d(x,y)=d(y,x)$

Távolság tulajdonságai

- Időnként feltesszük, hogy **d**-re teljesül a **háromszög-egyenlőtlenség**:
  - ha minden x,y,z ponthármasra **d(x,z)<=d(x,y)+d(y,z)**
  - ha egy távolságra teljesül a háromszög-egyenlőtlenség akkor azt **metrikának** nevezzük
  - a háromszög-egyenlőtlenség teljesülése nincs megkövetelve

Távolság tulajdonságai

- A távolságokat egy szimmetrikus **távolság-mátrix**-al reprezentáljuk

$$\begin{bmatrix} 0 & d(1,2) & d(1,3) & \cdots & d(1,n) \\ d(2,1) & 0 & d(2,3) & \cdots & d(2,n) \\ d(3,1) & d(3,2) & 0 & \cdots & d(3,n) \\ & \vdots & \vdots & \ddots & \vdots \\ d(n,1) & d(n,2) & d(n,3) & \cdots & 0 \end{bmatrix}$$

  - ahol **d(i,j)** adja meg az i-edik és a j-edik elem távolságát
  - **n** az adatpontok száma

Euklideszi-távolság

- Euklideszi-távolság

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

  - két pont távolsága egyenlő a pontok különbségének négyzetének gyökével (p és q vektorok)
  - több dimenzió esetén a négyzetre emelést minden dimenzióra el kell végezni -> eredményüket összegezni -->elvégezni a gyökvonást

Euklideszi-távolság

- Súlyozott Euklideszi-távolság

$$d(x,y) = \sqrt{w_1|x_1 - y_1|^2 + w_2|x_2 - y_2|^2 + \cdots + w_m|x_m - y_m|^2},$$

ahol w_i-vel jelöltük i-edik attribútum súlyát és legyen $\sum_{i=1}^{m} w_i = 1$.

Manhattan-távolság

- Manhattan-távolság

$$d(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{N} |p_i - q_i|$$

  - a koordináta különbségének abszolútértékének összege
  - p és q vektorok
    - $\mathbf{p}=(p_1,p_2,..,p_N)$ $\mathbf{q}=(q_1,q_2,...,q_N)$

Klaszterező algoritmusok jellemzői

- Skálázhatóság
  - tár és időszükséglet kezelhető méretű maradjon nagy adathalmazokon is
- Adattípusok
  - többfajta adattípusra is működjön (numerikus, bináris, kategórikus, és ezek keverékei)
- Klaszter-geometria
  - ne preferáljon a kialakítandó klaszterek között azok geometriai tulajdonságai alapján

Klaszterező algoritmusok jellemzői

- Robusztusság
  - ne legyen zajos adatokra érzékeny
- Sorrend-függetlenség
  - ne függjön az adatpontok beolvasási sorrendjétől

Klaszterező algoritmusok jellemzői

- Azon adatpontokat amelyek lényegesen különböznek minden más adatponttól, **kívülállónak** (**outlier**) nevezzük
  - keletkezésének oka: mérési hiba, zaj
  - két eset
    - az adatpont elhagyható
    - az adatpont fontos

Klaszterező algoritmusok jellemzői

- Skála-invariancia
  - ha minden elempár távolsága helyett annak α-szorosát vesszük alapul (α>0), akkor a klaszterező eljárás eredménye ne változzon meg
- Gazdagság
  - tetszőleges, előre megadott klaszterezéshez létezzen olyan távolságmátrix, hogy a klaszterező eljárás az előre megadott partícióra vezessen

Klaszterező algoritmusok jellemzői

- Konzisztencia
  - tekintsük egy adott adathalmazon egy klaszterező algoritmus eredményét
  - a pontok közötti távolságok megváltoztatása úgy, hogy
    - azonos csoportban lévő elempárok között a távolság nem nő
    - különböző csoportban lévő elempárok között a távolság nem csökken
  - ekkor az újonnan kapott távolságokra alkalmazott algoritmus az eredetivel megegyező eredményt (csoportosítást) ad

Particionáló klaszterező algoritmusok

- Alapötlet
  - a megfelelő klaszterezést a pillanatnyi eredményeként kapott klaszterezés **folyamatos pontosítás**ával, **iterálva** éri el
  - az adatpontok hozzárendelése a hozzá legközelebb eső klaszterhez
  - akkor ér véget ha az iteráció során, nem vagy csak alig változik meg a partíció

  - Pl.: k-means, k-medoid

K-means (k-közép) algoritmus

- Az egyik legelső és legelterjedtebb klaszterezési algoritmus
- K db kezdő középpont kiválasztása
  - K egy megadott paraméter: klaszterek száma
- Adatpontok a hozzá legközelebb eső középponthoz való rendelése
- Az így létrejött csoportok a klaszterek
- Klaszterközéppontok frissítése a hozzájuk rendelt pontok alapján
  - kezdeti középpontok elmozdítása a tényleges klaszterközéppont irányába
- Ismétlés míg egyetlen pont sem vált klasztert vagy a középpontok nem változnak (vándorolnak)

K-means (k-közép) algoritmus

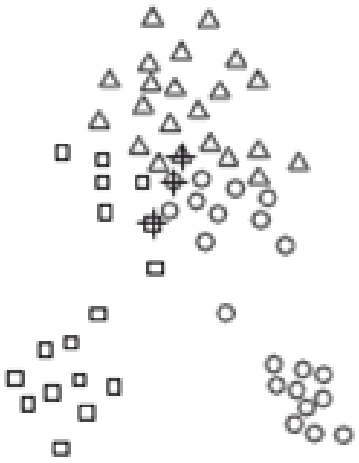
Algoritmizálva: (MacQueen, 1967)

1: Válasszunk ki K darab kezdeti középpontot
2:**repeat**
    3: Hozzunk létre K darab klasztert mindegyik adatpontnak a legközelebbi középponthoz rendelésével
    4: Számoljuk újra mindegyik klaszter középpontját
5: **until** a középpontok nem változnak

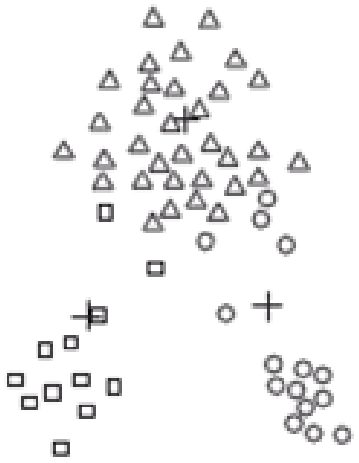
K-means (k-közép) algoritmus

1. **Kezdő középpontok inicializálása:** véletlenszerűen választunk k középpontot a d-dimenziós térben, ahol k a felhasználó által megadott kívánt csoportok száma.
2. **Középpontokhoz legközelebbi adatpontokhoz tartozó csoportok kialakítása:** az adatpontok csoportokba való besorolása a hozzájuk legközelebbi középpontok alapján. Ezt a clustering folyamatot a Euclidean távolság vagy más távolságmérő függvény segítségével lehet elvégezni.
3. **Középpontok frissítése:** Miután az összes adatpontot hozzárendeltük a hozzájuk legközelebbi középponthoz, a következő lépés az új középpontok kiszámítása az egyes csoportok átlaga alapján. Ez a lépés a csoportok centroidjainak (középpontjainak) kiszámítását jelenti.
4. **Ismétlés:** Az előző két lépést addig ismételjük, amíg a csoportok elrendezése stabilizálódik és már nem változik jelentősen.
5. **Kimenet:** az algoritmus kimenete az adatpontok csoportba sorolása, valamint a csoportok középpontjainak koordinátái.

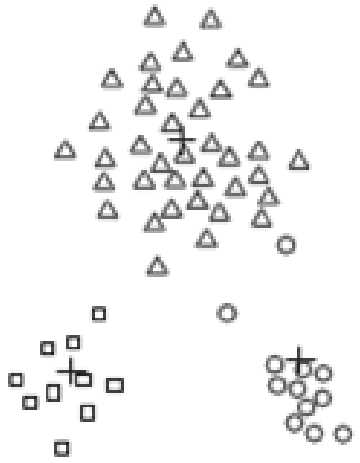
K-means (k-közép) algoritmus - példa



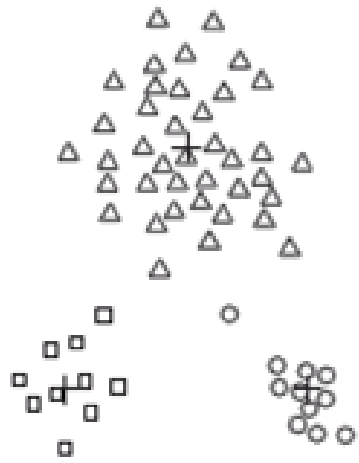
(a) Első iteráció



(b) Második iteráció

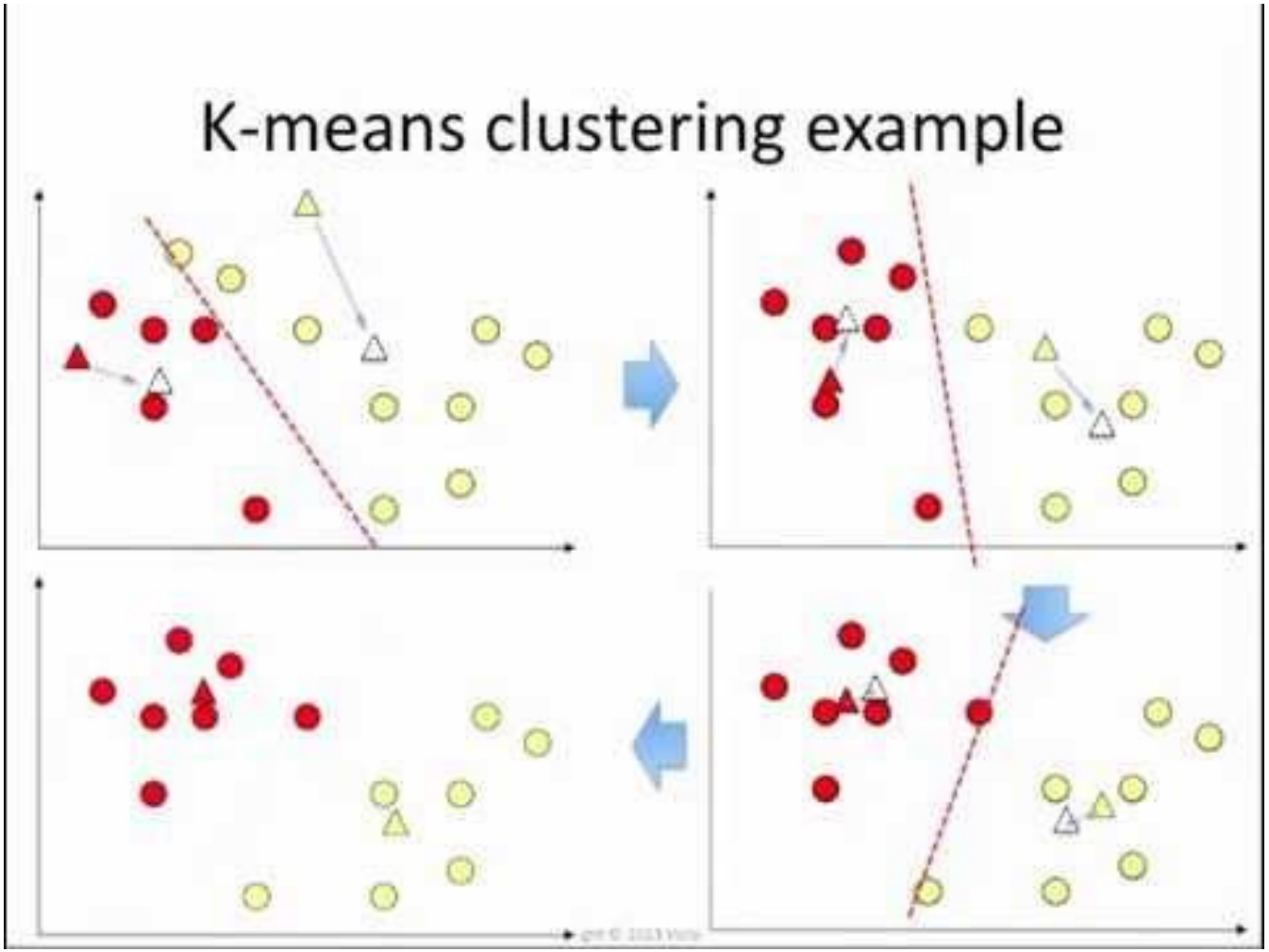


(c) Harmadik iteráció



(d) Negyedik iteráció

K-means (k-közép) algoritmus - példa



K-means (k-közép) algoritmus

- Futási idejét befolyásolja az alkalmazott távolság számítási módszer

- Előnyei
  - egyszerű
  - érzéketlen az adatpontok sorrendjére

- Hátrányai
  - előre megadott klaszterek száma (K)
  - érzékeny a kívülálló pontokra
  - K klaszterszám megtalálása többszöri futtatást igényelhet
  - a végeredmény függ a kezdeti partíciótól
  - nem képes kezelni nem gömb alakú, vagy különböző méretű és sűrűségű klasztereket

K-medoid algoritmus

- K-means algoritmus módosulata
  - a k-means csak olyan esetben használható mikor értelmezett a klaszterközéppont
  - k-medoid az adatpontok közötti távolságot összehasonlításán alapszik
  - a klasztereket **medoid**-al reprezentálja
  - medoid: az adott klaszter elemeihez legközelebb eső adatpont
    - „klaszterközéppont” egy létező adatelem
  - az algoritmus menete megegyezik a k-means menetével
    - kezdetben választunk k darab tetszőleges adatpontot: ezek lesznek a medoid-ok
    - adatpontok medoid-okhoz történő hozzárendelése
    - medoid-ok vándorlása

K-medoid algoritmus

1. **Kezdő középpontok inicializálása:** A k-medoid algoritmus kezdeti lépése az, hogy véletlenszerűen választunk k középpontot az adatok közül, ahol k a felhasználó által megadott kívánt csoportok száma.
2. **Középpontokhoz legközelebbi adatpontokhoz tartozó csoportok kialakítása:** az adatpontok csoportokba való besorolása a hozzájuk legközelebbi középpontok alapján. Ezt az úgynevezett clustering folyamatot a Manhattan vagy más távolságmérő függvény segítségével lehet elvégezni.
3. **Középpontok cseréje:** Az előző lépésben elkészült csoportokhoz tartozó középpontokat lehet újra kiválasztani a csoporton belül egy olyan adatpontra cserélve, amelynek cseréje javítja a csoportok homogenitását. A cserélendő középpont a csoporton belül olyan adatpont lesz, amelynek összes többi adatponthoz viszonyított távolsága minimális.

K-medoid algoritmus

4. **Ismétlés:** Az előző két lépést addig ismételjük, amíg a csoportok elrendezése stabilizálódik és már nem változik jelentősen.
5. **Kimenet:** A k-medoid algoritmus kimenete az adatpontok csoportba sorolása, valamint a csoportok középpontjainak koordinátái.

K-means és K-medoid jellemzők

- Az egyik fő különbség a k-means és a k-medoid algoritmusok között az, hogy milyen módon választják ki a csoportok reprezentatív elemeit. **A k-means algoritmus az adatpontok átlagát használja a középpontok kiválasztásához, míg a k-medoid algoritmus az adatpontok közül választja ki a középpontokat.**

- További különbség az algoritmusok érzékenysége a zajos adatokra. **A k-means algoritmus érzékeny a zajos adatokra**, mivel az átlagokat számítja a csoportok középpontjához, és egyetlen zajos adatpont jelentős hatással lehet az átlagra. **A k-medoid algoritmus azonban ellenállóbb a zajos adatokkal szemben**, mivel a középpontokat az adatok közül választja ki, és nem az átlagokból számítja ki őket.

Hierarchikus klaszterező algoritmusok

- Alapötlet
  - adatelemek egy **hierarchikus adatszerkezet**be rendezése
  - hierarchikus adatszerkezet: **fa**, dendogram
  - az adatpontok a fa leveleiben találhatóak
  - a fa minden belső pontja egy klaszternek felel meg
    - a fában az alatta lévő pontokból
    - -> a gyökér az összes pontot tartalmazza
  - az algoritmus lehet
    - felhalmozó (egyesítő, bottom-up)
    - lebontó (felosztó, top-down)

Hierarchikus klaszterező algoritmusok

- Felhalmozó (egyesítő, bottom-up)
  - kezdetben minden elem különálló klaszter
  - legközelebbi klaszterek egyesítése
    - a hierarchiában egy szinttel fentebb lévő klaszterek egyesítése
- Lebontó (felosztó, top-down)
  - kezdetben minden adatpont egy klasztert alkot
  - a klaszter további klaszterekre való bontása minden iterációban

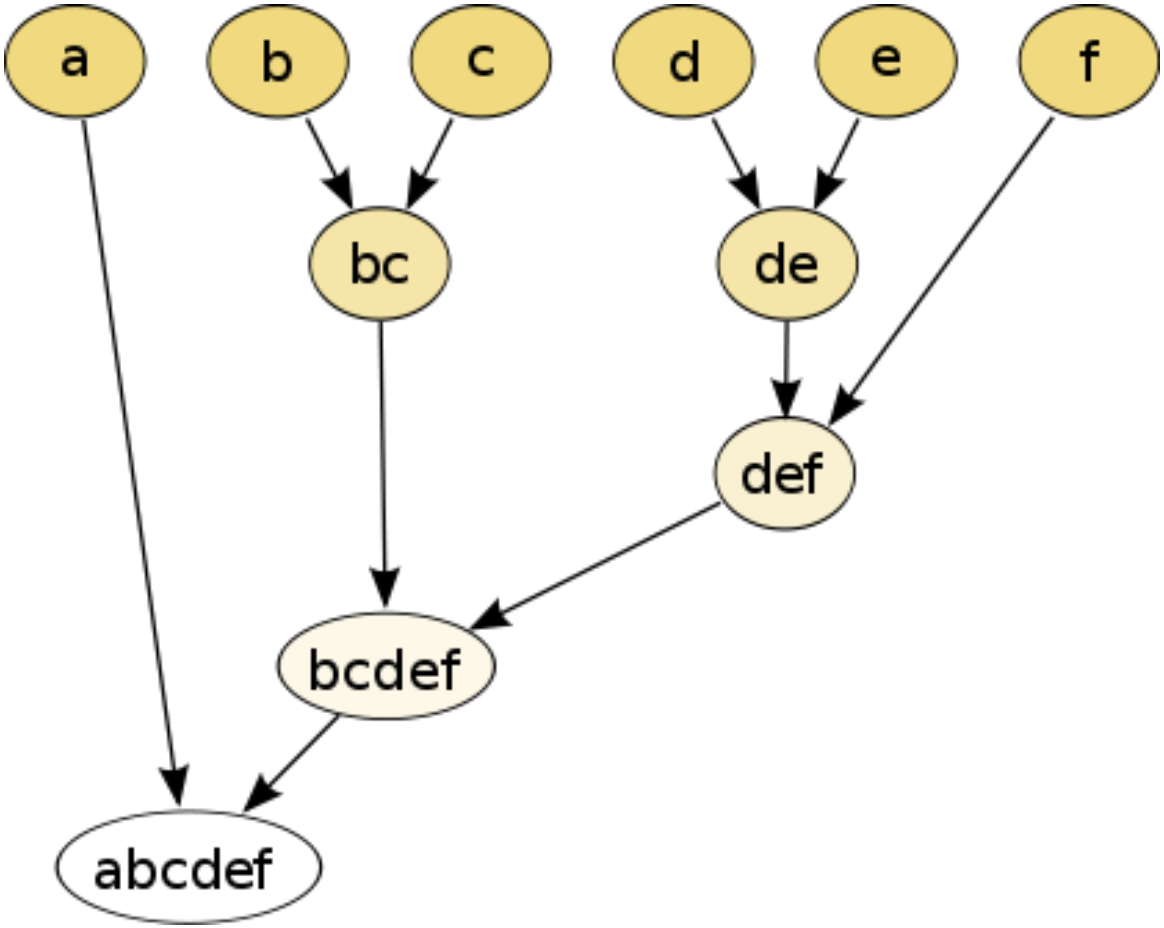
Hierarchikus klaszterező algoritmusok

- Kulcslépés
  - az egyesítendő vagy felosztandó klaszterek megválasztása
- Az összevonások/szétválasztások visszavonhatatlanok
- Probléma
  - zajos adatok
  - magas-dimenziójú adatok

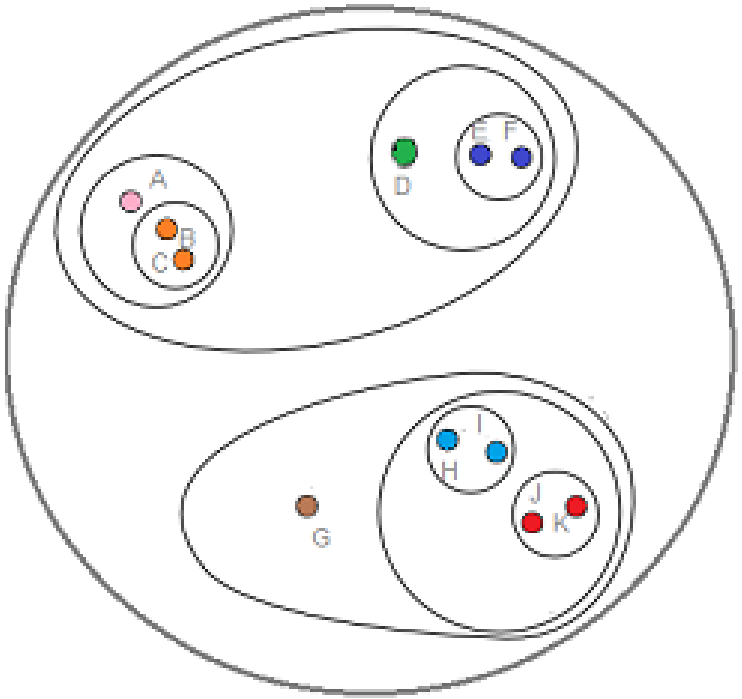
Hierarchikus klaszterező algoritmusok

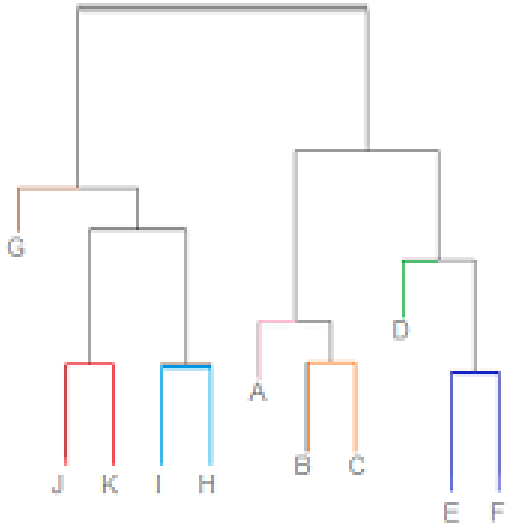
- Pl. egy egyszerű felhalmozó algoritmus
  - D adatpontok halmaza
  - $d_*(C_i,C_j)$ két klaszter távolságát mérő függvény
  - S(C) megállási szabály
  - Kezdetben minden pont önálló klaszter
  - d_* szerint egymáshoz 2 legközelebbi klaszter egyesítése
  - Iterálás míg S(C) miatt le nem áll
    - S(C) pl. küszöbérték d_*-ra
- Pl. Rock algoritmus (*RObust Clustering using linKs*)

Hierarchikus klaszterező algoritmusok

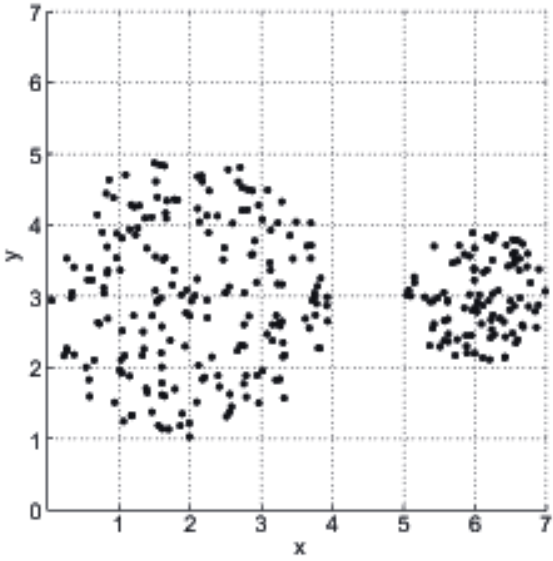


Hierarchikus klaszterező algoritmusok





További módszerek



- Rács-alapú
  - az adatteret rácscellákra bontja
  - klasztereket alakít ki azokból a cellákból, amelyek kellőképpen sűrűek
  - célja az objektumok olyan sűrű tartományainak megkeresésére, amelyeket kis sűrűségű tartományok vesznek körül

  - 1. Definiáljuk rácscellák egy halmazát
  - 2. Rendeljük hozzá az objektumokat a megfelelő cellákhoz és számoljuk ki az egyes cellák sűrűségét
  - 3. Távolítsuk el azokat a cellákat, amelyek sűrűsége adott küszöbértéknél kisebb
  - 4. Alkossunk klasztereket az érintkező, sűrű cellacsoportokból
  - Pl. DENCLUE

Klaszterező algoritmusok alkalmazásai

- Képfeldolgozás
  - fényképek kategorizálása
  - adott képhez hasonló képek keresése
- Keresőmotorok
- Szövegbányászat
- Mintaillesztés
- Számítógép-hálózatok
- Marketing, online ajánló rendszerek
- Robotika

Klaszterező algoritmusok alkalmazásai

- Matlab bemutató :)

Felhasznált irodalom

- Pang-Ning Tan, Michael Steinbach, Vipin Kumar - Bevezetés az adatbányászatba
  - http://www.tankonyvtar.hu/hu/tartalom/tamop425/0046_adatbanyaszat/ch08.html
- Jegyzet
  - http://web.cs.elte.hu/~lukacs/Papers/klaszterezes.pdf

Köszönöm a figyelmet!