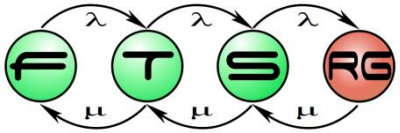
A legjobb MapReduce példa (a szószámlálás után): kmeans

Salánki Ágnes, Kocsis Imre

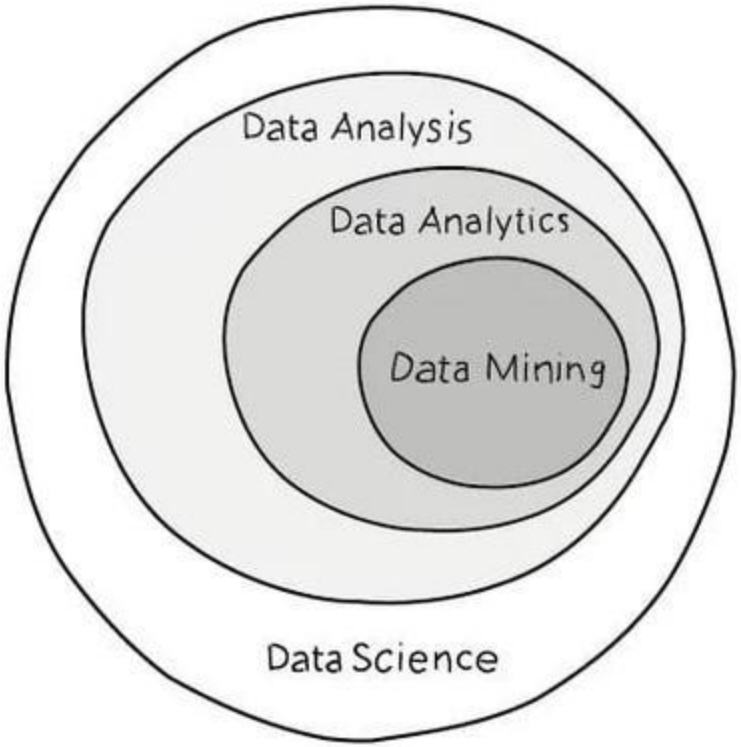
salanki@, ikocsis@

2015.10.06.



Adatelemzés

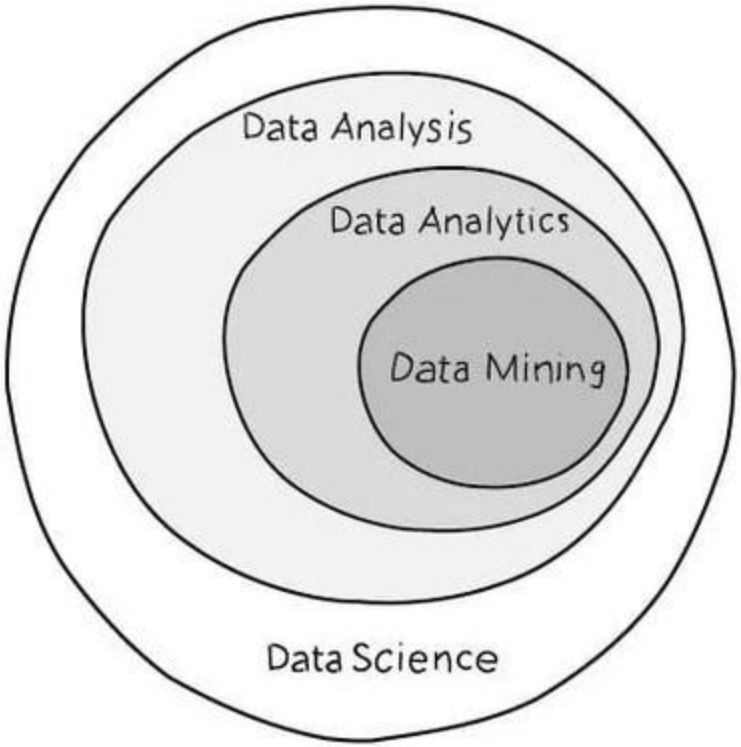
- Gépi tanulás (Machine Learning)
  - Előrejelző modellt épít
- Adatbányászat
  - ML eredményeket felhasználva nyer ki új információt nyer ki



Ábra forrása: https://onthe.io/learn

Adatelemzés

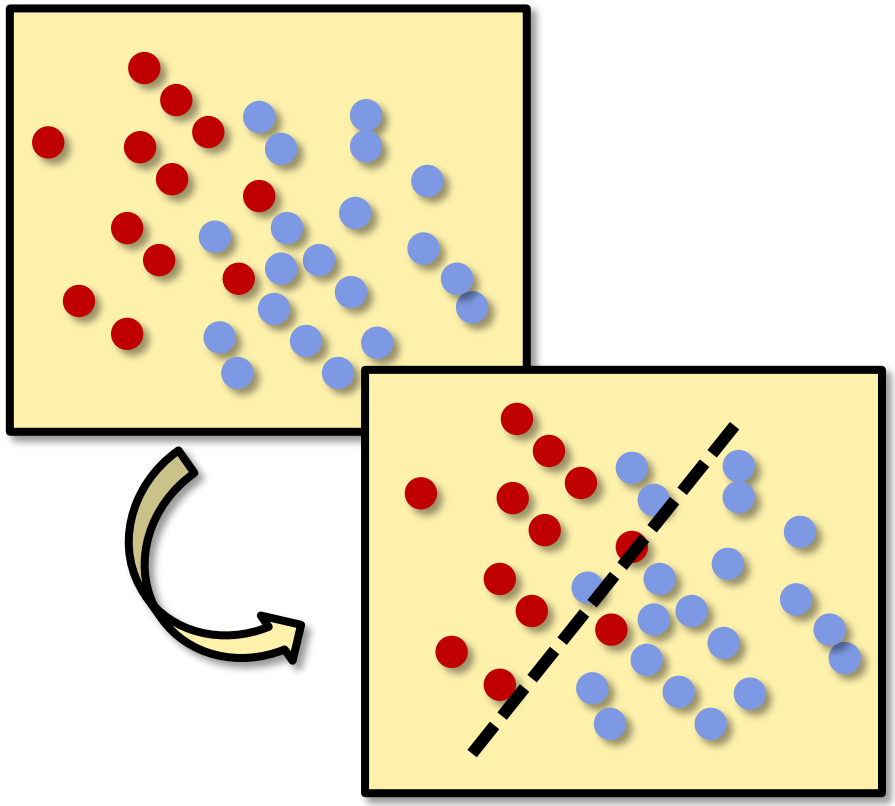
- Analytics
  - nagyrészt automatizálás
  - az adat teljes életciklusa
- Analysis
  - egy konkrét adatsor vizsgálata
  - szakértői is (nem kell ML)
- Data Science
  - strukturált/nemsukturált
  - az adat teljes életciklusa

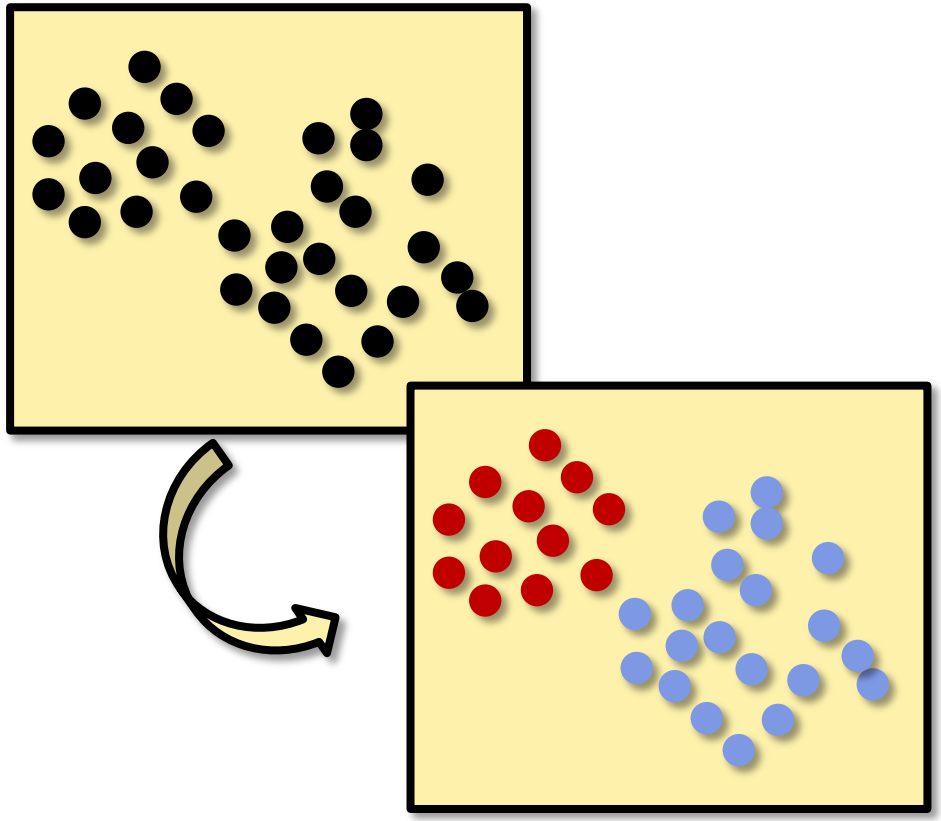


Ábra forrása: https://onthe.io/learn

ML módszerek egy csoportosítása

- Felügyelt tanulás
  - néhány pontra adott az elvárt kimenet
  - Fő feladat: általánosítás
- Nemfelügyelt tanulás
  - nincs súgásunk
  - Fő feladat: mintakeresés





Felügyelt vagy nemfelügyelt?

Feladat1

Melyik a kakukktojás?

„hosszú”, „könnyű”,

„hattyú”, „jobbra”, „pöttyös”

Feladat2

Ha 8809 -> 6

7111 -> 0

6666 -> 4

9313 -> 1, akkor

2581 -> ???

Felügyelt vagy nemfelügyelt?

Feladat1

Megmutatjuk 100 nő fényképét és 100 férfi fényképét. Aztán 10 fényképről el kell döntenünk , hogy nő vagy férfi van rajta.

Feladat2

Egy hatalmas mozifilm-néző értékelési mátrix alapján megpróbáljuk kideríteni, mik voltak a hasonló filmek

Feladat3

Két barátunk lefényképezte a kutyáját 100-szor különböző szögekből, majd a 200 képet összekeverték. Szét kell válogatnunk két kupacra.

Feladat4

A tavalyi mozijegy-bevételi statisztikánk alapján meg kell állapítanunk, hogy az új Éhezők viadala film mekkora bevételre számíthat.

k-means

- Adatpontok: vektortér
- Klaszter reprezentációja: súlyponttal / ”középponttal” (vektor-átlag)
- $r(C_i)$: i-edik klaszter reprezentánsa
- Minimalizálandó pontok négyzetes távolságösszege, mint hiba:

$$E(C) = \sum_{i=1}^{k} \sum_{u \in C_i} d\left(u, r(C_i)\right)^2$$

Egy megoldás

$\{r(C_1), r(C_2), \ldots, r(C_k)\} \leftarrow$ repr. kezdeti halmaza
while $r(C_i)$ változik
do for $\forall u \in D$ adott sorrendben
do
$h \leftarrow u$ klaszter-indexe
$j \leftarrow argmin_i d(u, r(C_i))$
if $h \neq j$ **then** {
$C_j \leftarrow C_j \cup \{u\}$
$C_h \leftarrow C_h \setminus \{u\}$
$r(C_j) \leftarrow \frac{1}{|C_j|} \sum_{v \in C_j} v$
$r(C_h) \leftarrow \frac{1}{|C_h|} \sum_{v \in C_h} v\}$
return C

Régi klaszter

Új klaszter

Itt rögtön újra is számoljuk

k-means: map

- **`rmr2/blob/master/pkg/tests/kmeans.R`**
- Kulcs: P ponthoz legközelebbi C klaszter-centrum
  - C „normál” R objektum
  - scoping miatt elérhető a map-ben
  - P: HDFS-ből
- Érték: P
- Ha még nincsenek klaszter-centrumok: mintavétel visszahelyezéssel
- Vektorizált keyval ismét

k-means: map

```
1  kmeans.map =
2        function(., P) {
3          nearest = {
4            if(is.null(C))
5              sample(1:num.clusters,nrow(P),
6                     replace = T)
7            else {
8              D = dist.fun(C, P)
9              nearest = max.col(-D)}}
10
11         if(!(combine || in.memory.combine))
12           keyval(nearest, P)
13         else
14           keyval(nearest, cbind(1, P))}
```

A Map kap néhány pontot

Első kör: inicializálás

Legközelebbi klaszter

k-means

P pont C_i klasztertől vett távolsága

```
15 dist.fun = function(C, P){
16     apply(C,
17           1,
18           function(x)
19             colSums((t(P) - x)^2))}
```

A klaszter középpontok mátrixának minden sorára

k-means: reduce

- Azonos kulcshoz (középpont) tartozó vektorok átlaga
- Azaz:
  - Map: a legközelebbi klaszterbe sorol (középpont)
  - Reduce: kialakult új középpontok
- Szemlétesen: a középpontokat „tologatjuk”
- Beragadhat lokális minimumba! (aut. megállásnál)
- Algoritmust lásd (aut. megállással): [8], p 1422

k-means: reduce

```
1 kmeans.reduce = {
2 if (!(combine||in.memory.combine))
3     function(., P)
4         t(as.matrix(apply(P, 2, mean)))
5 else
6     function(k, P)
7         keyval(
8           k,
9          t(as.matrix(apply(P,2,sum))))}
```

k klaszterközépponthoz lekérjük az összes P pontot

Előbb csak összeget számolunk..

kmeans.mr: törzs (1)

- Iterációk C felüldefiniálásával

- Minden menetben mapreduce-szal új középpontok

kmeans.mr: törzs (1)

```
1     C = NULL
2       for(i in 1:num.iter ) {
3         C =
4           values(
5             from.dfs(
6               mapreduce(
7                 P,
8                 map = kmeans.map,
9                 reduce = kmeans.reduce)))
10        if(combine || in.memory.combine)
11          C = C[, -1]/C[, 1]
```