Notițe Seminar 10, 11

December 12, 2019

Intro: După cum știți, de săptămâna trecută am început capitolul de **Clusterizare**. Mai concret, am făcut clusterizare **ierarhică**. De acum începem clusterizarea **neierarhică**. Mai concret, în următoarele două seminarii vom face algoritmul **k-means**.

1 k-means - noțiuni de bază

k va fi un număr natural nenul care va semnifica în câte grupuri/clustere vrem să împărțim instanțele.

– clusterizare **neierarhică**/plată (nu mai facem arbori)

– asignare **hard** a instanțelor la clustere (având o instanță spunem că ea aparține unui singur cluster și atât)

– **algoritm iterativ, la fiecare iterație trebuind să actualizăm: centroizii, desenul, clusterele în această ordine** (convenție: noi lucrăm în această ordine, deși am putea actualiza și clusterele și apoi centroizii)

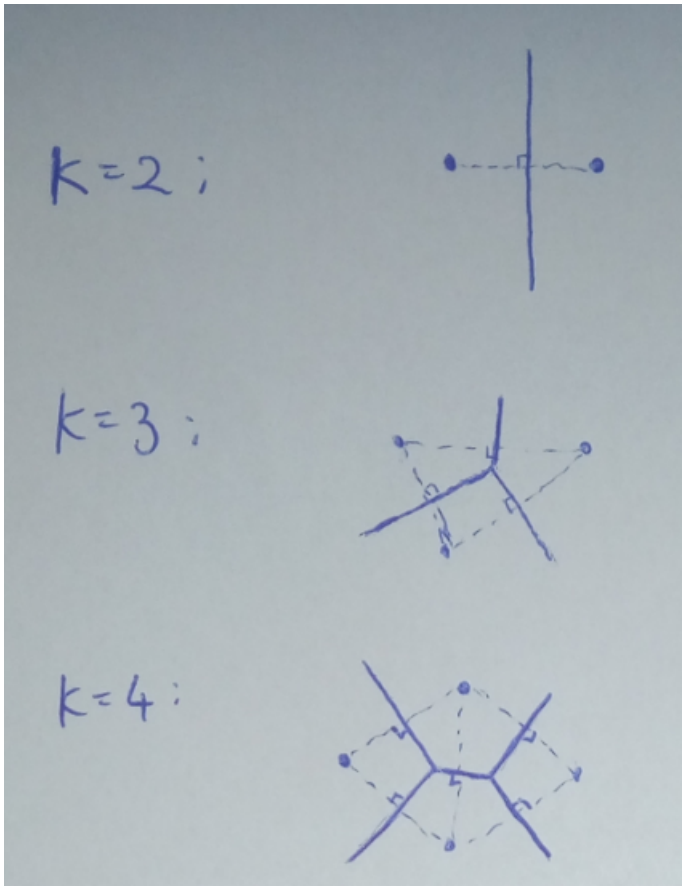
– **start/stop** algoritm:

- exemple de *euristici pentru iniţializarea centroizilor*:
  iniţializare arbitrară / random în $\mathbb{R}^d$ sau în $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^d$ (setul de date de clusterizat);
  aplicare în prealabil a unui algoritm de clusterizare ierarhică;
  folosind o anumită distribuţie probabilistă definită pe X: K-means++ (David Arthur, Sergei Vassilvitskii, 2007): ex. 43.
- exemple de *criterii de oprire*:
  după efectuarea unui număr maxim de iteraţii (fixat iniţial);
  când componenţa clusterelor nu se mai modifică de la o iteraţie la alta;
  când poziţiile centroizilor nu se mai modifică de la o iteraţie la alta;
  când descreşterea valorii criteriului J_K de la o iteraţie la alta nu mai este strictă sau nu mai este peste un anumit prag ε fixat în prealabil.

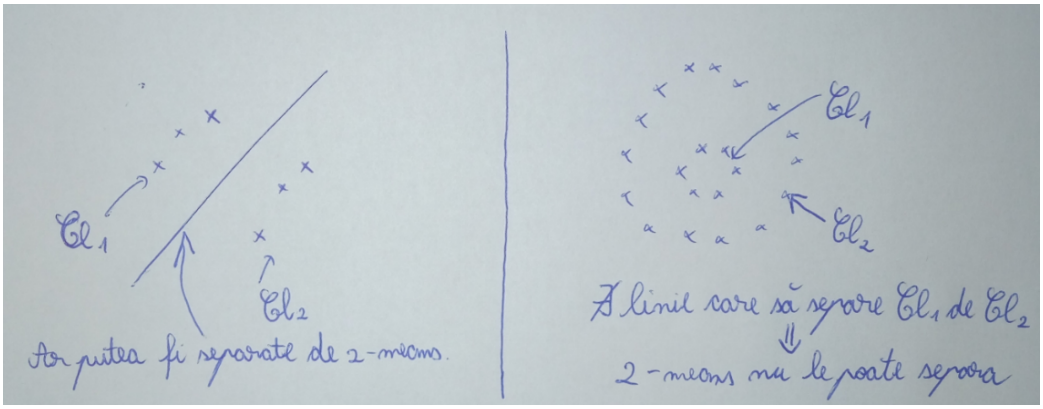
(preluat din `https://profs.info.uaic.ro/~ciortuz/ML.ex-book/book-book.8oct2019.pdf` pagina 463)

Aplicare: vezi ex. 7a/pag.481

– după cum se menționează la ex. 7a/pag.481 algoritmul poate fi aplicat în manieră ***analitică*** sau ***geometrică***: vezi ex. 7a/pag.481; de fapt, atunci când aplicați k-means în manieră geometrică, la o iterație voi desenați granițele de decizie corespunzătoare algoritmului **1NN** aplicat pe centroizi: considerați fiecare centroid un punct de o anumită etichetă, iar niciun centroid nu are aceeași etichetă ca un alt centroid



– **granițele de separare între clustere sunt liniare**



– din cauza inițializării care poate diferi, **algoritmul nu este deterministic**

Două noțiuni noi:
– **k-partiție** = cele k clustere
– **k-configurație** = cei k centroizi

2 k-means - algoritm de optimizare

Există **două rezultate** legate de optimizare referitoare la k-means: **ex. 12** (este rezolvat), **ex. 39** (este propus, dar are rezolvare în slide-uri).

2.1 Criteriul J

Pe ambele le veți aborda la curs, iar la seminar vom pune accent pe primul rezultat care spune că algoritmul k-means minimizează un **criteriu J al celor mai mici pătrate**:

$$J_k(C, \mu) = \sum_{i=1}^{n} \|x_i - \mu_{C(x_i)}\|^2$$

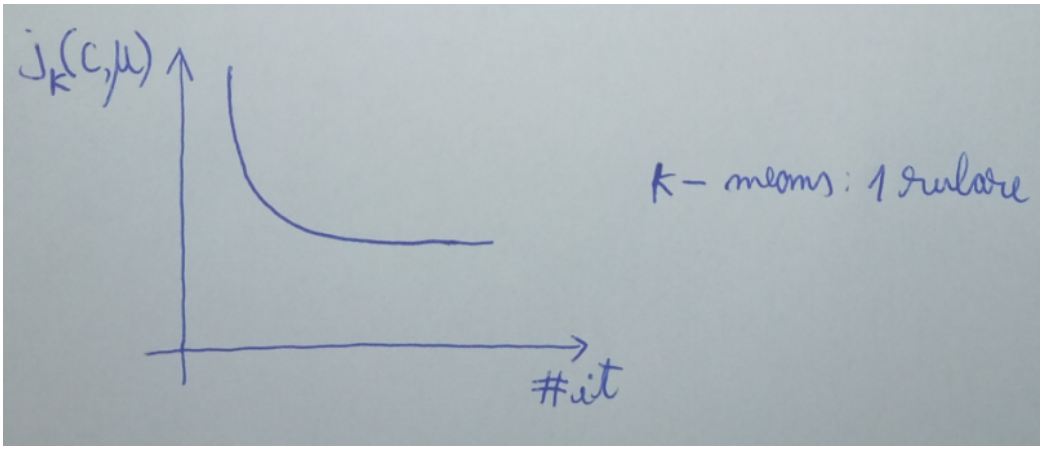
unde n reprezintă numărul de instanțe de clusterizat, x_i este o instanță de clusterizat, C este o împărțire a punctelor în clustere (adică o k-partiție), μ este o mulțime de *reprezentanți* (vectori) pentru fiecare cluster ($\mu = (\mu_1, \ldots, \mu_k)$), iar $C(x_i) \in \{1, \ldots, k\}$ este clusterul la care este asignată instanța x_i.

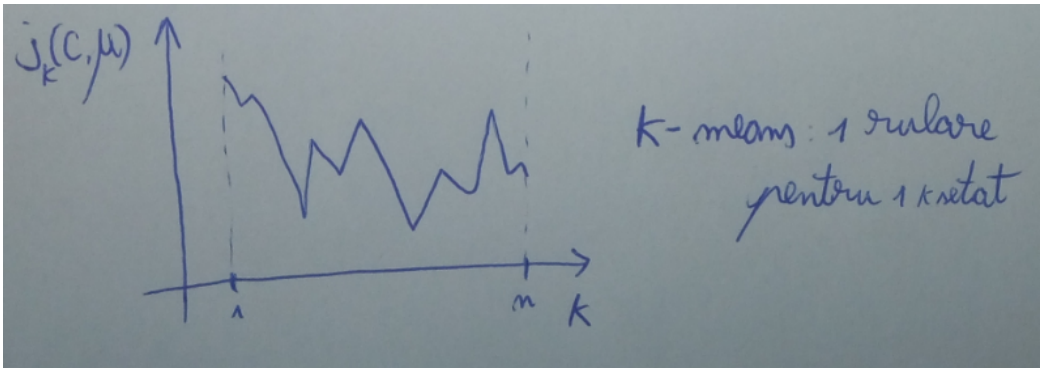
Altfel spus, având dat un set de date $X = (x_1, \ldots, x_n)$ și un număr natural nenul k, trebuie să găsim o împărțire în k clustere a instanțelor și pentru fiecare cluster, un *reprezentant* astfel încât criteriul $J_k(C, \mu)$ să fie minimizat. Algoritmul k-means încearcă să facă acest lucru printr-o metodă de căutare/optimizare care se numește **descreștere pe coordonate** (*coordinate descent*). Astfel, k-means reușește să găsească formule de actualizare pentru C și pentru μ la fiecare iterație. **Formulele pe care le găsește k-means pentru *reprezentanții* μ sunt chiar formulele pentru centroizi.** Din păcate, minimul la care ajunge k-means este un **minim local**. Ex. 12/pag. 494 conține mai multe detalii legate de acest subiect.

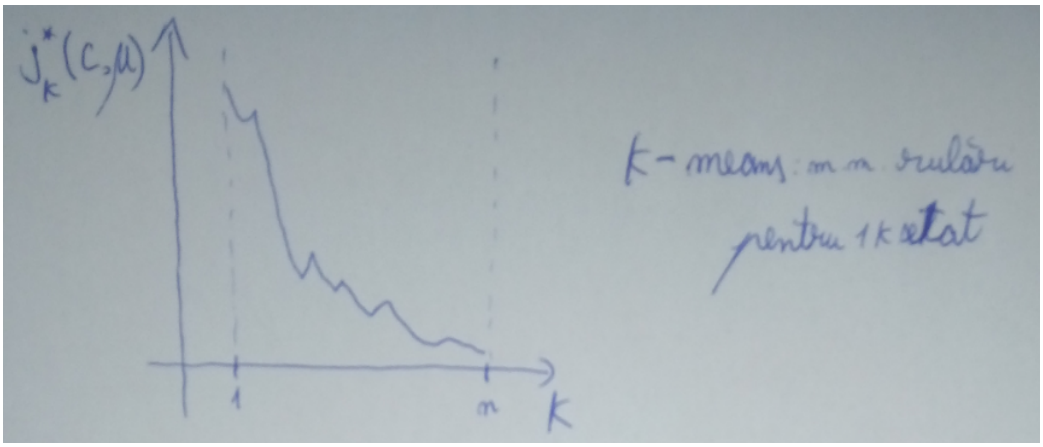
Să luăm un **exemplu de calcul pentru** $J_k(C, \mu)$.

Exemplu: $k = 2$, $X = (x_1, x_2, x_3)$
$C_1 = \{x_1, x_2\} \Leftrightarrow C(x_1) = C(x_2) = 1$,
$C_2 = \{x_3\} \Leftrightarrow C(x_3) = 2$,
$\mu = (\mu_1, \mu_2)$
Atunci: $J_2(C, \mu) = \|x_1 - \mu_1\|^2 + \|x_2 - \mu_1\|^2 + \|x_3 - \mu_2\|^2$ ■

Am spus că algoritmul k-means obține un **optim local** (și nu global) pentru criteriul $J_k(C, \mu)$. Din această cauză **este bine să restartăm algoritmul cu noi valori la inițializare** și să facem acest lucru de mai multe ori; astfel, la finalul fiecărei rulări obținem câte o valoare de minim local pentru criteriul $J_k(C, \mu)$; dintre toate aceste valori putem lua valoarea minimă pentru a fi mai aproape de optimul global. Dacă, pentru un k fixat, notăm cu $J_k^*(C, \mu)$ **valoarea minimă pentru** $J_k(C, \mu)$ **obținută după mai multe rulări**, atunci avem următoarele grafice:







Având în vedere pașii simpli din algoritmul k-means, se mai pot defini **alte două criterii** J, unul care depinde doar de μ și altul care depinde doar de C:

1.
$$J_k(\mu) = \sum_{i=1}^{n} \min_{j \in \{1,\dots,k\}} \|x_i - \mu_j\|^2$$

Exemplu: $k = 2$, $X = (x_1, x_2, x_3)$, $\mu = (\mu_1, \mu_2)$

Atunci:

$$\begin{aligned} J_2(\mu) = &\min(\|x_1 - \mu_1\|^2, \|x_1 - \mu_2\|^2) + \\ &+ \min(\|x_2 - \mu_1\|^2, \|x_2 - \mu_2\|^2) + \\ &+ \min(\|x_3 - \mu_1\|^2, \|x_3 - \mu_2\|^2) \blacksquare \end{aligned}$$

2.

$$J_k(C) = \sum_{i=1}^{n} \|x_i - \mu_j\|^2,$$

unde $\mu_j = \dfrac{\sum_{x_i \in X, C(x_i)=j} x_i}{\sum_{x_i \in X, C(x_i)=j} 1}$

Exemplu: $k = 2$, $X = (x_1, x_2, x_3)$,

$C_1 = \{x_1, x_2\} \Leftrightarrow C(x_1) = C(x_2) = 1$,

$C_2 = \{x_3\} \Leftrightarrow C(x_3) = 2$

Atunci:

$\mu_1 = \dfrac{x_1 + x_2}{2}$, $\mu_2 = x_3$

$$J_2(C) = \|x_1 - \mu_1\|^2 + \|x_2 - \mu_1\|^2 + \|x_3 - \mu_2\|^2 \blacksquare$$

Observație: $J_k(C^t, \mu^{t+1}) = J_k(C^t)$, $J_k(C^t, \mu^t) = J_k(\mu^t)$, unde C^a reprezintă k-partiția furnizată de k-means la iterația a, iar μ^a este k-configurația furnizată de k-means la iterația a.

Revenind la rezultatul de la ex. 12, avem faptul că

$$\underbrace{J_k(C^0, \mu^0)}_{J_k(\mu^0)} \geq \underbrace{J_k(C^0, \mu^1)}_{J_k(C^0)} \geq \underbrace{J_k(C^1, \mu^1)}_{J_k(\mu^1)} \geq \underbrace{J_k(C^1, \mu^2)}_{J_k(C^1)} \geq \dots$$

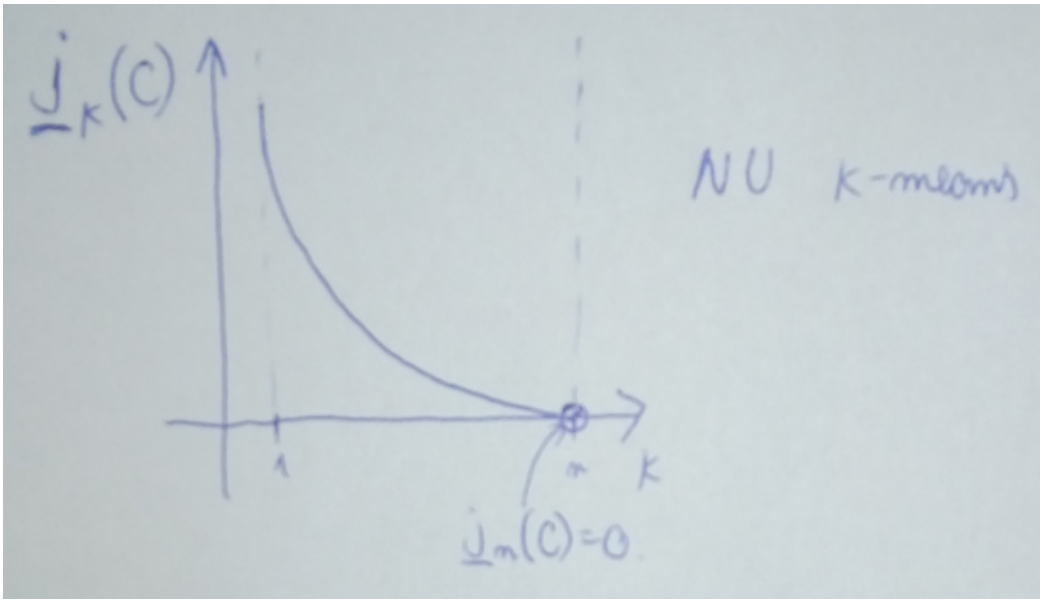
Astfel, se poate spune că algoritmul k-means nu minimiează doar $J_k(C, \mu)$, ci și $J_k(C)$, și $J_k(\mu)$.

Nu trebuie să știți toate aceste detalii, doar că în culegere când se vorbește despre criteriul J se poate referi la oricare din cele trei forme de mai sus ($J_k(C, \mu), J_k(\mu), J_k(C)$). De exemplu, la ex. 13/pag. 498, este vorba despre $J_k(C)$. De ce $J_k(C)$ este interesant? Pentru că putem merge prin toate cluterizările posibile C (care sunt în număr de k^n, unde k este numărul de clustere, iar n este numărul de instanțe) și putem lua minimul. Mai mult, observăm faptul că, *aproximativ*, **(mulțimea k-partițiilor lui X)** $\subseteq$ **(mulțimea (k+1)-partițiilor lui X)**.

De **exemplu**, dacă $X = (1, 2, 3)$, atunci:
1-partițiile lui X: $(\{1, 2, 3\})$
2-partițiile lui X:
$(\{\},\{1, 2, 3\})$
$(\{1\},\{2, 3\})$
$(\{2\},\{1, 3\})$
$(\{3\},\{1, 2\})$

Pentru că 1-partiția lui X o putem scrie și așa: $(\{\},\{1, 2, 3\})$, putem spune că (mulțimea 1-partițiilor lui X) $\subseteq$ (mulțimea 2-partițiilor lui X). Cu un raționament asemănător se demonstrează cazul general. ■

Din acest motiv, dacă notăm $\underline{J}_k(C) \overset{\text{not.}}{=}$ **minimul global al lui** $J_k(C)$ (mergând prin toate k-partițiile posibile ale lui X), vom avea următorul grafic:



2.2 k-medians

Să remarcăm faptul că în cadrul criteriului $J_k(C, \mu)$ am folosit **norma euclidiană** la pătrat. Ex. 42/pag. 546 schimbă această normă în norma cu $p = 1$ și nu mai ridică la pătrat norma:

$$J_k(C, \mu)_1 = \sum_{i=1}^{n} \|x_i - \mu_{C(x_i)}\|_1$$

Astfel, având dat un set de date $X = (x_1, \ldots, x_n)$ și un număr natural nenul k, trebuie să găsim o împărțire în k clustere a instanțelor și pentru fiecare cluster, un *reprezentant* astfel încât criteriul $J_k(C, \mu)_1$ să fie minimizat. În acest caz, dacă vom deriva un algoritm în aceeași manieră în care este derivat k-means (*coordinate descent*), vom da peste algoritmul k-medians. Astfel, **k-medians** reușește să găsească formule de actualizare pentru C și pentru μ la fiecare iterație. **Formulele pe care le găsește k-medians pentru *reprezentanții* μ nu mai sunt formulele pentru centroizi**, ci altele (pentru k-means, *reprezentantul* era media unor puncte; în k-medians, *reprezentantul* este mediana unor puncte). Din nou, minimul la care ajunge k-medians este un **minim local**.

2.3 Ex. 39

vezi ex. 39

Schemă de final

1. k-means - noțiuni de bază
2. k-means - algoritm de optimizare
   (a) criteriul J
      i. $J_k(C, \mu)$
      ii. $J_k^*(C, \mu)$
      iii. $J_k(\mu)$
      iv. $J_k(C)$
      v. $\underline{J}_k(C)$
   (b) k-medians
   (c) ex. 39