Sisteme de recunoaștere a formelor – Lab 6

Algoritmul de grupare K-means

1. Obiective

În această lucrare de laborator se va rezolva problema grupării unui set de puncte (*clustering*). Această sarcină face parte din învățare automată nesupervizată, în sensul că etichetele/clasele punctelor nu sunt cunoscute şi nici nu sunt necesare în procesul de învățare. Prin utilizarea unor metode adecvate se va identifica structura datelor și punctele similare vor fi grupate împreună.

2. Fundamente teoretice

Scopul metodelor de grupare (clusterizare) este de a partiționa o mulţime de obiecte in grupuri diferite, unde instanţele dintr-un grup sunt similare într-un anumit sens. Clusterizarea este folosită în multe domenii precum: învăţare cumputerizată, sisteme de recunoaștere a formelor, analiza imaginilor, bioinformatică, compresie, grafică.

Algoritmul k-means primește ca valori de intrare o listă de puncte $X = \{x_i, i = 1{:}\,n\}$. Fiecare punct este d-dimensional $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$. Obiectivul metodei k-means este gruparea punctelor în K mulțimi notate cu $S = \{S_k | k = 1{:}\,K\}$. Centroidul care reprezintă submulţimea k este notată cu m_k. Gruparea datelor trebuie realizată astfel încât să fie minimizată funcția obiectiv:

$$J(X,S) = \sum_{k=1}^{K} \sum_{x \in S_k} dist(x, m_k)$$

unde $dist(.,.)$ este distanţa Euclidiană în spațiul d-dimensional:

$$dist(x,y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

Aceasta este o problemă NP-hard, dar există anumite soluții aproximative prin care se obțin rezultate bune. Metoda lui Lloyd propune separarea problemei în două părți. Dacă se cunoaște modul în care punctele sunt partiționate se pot calcula centrele, dar nu se pot cunoaște partițiile dacă centrele grupurilor nu sunt cunoscute. Ideea este să se înceapă cu o mulțime aleatoare a centrelor grupurilor și apoi centrele să fie schimbate în mai multe iterații. Este demonstrat că algoritmul va găsi o partiționare care corespunde unui minim local al funcției *J*, dar care nu este însă întotdeauna minimul global [2].

Fie L funcția de apartenență pentru fiecare punct, deci $L(i) \in 1{:}\,K, i = 1{:}\,n$. Această funcţie returnează grupul din care face parte al *i*-lea punct. Se începe prin selectarea aleatoare a centrelor grupurilor din punctele din setul de date: $m_k = x_{r_k}$, unde r_k este o valoarea întreagă uniform distribuită între 1 şi n. Pentru a asigura convergența algoritmului se pot aplica tehnici mai complexe de inițializare. În [2] se definește metoda k-means++ bazată pe selectarea punctelor după o distribuție de probabilitate care penalizează punctele apropiate.

În continuare se aplică iterativ pașii de atribuire și de actualizare. Când apartenența punctelor la grupuri nu se mai modifică sau când se atinge un număr maxim de iterații, algoritmul se termină. Pașii metodei sunt prezentați în algoritmul următor:

Algoritmul K-means

Inițializare – Se selectează aleator *K* centre din lista punctelor de intrare. Fie r_k o variabilă aleatoare, întreagă și uniform distribuită în intervalul [1,*n*], atunci centrele inițiale sunt selectate ca:

$$m_k = x_{r_k}$$

Atribuire – Fiecare punct din lista de intrare este asociat cu centrul cel mai apropiat. Funcţia de apartenență va lua valoarea indexului celui mai apropiat centru:

$$L(i) = argmin_k dist(x_i, m_k)$$

Actualizare – Se recalculează centrele grupurilor pe baza funcției de apartenență. Noile centre ale grupurilor sunt calculate ca media punctelor din acel grup. În formula următoare se însumează toate punctele care aparțin grupului *k*, adică au funcția de apartenență $L(i) = k$.

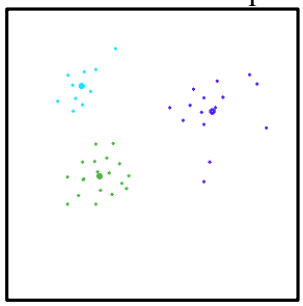
$$m_k = \frac{\sum_{L(i)=k} x_i}{\sum_{L(i)=k} 1} = \frac{\sum_{x \in S_k} x}{|S_k|}$$

Condiția de terminare – Dacă nu apare nici o schimbare în funcția de apartenență, atunci algoritmul poate fi oprit deoarece calculele viitoare nu vor produce nici o schimbare în valorile centrelor. De asemenea, se poate limita numărul maxim de iterații ale algoritmului. Dacă nici una din condițiile de mai sus nu este îndeplinită, atunci algoritmul continuă cu pasul de atribuire.

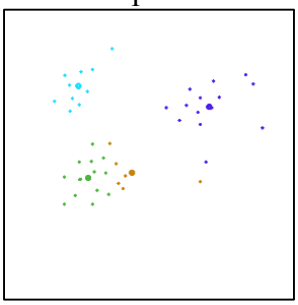
---

3. Exemple de rezultate

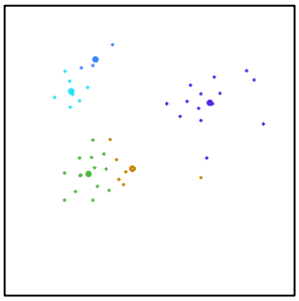
Pentru d=2, când algoritmul K-means este rulat pe un set de puncte 2D obţinem următoarele rezultate pe fişierul points2.bmp:



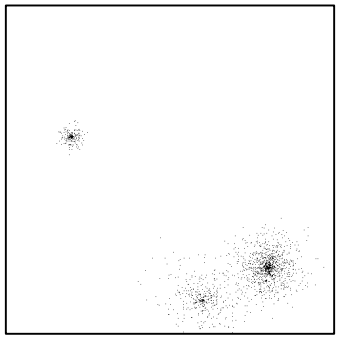
points2 - K = 3



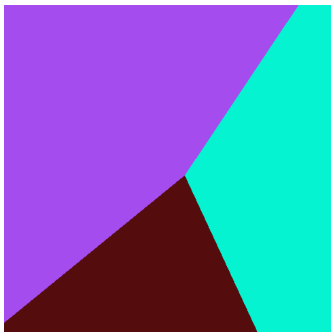
points2 - K = 4



points2 - K = 5



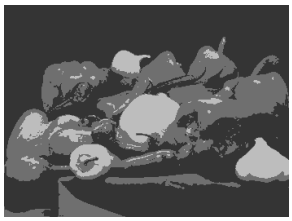
points4



points4 - mozaicare Voronoi K=3

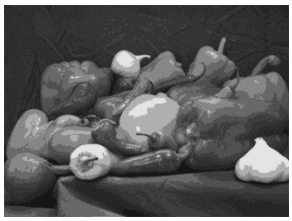
Pentru d=1, când algoritmul K-means este rulat pe imagini grayscale obţinem următoarele rezultate:

Imaginea originala



K=3

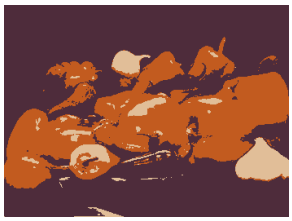
K=7



K=10

Pentru d=3, când algoritmul K-means este rulat pe imagini color obţinem următoarele rezultate:

Imaginea originala



K=3

K=7

K=10

4. Detalii de implementare

Generarea unei valori întregi aleatoare uniform distribuite din intervalul [*a, b*] (inclusiv):

```
#include <random>

default_random_engine gen;
uniform_int_distribution<int> distribution(a, b);
int randint = distribution(gen);
```

Crearea unei imagini color:

```
Mat img(height, width, CV_8UC3);
```

Construirea unui tablou de culori aleatoare pentru grupuri:

```
const int K = 3;
Vec3b colors[K];
for (int i = 0; i<K; i++)
     colors[i] = { (uchar)distribution(gen),
                   (uchar)distribution(gen),
                   (uchar)distribution(gen) };
```

Atribuirea unei culori colors[*k*] la poziția (*i*, *j*) în imagine:

```
img.at<Vec3b>(i,j) = colors[k];
```

5. Activitate practică

1. Implementaţi metoda de grupare K-means pe o listă de puncte *d* dimensionale. Aplicați iterativ algoritmul până când nu mai apare nici o modificare în funcția de apartenență sau până când numărul maxim de repetări ale algoritmului este atins. Numărul de grupuri *K* este specificat de utilizator în fiecare caz.
2. Aplicaţi K-means pe o mulţime de puncte din spaţiul 2D (imaginile points*.bmp) În acest caz *d=2*.
    a. Alegeţi culori în mod aleator pentru grupuri şi coloraţi punctele în funcţie de apartenenţa lor.
    b. Pentru o vizualizare mai bună coloraţi şi vecinătatea punctelor.
    c. Desenaţi mozaicarea Voronoi corespunzătoare centrelor grupurilor obţinute. Aceasta presupune colorarea fiecărui pixel din imagine (şi cei care aparțin fundalului) bazat pe care este centrul cel mai apropiat.
3. Aplicaţi K-means pe imagini grayscale. În acest caz se grupează intensităţile care apar în imagine. Duplicatele se păstrează, adică de exemplu, dacă intensitatea 0 apare de 10 ori, vom avea 10 puncte egale cu 0. În acest caz *d=1*.
    a. Recoloraţi imaginea în funcţie de intensitatea medie a fiecărui grup.
4. Aplicaţi K-means pe imagini color. În acest caz se grupează culorile care apar în imagine, adică componentele R, G, B . În acest caz *d=3*.
    a. Recoloraţi imaginea în funcţie de culoarea medie a fiecărui grup.
5. Opţional, implementați metoda de inițializare k-means++ din [3].

Referinţe

[1] Cluster analysis Wikipedia article - https://en.wikipedia.org/wiki/Cluster_analysis
[2] K-means Wikipedia article - https://en.wikipedia.org/wiki/K-means_clustering
[3] Arthur, David, and Sergei Vassilvitskii. "k-means++: The advantages of careful seeding." Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics, 2007.
[4] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. „Contour Detection and Hierarchical Image Segmentation", IEEE TPAMI, Vol. 33, No. 5, pp. 898-916, May 2011.
[5] Image segmentation dataset:
https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html