Stručni rad
Prihvaćeno 16. 9. 2016.

IVANČICA MIROŠEVIĆ

Algoritam k-sredina

k-means Algorithm

ABSTRACT

In this paper, k-means algorithm is presented. It is a heuristic algorithm for solving NP-hard optimisation problem of classifying a given data into clusters, with a number of clusters fixed apriori. The algorithm is simple and it's convergence is fast, what makes it widely used, despite its tendency of stoping in a local minimum and inability of recognizing clusters not separated by hyper-planes.

The method of the first variation as a tool for escaping from a local minimum is also presented in the paper.

Key words: k-means algorithm, clustering, first variation method

MSC2010: 65K10

Algoritam k-sredina

SAŽETAK

U članku je objašnjen algoritam k-sredina (k-means algorithm), heuristika koja rješava NP teški optimizacijski problem razvrstavanja podataka (točaka) u skupine (klastere) s unaprijed zadanim brojem skupina. Zbog jednostavnosti i brzine konvergencije, algoritam je u širokoj primjeni, unatoč tendenciji zapinjanja u lokalnom minimumu, te nemogućnosti prepoznavanja skupina koje nisu razdvojive hiperravninama.

U članku je također objašnjena i metoda prve varijacije, heuristika lokalnog traženja kojom algoritam "izvlačimo" iz lokalnog minimuma.

Ključne riječi: algoritam k-sredina, klasteriranje, metoda prve varijacije

1 Uvod

Osnovna ideja algoritma k-sredina je određivanje predstavnika k skupina, i pridruživanje svake točke skupini s najbližim predstavnikom tako da zbroj kvadrata udaljenosti točaka od predstavnika skupina kojima pripadaju bude minimalan. Drugim riječima, algoritam k-sredina generira skupine s minimalnom totalnom varijancom (najkompaktnije moguće skupine).

Nedostatak algoritma je u tome što se na izlazu dobiva samo stabilno, a ne nužno i optimalno rješenje. Odnosno, rješenje bitno ovisi o početnoj k-torci predstavnika skupina. Drugi nedostatak algoritma k-sredina je u tome što može prepoznati samo skupine odvojive hiperravninama.

2 Algoritam k-sredina

Neka je zadan skup točaka $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ u n dimenzionalnom Euklidskom prostoru $\mathbb{R}^n$. Cilj algoritma k-sredina je pronaći, za unaprijed zadani broj $k \geq 2$, optimalnu k-particiju skupa X, $\pi = \{C_1, C_2, ..., C_k\}$, odnosno razmjestiti m točaka skupa X u k skupina (podskupova, klastera) $C_1, C_2, ..., C_k$. Svakoj skupini C_i, $i = 1, \ldots, k$, pridružena je točka

$$\mathbf{c}_i = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x} \tag{1}$$

koju nazivamo predstavnikom skupine (predstavnik skupine ne mora biti element skupa X). Ovdje je $|C_i|$ kardinalnost od C_i, tj. broj točaka koje pripadaju skupini C_i. Algoritam k-sredina se zasniva na jednostavnoj činjenici da je optimalni izbor predstavnika skupine središte same skupine.

Svakoj particiji $\pi = \{C_1, C_2, ..., C_k\}$ skupa X pridružena je vrijednost ciljne funkcije

$$J = J(\pi) = \sum_{i=1}^{k} \sum_{x \in C_i} ||\mathbf{x} - \mathbf{c}_i||_2^2,$$

gdje su $\mathbf{c}_i$, $i = 1, \ldots, k$, definirani s (1). Algoritam k-sredina u nizu iteracija nastoji minimizirati vrijednost ciljne funkcije, odnosno pronaći particiju kojoj je pridružena najmanja vrijednost ciljne funkcije.

Ovdje je prikazana klasična verzija algoritma k-sredina koja koristi Euklidsku metriku na $\mathbb{R}^n$

$$d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2 = \left[\sum_{i=1}^{n} (x_i - y_i)^2 \right]^{\frac{1}{2}}, \ \mathbf{x}, \mathbf{y} \in \mathbb{R}^n. \tag{2}$$

Inače, algoritam za particioniranje može se lako prilagoditi tako da koristi kosinusnu sličnost među vektorima kao

metriku,

$$d(\mathbf{x},\mathbf{y}) = \frac{\mathbf{x}^T\mathbf{y}}{||\mathbf{x}||_2||\mathbf{y}||_2}, \ \mathbf{x},\mathbf{y} \in \mathbb{R}^n, \tag{3}$$

što može biti prikladniji izbor za particioniranje podataka od Euklidske metrike (vektori se prethodno normaliziraju, pa se ovakav algoritam k-sredina naziva sfernim algoritmom k-sredina).

Prije pokretanja algoritma k-sredina potrebno je na neki način zadati početne vrijednosti $\{\mathbf{c}_i\}$ predstavnika skupina, na primjer nasumičnim izborom k točaka iz zadanog skupa točaka X.

Algoritam k-sredina je iterativni algoritam koji ponavlja dva osnovna koraka dok se ne zadovolji neki kriterij konvergencije. U prvom koraku ("*pridruživanje*") svaka točka pridružuje se najbližem predstavniku (određuje se particija). U drugom koraku ("*prepravljanje*") skup predstavnika se prepravlja - za nove predstavnike se uzimaju središta skupina definiranih u koraku "pridruživanje". Algoritam se zaustavlja kada se vrijednost ciljne funkcije prestane smanjivati.

Algoritam 1 *Algoritam k-sredina*

1. ***Inicijalizacija.***
   Zadaj početni skup predstavnika $\{\mathbf{c}_i^0\}_{i=1}^k$. *Postavi brojač* $l = 0$;

2. ***Pridruživanje.***
   Za svaku točku $\mathbf{x} \in X \subset \mathbb{R}^n$ *odredi* $sk(\mathbf{x}) \in \{1,\dots,k\}$ *(redni broj skupine točke* $\mathbf{x}$*) takav da je*

   $$||\mathbf{c}^l_{sk(\mathbf{x})} - \mathbf{x}||_2 = \min_{j\in\{1,2,\dots,k\}} ||\mathbf{c}^l_j - \mathbf{x}||_2.$$

   Definiraj skupine

   $$C_i^{(l+1)} = \{\mathbf{x} : sk(\mathbf{x}) = i\}, \ i = 1,\dots,k.$$

3. ***Prepravljanje.***
   Izračunaj predstavnike novih skupina definiranih u 2. koraku:

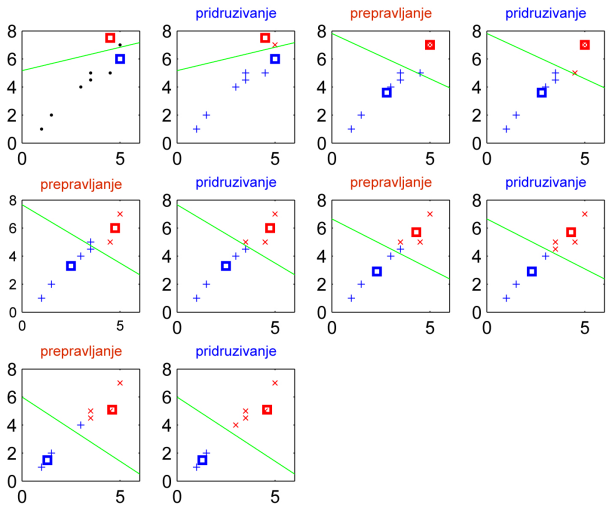
   $$\mathbf{c}_i^{l+1} = \frac{1}{n_i} \sum_{\mathbf{x}\in C_i^{(l+1)}} \mathbf{x},$$

   gdje je $n_i = |C_i^{(l+1)}|$.
   Izračunaj vrijednost ciljne funkcije

   $$J^{(l+1)} = \sum_{i=1}^k \sum_{x\in C_i^{(l+1)}} ||\mathbf{x} - \mathbf{c}_i^{l+1}||_2^2;$$

   Stavi $l = l+1$*; Ponavljaj korake 2. i 3. sve dok se vrijednost J ne prestane smanjivati.*

Primjer 1 Na Slici 1 dan je primjer skupa od sedam točaka $X = \{\mathbf{x}_1,\dots,\mathbf{x}_7\} = \{(1,1),(1.5,2),(3,4),(3.5,4.5),(3.5,5),(4.5,5),(5,7)\}$, s inicijalnim skupom predstavnika $\{\mathbf{c}_1,\mathbf{c}_2\} = \{(4.5,7.5),(5,6)\}$ za kojeg algoritam k-sredina u pet iteracija daje optimalnu particiju $\pi = \{C_1,C_2\}$, $C_1 = \{\mathbf{x}_1,\mathbf{x}_2\}$ i $C_2 = \{\mathbf{x}_3,\dots,\mathbf{x}_7\}$.



Slika 1: *Algoritam k-sredina pronalazi optimalnu particiju za zadane inicijalne predstavnike u 5 iteracija. Crvenim znakom 'x' označene su točke prve skupine, a plavim znakom '+' točke druge skupine. Kvadrati označavaju predstavnike skupina.*

Teorem 1 ([3]) *Vrijednost J ciljne funkcije algoritma k-sredina monotono se smanjuje.*

Dokaz. Označimo s $J^{(l)}$ vrijednost ciljne funkcije u l-tom ponavljanju. Vrijedi:

$$J^{(l)} = \sum_{i=1}^k \sum_{x\in C_i^{(l)}} ||\mathbf{x} - \mathbf{c}_i^l||_2^2 \geq \sum_{i=1}^k \sum_{x\in C_i^{(l)}} ||\mathbf{x} - \mathbf{c}^l_{sk(\mathbf{x})}||_2^2 =$$

$$= \sum_{i=1}^k \sum_{x\in C_i^{(l+1)}} ||\mathbf{x} - \mathbf{c}^l_{sk(\mathbf{x})}||_2^2 \geq \sum_{i=1}^k \sum_{x\in C_i^{(l+1)}} ||\mathbf{x} - \mathbf{c}_i^{l+1}||_2^2 = J^{(l+1)}.$$

Druga nejednakost slijedi iz činjenice da vektor središta skupine minimizira kvadratnu devijaciju. □

Algoritam se zaustavlja ako je $J^{(l)} = J^{(l+1)}$.

Primijetimo da su rezultat algoritma k-sredina skupovi (klasteri) odijeljeni hiperravninama. Na primjer, pogledajmo slučaj za $k = 2$:

$\mathbf{x} \in C_1$ akko $||\mathbf{x} - \mathbf{c}_1||_2^2 \leq ||\mathbf{x} - \mathbf{c}_2||_2^2$.

Skup točaka koje su jednako udaljene od $\mathbf{c}_1$ i $\mathbf{c}_2$ dijeli točke na ovaj način. A taj skup točaka je hiperravnina okomita na $\mathbf{c}_1 - \mathbf{c}_2$ koja prolazi polovištem kojeg određuje $\mathbf{c}_1 - \mathbf{c}_2$:

$$(\mathbf{x} - \mathbf{c}_1)^T (\mathbf{x} - \mathbf{c}_1) = (\mathbf{x} - \mathbf{c}_2)^T (\mathbf{x} - \mathbf{c}_2)$$
$$\Longrightarrow 2(\mathbf{c}_1 - \mathbf{c}_2)^T \mathbf{x} + (\mathbf{c}_2^T \mathbf{c}_2 - \mathbf{c}_1^T \mathbf{c}_1) = 0.$$

Općenito, algoritam k-sredina razdjeljuje točke skupom hiperravnina. Rezultirajuća particija prostora odgovara tzv. Voronojevom dijagramu skupa predstavnika.

Definicija 1 *Neka je zadan skup* $S = \{\mathbf{p}_i \in \mathbb{R}^n, i = 1, \ldots, q\}$ *za neki* $q \in \mathbb{N}$. *Voronojeva ćelija (engl. Voronoi cell) pridružena točki* p_i *je skup*

$$V(\mathbf{p}_i) = \{\mathbf{x} \in \mathbb{R}^n : ||\mathbf{x} - \mathbf{p}_i||_2 \leq ||\mathbf{x} - \mathbf{p}_j||_2, \forall j = 1, \ldots, q\}.$$

Voronojev dijagram skupa S je unija Voronojevih ćelija, odnosno

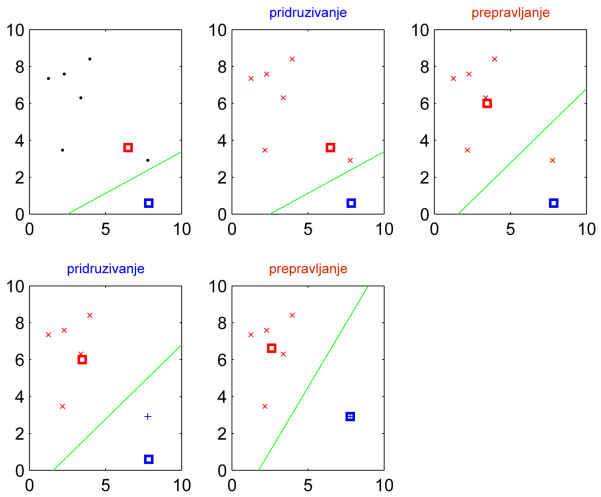
$$V(S) = \bigcup_{i \in \{1,\ldots,q\}} V(\mathbf{p}_i).$$

Primjer 2 Na Slici 1 zelenim crtama naznačeni su pravci kojima algoritam k-sredina razdjeljuje zadane točke na dvije skupine. Dvije rezultirajuće poluravnine predstavljaju dvije Voronojeve ćelije pridružene predstavnicima skupina.

Prilikom izvršavanja algoritma k-sredina može se dogoditi da neki predstavnik ostane bez pridruženih mu točaka (ako su sve točke bliže ostalim predstavnicima). Sama heuristika k-sredina ne govori o tome kako postupiti u situaciji kada skupina u nekom koraku ostane prazna. Uobičajene strategije u praksi su:

- premiještanje predstavnika ispražnjene skupine slučajnim izborom ili proizvoljno
- uzimanje točke koja je najudaljenija od svog predstavnika za novu jednočlanu skupinu
- zadržavanje predstavnika (ostavlja se mogućnost da skupina ponovo primi točke);
- brisanje predstavnika skupine koja je ostala prazna.

Primjer 3 Slika 2 prikazuje primjer particioniranja algoritmom k-sredina u kojem jedan predstavnik ostaje bez pridruženih točaka. Sam predstavnik je zadržan, i već u sljedećem koraku točke su razdijeljene na dvije skupine. Primijetimo da bi u ovom primjeru pomicanje predstavnika u točku najudaljeniju od pridruženog predstavnika rezultiralo većim brojem iteracija.



Slika 2: *Primjer inicijalizacije algoritma k-sredina koja rezultira praznom skupinom. Predstavnik je zadržan, i u sljedećem koraku skupina prima točke.*

3 Dvojnost ciljne funkcije algoritma k-sredina

Pokazat ćemo da algoritam k-sredina smanjivanjem vrijednosti ciljne funkcije automatski povećava prosječnu kvadratnu udaljenost među parovima predstavnika skupina, a smanjuje prosječni zbroj kvadrata razlika udaljenosti parova točaka unutar skupina.

Za $\mathbf{x} \in \mathbb{R}^n$ vrijedi

$$\operatorname{tr}\left(\sum \mathbf{x}\mathbf{x}^T\right) = \sum \mathbf{x}^T \mathbf{x},$$

gdje je

$$\operatorname{tr} A = \sum_{i=1}^{n} a_{ii} \quad \text{za} \quad A = [a_{ij}] \in \mathbb{R}^{n \times n}.$$

Stoga se ciljna funkcija algoritma k-sredina može shvatiti kao $\operatorname{tr}(S_W)$, gdje je

$$S_W = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{c}_i)(\mathbf{x} - \mathbf{c}_i)^T.$$

Matrica S_W naziva se matricom rasipanja unutar skupina (engl. within-class scatter matrix).

Matrica ukupnog rasipanja je dana izrazom

$$S_T = \sum_{\mathbf{x}\in X} (\mathbf{x}-\mathbf{c})(\mathbf{x}-\mathbf{c})^T,$$

gdje je $\mathbf{c}$ srednja vrijednost cijelog skupa podataka. Može se dokazati da je $S_T = S_W + S_B$, gdje je S_B matrica rasipanja između skupina definirana izrazom

$$S_B = \sum_{i=1}^{k} |C_i| (\mathbf{c}_i-\mathbf{c})(\mathbf{c}_i-\mathbf{c})^T.$$

Propozicija 1 *Vrijedi*

$S_T = S_W + S_B$.

Dokaz. Definirajmo za proizvoljni vektor $\mathbf{a}$

$$\begin{aligned} S_T(\mathbf{a}) &= \sum_{\mathbf{x}} (\mathbf{x}-\mathbf{a})(\mathbf{x}-\mathbf{a})^T \\ &= \sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{x}-\mathbf{a}+\mathbf{c}_i-\mathbf{c}_i)(\mathbf{x}-\mathbf{a}+\mathbf{c}_i-\mathbf{c}_i)^T \\ &= \sum_i \sum_{\mathbf{x}\in C_i} ((\mathbf{x}-\mathbf{c}_i)+(\mathbf{c}_i-\mathbf{a}))\left((\mathbf{x}-\mathbf{c}_i)^T+(\mathbf{c}_i-\mathbf{a})^T\right) \\ &= \sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{x}-\mathbf{c}_i)(\mathbf{x}-\mathbf{c}_i)^T + \sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{x}-\mathbf{c}_i)(\mathbf{c}_i-\mathbf{a})^T \\ &\quad + \sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{c}_i-\mathbf{a})(\mathbf{x}-\mathbf{c}_i)^T + \sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{c}_i-\mathbf{a})(\mathbf{c}_i-\mathbf{a})^T \end{aligned}$$

Vrijedi

$$\begin{aligned} S_T(\mathbf{a}) &= \sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{x}-\mathbf{c}_i)(\mathbf{x}-\mathbf{c}_i)^T + \sum_i |C_i|(\mathbf{c}_i-\mathbf{a})(\mathbf{c}_i-\mathbf{a})^T \\ &= S_W + S_B(\mathbf{a}), \end{aligned}$$

jer je $\sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{c}_i-\mathbf{a})(\mathbf{x}-\mathbf{c}_i)^T = 0 = \sum_i \sum_{\mathbf{x}\in C_i} (\mathbf{x}-\mathbf{c}_i)(\mathbf{c}_i-\mathbf{a})^T$.

Prethodna relacija je točna za proizvoljni vektor $\mathbf{a}$, pa je točna i za $\mathbf{a} = \mathbf{c}$, što dokazuje tvrdnju. □

Možemo, također, pokazati da je

$$\begin{aligned} S_T &= \frac{1}{2m} \sum_{i=1}^{m} \sum_{j=1}^{m} (\mathbf{x}_i-\mathbf{x}_j)(\mathbf{x}_i-\mathbf{x}_j)^T \\ S_W &= \sum_{i=1}^{k} \frac{1}{2|C_i|} \sum_{\mathbf{x}\in C_i} \sum_{\mathbf{y}\in C_i} (\mathbf{x}-\mathbf{y})(\mathbf{x}-\mathbf{y})^T \\ S_B &= \frac{1}{2m} \sum_{i=1}^{k} \sum_{j=1}^{k} |C_i||C_j| (\mathbf{c}_i-\mathbf{c}_j)(\mathbf{c}_i-\mathbf{c}_j)^T. \end{aligned}$$

Budući da je $\operatorname{tr}(S_W)$ ciljna funkcija algoritma k-sredina, on nastoji minimizirati prosječni zbroj kvadrata razlika parova točaka unutar skupina.

Prema tome, budući da je $\operatorname{tr}(S_T) = \operatorname{tr}(S_W) + \operatorname{tr}(S_B)$, a $\operatorname{tr}(S_T)$ je fiksan, ciljnu funkciju algoritma k-sredina možemo promatrati kao

$$\begin{aligned} \max \operatorname{tr}(S_B) &= \max \sum_i |C_i| \cdot ||\mathbf{c}_i-\mathbf{c}||_2^2 \\ &= \max \frac{1}{2m} \sum_i \sum_j |C_i||C_j| \cdot ||\mathbf{c}_i-\mathbf{c}_j||_2^2, \end{aligned}$$

što znači da algoritam k-sredina pokušava maksimizirati prosječnu kvadratnu udaljenost među parovima predstavnika skupina.

4 O međama broja iteracija algoritma k-sredina u jednodimenzionalnom modelu

Računsku složenost algoritma k-sredina po jednoj iteraciji na m-članom skupu $X \subset \mathbb{R}^n$ možemo razložiti po koracima:

- U koraku "*pridruživanje*" algoritam k-sredina mk puta računa udaljenosti, za što mu treba $3mkn$ osnovnih operacija (zbrajanja, množenja ili uspoređivanja). Za pronalaženje najbližeg predstavnika treba mu mk operacija. Prema tome, složenost pridruživanja je $O(mkn)$.

- U koraku "*prepravljanje*" algoritam k-sredina izvršava mn zbrajanja i kn dijeljenja. Složenost prepravljanja je $O(mn)$ $(k \leq m)$.

- Za računanje vrijednosti ciljne funkcije algoritmu k-sredina treba m zbrajanja udaljenosti izračunatih u prvom koraku, pa je složenost računanja vrijednosti ciljne funkcije $O(m)$.

Prema tome, složenost algoritma k-sredina je $O(mknt)$, gdje je t broj iteracija. Sam broj iteracija može varirati od nekoliko do nekoliko tisuća, ovisno o broju i distribuciji točaka, te o broju skupina, i još uvijek nije u potpunosti teoretski razjašnjen broj iteracija koje algoritam k-sredina može izvršiti u najgorem slučaju. U [4] je dokazano da različitih Voronojevih dijagrama zadanih s k centara, koji m-člani skup $X \in \mathbb{R}^n$ dijele na k podskupova, ima najviše $O\left(m^{kn}\right)$, što predstavlja i trivijalnu gornju među za broj iteracija algoritma k-sredina. Međutim, činjenica da u uobičajenim aplikacijama k može biti veličine nekoliko stotina, i da algoritam k-sredina u praksi relativno brzo konvergira, čini ovu među beznačajnom.

5 Prva varijacija - algoritam za profinjenje rezultata algoritma *k*-sredina

Veliki nedostatak algoritma *k*-sredina je u tome što može "zapeti" u lokalnom minimumu i rezultirati lošom particijom. Navest ćemo primjer kada loša inicijalizacija algoritma rezultira particijom koja očigledno nije optimalna. Prva varijacija je metoda lokalnog traženja kojom se omogućava izbjegavanje lokalnog minimuma.

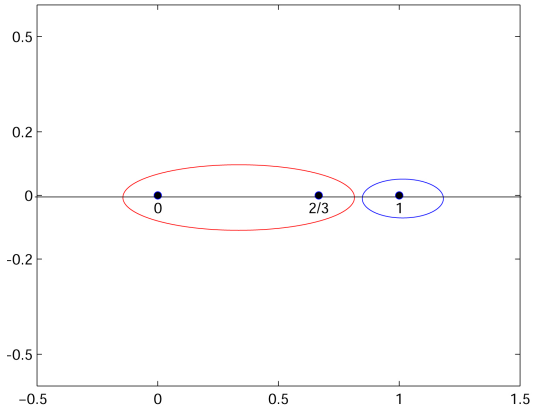
Inače, u praksi se pokazalo da se nedostatci algoritma *k*-sredina najčešće očituju kod malih skupova točaka (do otprilike 200 točaka) [2].

Primjer 4 Neka je zadan skup $X = \{x_1, x_2, x_3\}$, $x_1 = 0$, $x_2 = \frac{2}{3}$, $x_3 = 1$ i inicijalni par predstavnika $c_1^0 = \frac{1}{3}$, $c_2^0 = 1$ (Slika 3). To znači da je u početnoj particiji $C_1^0 = \{x_1, x_2\}$ i $C_2^0 = \{x_3\}$ i vrijedi

$$\begin{aligned} J\left(\{C_1^0, C_2^0\}\right) &= \sum_{i=1}^{2} \sum_{x \in C_i^0} ||\mathbf{x} - \mathbf{c}_i^0||_2^2 \\ &= ||\mathbf{x}_1 - \mathbf{c}_1^0||_2^2 + ||\mathbf{x}_2 - \mathbf{c}_1^0||_2^2 + ||\mathbf{x}_3 - \mathbf{c}_2^0||_2^2 \\ &= \left(\frac{1}{3}\right)^2 + \left(\frac{1}{3}\right)^2 + 0 = \frac{2}{9}. \end{aligned}$$

Algortam *k*-sredina neće promijeniti ovakvu početnu particiju jer je $||x_2 - c_1^0||_2^2 = ||x_2 - c_2^0||_2^2$, iako za particiju $\{C_1^1, C_2^1\}$, $C_1^1 = \{x_1\}$ i $C_2^1 = \{x_2, x_3\}$, odnosno za $c_1^1 = 0$, $c_2^1 = \frac{5}{6}$, vrijedi

$$\begin{aligned} J\left(\{C_1^1, C_2^1\}\right) &= \sum_{i=1}^{2} \sum_{x \in C_i^1} ||\mathbf{x} - \mathbf{c}_i^1||_2^2 \\ &= ||\mathbf{x}_1 - \mathbf{c}_1^1||_2^2 + ||\mathbf{x}_2 - \mathbf{c}_2^1||_2^2 + ||\mathbf{x}_3 - \mathbf{c}_2^1||_2^2 \\ &= 0 + \left(\frac{1}{6}\right)^2 + \left(\frac{1}{6}\right)^2 = \frac{1}{18}. \end{aligned}$$



Slika 3: *Primjer početne particije koju algoritam k-sredina ne mijenja, iako nije optimalna.*

Problem zaustavljanja algoritma *k*-sredina u lokalnom minimumu može se djelomično riješiti algoritmom prve varijacije, primjerom heuristike lokalnog traženja.

Definicija 2 *Prva varijacija particije* $\pi = \{C_1, \ldots, C_k\}$ *skupa X je particija* $\pi' = \{C_1', \ldots, C_k'\}$*, koja se dobije pomicanjem jedne točke* $\mathbf{x} \in X$ *iz skupine* $C_i \in \pi$ *u skupinu* $C_j \in \pi$*. Skup svih prvih varijacija particije* $\pi = \{C_1, \ldots, C_k\}$ *označavamo s* $\mathcal{V}(\pi)$*.*

Među svim elementima skupa $\mathcal{V}(\pi)$ tražimo particiju s najmanjom vrijednošću ciljne funkcije.

Definicija 3 *Particija* π^* *je prva varijacija particije* π *skupa X takva da je za svaku prvu varijaciju* π' *skupa X*

$$J(\pi^*) \leq J(\pi').$$

Particija π^* *zove se sljedeća prva varijacija.*

Algoritam prve varijacije generira niz particija $\pi^{(l)} = \{C_1^{(l)}, \ldots, C_k^{(l)}\}$, $l \geq 0$, takav da je $\pi^{(l+1)} = \pi^{(l)*}$, $l = 0, 1, \ldots$

Promotrimo razliku između iteracije algoritma *k*-sredina i iteracije algoritma prve varijacije. Neka je zadana bipartcija $\pi = \{Z, Y\}$ skupa $X \subset \mathbb{R}^n$, pri čemu je $Z = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ i $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$. Želimo utvrditi treba li jedan vektor, npr. $\mathbf{z}_n$, premjestiti iz Z u Y. Definirajmo potencijalne nove skupine sa

$$Z^- = \{\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}\} \quad \text{i} \quad Y^+ = \{\mathbf{y}_1, \ldots, \mathbf{y}_m, \mathbf{z}_n\}.$$

Algoritam *k*-sredina provjerava vrijednost

$$\Delta_{km} = ||\mathbf{z}_n - \mathbf{c}(Y)||_2^2 - ||\mathbf{z}_n - \mathbf{c}(Z)||_2^2. \tag{4}$$

Ako je $\Delta_{km} < 0$, algoritam *k*-sredina pomiče $\mathbf{z}_n$ iz Z u Y. Inače $\mathbf{z}_n$ ostaje u Z.

Algoritam prve varijacije, međutim, provjerava stvarnu promjenu u vrijednosti ciljne funkcije.

$$\begin{aligned}
\Delta_{pv} &= \sum_{i=1}^{m} ||\mathbf{y}_i - \mathbf{c}\left(Y^+\right)||_2^2 + ||\mathbf{z}_n - \mathbf{c}\left(Y^+\right)||_2^2 \\
&\quad + \sum_{i=1}^{n} ||\mathbf{z}_i - \mathbf{c}\left(Z^-\right)||_2^2 - ||\mathbf{z}_n - \mathbf{c}\left(Z^-\right)||_2^2 \\
&\quad - \sum_{i=1}^{m} ||\mathbf{y}_i - \mathbf{c}\left(Y\right)||_2^2 - \sum_{i=1}^{n} ||\mathbf{z}_i - \mathbf{c}\left(Z\right)||_2^2 \\
&= m||\mathbf{c}(Y) - \mathbf{c}\left(Y^+\right)||_2^2 + n||\mathbf{c}(Z) - \mathbf{c}\left(Z^-\right)||_2^2 \\
&\quad + ||\mathbf{z}_n - \mathbf{c}\left(Y^+\right)||_2^2 - ||\mathbf{z}_n - \mathbf{c}\left(Z^-\right)||_2^2 \\
&= m||\frac{\sum_{i=1}^{m}\mathbf{y}_i}{m} - \frac{\sum_{i=1}^{m}\mathbf{y}_i + \mathbf{z}_n}{m+1}||_2^2 + n||\frac{\sum_{i=1}^{n}\mathbf{z}_i}{n} - \frac{\sum_{i=1}^{n}\mathbf{z}_i - \mathbf{z}_n}{n-1}||_2^2 \\
&\quad + ||\mathbf{z}_n - \frac{\sum_{i=1}^{m}\mathbf{y}_i + \mathbf{z}_n}{m+1}||_2^2 - ||\mathbf{z}_n - \frac{\sum_{i=1}^{n}\mathbf{z}_i - \mathbf{z}_n}{n-1}||_2^2 \\
&= m||\frac{\sum_{i=1}^{m}\mathbf{y}_i - m\mathbf{z}_n}{m(m+1)}||_2^2 + n||\frac{n\mathbf{z}_n - \sum_{i=1}^{n}\mathbf{z}_i}{n(n-1)}||_2^2 \\
&\quad + ||\frac{m\mathbf{z}_n - \sum_{i=1}^{m}\mathbf{y}_i}{m+1}||_2^2 - ||\frac{n\mathbf{z}_n - \sum_{i=1}^{n}\mathbf{z}_i}{n-1}||_2^2 \\
&= \left[\frac{m}{(m+1)^2} + \frac{m^2}{(m+1)^2}\right]||\mathbf{z}_n - \mathbf{c}(Y)||_2^2 \\
&\quad + \left[\frac{n}{(n-1)^2} - \frac{n^2}{(n-1)^2}\right]||\mathbf{z}_n - \mathbf{c}(Z)||_2^2 \\
&= \frac{m}{m+1}||\mathbf{z}_n - \mathbf{c}(Y)||_2^2 - \frac{n}{n-1}||\mathbf{z}_n - \mathbf{c}(Z)||_2^2 \qquad (5)
\end{aligned}$$

Razlika između (4) i (5),

$$\Delta_{km} - \Delta_{pv} = \frac{1}{m+1}||\mathbf{z}_n - \mathbf{c}(Y)||_2^2 + \frac{1}{n-1}||\mathbf{z}_n - \mathbf{c}(Z)||_2^2 \geq 0$$

je zanemariva kada su skupine Z i Y velike. Međutim, $\Delta_{km} - \Delta_{pv}$ može postati bitna kod malih skupina.

Primjer 5 Za $\mathbf{z}_n = \mathbf{x}_2$ iz primjera 4 je $\Delta_{km} = 0$, a $\Delta_{pv} = -\frac{3}{18}$ i to je razlog zašto algoritam k-sredina propušta optimalnu particiju $\{C_1^1, C_2^1\}$.

Algoritam 2 *Algoritam prve varijacije*

1. *Zadaj početnu particiju* $\pi^{(0)} = \{C_1^{(0)}, \ldots, C_k^{(0)}\}$. *Postavi brojač iteracija* $l = 0$.
2. *Generiraj sljedeću prvu varijaciju* $\pi^{(l)*}$.
   Ako je $J\left(\pi^{(l)*}\right) - J\left(\pi^{(l)}\right) < 0$, *postavi* $\pi^{(l+1)} = \pi^{(l)*}$, *povećaj* l *za* 1, *i vrati se na korak* 2.
3. *Stani.*

Prilikom računanja sljedeće prve varijacije algoritam izvršava $3mkn$ operacija za računanje udaljenosti, $3mk$ operacija za računanje vrijednosti ciljne funkcije za sve prve varijacije particije, te mk operacija za određivanje sljedeće prve varijacije, pa je složenost jedne iteracije jednaka složenosti iteracije algoritma k-sredina $(O(mkn))$. Međutim, promjene vrijednosti ciljne funkcije su u svakoj iteraciji jako male jer se pomiče samo jedna točka, dok algoritam k-sredina daje značajnija poboljšanja po iteraciji. Stoga pogledajmo kombinaciju algoritma k-sredina i algoritma prve varijacije:

Algoritam 3 *Algoritam k-sredina poboljšan algoritmom prve varijacije*

1. *Zadaj početnu particiju* $\pi^{(0)} = \{C_1^{(0)}, \ldots, C_k^{(0)}\}$. *Postavi brojač iteracija* $l = 0$.
2. *Generiraj sljedeću particiju* $\pi^{(l)'}$ *algoritmom k-sredina.*
   Ako je $J\left(\pi^{(l)'}\right) - J\left(\pi^{(l)}\right) < 0$, *postavi* $\pi^{(l+1)} = \pi^{(l)'}$, *povećaj* l *za* 1, *i vrati se na korak* 2.
3. *Generiraj sljedeću prvu varijaciju* $\pi^{(l)*}$.
   Ako je $J\left(\pi^{(l)*}\right) - J\left(\pi^{(l)}\right) < 0$, *postavi* $\pi^{(l+1)} = \pi^{(l)*}$, *povećaj* l *za* 1, *i vrati se na korak* 2.
4. *Stani.*

Algoritam 3 alternira između dviju faza:

(a) algoritma k-sredina

(b) algoritma prve varijacije.

U trećem koraku pomiče se samo jedna točka iz jedne skupine u drugu ako to pomicanje rezultira smanjujem vrijednost ciljne funkcije. Niz koraka prve varijacije omogućava izbjegavanje lokalnog minimuma, nakon čega nove iteracije algoritma k-sredina mogu nastaviti brže smanjivati vrijednost ciljne funkcije. Ova ping-pong strategija daje algoritam za profinjenje skupina, koji često poboljšava sam algoritam k-sredina, a računski nije prezahtjevan (složenost je još uvijek $O(mkn)$). Naime, udaljenosti točaka od svih predstavnika potrebne za generiranje sljedeće prve varijacije u trećem koraku već su izračunate u drugom koraku, pa se ne moraju ponovo računati.

Međutim, ni ovako poboljšan algoritam ne daje uvijek optimalno rješenje.

U ovom radu je korištena MatLabova *kmeans* funkcija. Ona se zasniva na dvofaznom iterativnom algoritmu čija

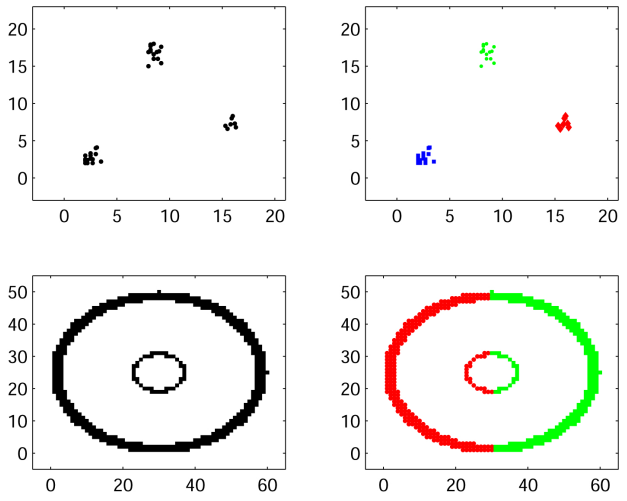
prva faza odgovara klasičnom algoritmu *k*-sredina. U drugoj fazi se pojedinačno premještaju sve točke čije premještanje rezultira smanjenjem ciljne funkcije (zbroja udaljenosti točaka od predstavnika skupina), nakon čega se ponovo računa skup predstavnika. Svaka iteracija se u drugoj fazi sastoji od jednog prolaska kroz sve zadane točke. Ovo je varijanta algoritma prve varijacije. Pritom se samo u prvoj iteraciji prve faze računaju sve udaljenosti točaka od predstavnika. U ostalim iteracijama udaljenosti se računaju samo za točke odnosno predstavnike koji su se pomicali. Budući da se broj pomaknutih točaka relativno mali nakon prvih par iteracija, ovim se bitno smanjuje vrijeme izvršavanja algoritma.

MatLabova *kmeans* funkcija dopušta unos vlastitog inicijalnog skupa predstavnika, ili ga sama računa na osnovi ulaznog skupa točaka na jedan od tri ponuđena načina:

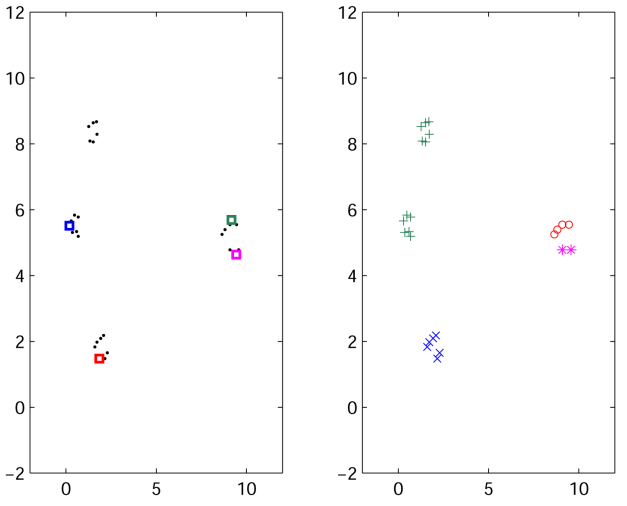
- "sample" - nasumičnim izborom bira *k* predstavnika iz ulaznog skupa točaka (uobičajeni način)
- "uniform" - nasumičnim izborom bira *k* uniformno distribuiranih predstavnika iz ulaznog skupa točaka
- "cluster" - particionira 10% nasumce izabranih točaka i dobivene predstavnike uzima za inicijalne predstavnike cijelog skupa točaka. U uvodnom particioniranju koristi "sample" za inicijalizaciju.

Primjer 6 Na Slici 4 su dva primjera particioniranja algoritmom *k*-sredina. U prvom slučaju algoritam je u 100 pokretanja sa "sample" inicijalizacijom 70 puta dao očiglednu optimalnu particiju. U primjeru s dva koncentrična prstena optimalne skupine nisu linearno odvojive, pa algoritam *k*-sredina daje najbolje što može - dijeli svaki prsten na dva dijela.

Primjer 7 Na Slici 5 (lijevo) je primjer skupa točaka koje su prirodno podijeljene na četiri skupine, te početni skup predstavnika. U ovom slučaju ni algoritam *k*-sredina ni algoritam prve varijacije ne daju optimalnu particiju.



Slika 4: *Rezultat particioniranja točaka MatLabovom* kmeans *funkcijom. Bojom je istaknuta pripadnost točaka skupinama.*



Slika 5: *Primjer inicijalizacije skupa predstavnika (kvadratići) za koju i algoritam k-sredina i algoritam prve varijacije daju lošu particiju. Lijevo su neparticionirane, a desno particionirane točke. Bojom i znakom je istaknuta pripadnost točaka skupinama.*

Literatura

[1] K. Alsabti, S. Ranka, V. Singh, An Efficient K-Means Clustering Algorithm, *IPPS/SPDP Workshop on High Performance Data Mining*, Orlando, Florida, 1998.

[2] I. Dhillon, Y. Guan, J. Kogan, Iterative Clustering of High Dimensional Text Data Augmented by Local Search, *Proceedings of the 2nd IEEE International Conference on Data Mining*, 131–138, Maebishi, Japan, 2002.

[3] I. S. DHILLON, D. S. MODHA, Concept Decompositions for Large Sparse Text Data using Clustering, *Machine Learning*, 42/1, 143–175, 2001.

[4] M. INABA, N. KATOH, H. IMAI, Applications of Weighted Voronoi Diagrams and Randomization to Variance-Based k-Clustering, *Proceedings of the 10th ACM Symposium on Computational Geometry*, 332–339, ACM, 1994.

[5] T. KANUNGO, D. M. MOUNT, N. S. NETANYAHU, C. D. PIATKO, R. SILVERMAN, A. Y. WU, A Local Search Approximation Algorithm for *k*-Means Clustering, *Proceedings of the 18th Annual ACM Symposium on Computational Geometry*, 10–18, 2002.

[6] S. HAR-PELED, B. SADRI, On Lloyd's *k*-means Method, *ACM-SIAM Symposium on Discrete Algorithms*, 2005.

Ivančica Mirošević
ivancica.mirosevic@fesb.hr

Fakultet elektrotehnike, strojarstva i brodogradnje
Sveučilište u Splitu
Ruđera Boškovića 32, 21000 Split