Algoritam K-sredina i modifikacije

Bratić, Tea

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Tea Bratić

ALGORITAM K-SREDINA I MODIFIKACIJE

Diplomski rad

Voditelj rada:
doc. dr. sc. Pavle Goldstein

Zagreb, veljača, 2020

Ovaj diplomski rad obranjen je dana ____________________ pred ispitnim povjerenstvom u sastavu:

1. ____________________________, predsjednik
2. ____________________________, član
3. ____________________________, član

Povjerenstvo je rad ocijenilo ocjenom _______________.

Potpisi članova povjerenstva:

1. ____________________________
2. ____________________________
3. ____________________________

Zahvaljujem mentoru doc. dr. sc. Pavlu Goldsteinu na zanimljivoj temi, savjetima i što je uvijek imao strpljenja i vremena za moje brojne upite pri izradi diplomskog rada. Veliko hvala mojim roditeljima, bratu i prijateljima na podršci koju su mi nesebično pružali tokom svih ovih godina studiranja.

Sadržaj

Uvod

Algoritam k-sredina jedan je od najpopularnijih algoritama za klasteriranje podataka koji se koristi u mnogim područjima kao što su strojno učenje, bioinformatika, prepoznavanje uzoraka i mnogim drugima. Klasteriranje je tehnika statističke analize podataka koja se koristi kad želimo dani skup podijeliti na podskupove (najčešće disjunktne), tako da unutar skupova maksimiziramo sličnost (ili minimiziramo udaljenost). U algoritmu k-sredina podaci se grupiraju u k klastera tako da se minimizira suma kvadrata udaljenosti u svakom pojedinom klasteru. Cilj ovog diplomskog rada je modificirati algoritam k-sredina koji se ne zaglavi.

U prvom poglavlju ćemo definirati osnovne pojmove u klaster analizi. Opisat ćemo algoritam k-sredina, s naglaskom na matematičke aspekte, kao što su težište skupa i konvergencija algoritma.

U drugom poglavlju će se opisati problem zaglavljivanja algoritma k-sredina. Navode se tri moguće modifikacije algoritma koje se najčešće ne zaglavljuju. Nadalje, za svaku od modifikacija ćemo provjeriti je li ona poboljšana verzija u odnosu na algoritam k-sredina.

Na kraju, u trećem poglavlju su prikazani rezultati testiranja na primjerima u kojima algoritam k-sredina ne uspijeva odrediti optimalnu particiju, dok modifikacije algoritma pronalaze optimalnu konfiguraciju za dani k.

Poglavlje 1

Klasteriranje k-sredinama

Klasteriranje ili grupiranje je tehnika statističke analize podataka koja se koristi u mnogim područjima uključujući strojno učenje, prepoznavanje obrazaca, slikovnu analizu i bioinformatiku. Klasteriranje je dijeljenje skupa podataka u podskupove tako da podaci u svakom podskupu dijele neko zajedničko obilježje - često približnost prema nekoj definiranoj veličini udaljenosti.

Definicija 1.0.1. *Neka je $C = \{x_1, x_2, \ldots, x_n\}$ skup točaka i $k \in \mathbb{N}$. Klasteriranje je particija skupa C, čije elemente zovemo klasteri. Podjela skupa C na k klastera je podjela skupa C na particiju od k elemenata.*

Klasteriranje k-sredinama (engl. *k- means clustering*) je jedna od najpopularnijih tehnika klasteriranja. Algoritam je iterativan i njegov cilj je particionirati skup podataka u k disjunktnih grupa (svaka točka pripada točno jednom klasteru od k klastera). Pripada kategoriji nenadziranog strojnog učenja (engl. *unsupervised learning*).

1.1 Funkcija cilja

Neka je $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathbb{R}^m$ skup točaka. Za $k \in \mathbb{N}$ sa $\{C_1, C_2 \dots, C_k\}$ označimo k klastera, te sa $c_1, c_2 \dots, c_k$ pripadne centre. Klasteriranjem k-sredina se treba naći optimalna k-particija skupa X. To se postiže minimizacijom funkcije cilja f iz Definicija 1.1.2, koja je definirana u terminima klastera $C_1, C_2 \dots, C_k$ i centara klastera $c_1, c_2 \dots, c_k$.

Definicija 1.1.1. *Neka su $x = (x_1, x_2, \dots, x_m)$, $y = (y_1, y_2, \dots, y_m)$ bilo koje dvije točke iz $\mathbb{R}^m$. Neka je $d : X \times X \to \mathbb{R}$ Euklidska udaljenost definirana formulom*

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_m - y_m)^2} = \sqrt{\left(\sum_{i=1}^{m} (x_i - y_i)^2\right)} \tag{1.1}$$

Definicija 1.1.2. *Neka je c_i centar klastera C_i, za svaki $i = 1, \dots, k$, gdje je k broj klastera. Definiramo funkciju cilja sa*

$$f(C_1, C_2, \dots, C_k, c_1, c_2 \dots, c_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, c_i), \tag{1.2}$$

gdje je $d(\cdot, \cdot)$ Euklidska udaljenost.

1.2 Težište skupa

Definicija 1.2.1. *Neka je $X = \{x_1, x_2 \dots, x_n\} \subseteq \mathbb{R}^m$, gdje je $x_l = (x_{l1}, x_{l2}, \dots, x_{lm}) \in \mathbb{R}^m$, $l = 1, 2, \dots, n$. Težište skupa X je točka $t_X = (t_1, t_2, \dots, t_m) \in \mathbb{R}^m$ u kojoj funkcija*

$$f(t_X) := \sum_{i=1}^{n} d^2(t_X, x_i) \tag{1.3}$$

postiže minimum.

Lema 1.2.2. *Neka je $X = \{x_1, x_2, \dots, x_n\} \subseteq \mathbb{R}^m$, gdje je $x_l = (x_{l1}, x_{l2}, \dots, x_{lm}), \in \mathbb{R}^m$, $l = 1, 2, \dots, m$. Tada je*

$$t_X = (t_1, t_2, \dots, t_m) = \left(\frac{\sum_{i=1}^{n} x_{i1}}{n}, \frac{\sum_{i=1}^{n} x_{i2}}{n}, \dots, \frac{\sum_{i=1}^{n} x_{im}}{n} \right) \in \mathbb{R}^m \tag{1.4}$$

točka u kojoj funkcija (1.3) postiže minimum.

Dokaz. Računamo gradijent funkcije f:

$$\nabla f = \left(\frac{\partial f}{\partial t_1}, \frac{\partial f}{\partial t_2}, \ldots, \frac{\partial f}{\partial t_m}\right). \tag{1.5}$$

Za svaki $l \in \{1, 2, \ldots, m\}$ vrijedi:

$$\frac{\partial f}{\partial t_l}(t) = \sum_{i=1}^{n}(-2x_{il} + 2t_l) \tag{1.6}$$

$$= -2\sum_{i=1}^{n} nx_{il} + 2nt_l. \tag{1.7}$$

Ako gradijent (1.5) izjednačimo s nulom, onda dobijemo koordinatne kritične točke t_X funkcije f:

$$t_l = \frac{\sum_{i=1}^{n} x_{il}}{n}.$$

Na dva načina je moguće pokazati da je dobivena točka minimuma funkcije f.
Prvi način je da odredimo Hesseovu matricu H. Lako se vidi da su sve mješovite parcijalne derivacije jednake nuli, dok za ostale vrijedi

$$\frac{\partial^2 f}{\partial t_l \partial t_l}(t) = 2n, \forall l \in \{1, 2, \ldots, m\},$$

što je strogo veće od nule jer je n broj točaka u skupu X. Dakle, matrica H izgleda ovako:

$$H = \begin{bmatrix} 2n & 0 & \cdots & 0 \\ 0 & 2n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2n \end{bmatrix},$$

i ona je pozitivno definitna matrica. Dakle, dobivena kritična točka je točka minimuma funkcije f. Drugi način je da uočimo da gradijent funkcije f ima jedinstvenu nultočku, te da vrijedi

$$\lim_{\|t\| \to \infty} f(t) = +\infty,$$

iz čega možemo zaključiti da dotična nultočka mora biti točka minimuma. Dakle, funkcija f postiže minimum u točki

$$t_X = \left(\frac{\sum_{i=1}^{n} x_{i1}}{n}, \frac{\sum_{i=1}^{n} x_{i2}}{n}, \ldots, \frac{\sum_{i=1}^{n} x_{im}}{n}\right).$$

□

Napomena 1.2.3. *Uočimo da su koordinate težišta t_X aritmetička sredina odgovarajućih točaka iz skupa X.*

1.3 Lloydov algoritam

Lloydov algoritam particionira skup podataka u klastere, računa njihove centre, te onda razmješta elemente skupa po klasterima ovisno o njihovoj udaljenosti do centra klastera. Neka je $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^m$ skup točaka. Cilj algortima je particionirati skup X na k-klastera. Algoritam ponavlja korake:

1. "**assignment**" korak: pridruživanja točke klasteru s najbližim centrom

$$C_i^{t+1} = \{x : d(x, c_i^{(t)}) \leq d(x, c_j^{(t)}), \forall j\} \tag{1.8}$$

2. "**update**" korak: određivanje centara za novi raspored klastera

$$c_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x \in C_i^{(t)}} x \tag{1.9}$$

Ovdje $C_i^{(t)}$ i $c_i^{(t)}$ označavaju i-ti klaster i i-ti centar u t-toj iteraciji, dok $|C_i^{(t)}|$ označava veličinu skupa $C_i^{(t)}$.

Algoritam 1 Lloydov algoritam

1: Slučajnim odabirom ili na neki drugi način odrediti inicijalne c_i, $i = 1, \ldots, k$, gdje su $c_i \in X$.
2: Za svaki $x \in X$ pronaći njemu najbliži centar c_i i pridružiti x klasteru C_i. Za računanje udaljenosti među točkama koristi se Euklidska udaljenost 1.1 .
3: Za svaki klaster C_i koristeći 1.9 izračunati novi centar c_i.
4: Ponavljati korake 2 i 3 do konvergencije.

Cilj algoritma je pronaći particiju tako da funkcija cilja ima minimalnu vrijednost. Algoritam se može zaustaviti na dva načina, ako se particija više ne mjenja ili ako je zadan broj iteracija. Koraci (2) i (3) u Algoritam 1 smanjuju vrijednost funkcije cilja f. Koraci (2) i (3) su optimalni odabiri u smislu da oni (lokalno) maksimalno poboljšavaju danu konfiguraciju. Dakle, ako označimo s $f^{(i)}$ vrijednost funkcije cilja u i-toj iteraciji, dobivamo padajući niz

$$f^{(1)} \geq f^{(2)} \geq \cdots \geq 0 \tag{1.10}$$

Iz toga slijedi da će algoritam dostići minimum od f u konačnom broju koraka, ali iz (1.10) također slijedi da će taj minimum često biti lokalan.

1.4 Konvergencija algoritma

Sljedeće dvije leme pokazuju da Lloydov algoritam optimizira funkciju cilja, odnosno da funkcija cilja konvergira.

Lema 1.4.1. *Neka su $C_1, C_2 \dots, C_k$ klasteri dobiveni u nekoj iteraciji algoritma k-sredina, te neka su $c'_1, c'_2 \dots, c'_k$ novi centri i $C'_1, C'_2 \dots, C'_k$ novi klasteri. Tada vrijedi:*

(a) $c'_i = arg\min_t \sum_{x \in C_i} d^2(x,t), \forall i \in \{1, \dots, k\}$

(b) $\forall x \in X, x \in C'_i \Leftrightarrow d^2(x, c'_i) \leq d^2(x, c'_j), \forall j = 1, \dots, k$

Dokaz. Tvrdnja (a) je dokazana u Lemi 1.2.2. Tvrdnja (b) slijedi iz činjenice da smo klaster C'_i definirali kao $C'_i := \{x \in X : i = arg\min_{1 \leq j \leq k} d^2(x, c'_j)\}$, tj. klaster C'_i sastoji se od točaka za koje je c'_i najbliži cenatar. □

Lema 1.4.2. *Neka su $c_1, c_2 \dots, c_k$ centri dobiveni u nekoj iteraciji algoritma k-sredina i f vrijednost funkcije cilja u toj iteraciji, te neka su $c'_1, c'_2 \dots, c'_k$ centri dobiveni u sljedećoj iteraciji i f' funkcija cilja u toj iteraciji. Tada vrijedi:*

$$f' = \sum_{i=1}^{k} \sum_{x \in C'_i} d^2(x, c'_i) \leq f = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, c_i)$$

Dokaz. Koristimo pomoćne tvrdnje dokazane u Lemi 1.4.1. Prema tvrdnji (b) vrijedi

$$\forall x \in X, x \in C'_i \Leftrightarrow d^2(x, c'_i) \leq d^2(x, c'_j), \forall j = 1, \dots, k$$

tj. za svaku točku, od svih centara kandidata, odabiremo najbliži centar. Kako ovo vrijedi za svaku točku c'_j, onda specijalno vrijedi i za $c'_j = c_i$, a kako tvrdnja vrijedi za svaku točku x iz klastera C_i, imamo

$$\sum_{x \in C_i} d^2(x, c'_i) \leq \sum_{x \in C_i} d^2(x, c_i), \forall j = 1, 2, \dots, k$$

Kako ovo vrijedi za svaki klaster $i = 1, 2, \dots, k$, dobivamo

$$f' = \sum_{i=1}^{k} \sum_{x \in C'_i} d^2(x, c'_i) \leq f = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, c_i),$$

što pokazuje da funkcija cilja pada. Prema tvrdnji (a) iz Leme 1.4.1 vrijedi

$$c'_i = arg\min_t \sum_{x \in C_i} d^2(x, t).$$

Dakle, za svako t vrijedi

$$\sum_{x \in C_i} d^2(x, c_i') \leq \sum_{x \in C_i} d^2(x, t),$$

pa specijalno to vrijedi i za $t = c_i$. Budući da to vrijedi $\forall i = 1, 2, \ldots, k$ dobivamo

$$f' = \sum_{i=1}^{k} \sum_{x \in C_i'} d^2(x, c_i') \leq f = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, c_i).$$

Prema tome slijedi $f' \leq f$. $\square$

U nastavku je pokazano kako radi algoritam na jednostavnom primjeru.

Primjer 1.4.3. *Neka je:*

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\} \subseteq \mathbb{R}^2$$

$$x_1 = (1, 1), x_2 = (4, 5), x_3 = (2, 2), x_4 = (4, 3), x_5 = (1, 2), x_6 = (4, 4)$$

Neka je $k = 2$, željeni broj klastera, te neka su $c_1 = x_1$ i $c_2 = x_2$ inicijalni centri.

1. $c_1 = x_1, c_2 = x_2$

$$\begin{array}{ccc} d^2(x_3, c_1) = 2 & d^2(x_3, c_2) = 13 & x_3 \in C_1 \\ d^2(x_4, c_1) = 13 & d^2(x_4, c_2) = 4 & x_4 \in C_2 \\ d^2(x_5, c_1) = 1 & d^2(x_5, c_2) = 18 & x_5 \in C_1 \\ d^2(x_6, c_1) = 18 & d^2(x_6, c_2) = 1 & x_6 \in C_2 \end{array}$$

$\Rightarrow C_1 = \{x_1, x_3, x_5\}, C_2 = \{x_2, x_4, x_6\}$

2. $c_1 = t_{C_1} = (\frac{4}{3}, \frac{5}{3}), c_2 = t_{C_2} = (4, 4)$

$$\begin{array}{ccc} d^2(x_1, c_1) = 0.\dot{5} & d^2(x_1, c_2) = 18 & x_1 \in C_1 \\ d^2(x_2, c_1) = 18.\dot{2} & d^2(x_2, c_2) = 1 & x_2 \in C_2 \\ d^2(x_3, c_1) = 0.\dot{5} & d^2(x_3, c_2) = 8 & x_3 \in C_1 \\ d^2(x_4, c_1) = 8.\dot{8} & d^2(x_4, c_2) = 1 & x_4 \in C_2 \\ d^2(x_5, c_1) = 0.\dot{2} & d^2(x_5, c_2) = 13 & x_5 \in C_1 \\ d^2(x_6, c_1) = 12.\dot{5} & d^2(x_6, c_2) = 0 & x_5 \in C_2 \end{array}$$

$\Rightarrow C_1 = \{x_1, x_3, x_5\}, C_2 = \{x_2, x_4, x_6\}$

3. *Algoritam može stati, jer se c_1 i c_2 ne mijenjaju.*

1.5 Minimizacija sume kvadrata udaljenosti između točaka

Lema 1.5.1. *Neka je $X = \{x_1, x_2, ..., x_n\} \subseteq \mathbb{R}^m$ neprazan, konačan skup i $|X| = n$. Za svaki $y \in \mathbb{R}$ vrijedi:*

$$\sum_{i=1}^{n} d^2(x_i, y) = \sum_{i=1}^{n} d^2(x_i, t) + n \cdot d^2(t, y) \tag{1.11}$$

gdje je t težište skupa X.

Dokaz.

$$\begin{aligned}\sum_{i=1}^{n} d^2(x_i, y) &= \sum_{i=1}^{n}(x_i - y)^2 \\ &= \sum_{i=1}^{n}(x_i - t + t - y)^2 \\ &= \sum_{i=1}^{n}(x_i - t)^2 + (t - y)\sum_{i=1}^{n}(x_i - t) + n \cdot (t - y)^2\end{aligned}$$

Kako je t težište vrijedi: $\sum_{i=1}^{n}(x_i - t) = 0$. Iz $t = \frac{1}{n}\sum_{i=1}^{n} x_i$ slijedi:

$$n \cdot t = n \cdot \frac{1}{n}\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i$$

nadalje imamo:

$$\sum_{i=1}^{n}(x_i - t) = \sum_{i=1}^{n} x_i - n \cdot t = nt - nt = 0$$

iz čega slijedi:

$$\begin{aligned}&= \sum_{i=1}^{n}(x_i - t)^2 + (t - y) \cdot 0 + n \cdot (t - y)^2 \\ &= \sum_{i=1}^{n} d^2(x_i, t) + n \cdot d^2(t, y)\end{aligned}$$

□

Posljedica Leme 1.5.1 je da centar smanjuje sumu kvadrata udaljenosti budući da je $n \cdot d^2(t, y)$ uvijek pozitivan.

Napomena 1.5.2. *Treba pripaziti na notaciju, jer za skup točaka* $X = \{x_1, x_2, \dots, x_n\}$*, izraz* $\sum_{i=1}^{n} \sum_{j=i+1}^{n} d^2(x_i, x_j)$ *broji količinu* $d^2(x_i, x_j)$ *za svaki par* $(i, j), j > i$*, dok* $\sum_{i,j} d^2(x_i, x_j)$ *broji svaki* $d^2(x_i, x_j)$ *dva puta, stoga je suma drugog izraza dvostruko veća od sume prvog izraza.*

Druga mogućnost za minimizaciju funkcije cilja koja računa sume kvadrata udaljenosti točke skupa X i centra je minimizacija sume kvadrata udaljenosti svih parova točaka iz X.

Korolar 1.5.3. *Neka je* $X = \{x_1, x_2, ..., x_n\} \subseteq \mathbb{R}^m$*. Za svaki* $t \in \mathbb{R}$ *vrijedi:*

$$\sum_{i=1}^{n} \sum_{j>i} d^2(x_i, x_j) = n \sum_{i} d^2(x_i, t). \tag{1.12}$$

Dokaz. Ako u tvrdnju (1.11) umjesto y uvrstimo sve točke skupa X imamo sljedeće:

$$\begin{aligned} \sum_{i,j=1}^{n} d^2(x_i, x_j) &= n \sum_{i=1}^{n} d^2(x_i, t) + n \sum_{i=j}^{n} d^2(x_j, t) \\ &= 2n \sum_{i=1}^{n} d^2(x_i, t) \end{aligned}$$

iz napomene (1.5.2) imamo:

$$\sum_{i,j} d^2(x_i, t) = 2 \sum_{i} \sum_{j>i} d^2(x_i, x_j)$$

iz čega slijedi:

$$\sum_{i} \sum_{j>i} d^2(x_i, x_j) = n \sum_{i=1}^{n} d^2(x_i, t)$$

□

Sad smo pokazali da je zbroj kvadrata udaljenosti između svih parova točaka jednak umnošku broju točaka i zbroju kvadrata udaljenosti od točke do središta.

Poglavlje 2

Modifikacije Lloydovog algortima

2.1 Nedostaci Lloydovog algortima

Kao što je ranije navedeno, Lloyodov algoritam particionira polazni skup točaka u k klastera, gdje je cilj minimizacija funkcije cilja 1.1.2. Osigurava konvergenciju, jer je pohlepan algoritam koji u oba koraka (1.3) i (1.4) smanjuje vrijednost ciljne funkcije. Lloydov algoritam ne mora nužno pronaći optimalnu konfiguraciju klastera koja odgovara globalnom minimumu ciljne funkcije. Za pronalazak optimalne konfiguracije klastera često je potreban dobar izbor polazne konfiguracije. Ako algoritam završi u lokalnom minimumu, a ne u globalnom minimumu, onda kažemo da se algoritam "zaglavio".

2.2 Varijacije Lloydovog algoritma

U ovom potpoglavlju će se opisati dvije verzije algortima za rješavanje problema zaglavljivanja. Ideje sljećih algoritama su:

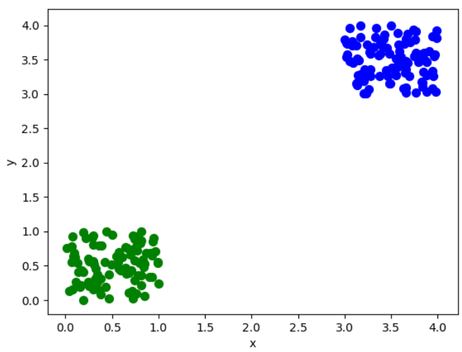
1. **Algoritam 1**: Pridružiti točke polaznog skupa onom klasteru s čijim točkama ima manju udaljenost od prosječne.
2. **Algoritam 2**: Odabrati dvije točke polaznog skupa koje imaju najveću udaljenost i pridružiti ih različitim klasterima. Svaku iduću točku pridružiti onom klasteru čijim je točkama bliža.

Neka je $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^m$ polazni skup podataka koje želimo razvrstati u klastere $C_1, C_2, \ldots, C_k$ za dani $k \in \mathbb{N}$ željeni broj klastera.

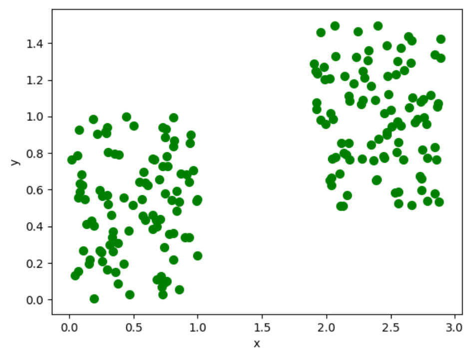
Algoritam 1

1: Prvu točku skupa X dodijeliti prvom klasteru, $x_1 \in C_1$.
2: Izračunati prosječnu udaljenost između svih točaka skupa X, oznaka p.
3: Za iduću točku $x_2 \in X$ provjeriti je li udaljenost između x_1, x_2 manja od p. Ako je tvrdnja istina onda obje točke pripadaju istom klasteru, $x_1, x_2 \in C_1$. U protivnom pripadaju različitim klasterima, $x_1 \in C_1, x_2 \in C_2$
4: Za svaki $i = 1, 2, \ldots, k$, za svaki $j = 1, 2, \ldots, |C_i|$ izračunati udaljenost od svake točke iz C_i do nove točke $x_l \in X$. Ako postoji neka točka iz C_i tako da je njihova udaljenost manja od p one pripadaju istom klasteru, u protivnom, one pripadaju različitim klasterima.
5: Za svaku iduću točku skupa X tj. za svaki $l = 3, 4 \ldots, n$, ponavljamo korak 4.

Ovakav pristup problemu daje dobro rješenje jedino u slučaju kad je udaljenost između dvije najbliže točke različitih klastera veća od prosječne udaljenosti između svih točaka promatranog skupa. Slika 2.1 i Slika 2.2 pokazuju primjer kad će algoritam uspješno klasterirati i primjer u kojem neće.



Slika 2.1: Primjer 1



Slika 2.2: Primjer 2

Na prvoj slici vidi se primjer uspješnog klasteriranje jer je prosječna udaljenost između svih točaka 2.36, a udaljenost između klastera (između dvije najbliže točke različitih klastera) je 3.09, kao što je opisano u Algoritam 1. Algoritam će primjer sa slike 2.1 uspješno klasterirati jer je prosječna udaljenost između svih točaka veća od udaljenosti između klastera. Na drugom primjeru prosječna udaljenost između svih točaka je 1.23, a udaljenost između klastera (između dvije najbliže točke različitih klastera) je 0.98, te ne uspjeva klasterirati podatke u dvije grupe.

Algoritam 2

1: Izračunati udaljenosti između svih točaka skupa X.
2: Dvije točke $x_s, x_t \in X$ koje imaju najveću udaljenost pridružiti različitim klasterima, $x_s \in C_1, x_t \in C_2$
3: Za svaku točku $x_i \in X \setminus \{x_s, x_t\}$ provjeriti kojoj od dvije najudaljenije točke x_s, x_t bliža, te pridružiti klasteru čijoj je točki bliža, $x_i \in C_1$ ili $x_i \in C_2$.
4: Za svaki klaster izračunamo sumu kvadrata udaljenosti između točaka istog klastera i odaberemo klaster koji ima veću ukupnu sumu, C_l.
5: Izračunati udaljenosti između svih točaka klastera C_l.
6: Dvije najudaljenije točke $x'_s, x'_l \in C_l$ pridružiti različitim klasterima, $x'_s \in C'_l, x'_t \in C_j$
7: Za svaku točku $x'_i \in C_l \setminus \{x'_s, x'_t\}$ provjeriti kojoj od dvije najudaljenije točke x'_s, x'_t bliža, te pridružiti klasteru čijoj je točki bliža, $x'_i \in C'_l$ ili $x'_i \in C_j$.
8: Ako je $k > 2$, za $j = 3, 4, \ldots, k$ ponavljamo korake $4, 5, 6, 7$.

Drugi algoritam staje kad dobijemo željeni broj klastera.

Primjer 2.2.1. *Neka je:*

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\} \subseteq \mathbb{R}^2$$

$$x_1 = (1, 1), x_2 = (4, 3), x_3 = (1, 2), x_4 = (5, 2), x_5 = (6, 2), x_6 = (4, 4)$$

Neka je $k = 3$, željeni broj klastera.
Prvo moramo izračunati sve udaljenosti između svih točaka. Udaljenosti između svih točaka skupa X prikazujemo matricom udaljenosti:

$$\begin{bmatrix} 0 & 3.6 & 1 & 4.1 & 5.1 & 4.2 \\ 3.6 & 0 & 3.2 & 1.4 & 2.2 & 1 \\ 1 & 3.2 & 0 & 4 & 5 & 3.6 \\ 4.1 & 1.4 & 4 & 0 & 1 & 2.2 \\ 5.1 & 2.2 & 5 & 1 & 0 & 2.8 \\ 4.2 & 1 & 3.6 & 2.2 & 2.8 & 0 \end{bmatrix}$$

Vidimo da točke x_1, x_5 imaju najveću udaljenost, te ih pridružiti različitim klasterima: $x_1 \in C_1$, $x_5 \in C_2$. Sve iduće točke prirdužimo klasterima C_1, C_2, ovisno jesu li bliže x_1 ili x_5. Dobijemo sljedeće:

$$C_1 = \{x_1, x_3\}, C_2 = \{x_2, x_4, x_5, x_6\}$$

Sad izračunamo:

$$S_l = \sum_{i=1}^{|C_l|} \sum_{j>i} d^2(x_i, x_j),$$

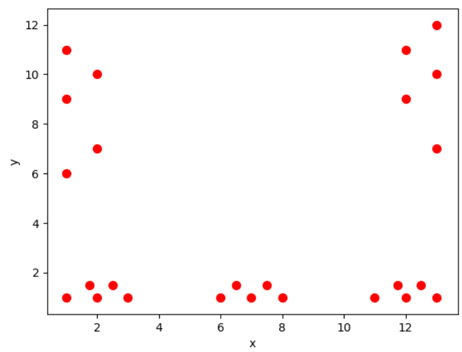
za l=1,2. Dobijemo da je $S_1 = 1$ *i* $S_2 = 42.96$*. Stoga dalje grupiramo točke iz klastera* C_2*. Iz matrice udaljenosti vidimo da su* x_5 *i* x_6 *najudaljenije točke u klasteru* C_2*, vrijedi:* $x_5 \in C_2$ *i* $x_6 \in C_3$*. Sad pridružujemo preostale točke polaznog klastera* C_2 *u novonastala dva, ovisno o tome jesu li bliže točki* x_5 *ili* x_6*.*
Dobivamo:

$$C_2 = \{x_2, x_6\}, C_3 = \{x_4, x_5\}.$$

Algoritam staje jer smo dobili tri klastera.

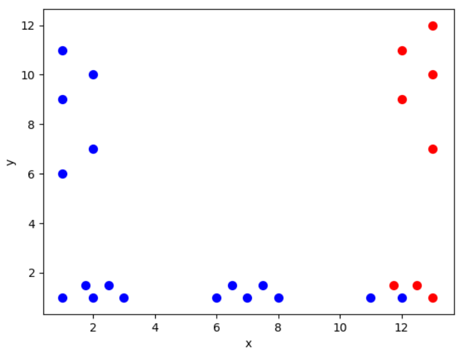
Ovaj algoritam će uspješno riješiti primjere sa Slika 2.1 i Slika 2.2, jer algoritam daje zadovoljavajuće rezultate kad je željeni broj klastera dva. U nastavku slijedi primjer na kojem algoritam neće grupirati točke tako da funkcija cilja bude minimalna.

Na slici 2.3 su prikazane točke polaznog skupa. Taj skup je potrebno klasterirati u pet klastera, postupak stvaranja klastera je prikazan slikama 2.4, 2.5, 2.6, 2.7.

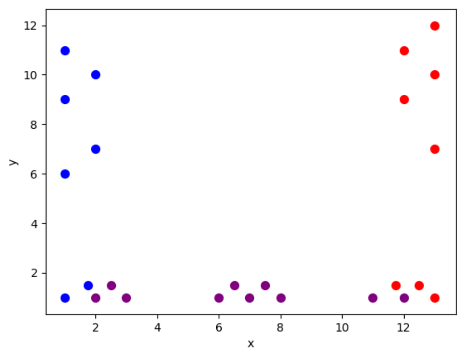


Slika 2.3: Polazni skup

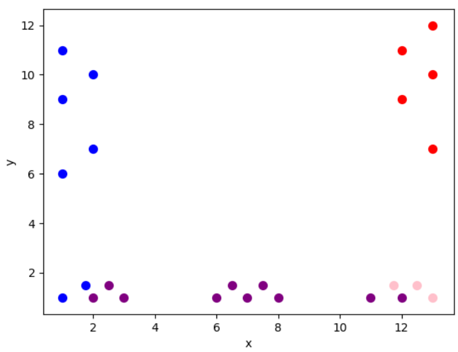
Već prvom grupacijom točaka vidimo da se neće formirati klasteri koje očekujemo. Do toga dolazi jer se točke pridružuju klasteru samo po jednom kriteriju, a to je udaljenost između točaka. Na slici 2.8 je prikazano kako očekujemo da se točke grupiraju.



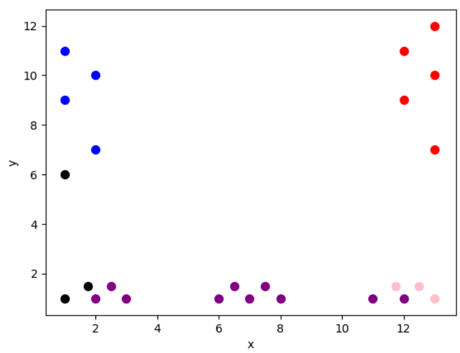
Slika 2.4: Dva klastera



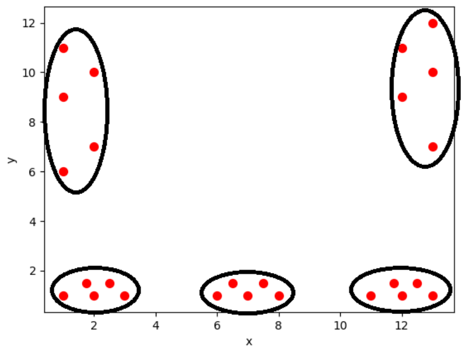
Slika 2.5: Tri klastera



Slika 2.6: Četiri klastera



Slika 2.7: Pet klastera



Slika 2.8: Očekivano grupiranje

2.3 Mješoviti algoritam

Neka je $X = \{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{R}^m$ skup podataka koje želimo razvrstati u klastere $C_1, C_2, \ldots, C_k$. Mješoviti algoritam se sastoji od sljedeća četiri koraka:

1. Dijeliti klaster koji ima veću sumu kvadratnih udaljenosti od točke do težišta na dva klastera, tako da se svakom klasteru pridruži najudaljenija točka. Svaku iduću točku tog klastera pridružiti jednom od nova dva klastera s kojim ima najmanju sumu kvadratnih udaljenosti od nje do težišta.

2. Svaku točku skupa X pridružiti klasteru čijem je težištu bliža.

3. Izračunati nova težišta klastera u odnosu na trenutnu particiju.

4. Izračunati nove sume kvadrata udaljenosti od točke do težišta za sve dobivene klastere.

Skup X se može promatrati kao jedan klaster, dakle njega dijelimo u prvom koraku algoritma. Funkcija cilja je ranije definirana kao

$$f(C_1, \ldots, C_k, c_1, \ldots, c_k) = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(x, c_i),$$

gdje je c_i centar klastera C_i i $d(\cdot, \cdot)$ Euklidska udaljenost. U mješovitom algoritmu drugi i treći korak su upravo koraci "assignment " i "update " u Lloydovom algoritmu. Za svaki $k > 1$, algoritam će napraviti $(k-1)$ put opisana četiri koraka. Algoritam je deterministički, dakle za iste ulazne podatke daje iste rezultate. Ako se u Lloydovom algortimu (1) u prvom koraku na slučajan način određuju inicijalni centri c_i, $i = 1, \ldots, k$, algoritam neće biti deterministički, dakle za različiti odabir polaznih centara dobiveni klasteri nisu nužno jednaki.

Algoritam 3 Mješoviti algoritam

1: Izračunati udaljenosti između svih točaka skupa X.
2: Dvije točke $x_s, x_t \in X$ koje imaju najveću udaljenost pridružiti različitim klasterima, $x_s \in C_1, x_t \in C_2$.
3: Ako je udaljenost od sljedeće točke $x_a \in X \setminus \{x_s, x_t\}$ do x_s manja od udaljenosti od x_a do x_t pridružiti je klasteru C_1, u protivnom, pridružiti je klasteru C_2.
4: Izračunati težišta skupova C_1 i C_2
5: Za svaku iduću točku skupa $x_b \in X \setminus \{x_s, x_t, x_a\}$ izračunati vrijednosti težišta skupova C_1, C_2 s točkom x_b i sumu kvadratnih udaljenosti od točka pripadnog skupa do težišta. Pridružiti točku x_b klasteru čija je sumu kvadratnih udaljenosti od točka pripadnog skupa do težišta manja.
6: Za svaki $i = 1, 2, \ldots, n$, točku $x_i \in X$ pridružiti klasteru čijem je težištu bliža.
7: Izračunati koji od dobivenih klastera ima najveću sumu kvadratnih udaljenosti od točke do težišta, oznaka C_s
8: Dvije točke $x'_s, x'_t \in C_s$ koje imaju najveću udaljenost pridružiti različitim klasterima, $x'_s \in C'_s, x'_t \in C_j$
9: Ako je udaljenost od sljedeće točke $x'_a \in C_s \setminus \{x'_s, x,'_t\}$ do x'_s manja od udaljenosti od x'_a do x'_t pridružiti klasteru C'_s, u protivnom, pridružiti je klasteru C_j.
10: Izračunati težišta skupova C'_s i C_j
11: Za svaku iduću točku skupa $x'_b \in C_s \setminus \{x'_s, x'_t, x'_a\}$ izračunati vrijednosti težišta skupova C'_s, C_j s točkom x'_b i sumu kvadrata udaljenosti točaka pripadnog skupa i njegovog težišta.
12: Pridružiti točku x'_b onom klasteru čija je suma kvadrata udaljenosti manja.
13: Za svaki $i = 1, 2, \ldots, n$, točku $x_i \in X$ pridružiti klasteru čijem je težištu bliža.
14: Ako je $k > 2$, za $j = 3, 4, \ldots, k$ ponavljati korake od 7 do 13.

U prvom koraku u Algoritam 3 za izračunavanje udaljenosti između svih točaka skupa koristi se Euklidska udaljenost tj. za svaki $x, y \in X$ izračunati

$$d_j = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2},$$

za $j = 1, 2, \ldots, \frac{(n+1)n}{2}$.

Težište klastera $C = \{x_1, x_2, \ldots, x_l\} \subseteq \mathbb{R}^m$, gdje je $x_1 = (x_{11}, x_{12}, \ldots, x_{1m})$, $x_2 = (x_{21}, x_{22}, \ldots, x_{2m}), \ldots, x_l = (x_{l1}, x_{l2}, \ldots, x_{lm})$ je

$$t_C = (t_1, t_2, \ldots, t_m) = \left(\frac{\sum_{i=1}^{l} x_{i1}}{|C|}, \frac{\sum_{i=1}^{l} x_{i2}}{|C|}, \ldots, \frac{\sum_{i=1}^{l} x_{im}}{|C|} \right). \tag{2.1}$$

Primijetimo da dodavanjem nove točke $x_b \in X$ klasteru C s težištem t_c kao što je u koracima 5 i 11 u algoritmu 3 nije potrebno računati novo težište t'_c kao u (2.1), dovoljno je sljedeće:

$$t'_{C'} = \frac{|C|}{|C|+1}\left(\frac{\sum_{x\in C} x}{|C|} + x_b\right). \tag{2.2}$$

Primjer 2.3.1. *Neka je:*

$$X = \{x_1, x_2, x_3, x_4, x_5, x_6\} \subseteq \mathbb{R}^2$$

$$x_1 = (1,2), x_2 = (2,1), x_3 = (4,7), x_4 = (5,6), x_5 = (11,12), x_6 = (10,11)$$

Neka je $k = 3$, željeni broj klastera.

Prvo moramo izračunati udaljenosti između svih parova točaka. Udaljenosti između svih točaka skupa X prikazujemo matricom udaljenosti:

$$\begin{bmatrix} 0 & 1.4 & 5.8 & 5.7 & 14.1 & 12.7 \\ 1.4 & 0 & 6.3 & 5.8 & 14.2 & 12.8 \\ 5.8 & 6.3 & 0 & 1.4 & 8.6 & 7.2 \\ 5.7 & 5.8 & 1.4 & 0 & 8.5 & 7.1 \\ 14.1 & 14.2 & 8.6 & 8.5 & 0 & 1.4 \\ 12.7 & 12.8 & 7.2 & 7.1 & 1.4 & 0 \end{bmatrix}$$

Dvije najudaljenije točke su x_2, x_5, te ih pridružiti različitim klasterima: $x_2 \in C_1, x_5 \in C_2$. Za iduću točku $x_1 \in X$ usporediti udaljenosti: $d(x_1, x_2) < d(x_1, x_5)$, te vrijedi $x_1 \in C_1$. Dobivena težišta: $t_{C_1} = (1.5, 1.5), t_{C_2} = (11, 12)$. Za svaku iduću točku izračunati t_{C_1} i t_{C_2} i izračunati sume kvadrata udaljenosti od točke do težišta S_1, S_2 i pridružiti je klasteru koji s njom ima manju sumu kvadratnih udaljenosti od točke do težišta.

$$\begin{array}{llllll} x_3: & t_{C_1} = (2.3, 3.3) & t_{C_2} = (7.5, 9.5) & S_1 = 25.3 & S_2 = 37 & \Rightarrow x_3 \in C_1 \\ x_4: & t_{C_1} = (3, 4) & t_{C_2} = (8, 9) & S_1 = 36 & S_2 = 35.9 & \Rightarrow x_4 \in C_2 \\ x_6: & t_{C_1} = (4.25, 5.25) & t_{C_2} = (8.7, 9.7) & S_1 = 113.5 & S_2 = 41.4 & \Rightarrow x_6 \in C_2 \end{array}$$

Vrijedi $C_1 = \{x_1, x_2, x_3\}, C_2 = \{x_4, x_5, x_6\}$. Treba provjeriti za svaku točku skupa X kojem je težištu bliža:

$$\begin{array}{ll} d(x_1, t_{C_1}) = 1.9 & d(x_1, t_{C_2}) = 10.8 \\ d(x_2, t_{C_1}) = 2.4 & d(x_2, t_{C_2}) = 10.9 \\ d(x_3, t_{C_1}) = 4 & d(x_3, t_{C_2}) = 5.4 \\ d(x_4, t_{C_1}) = 3.8 & d(x_4, t_{C_2}) = 5.2 \\ d(x_5, t_{C_1}) = 12.3 & d(x_5, t_{C_2}) = 3.3 \\ d(x_6, t_{C_1}) = 10.8 & d(x_6, t_{C_2}) = 1.9 \end{array}$$

Imamo $C_1 = \{x_1, x_2, x_3, x_4\}$,$C_2 = \{x_5, x_6\}$, $S_1 = 36$,$S_2 = 1$,$t_{C_1} = (3, 4)$, $t_{C_2} = (10.5, 11.5)$. *Dalje nastavljamo dijeliti klaster* C_1. *Dvije najudaljenije točke u klasteru* C_1 *su* x_2, x_3, *stavljamo* $x_2 \in C_1'$, $x_3 \in C_3$. *Za iduću točku* x_1 *je* $d(x_1, x_2) < d(x_1, x_3)$, *te vrijedi* $x_1 \in C_1'$. *Dobivena težišta:* $t_{C_1'} = (1.5, 1.5)$,$t_{C_3} = (4, 7)$. *Za svaku iduću točku izračunati* $t_{C_1'}$ *i* t_{C_3}, *sume kvadrata udaljenosti od točke do težišta* S_1, S_2 *i pridružiti je klasteru koji s njom ima manju sumu kvadratnih udaljenosti od točke do težišta.*

$$x_4: \quad t_{C_1'} = (2.7, 3) \quad t_{C_3} = (4.5, 6.5) \quad S_1 = 22.7 \quad S_2 = 1 \quad => x_4 \in C_3$$

Imamo $C_1 = \{x_1, x_2\}$,$C_2 = \{x_5, x_6\}$,$C_3 = \{x_3, x_4\}$ *Sad provjeravamo za svaku točku skupa X kojem je težištu bliža:*

$$\begin{array}{lll}
d(x_1, t_{C_1}) = 0.7 & d(x_1, t_{C_2}) = 13.4 & d(x_1, t_{C_3}) = 5.7 \\
d(x_2, t_{C_1}) = 0.7 & d(x_2, t_{C_2}) = 13.5 & d(x_1, t_{C_3}) = 6 \\
d(x_3, t_{C_1}) = 6 & d(x_3, t_{C_2}) = 7.9 & d(x_1, t_{C_3}) = 0.7 \\
d(x_4, t_{C_1}) = 5.7 & d(x_4, t_{C_2}) = 7.8 & d(x_1, t_{C_3}) = 0.7 \\
d(x_5, t_{C_1}) = 14.1 & d(x_5, t_{C_2}) = 0.7 & d(x_1, t_{C_3}) = 8.5 \\
d(x_6, t_{C_1}) = 12.8 & d(x_6, t_{C_2}) = 0.7 & d(x_1, t_{C_3}) = 7.1
\end{array}$$

Klasteri su ostali nepromijenjeni, $C_1 = \{x_1, x_2\}, C_2 = \{x_5, x_6\}, C_3 = \{x_3, x_4\}$. *Algoritam staje jer smo došli do željenog broja klastera.*

U korolaru 1.5.3 je pokazano da suma kvadrata udaljenosti između svih parova točaka jednaka umnošku broja točaka i sumi kvadrata udaljenosti od točke do težišta, dakle:

$$\sum_i \sum_{j>i} d^2(x_i, x_j) = n \sum_{i=1}^{n} d^2(x_i, t).$$

Iz toga slijedi da je mješoviti algoritam moguće izvesti i u drugoj verziji, tako da se minimizira funkcija cilja:

$$f(C_1, \ldots, C_k) = \sum_{i=1}^{k} \sum_{x,y \in C_i} d^2(x, y). \tag{2.3}$$

Poglavlje 3

Implementacija i rezultati

3.1 Implementacija Mješovitog algoritma

Za implementaciju mješovitog algoritma korišten je programski jezik Python. Za analizu Mješovitog algoritma se koristi skup podataka koji sadržava 10000 točaka u 15 dimenziji. Promatrani skup podataka je zanimljiv za analizu jer bi se klasteriranjem tih točaka trebalo dobiti 25 klastera s jednakim brojem točaka u svakom klasteru. Primjena mješovitog algoritma na navedeni skup podataka iziskuje veliki broj računskih operacija i manipulaciju s višedimenzionalnim nizovima, te kako je Python interpreterski jezik, programi napisani u njemu se vrše malo sporije nego programi napisani u kompajlerskim jezicima. Stoga primjenom biblioteke kao što je *NumPy* se znatno ubrzava vrijeme izvršavanja algoritma. *NumPy* je biblioteka za programski jezik Python, koja daje podršku za velike višedimenzionalne nizove i matrice, također podržava veliki broj matematičkih funkcija koje se koriste na *NumPy* objektima.

U nastavku su prikazani isječci koda mješovitog algoritma u dvije verzije; prva verzija je s korištenjem *NumPy* biblioteke, druga je pisana bez korištenja dodatnih biblioteka. Varijabla *nopt* označava broj točaka polaznog skupa.

- Inicijalizacija matrice:

```
matrica=numpy.zeros((nopt,nopt))
```

Isječak koda 3.1: Inicijalizacija matrice s *NumPy* bibliotekom

```
matrica=[[0 for i in range(nopt)]*nopt]
```

Isječak koda 3.2: Inicijalizacija matrice bez korištenja dodatnih biblioteka

- Računanje udaljenosti između svih parova točaka:

```
for i in range(nopt):
    for j in range (i,nopt):
        udaljenost = numpy.linalg.norm(x[i]-x[j])
```

Isječak koda 3.3: Računanje udaljenosti s *NumPy* bibliotekom

```
for i in range(nopt):
    for j in range (i,nopt):
        udaljenost=0
        for k in range(len(x[i])):
            udaljenost+=(x[i][k]-x[j][k])**2
        udaljenost=udaljenost**(0.5)
```

Isječak koda 3.4: Računanje udaljenosti bez korištenja dodatnih biblioteka

U tablici 3.1 se vidi koliko je vremena potrebno da se izvedu navedeni isječci koda s opisanim skupom podataka. Za mjerenje vremenskog izvršavanja koda, koristila se biblioteka *timeit* tako da se spremi vrijeme prije i poslije izvršavanja isječka koda. Korištena verzija programa je Python 3.7. Specifikacije korištenog računala su: Procesor- Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.59 GHz, RAM- 8,00 GB i vrsta sustava- 64-bitni operacijski sustav, procesor x64.

Testirani dio		Vrijeme izvršavanja [s]
Inicijalizacija matrice	(3.1)	0.002033600000000008
	(3.2)	0.3334594000000001
Računanje udaljenosti za sve točke	(3.3)	410.3942073
	(3.4)	1455.0492153

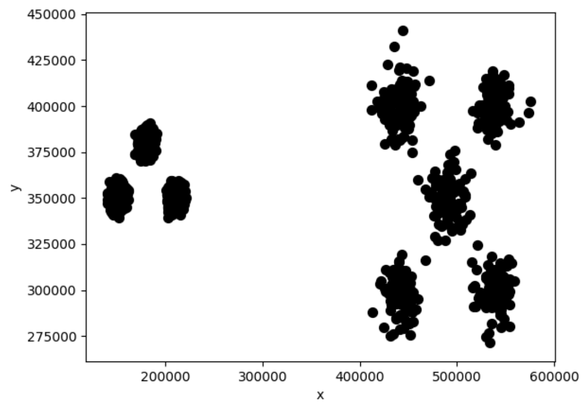
Tablica 3.1: Vrijeme izvršavanja dijelova koda

Također su još korištene biblioteke:

- *StringIO* - biblioteka za čitanje/pisanje u datoteku.
- *matplotlib* - biblioteka koja omogućuje crtanje grafova.

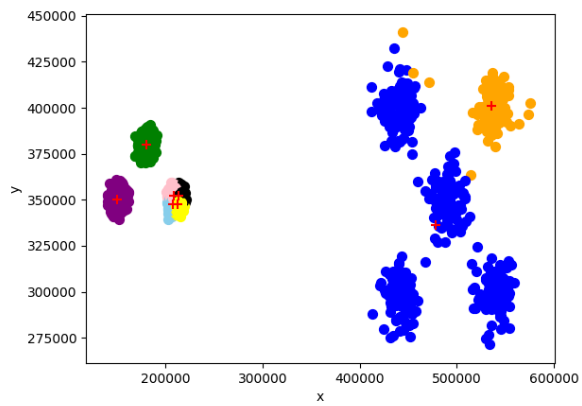
3.2 Primjeri

Primjer 3.2.1. *Polazni skup sadrži 6500 dvodimenzionalnih točaka. Broj klastera je 8, od čega 3 klastera sadrže po 2000 točaka, preostalih 5 klastera sadrže po 100 točaka.*

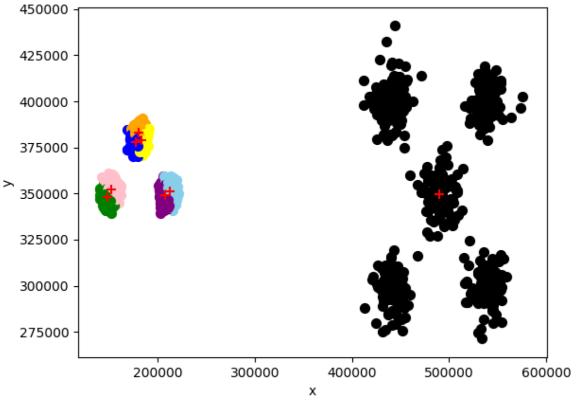


Slika 3.1: Polazni skup

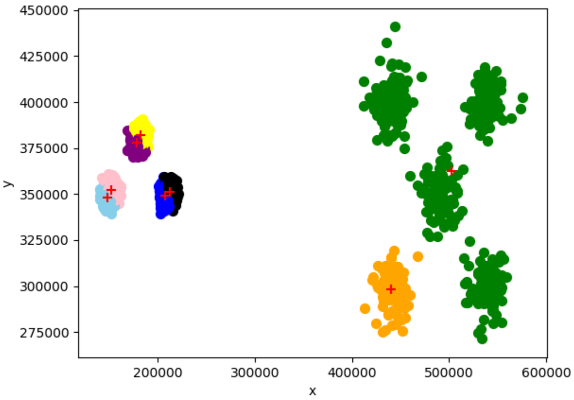
Prvo testiramo Lloydov algoritam opisan u Algoritam 1.3 tako da se na nasumičan način odrede inicijalni centri. Slika 3.2, Slika 3.3 i Slika 3.4 prikazuju dobivene klastere s tri različite polazne konfiguracije. Simbol "+" označava centar klastera, a sve točke istog klastera imaju istu boju.



Slika 3.2: Dobiveni klasteri s 1. polaznom konfiguracijom



Slika 3.3: Dobiveni klasteri s 2. polaznom konfiguracijom



Slika 3.4: Dobiveni klasteri s 3. polaznom konfiguracijom

Broj točaka u svakom klasteru nakon:

- *prve početne konfiguracije: 2000, 396, 567, 2000, 547, 446, 104, 440*
- *druge početne konfiguracije: 961, 737, 609, 981, 1039, 500, 654, 1019*
- *treće početne konfiguracije: 400, 977, 998, 1002, 1043, 1023, 100, 957.*

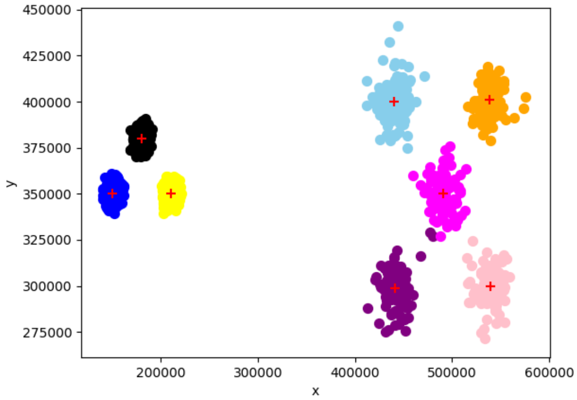
Tablica 3.2 pokazuje dobivene vrijednosti funkcije cilja koristeći Lloydov algoritam s tri različite polazne konfiguracije:

Br. konfiguracije	Vrijednost ciljne funkcije
1	$1.5670084504646458 \times 10^{12}$
2	$2.1719501842064883 \times 10^{12}$
3	$1.5443914537218945 \times 10^{12}$

Tablica 3.2: Vrijednosti ciljne funkcije s različitim polaznim konfiguracijama

Očigledno se Lloydov algoritam zaglavi, dakle nije uspio pronaći optimalnu particiju.

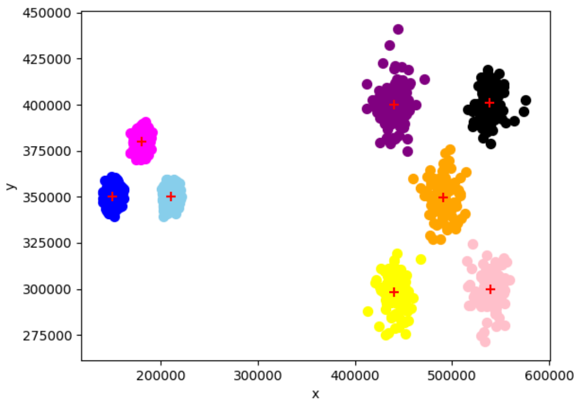
Testiranje mješovitog algoritma opisanog u Algoritam 3 daje klastere prikazane sljedećom slikom:



Slika 3.5: 1. Rezultat primjene mješovitog algoritma

Broj točaka nakon izvršavanja algoritma u svakom klasteru: 2000, 2000, 102, 100, 2000, 100, 100, 98. Vrijednost funkcije cilja je $2.177554430876221 \times 10^{11}$. *Očigledno niti mješoviti algoritam nije uspio odrediti optimalnu particiju.*

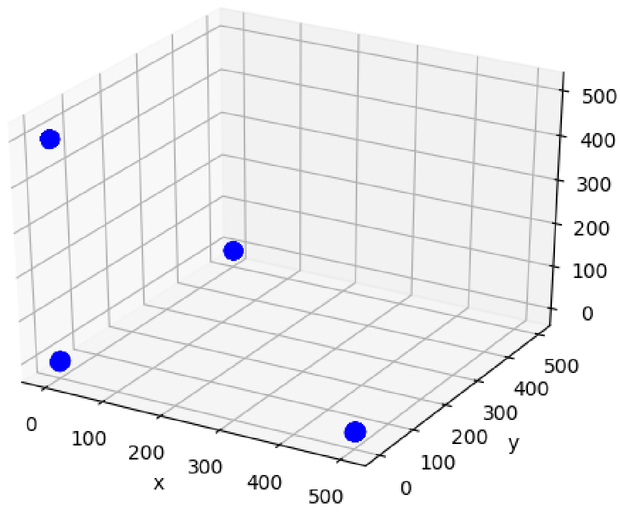
Preostalo je još testirati poboljšanu verziju mješovitog algoritma, gdje je cilj minimizirati funkciju cilja (2.3). Testiranje s ovom verzijom algoritma daje klastere prikazane sljedećom slikom:



Slika 3.6: 2. Verzija mješovitog algoritma

Broj točaka u svakom klasteru nakon izvršavanja algoritma je 2000, 2000, 100, 100, 2000, 100, 100, 100. Vrijednost funkcije cilja:$2.14492062847683 \times 10^{11}$. *Poboljšana verzija mješovitog algoritma je uspjela odrediti optimalnu particiju.*

Primjer 3.2.2. *Polazni skup sadrži 10000 točaka, gdje je svaka točka iz* R^{15}*. Broj klastera je 25, gdje svaki klaster sadrži točno 400 točaka. Slika 3.7 prikazuje prve tri koordinate skupa u trodimenzionalnom grafu.*



Slika 3.7: Prikaz prve tri koordinate skupa

Prvo testiramo Lloydov algoritam opisan u Algoritam 1.3 tako da na nasumičan način se odrede inicijalni centri. Broj točaka u svakom klasteru nakon:

- *prve početne konfiguracije: 1200, 0, 0, 400, 133, 400, 800, 400, 400, 800, 400, 134, 400, 400, 0, 400, 193, 207, 0, 400, 133, 400, 800, 800, 800*
- *druge početne konfiguracije: 400, 800, 400, 400, 400, 800, 400, 400, 400, 400, 400, 800, 400, 400, 0, 400, 400, 0, 400, 0, 400, 400, 400, 400, 400*
- *treće početne konfiguracije: 400, 0, 400, 400, 400, 800, 400, 0, 400, 192, 217, 800, 800, 187, 400, 800, 213, 400, 400, 400, 400, 400, 208, 183, 800.*

Tablica 3.3 pokazuje dobivene vrijednosti funkcije cilja koristeći Lloydov algoritam s tri različite polazne konfiguracije:

Br. konfiguracije	Vrijednost ciljne funkcije
1	$5.335137755309479 \times 10^8$
2	$1.501935017525435 \times 10^8$
3	$3.501123633439091 \times 10^8$

Tablica 3.3: Vrijednosti ciljne funkcije s različitim polaznim konfiguracijama

Očigledno se Lloydov algoritam zaglavi, dakle nije uspio pronaći optimalnu particiju.

Primjenom mješovitog algoritma opisanog u Algoritam 3 vrijednost ciljne funkcije je $1.4930212311320295 \times 10^8$. *Također se dobije ista vrijednost ciljne funkcije ako se minimizira suma kvadrata udaljenosti svih parova točaka (1.12).*

Bibliografija

[1] P. Goldstein, *Bioinformatika*, https://web.math.pmf.unizg.hr/nastava/bioinformatika/, Accessed: 2019-25-8.

[2] M. Kelava, *Algoritam k sredina u prostoru Minkowskog (diplomski rad)*, (2017).

[3] S. Sieranoja i P. Fränti, *Clustering basic benchmark*, http://cs.joensuu.fi/sipu/datasets/, Accessed: 2019-18-10.

[4] S. Šeperić, *Kriteriji kompleksnosti za k-means algoritam (diplomski rad)*, (2014).

Sažetak

U ovom radu su razvijane modifikacije algoritma k-sredina, za koje smo pokazali da uspijevaju odrediti optimalnu particiju skupa podataka i onda kad algoritma k-sredina ne uspijeva. Nadalje, dokazano je da suma kvadrata udaljenosti između svih parova točaka jednaka umnošku broju točaka i sumi kvadrata udaljenosti od točke do težišta. Na kraju, smo testirali modifikacije algoritma k-sredina i usporedili rezultate s algoritmom k-sredina.

Summary

In certain cases, where the k-means algorithm failed, these modified versions were successful in finding the optimal data set partition. We have explored the relationship between the sum of squared distances in a set and the sum of squared distances from the center of mass(of a set). Finally, we tested our modified algorithm on several examples.

Životopis

Rođena sam 8. rujna 1993. godine u Splitu. Nakon završene osnovne škole Spinut u Splitu, upisala sam Prirodoslovnu školu u Splitu. Godine 2014. sam upisala preddiplomski studij Informatike na Prirodoslovno-matematičkom fakultetu u Splitu. Preddiplomski studij sam završila 2017. godine. Iste godine sam upisala diplomski sveučilišni studiji Računarstvo i matematika na Prirodoslovno-matematičkom fakultetu u Zagrebu.