UNIVERZA V LJUBLJANI
FAKULTETA ZA MATEMATIKO IN FIZIKO

Finančna matematika – 1. stopnja

Lena Benčina

Metoda k-voditeljev

Delo diplomskega seminarja

Mentor: izred. prof. dr. Marjetka Krajnc

Ljubljana, 2016

Kazalo

Metoda k-voditeljev

Povzetek

V delu diplomskega seminarja je kot ena izmed popularnejših metod particijskega grupiranja predstavljena kvadratična metoda k-voditeljev, tj. metoda k-voditeljev z uporabo evklidske razdalje. Za dobro delovanje algoritma je potrebna premišljena predpriprava podatkov, zato se najprej posvetimo problemom, kot so manjkajoče vrednosti v podatkih, šumi na podatkih, prevladovanje atributov z veliki razponi vrednosti in problem večdimenzionalnosti.

Glavni del diplomskega seminarja predstavlja izpeljava kvadratične metode k-voditeljev, ki je sestavljena iz dveh delov. Prvi del predstavlja paketna verzija, drugi del pa izboljšava le te, tj. inkrementalna verzija z združitvijo. Kljub enostavni implementaciji metode se pri njeni uporabi soočamo z dvema velikima problemoma, ki imata ključno vlogo pri kvaliteti končne particije; izbira začetne particije in določitev števila gruč. V ta namen so na kratko predstavljene še metode naključno seme, naključno razčlenjevanje, hierarhično razdvajanje na osnovi analize glavnih komponent in komolčna metoda. V zadnjem delu diplomskega seminarja je predstavljena uporaba kvadratične metode k-voditeljev na primeru grupiranja držav glede na pomembnejše ekonomske indikatorje.

k-means algorithm

Abstract

This work presents one of the most popular methods for partitioning clustering, quadratic k-means algorithm, which uses euclidian distance for its similarity measure. Since a large part of dealing with data is preprocessing it, we first focuse on problems such as missing values, noisy data, different scales of attribute values and curse of dimensionality.

Main part of this work features the derivation of quadratic k-means algorithm. First we present the batch k-means algorithm and after the final version of k-means algorithm, which is the merger of batch and incremental version. Despite the simplicity of implementation, we deal with two major issues, which can affect the quality of final partition; choosing the initial partition and determining the number of clusters. For this purpose, a brief introduction of random seed, random partitioning, PCA based divisive hierarchical approach and elbow method is given. In the last part, reader is introduced to the practical use of k-means algorithm by clustering countries using some of the important economic indicators.

Math. Subj. Class. (2010): 68P10, 68W40

Ključne besede: grupiranje, metoda k-voditeljev, evklidska razdalja, center, optimalna particija

Keywords: clustering, k-means algorithm, euclidean distance, centroid, optimal partition

1. Uvod

Zaradi eksponente tehnološke rasti se iz dneva v dan veča potreba po avtomatizirani obdelavi podatkov, kar je razlog za prevladovanje uporabe informacijske tehnologije v podjetjih, kot tudi pri posameznikih. Na tem mestu se lahko kot pomembni veji analize podatkov zatečemo k podatkovnem rudarjenju, katerega uporaba je v zadnjih letih vedno bolj priljubljena. Uporablja se na raznih področjih, kot so marketing, zdravstvo, proizvodni procesi, upravljanje odnosov s strankami, odkrivanje prevar, bančništvo in bioinformatika. Ena pomembnejših nalog podatkovnega rudarjenja je grupiranje, ki prav tako postaja vedno bolj prisotno na omenjenih področjih. Za lažje razumevanje uporabe metode k-voditeljev sledi opis podatkovnega rudarjenja in grupiranja ter umestitev metode v omenjena pojma.

2. Podatkovno rudarjenje in grupiranje

Podatkovno rudarjenje (ang. data mining) je proces analiziranja in iskanja vzorcev v naboru podatkov, ki na prvi pogled niso vidni. Uporaba le tega zaradi fleksibilnosti postane zelo pomembna, ko statistična analiza ni dovolj, z drugimi besedami, ko s pomočjo samih statističnih metod v naboru ne najdemo koristnih informacij. Iskanje koristnih informacij ima več pomenov, in sicer lahko predstavlja odkrivanje anomalij v naboru podatkov (ang. deviation detection), iskanje povezanosti spremenljivk (ang. association rule learning), razvrščanje nabora podatkov v že poznane strukture (ang. classification), iskanje funkcije, ki pojasnjuje podatke s čim manjšo napako (ang. regression), zagotavljanje bolj kompaktne predstavitve nabora podatkov (ang. summarization) in grupiranje v še neznane strukture (ang. clustering). Naštete naloge pa so le del celotnega procesa iskanja informacij, ki sestoji iz naslednjih faz.

(1) Opredelitev cilja procesa z vidika tistega, ki bo uporabil rezultate.
(2) Razumevanje podatkovne množice in zahtevanega znanja za nadaljno obdelavo.
(3) Izbira in predstavitev ciljnega nabora, na katerem se izvajajo nadaljni koraki procesa. Ciljni nabor je lahko le del celotnega nabora podatkov.
(4) Predprocesiranje izbranega nabora s primernimi metodami.
(5) Pregled in izbira naloge rudarjenja, ki se ujema s cilji, ki so bili zastavljeni na začetku (npr. grupiranje).
(6) Izbira algoritma (npr. metoda k-voditeljev) ter določanje optimalnih parametrov v modelu.
(7) Interpretacija dobljenih informacij.
(8) Vključitev pridobljenega znanja v sistem za nadaljno uporabo.

Pomembno se je zavedati, da ni univerzalnega pravila za dobro analizo, saj raznolikost podatkov zahteva različne pristope za vsako od omenjenih faz. Vzorci najdeni s podatkovnim rudarjenjem lahko predstavljajo končen rezultat, velikokrat pa služijo nadaljnim analizam, kot na primer strojnemu učenju. Podatkovno rudarjenje je v tesni povezavi s strojnim učenjem, ki prav tako predstavlja iskanje informacij v podatkovnih množicah, vendar uporablja drugačen pristop. Grupiranje uvrščamo med nenadzorovano strojno učenje, ki se od nadzorovanega razlikuje v tem, da ima

slednji prisotno ciljno spremenljivko, s pomočjo katere nadzoruje učenje. Učinkovitost končnega rezultata je tako lažje ocenjena, medtem ko pri nenadzorovanem strojnem učenju ni povratne informacije o končem rezultatu.

Grupiranje, kot ena izmed pomembnejših metod podatkovnega rudarjenja, je razvrščanje podatkovne množice v gruče (ang. clusters), ki opisujejo lastnosti pripadajočih elementov. Cilj je, da so si elementi v posamezni gruči bolj podobni med seboj, kot so podobni elementom v ostalih gručah in da je vsota razdalj med elementi v posameznih gručah čim manjša. Večja kot je podobnost oz. homogenost v posamezni gruči in bolj kot se razlikujejo gruče med seboj, boljše in bolj izrazito je grupiranje. Obstajajo številni algoritmi, ki rešujejo problem grupiranja, ki pa se močno razlikujejo med seboj. Po najbolj osnovni delitvi se grupiranje deli na hierarhično in particijsko (ang. hierarhical and partitioning clustering). Algoritmi za hierarhično grupiranje razbijejo podatke v hierarhično gnezdene gruče, particijski pa jih razdelijo v enonivojske medsebojno disjunktne gruče. Razvrščanje z metodo k-voditeljev uvrščamo med particijske algoritme.

3. Uvod v metodo k-voditeljev

Metoda k-voditeljev je eden izmed popularnejših algoritmov za grupiranje podatkov. Glavna razloga za to sta sigurno preprostost implementacije in hitrost računanja. Predpostavimo, da imamo n podatkov, ki jih želimo razvrstiti v k disjunktnih gruč, tako da so elementi v posamezni gruči bolj podobni ostalim elementom v isti gruči kot tistim v ostalih gručah. Algoritem na vhodu sprejme nabor podatkov in parameter k, ki predstavlja število gruč, vrne pa particijo, tj. informacijo o pripadajočih gručah za vsak podatek. V grobem algoritem deluje na naslednji način.

(1) Izberemo k naključnih točk znotraj prostora določenega s podatkovno množico in jih določimo za centre posameznih gruč.
(2) Vsakemu podatku pripišemo najbližji center in s tem pripadajočo gručo.
(3) Preračunamo pozicije centrov in s tem zmanjšamo razpršenosti posameznih gruč.
(4) Ponavljamo koraka (2) in (3) dokler centri ne ostanejo nespremenjeni oz. dokler ni dosežen zaustavitveni pogoj.

Algoritem s pomočjo iteracij minimizira objektno funkcijo, ki predstavlja vsoto kvadratov razdalj med elementi v posameznih gručah. Torej, manjša kot je vsota razdalj znotraj posamezne gruče, manjša je razpršenost gruč, bolj učinkovito je grupiranje. Čeprav se algoritem vedno ustavi, ni nujno da vedno doseže optimalno rešitev. Minimizacija objektne funkcije je dosežena s pomočjo gradientnega spusta, kar pomeni, da se na vsakem koraku premaknemo v smeri gradienta, kar nas pripelje do lokalnega minimuma, ne pa nujno do globalnega. Rešitev za to je izbira drugih začetnih centrov. Drug problem, ki se pojavi, je izbira optimalnega števila gruč. Kako izbrati začetne centre in parameter k je opisano v poglavju 6.

Iskanje najbližjega centra je odvisno od tega, kako merimo podobnost podatkov v posameznih gručah. Obstaja več verzij algoritma, ki se razlikujejo v meri podobnosti. Najbolj osnovna verzija algoritma uporablja kvadrat evklidske razdalje. Posledično je za določitev novega centra za določeno gručo izračunana aritmetična sredina vseh podatkov v tej gruči. Razvrščanje z metodo k-voditeljev z uporabo kvadratne evklidske razdalje je podrobneje opisano v poglavju 5. Še ena pomembnejših

verzij algoritma je sferična metoda k-voditeljev, ki za razdaljo uporabi kosinusno mero, izpeljava le tega pa je zahtevnejša. Obstajajo tudi druge mere, ki se lahko uporabijo, kot na primer L^1 in še nekatere druge.

Očitno je, da morajo biti vrednosti vseh spremenljivk numerične. V primeru kategoričnih vrednosti se pojavi problem, saj metoda k-voditeljev z evklidsko razdaljo ne zna določiti, kako podobne oz. različne so si vrednosti med seboj. V takem primeru se raje zatečemo h kakšni drugi metodi, ki na takšne vrednosti ni občutljiva.

4. Priprava podatkov

Pomemben del pri uporabi algoritma je priprava podatkov, saj lahko le ta drastično vpliva na končen rezultat.

4.1. **Predstavitev podatkov.**

Prvo vprašanje, s katerim se srečujemo pri pripravi podatkov je, kako jih bomo predstavili. Množico podatkov predstavimo s pomočjo matrike podatkov velikosti $m \times n$, kjer vsaka izmed m-tih vrstic predstavlja en podatek in vsak izmed n-tih stolpcev en atribut. Grafično pa jih ponavadi prikažemo kar s točkami (vektorji), kjer vsaka dimenzija predstavlja eno spremenljivko oz. atribut, ki opisuje podatek. Podatki predstavljeni z matriko

$$M = \begin{bmatrix} -3 & 0 \\ -2 & -1 \\ -2 & 1 \\ -1 & 0 \\ 1 & 0 \\ 2 & -1 \\ 2 & 1 \\ 3 & 0 \end{bmatrix}$$

so grafično prikazani na sliki 1.

Drug način predstavitve podatkov je s pomočjo razdaljne matrike (ang. proximity matrix), kjer so zapisane razdalje za vsak par podatkov. Ker metoda k-voditeljev v vsaki iteraciji preračunava centre gruč, implementacija z razdaljno matriko potrebuje nekaj dodatnega računanja s pomočjo kvadratne norme, kar pa poveča časovno zahtevnost. Vse verzije algoritma v diplomskem seminarju bodo podatke sprejele v obliki matrike podatkov ([7]).

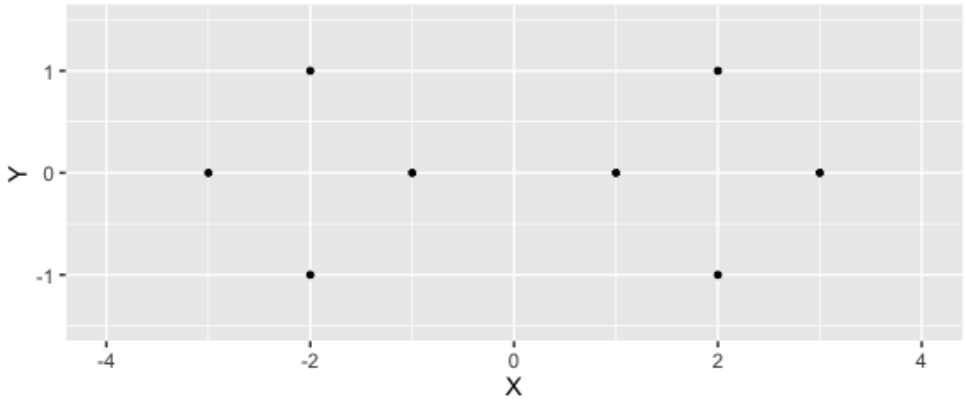
4.2. **Čiščenje.**

Čiščenje podatkov predstavlja najpomembnejšo nalogo pri predprocesiranju podatkov, saj se v realnem življenju velikokrat soočamo z 'umazanimi' podatki.

V podatkih, ki jih želimo analizirati, se pogosto pojavljajo manjkajoče vrednosti. Prvi in najenostavnejši način, kako se soočiti z manjkajočimi vrednostmi, je brisanje. V matriki podatkov poiščemo manjkajoče vrednosti in izbrišemo vse nepopolne vrstice, kar pa seveda lahko privede do neefektivnega grupiranja, še posebej, če je število le teh relativno veliko. V primeru, da se pojavi relativno malo manjkajočih vrednostih, je brisanje najboljša izbira.

Drug način soočanja z omenjenim problemom je imputacija, ki dopolni manjkajoče vrednosti z smiselnimi ocenami. V primerjavi z brisanjem so tovrstne metode

Slika 1. Graf podatkov.



bistveno bolj zapletene, saj efektivna imputacija zahteva informacije o skupni porazdelitvi manjkajočih vrednosti in podatkov. Čeprav so verjetnostni modeli lahko zelo učinkoviti, če uporabimo pravilne predpostavke, so izračuni le teh lahko zelo zamudni.

Pogosto se soočamo tudi s šumi na podatkih (ang. noisy data), kar pomeni, da se v podatkih pojavljajo naključne napake ali pa osamelci (ang. outliers). Podobno kot pri problemu manjkajočih vrednostih se relativno mali šumi lahko izbrišejo, če le te opazimo, vendar to ni najbolj učinkovit pristop, saj le te težko najdemo brez ustrezne metode. Metoda k-voditeljev je zelo občutljiva na šume, saj minimizira vsoto kvadratov razdalj in tako npr. osamelci, ki so zelo oddaljeni, veliko doprinesejo k objektni funkciji. Obstajajo številni pristopi, kako se soočiti z omenjenim problemom. Eden izmed njih je, da 'kaznujemo' primere, ki jih zaznamo kot šume, kar pomeni, da jim pripišemo manjšo težo in s tem zmanjšamo njihov vpliv na objektno funkcijo.

4.3. **Normalizacija.**

Velik del priprave podatkov je transformacija le teh. Razlog za to so lahko različno merjene vrednosti posameznih atributov. V primerih, ko razpon podatkov močno varira od atributa do atributa, lahko to vpliva na končen rezultat grupiranja, saj lahko en atribut prevlada nad drugim. V takih primerih je nujno potrebna normalizacija podatkov, saj prepreči, da atribut z velikim razponom, kot je na primer dohodek, prevlada nad atributom z majhnim razponom, kot je na primer starost. Cilj je, da izenačimo velikosti razponov in s tem odpravimo raznolikost atributov. Še posebej je to potrebno, če se podatke grupira z uporabo razdalje, ki je občutljiva na razlike v razponih atributov. Taka je na primer evklidska razdalja. Podatke lahko normaliziramo na naslednje načine.

Min-max normalizacija je linearna transformacija originalnih podatkov. Naj bo $\min_X$ vrednost najmanjšega podatka atributa X in $\max_X$ vrednost največjega. Transformacija za vrednost v je naslednje oblike:

$$\widetilde{v} = \frac{v - \min_X}{\max_X - \min_X}. \tag{1}$$

Po uporabi (1) so transformirane vrednosti na intervalu $[0, 1]$.

Pri Z-score normalizaciji so vrednosti atributa X transformirane s pomočjo povprečja in standardnega odklona atributa X. Vrednost v atributa X je normalizirana z izračunom

$$\widetilde{v} = \frac{v - \overline{X}}{\sigma_X}, \tag{2}$$

kjer $\overline{X}$ predstavlja povprečno vrednost, σ_X pa standardni odklon atributa X. Po uporabi (2) vrednosti atributa pridobijo lastnosti standardne normalne porazdelitve, kar pomeni, da je $\overline{X} = 0$ in $\sigma_X = 1$.

Normalizacija z decimalnim normiranjem vrednosti normalizira s premikanjem decimalne vejice vrednosti atributa X. Število mest za decimalno vejico je odvisno od absolutne maksimalne vrednosti atributa X. Normalizacija je oblike

$$\widetilde{v} = \frac{v}{10^j}, \tag{3}$$

kjer j predstavlja najmanjše celo število, da je $\max(|\widetilde{v}|) < 1$. Transformirane vrednosti po uporabi (3) ležijo na intervalu [-1,1]

Izbira metode je odvisna zgolj od podatkov. Metoda (1) ne doprinese k dobrim rezultatom, če podatki niso enakomerno porazdeljeni. Metodi (1) in (2) sta zelo občutljivi na osamelce, čeprav včasih osamelce odstranimo že pred normaliziranjem. Metoda (2) je še posebej uporabna, če ne poznamo minimalne in maksimalne vrednosti ([7]).

4.4. **Redukcija.**

Razvrščanje z metodo k-voditeljev na visokorazsežni podatkovni množici včasih ne doseže učinkovitega grupiranja. V takem primeru lahko nabor zreduciramo, to pomeni, da izločimo nepotrebne atribute. To lahko storimo s pomočjo različnih statističnih metod. Ena izmed popularnejših je analiza glavnih komponent (ang. principal component analysis), ki s pomočjo lastnih vrednosti variančno kovariančne matrike odstrani spremenljivke, ki pojasnjujejo najmanjši delež variance ([10]).

5. Kvadratična metoda k-voditeljev

Vse izpeljave in primeri v tem poglavju so povzeti po viru [1].

5.1. **Splošna formulacija problema.**

Z $A = \{\mathbf{a}^1, \mathbf{a}^2, \ldots, \mathbf{a}^m\} \subseteq S \subseteq \mathbb{R}^n$ označimo množico podatkov, kjer $\mathbf{a}^i = (a_1^i, \ldots, a_n^i)^T \in \mathbb{R}^n$ predstavlja en podatek. Cilj je razvrstiti podatke v particijo, sestavljeno iz k disjunktnih gruč. To particijo označimo s $\Pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$, zanjo pa mora veljati

$$\pi_1 \cup \ldots \cup \pi_k = A,\ \pi_i \subseteq A \quad \text{in} \quad \pi_i \cap \pi_j = \emptyset \text{ za } i \neq j.$$

Ena pomembnejših komponent samega algoritma je mera podobnosti oz. razdalja, katero označimo z $d(u, v)$. Zanjo zahtevamo, da zadošča naslednjemu aksiomu razdalje,

$$d(u, v) \geq 0 \text{ za vse } (u, v) \in S, \quad d(u, v) = 0 \Longleftrightarrow u = v,$$

ni pa nujno, da zadošča simetričnosti ter trikotniški neenakosti. Zato pojem razdalje zaradi zlorabe notacije zamenjamo s pojmom razdaljne funkcije (ang. distance-like function).

Kako dobro je grupiranje merimo s funkcijo Q: $A \to \mathbb{R}$, ki ji pravimo tudi kvaliteta particije, za katero velja

$$Q(\Pi) = Q(\pi_1) + \ldots + Q(\pi_k).$$

V nadaljevanju bomo Q imenovali objektna funkcija. Naš problem se nanaša na iskanje optimalne particije $\Pi^{\min} = \pi_1^{\min} \cup \ldots \cup \pi_k^{\min}$, tj. tiste particije, ki minimizira objektno funkcijo. Za opis povezave objektne funkcije z razdaljno funkcijo definiramo center gruče π z naslednjo zvezo:

$$\mathbf{c} = \mathbf{c}(\pi) = \arg\min_{\mathbf{x} \in S} \sum_{\mathbf{a} \in \pi} d(\mathbf{x}, \mathbf{a}).$$

Definiramo $Q(\pi)$ kot vsoto razdalj med centrom in podatki gruče π:

$$Q(\pi) = \sum_{\mathbf{a} \in \pi} d(\mathbf{c}(\pi), \mathbf{a}). \tag{4}$$

Povezave med gručami in njihovimi centri lahko zdaj opišemo na dva načina.

(1) Za dano particijo $\{\pi_1, \pi_2, \ldots, \pi_k\}$ množice A definiramo pripadajoče centre $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ z

$$\mathbf{c}_i = \mathbf{c}(\pi_i) = \arg\min_{\mathbf{x} \in S} \sum_{\mathbf{a} \in \pi_i} d(\mathbf{x}, \mathbf{a}). \tag{5}$$

(2) Za vsako množico k centrov $\{\mathbf{c}_1, \ldots, \mathbf{c}_k\}$ definiramo particijo $\{\pi_1, \pi_2, \ldots, \pi_k\}$ množice A z

$$\pi_i = \{\mathbf{a} : \mathbf{a} \in A,\ d(\mathbf{c}_i, \mathbf{a}) \leq d(\mathbf{c}_\ell, \mathbf{a}),\ \ell = 1, \ldots, k,\ \ell \neq i\}. \tag{6}$$

Problem, ki ga rešujemo, je torej minimizacija objektne funkcije. Opišemo ga z zvezo

$$\min Q(\Pi) = \min \sum_{i=1}^{k} Q(\pi_i) = \min \sum_{i=1}^{k} \sum_{\mathbf{a} \in \pi_i} d(\mathbf{c}_i, \mathbf{a}),$$

pri čemer iščemo minimum po vseh možnih particijah množice A.

5.2. **Paketna verzija metode k-voditeljev.**

Osnovna verzija algoritma, ki ji pravimo tudi paketna verzija, za doseg optimalne particije iterira med prej omenjenima korakoma (5) in (6). Medtem ko korak 2-(6) ni preveč zahteven za izračun, korak 1-(5) predstavlja optimizacijski problem z omejitvami. Stopnja težavnosti je odvisna od izbire razdalje $d(\cdot, \cdot)$ in podatkovne množice S. V nadaljevanju naj bo $S = \mathbb{R}^n$. Najbolj osnovna verzija algoritma za razdaljo uporabi kvadrat evklidske razdalje,

$$d(\mathbf{x}, \mathbf{a}) = \|\mathbf{x} - \mathbf{a}\|^2 = |x_1 - a_1|^2 + \ldots + |x_n - a_n|^2,$$

kjer je $\mathbf{x} = (x_1, \ldots, x_n)^T$ in $\mathbf{a} = (a_1, \ldots, a_n)^T$. Očitno je, da d ne zadošča trikotniški enakosti, saj je npr.

$$1 = d(0, 1) > d(0, 0.5) + d(0.5, 1) = 0.5.$$

Za lažjo predstavo naj bo $\{a^1, \ldots, a^m\}$ množica skalarjev in δ kvadratna funkcija skalarja x oblike

$$\delta(x) = \sum_{i=1}^{m} |x - a^i|^2 = mx^2 - 2\left(\sum_{i=1}^{m} a^i\right)x + \sum_{i=1}^{m} (a^i)^2. \tag{7}$$

Odvajamo (7) in dobimo $\delta'(x) = 2mx - 2\sum_{i=1}^{m} a^i$. Ker je $\delta(x)$ konveksna funkcija, je $\delta(c) = \min_x \delta(x)$ natanko tedaj, ko $\delta'(c) = 0$. Obrnemo enačbo in dobimo

$$c = \frac{a^1 + \ldots + a^m}{m}, \tag{8}$$

kar pa je ravno aritmetična sredina vrednosti podatkov.

Naj bo sedaj $A = \{\mathbf{a}^1, \ldots, \mathbf{a}^m\}$ množica n-dimenzionalnih vektorjev. Če je

$$\sum_{i=1}^{m} \|\mathbf{x} - \mathbf{a}^i\|^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} (\mathbf{x}_j - \mathbf{a}^i_j)^2 = \sum_{j=1}^{n}\left(\sum_{i=1}^{m} (\mathbf{x}_j - \mathbf{a}^i_j)^2\right),$$

potem zveza (8) velja za vsako komponento in dobimo

$$\mathbf{c} = \mathbf{c}(A) = \frac{\mathbf{a}^1 + \ldots + \mathbf{a}^m}{m}. \tag{9}$$

Naj bo $\Pi = \{\pi_1, \ldots, \pi_k\}$ particija množice A s pripadajočimi centri $\mathbf{c}_i = \mathbf{c}(\pi_i)$ za $i = 1, \ldots, k$. Zvezo (4) lahko sedaj zapišemo kot

$$Q(\Pi) = \sum_{i=1}^{k}\sum_{\mathbf{a}\in\pi_i} \|\mathbf{c}_i - \mathbf{a}\|^2.$$

Za podatek $\mathbf{a} \in \pi_i \subseteq A$ definiramo indeks i gruče π_i s $\texttt{trenutni}(\mathbf{a})$ in indeks j, indeks najbližjega centra podatka $\mathbf{a}$, z $\texttt{min}(\mathbf{a})$, tako da velja

$$\|\mathbf{c}_{\texttt{min}(\mathbf{a})} - \mathbf{a}\| = \|\mathbf{c}_j - \mathbf{a}\| \leq \|\mathbf{c}_\ell - \mathbf{a}\|,\ \ell = 1, \ldots, k,\ \ell \neq j.$$

Definiramo naslednjo particijo $\texttt{nextBKM}(\Pi) = \Pi' = \{\pi'_1, \ldots, \pi'_k\}$ kot

$$\pi'_i = \{\mathbf{a} : \texttt{min}(\mathbf{a}) = i\}.$$

Trditev 5.1. *Za particijo* Π *in naslednjo particijo* $\texttt{nextBKM}(\Pi)$ *velja*

$$Q(\Pi) \geq Q(\texttt{nextBKM}(\Pi)). \tag{10}$$

Dokaz. Res,

$$Q(\Pi) = \|\mathbf{c}_{\texttt{trenutni}(\mathbf{a}^1)} - \mathbf{a}^1\|^2 + \cdots + \|\mathbf{c}_{\texttt{trenutni}(\mathbf{a}^m)} - \mathbf{a}^m\|^2.$$

Če označimo

$$q(\Pi) = \|\mathbf{c}_{\texttt{min}(\mathbf{a}^1)} - \mathbf{a}^1\|^2 + \cdots + \|\mathbf{c}_{\texttt{min}(\mathbf{a}^m)} - \mathbf{a}^m\|^2,$$

potem zaradi $\|\mathbf{c}_{\texttt{trenutni}(\mathbf{a}^i)} - \mathbf{a}^i\|^2 \geq \|\mathbf{c}_{\texttt{min}(\mathbf{a}^i)} - \mathbf{a}^i\|^2,\ i = 1, \ldots, k$ velja

$$Q(\Pi) \geq q(\Pi). \tag{11}$$

Za vsak $\mathbf{a} \in \pi'_1$ je $\texttt{min}(\mathbf{a}) = 1$ in $\mathbf{c}_{\texttt{min}(\mathbf{a})} = \mathbf{c}_1$. Še več,

$$\begin{aligned}\sum_{\mathbf{a}\in\pi'_1} \|\mathbf{c}_{\texttt{min}(\mathbf{a})} - \mathbf{a}\|^2 &= \sum_{\mathbf{a}\in\pi'_1} \|\mathbf{c}_1 - \mathbf{a}\|^2 \geq \sum_{\mathbf{a}\in\pi'_1} \|\mathbf{c}(\pi'_1) - \mathbf{a}\|^2 \\ &= \sum_{\mathbf{a}\in\pi'_1} \|\mathbf{c}'_1 - \mathbf{a}\|^2 = Q(\pi'_1).\end{aligned}$$

Očitno je, da neenakost drži, če 1 zamenjamo s katerimkoli indeksom $\ell = 2, \ldots, k$, zato je

$$\begin{aligned}q(\Pi) &= \sum_{\ell=1}^{k}\sum_{\mathbf{a}\in\pi'_\ell} \|\mathbf{c}_{\texttt{min}(\mathbf{a})} - \mathbf{a}\|^2 \geq \sum_{\ell=1}^{k}\sum_{\mathbf{a}\in\pi'_\ell} \|\mathbf{c}'_\ell - \mathbf{a}\|^2 \\ &= Q(\texttt{nextBKM}(\Pi)).\end{aligned} \tag{12}$$

Neenakosti (11) in (12) dokazujeta trditev 5.1. □

ALGORITEM 1. Paketna verzija metode k-voditeljev.

VHOD: matrika podatkov, k (število gruč), `tol` (toleranca oz. zaustavitveni pogoj)
IZHOD: particija podatkov

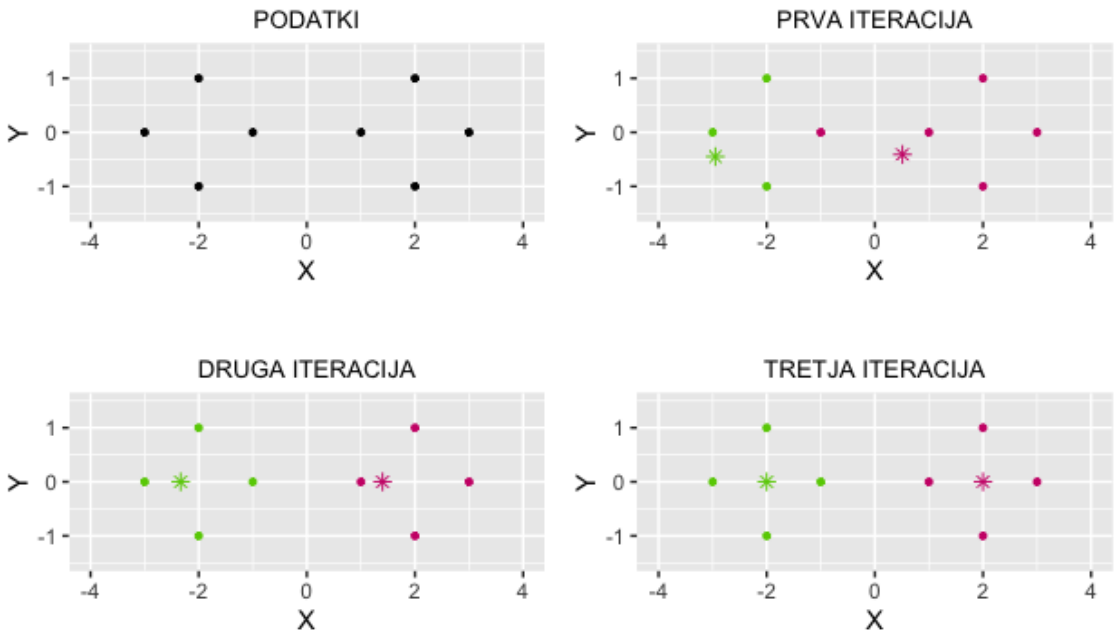
1 Določi poljubno začetno particijo $\Pi^{(0)} = \{\pi_1^{(0)}, \ldots, \pi_k^{(0)}\}$.
Nastavi $t = 0$.

2 Generiraj particijo `nextBKM`$(\Pi^{(t)})$.
if $[Q(\Pi^{(t)}) - Q($`nextBKM`$(\Pi^{(t)})) >$ `tol`$]$
nastavi $\Pi^{(t+1)} =$ `nextBKM`$(\Pi^{(t)})$
$t = t + 1$
Vrni se na korak 2.

3 **stop**

Grafi na sliki 2 ponazarjajo iteracije algoritma za $k = 2$.

SLIKA 2. Iteracije paketne verzije metode k-voditeljev. Centri so označeni z zvezdicami.



Pod blagimi predpostavkami ima končna particija, ki jo generira zgornji algoritem s `tol` $= 0$, konveksne lastnosti. Če so centri $\{\mathbf{c}(\pi_1), \ldots, \mathbf{c}(\pi_k)\}$ paroma različni, potem za vsak par centrov $\mathbf{c}(\pi_i), \mathbf{c}(\pi_j)$ obstaja daljica, ki ju povezuje. Hiperravnina

H_{ij}, ki gre skozi točko $\frac{\mathbf{c}(\pi_i)+\mathbf{c}(\pi_j)}{2}$ in je pravokotna na daljico, razdeli prostor na dva podprostora H_{ij}^- in H_{ij}^+. Če poenostavimo, naj bo $\mathbf{c}_i = \mathbf{c}(\pi_i) \in H_{ij}^-$ in $\mathbf{c}_j = \mathbf{c}(\pi_j) \in H_{ij}^+$. Torej vsak $\mathbf{a} \in \pi_i$ pripada H_{ij}^- in vsak $\mathbf{a} \in \pi_j$ pripada H_{ij}^+. Če predpostavimo še, da sta za vsak par indeksov i, j, bodisi π_i in H_{ij}, bodisi π_j in H_{ij} disjunktni, potem sledi naslednja lema.

Lema 5.2. *Za vsak* $1 \leq i \neq j \leq k$ *je* $conv\{\pi_i\} \cap conv\{\pi_j\} = \emptyset$*, kjer* $conv\{\pi_i\}$ *predstavlja konveksno ovojnico točk iz* π_i*.*

Pod predpostavko, da končna particija vsebuje k različnih centrov, je celoten prostor $\mathbb{R}^n$ razdeljen v regije $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$, tako da je $\mathbf{c}_i$ najbližji center elementom iz $\mathcal{V}_i$. Množici teh regij rečemo tudi Voronojeva particija definirana s centri. Particije generirane z zgornjim algoritmom ne zadoščajo nujno predpostavkama, ki ju potrebujemo za lemo 5.2 in niso nujno separabilne, medtem ko particije generirane z izboljšano verzijo algoritma, predstavljeno v podpoglavju 5.3, vedno zadoščajo omenjenim predpostavkam in so vedno separabilne.

Nekatere pomanjkljivosti osnovne paketne verzije algoritma so naslednje.

- Izbrati je potrebno optimalno število gruč.
- Izbrati je potrebno začetno particijo $\Pi^{(0)} = \{\pi_1^0, \ldots, \pi_k^0\}$.
- Končna particija, ki jo vrne algoritem, ne vsebuje nujno k gruč. Včasih algoritem generira prazne gruče (primer 5.3).
- Generirana končna particija ponavadi ni optimalna (primer 5.4).

Primer 5.3. Generiranje praznih gruč

Naj bo $A = \{-7, -5, -4, 4, 5, 7\}$ množica skalarjev, ki jo želimo razvrstiti v tri disjunktne gruče in $\Pi^{(0)} = \{\pi_1^{(0)}, \pi_2^{(0)}, \pi_3^{(0)}\}$ začetna particija,

$\pi_1^{(0)}$	$\pi_2^{(0)}$	$\pi_3^{(0)}$
-7,-5	-4,4	5,7

.

Iteracija paketne verzije nam da končno particijo $\Pi^{(1)} = \{\pi_1^{(1)}, \pi_2^{(1)}, \pi_3^{(1)}\}$ oblike

$\pi_1^{(1)}$	$\pi_2^{(1)}$	$\pi_3^{(1)}$
-7,-5,-4	$\emptyset$	4,5,7

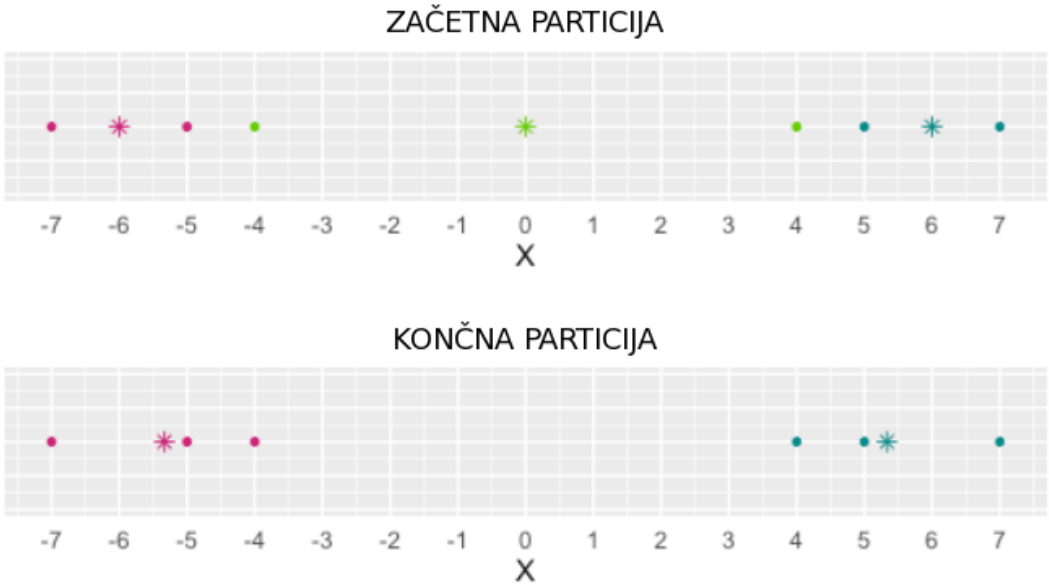
.

Opazimo, da je druga gruča prazna, kar pomeni, da se je število nepraznih gruč zmanjšalo. Na sliki 3 je grafično prikazana sprememba gruč, ki se zgodi v prvi iteraciji algoritma. ♢

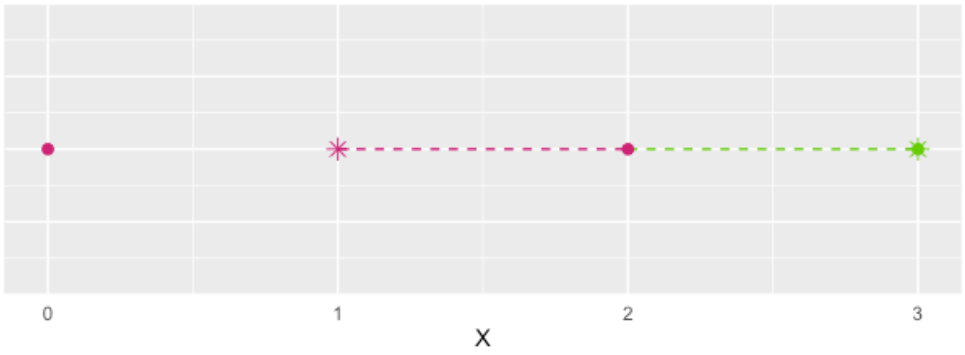
Primer 5.4. Neoptimalna končna particija

Naj bo $A = \{0, 2, 3\}$ množica skalarjev in $\Pi^{(0)} = \{\pi_1^0, \pi_2^0\}$ začetna particija, kjer je $\pi_1^{(0)} = \{0, 2\}$ in $\pi_2^{(0)} = \{3\}$. Iteracija osnovne verzije algoritma ne spremeni začetne particije. Iz slike 4 je razvidno, da bi bila bolj optimalna particija $\Pi^{(1)} = \{\{0\}, \{2, 3\}\}$, ki pa je algoritem na najde, saj je $d(2, c_1) = d(2, c_2)$, kjer c_1 predstavlja center gruče π_1, c_2 pa center gruče π_2. ♢

Slika 3. Sprememba števila gruč.



Slika 4. Neoptimalna končna particija.



5.3. **Inkrementalna verzija metode k-voditeljev in združitev.**

Druga verzija algoritma je inkrementalna verzija, ki skupaj z osnovno verzijo, opisano v prejšnjem poglavju, tvori končno verzijo algoritma. Najprej si bomo pogledali kako izpeljati formulo za spremembo objektne funkcije Q pri premiku vektorja iz ene gruče v drugo.

Naj bo $A = \{a^1, \ldots, a^m\}$ množica skalarjev. Spomnimo se kvadratne funkcije $\delta(x)$ podane z (7). Za vsak x je

$$\delta'(x) = 2mx - 2\sum_{i=1}^{m} a^i, \quad \delta''(x) = 2m \quad \text{in} \quad \frac{d^\ell}{dx^\ell}\delta(x) = 0, \quad \ell = 3, 4, \ldots$$

Nadalje za $c = c(\{a^1, \ldots, a^m\})$ in vsak x velja

$$\begin{aligned}\sum_{i=1}^{m} |x - a^i|^2 = \delta(x) &= \delta(c) + \delta'(c)(x-c) + \frac{1}{2}\delta''(c)(x-c)^2 \\ &= \sum_{i=1}^{m} |c - a^i|^2 + m|c - x|^2.\end{aligned} \tag{13}$$

Iz (13) dobimo

$$\sum_{i=1}^{m} \|\mathbf{x} - \mathbf{a}^i\|^2 = \sum_{i=1}^{m} \|\mathbf{c} - \mathbf{a}^i\|^2 + m\|\mathbf{c} - \mathbf{x}\|^2, \tag{14}$$

od koder sledi naslednja lema.

Lema 5.5. *Če sta* $A = \{\mathbf{a}^1, \ldots, \mathbf{a}^p\}$ *in* $B = \{\mathbf{b}^1, \ldots, \mathbf{b}^q\}$ *disjunktni množici iz* $\mathbb{R}^n$*, potem je*

$$Q\Big(A \bigcup B\Big) = Q(A) + Q(B) + p\|\mathbf{c} - \mathbf{c}(A)\|^2 + q\|\mathbf{c} - \mathbf{c}(B)\|^2, \tag{15}$$

kjer je

$$\mathbf{c} = \mathbf{c}\Big(A \bigcup B\Big) = \frac{p}{p+q}\mathbf{c}(A) + \frac{q}{p+q}\mathbf{c}(B).$$

Dokaz. Z uporabo (14) dobimo

$$\begin{aligned}Q\Big(A \bigcup B\Big) &= \sum_{i=1}^{p} \|\mathbf{c} - \mathbf{a}^i\|^2 + \sum_{i=1}^{q} \|\mathbf{c} - \mathbf{b}^i\|^2 \\ &= \sum_{i=1}^{p} \|\mathbf{c}(A) - \mathbf{a}^i\|^2 + p\|\mathbf{c} - \mathbf{c}(A)\|^2 \\ &+ \sum_{i=1}^{q} \|\mathbf{c}(B) - \mathbf{b}^i\|^2 + q\|\mathbf{c} - \mathbf{c}(B)\|^2 \\ &= Q(A) + Q(B) + p\|\mathbf{c} - \mathbf{c}(A)\|^2 + q\|\mathbf{c} - \mathbf{c}(B)\|^2.\end{aligned}$$

Izpeljimo še formulo za center množice $A \cup B$. Naj bosta najprej $A = \{a^1, \ldots, a^p\}$ in $B = \{b^1, \ldots, b^q\}$ disjunktni množici skalarjev in γ kvadratna funkcija

$$\begin{aligned}\gamma(x) &= p|x - c(A)|^2 + q|x - c(B)|^2 \\ &= p(x^2 - 2xc(A) + c(A)^2) + q(x^2 - 2xc(B) + c(B)^2).\end{aligned} \tag{16}$$

Odvajamo (16) in dobimo $\gamma'(x) = 2px - 2pc(A) + 2qx - 2qc(B)$. Ker je $\gamma(x)$ konveksna funkcija, je $\gamma(c) = \min_x \gamma(x)$ natanko tedaj, ko je $\gamma'(x) = 0$. Obrnemo enačbo in dobimo

$$c = \frac{p}{p+q}c(A) + \frac{q}{p+q}c(B). \tag{17}$$

Naj bosta sedaj $A = \{\mathbf{a}^1, \ldots, \mathbf{a}^p\}$ in $B = \{\mathbf{b}^1, \ldots, \mathbf{b}^q\}$ disjunktni množici n-dimenzionalnih vektorjev. Ker je

$$\begin{aligned}p\|\mathbf{c} - \mathbf{c}(A)\|^2 + q\|\mathbf{c} - \mathbf{c}(B)\|^2 &= p\sum_{i=1}^{n}(\mathbf{c}_i - \mathbf{c}(A))^2 + q\sum_{i=1}^{n}(\mathbf{c}_i - \mathbf{c}(B))^2 \\ &= \sum_{i=1}^{n}\Big(p(\mathbf{c}_i - \mathbf{c}(A))^2 + q(\mathbf{c}_i - \mathbf{c}(B))^2\Big),\end{aligned}$$

potem velja zveza (17) po komponentah in dobimo

$$\mathbf{c} = \frac{p}{p+q}\mathbf{c}(A) + \frac{q}{p+q}\mathbf{c}(B).$$

□

Izrek 5.6. *Če je* $A = \pi_1 \cup \pi_2 \cup \cdots \cup \pi_k$, $m_i = |\pi_i|$ *in* $\mathbf{c}_i = \mathbf{c}(\pi_i)$, $i = 1, \ldots, k$, *potem je*

$$\mathbf{c} = \mathbf{c}(A) = \frac{m_1}{m}\mathbf{c}_1 + \cdots + \frac{m_k}{m}\mathbf{c}_k, \tag{18}$$

kjer je $m = m_1 + \cdots + m_k$.

Če $\pi_i \cap \pi_j = \emptyset$, *ko* $i \neq j$, *potem je*

$$Q(A) = \sum_{i=1}^{k} Q(\pi_i) + \sum_{i=1}^{k} m_i \|\mathbf{c} - \mathbf{c}_i\|^2. \tag{19}$$

Opomba 5.7. Zvezi (18) in (19) sta očitni iz dokaza leme 5.5.

Poglejmo posebna primera leme 5.5. Prvič, naj bo $B = \{\mathbf{b}\}$ singelton, množico $A \cup B = \{\mathbf{a}^i, \ldots, \mathbf{a}^p, \mathbf{b}\}$ pa označimo z A^+. Iz leme 5.5 sledi

$$Q(A^+) = Q(A) + p\|\mathbf{c}(A^+) - \mathbf{c}(A)\|^2 + \|\mathbf{c}(A^+) - \mathbf{b}\|^2. \tag{20}$$

Ker je

$$\mathbf{c}(A^+) = \frac{p}{p+1}\mathbf{c}(A) + \frac{1}{p+1}\mathbf{b},$$

sledi

$$p\|\mathbf{c}(A^+) - \mathbf{c}(A)\|^2 = \frac{p}{(p+1)^2}\|\mathbf{c}(A) - \mathbf{b}\|^2 \tag{21}$$

in

$$\|\mathbf{c}(A^+) - \mathbf{b}\|^2 = \frac{p^2}{(p+1)^2}\|\mathbf{c}(A) - \mathbf{b}\|^2. \tag{22}$$

V (20) vstavimo (21) in (22) in dobimo

$$Q(A^+) = Q(A) + \frac{p}{p+1}\|\mathbf{c}(A) - \mathbf{b}\|^2. \tag{23}$$

Naj bo sedaj $B = \{\mathbf{b}^1, \ldots, \mathbf{b}^{q-1}, \mathbf{b}^q\}$. Iz B odstranimo vektor $\mathbf{b}^q$ in označimo $\{\mathbf{b}^1, \ldots, \mathbf{b}^{q-1}\}$ z B^-. Iz leme 5.5 sledi

$$Q(B) = Q(B^-) + (q-1)\|\mathbf{c}(B) - \mathbf{c}(B^-)\|^2 + \|\mathbf{c}(B) - \mathbf{b}^q\|^2. \tag{24}$$

Podobno kot prej, zaradi

$$\mathbf{c}(B^-) = \frac{q}{q-1}\mathbf{c}(B) - \frac{1}{q-1}\mathbf{b}^q \tag{25}$$

sledi

$$(q-1)\|\mathbf{c}(B) - \mathbf{c}(B^-)\|^2 = \frac{1}{q-1}\|\mathbf{c}(B) - \mathbf{b}^q\|^2. \tag{26}$$

V (24) vstavimo (26) in dobimo

$$Q(B) = Q(B^-) + \frac{q}{q-1}\|\mathbf{c}(B) - \mathbf{b}^q\|^2. \tag{27}$$

Opomba 5.8. Če upoštevamo, da je $B = (B^-)^+$, potem iz (23) sledi

$$Q(B) = Q((B^-)^+) = Q(B^-) + \frac{q-1}{q}\|\mathbf{c}(B^-) - \mathbf{b}^q\|^2.$$

Zaradi (25) lahko $\mathbf{c}(B^-) - \mathbf{b}^q$ zamenjamo z $\frac{q}{q-1}(\mathbf{c}(B) - \mathbf{b}^q)$, kar nam da ravno (27).

Enakosti (23) in (27) nam data naslednji izrek.

Izrek 5.9. *Naj bosta* $A = \{\mathbf{a}^1, \ldots, \mathbf{a}^p\}$ *in* $B = \{\mathbf{b}^1, \ldots, \mathbf{b}^q\}$ *disjunktni množici iz* $\mathbb{R}^n$. *Če je* $A^+ = \{\mathbf{a}^1, \ldots, \mathbf{a}^p, \mathbf{b}^q\}$ *in* $B^- = \{\mathbf{b}^1, \ldots, \mathbf{b}^{q-1}\}$, *potem je*

$$[Q(A) - Q(A^+)] + [Q(B) - Q(B^-)] = \frac{q}{q-1}\|\mathbf{c}(B) - \mathbf{b}^q\|^2 - \frac{p}{p+1}\|\mathbf{c}(A) - \mathbf{b}^q\|^2. \quad (28)$$

Izraz (28) prikazuje spremembo objektne funkcije $Q(A, B) - Q(A^+, B^-)$ po tem, ko odstranimo vektor $\mathbf{b}^q$ iz B in ga premaknemo v A.

Kdaj premakniti vektor $\mathbf{a} \in \pi_i$ iz gruče π_i z m_i vektorji v gručo π_j z m_j vektorji nam pove paketna verzija algoritma, ki to odločitev sprejme na podlagi izraza $\|\mathbf{c}(\pi_i) - \mathbf{a}\|^2 - \|\mathbf{c}(\pi_j) - \mathbf{a}\|^2$. Če je predznak

$$\Delta_k = \|\mathbf{c}(\pi_i) - \mathbf{a}\|^2 - \|\mathbf{c}(\pi_j) - \mathbf{a}\|^2$$

pozitiven, se zgodi premik. Spremembo prikazano v (28) zdaj zapišemo z

$$\Delta = \frac{m_i}{m_i - 1}\|\mathbf{c}(\pi_i) - \mathbf{a}\|^2 - \frac{m_j}{m_j + 1}\|\mathbf{c}(\pi_j) - \mathbf{a}\|^2. \quad (29)$$

Razlika med izrazoma je enaka

$$\Delta - \Delta_k = \frac{1}{m_i - 1}\|\mathbf{c}(\pi_i) - \mathbf{a}\|^2 + \frac{1}{m_j + 1}\|\mathbf{c}(\pi_j) - \mathbf{a}\|^2 \geq 0 \quad (30)$$

in je zanemarljiva, če sta gruči π_i in π_j relativno veliki. V nasprotnem primeru pa je razlika pomembna. Še posebej, če je vrednost Δ_k nenegativna in zato vektor $\mathbf{a}$ ostane v gruči π_i. Istočasno pa je vrednost Δ pozitivna, kar pomeni, da bi premik $\mathbf{a}$ v π_j zmanjšal vrednost objektne funkcije Q. Spomnimo se primera 5.4 in poglejmo Δ_k in Δ za $\mathbf{a} = \mathbf{a}_2$. Dobimo, da je

$$\Delta_k = \|\mathbf{c}_1 - \mathbf{a}_2\|^2 - \|\mathbf{c}_2 - \mathbf{a}_2\|^2 = 1 - 1 = 0$$

in

$$\Delta = \frac{2}{1}\|\mathbf{c}_1 - \mathbf{a}_2\|^2 - \frac{1}{2}\|\mathbf{c}_2 - \mathbf{a}_2\|^2 = 2 - \frac{1}{2} = \frac{3}{2}.$$

Očitno je, da je potrebna izpopolnitev osnovne verzije. Naslednja definicija opisuje inkrementalni korak, ki predstavlja to izpopolnitev.

Definicija 5.10. Prva različica particije Π je particija Π' dobljena iz Π s premikom vektorja $\mathbf{a}$ iz gruče π_i iz particije Π v že obstoječo gručo π_j iz particije Π.

Upoštevamo, da:

(1) je particija Π sama po sebi prva različica particije,
(2) prva različica ponovitve (tj. prva iteracija algoritma) lahko zmanjša število gruč v particiji.

Definicija 5.11. Particija $\texttt{nextFV}(\Pi)$ je prva različica particije Π, tako da za vsako prvo različico Π' velja

$$Q(\texttt{nextFV}(\Pi)) \leq Q(\Pi'). \quad (31)$$

Naslednji algoritem prikazuje združitev obeh verzij, kar predstavlja končno verzijo algoritma.

ALGORITEM 2. Kvadratična metoda k-voditeljev.

VHOD: matrika podatkov, k (število gruč), $\texttt{tol}_1$ in $\texttt{tol}_2$

IZHOD: particija podatkov

1 Določi poljubno začetno particijo $\Pi^{(0)} = \{\pi_1^{(0)}, \ldots, \pi_k^{(0)}\}$.
Nastavi $t = 0$.

2 Generiraj particijo $\texttt{nextBKM}(\Pi^{(t)})$.
if $[Q(\Pi^{(t)}) - Q(\texttt{nextBKM}(\Pi^{(t)})) > \texttt{tol}_1]$
nastavi $\Pi^{(t+1)} = \texttt{nextBKM}(\Pi^{(t)})$
$t = t + 1$
Vrni se na korak 2.

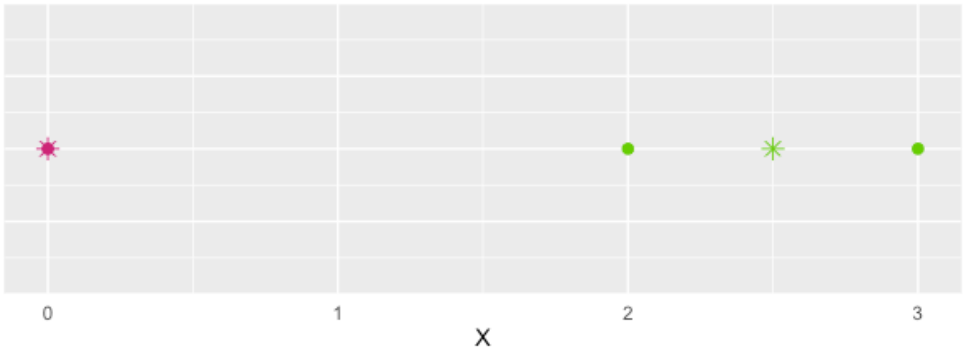
3 Generiraj particijo $\texttt{nextFV}(\Pi^{(t)})$.
if $[Q(\Pi^{(t)}) - Q(\texttt{nextFV}(\Pi^{(t)})) > \texttt{tol}_2]$
nastavi $\Pi^{(t+1)} = \texttt{nextFV}(\Pi^{(t)})$
$t = t + 1$
Vrni se na korak 2.

4 **stop**

Zgoraj opisana združitev algoritmov podeduje natančnost inkrementalnega koraka in hitrost paketne verzije, saj se vsi izračuni v koraku 3 izvedejo že v koraku 2 in zato dodatno računanje ni potrebno. Časovna zahtevnost algoritma je $\mathcal{O}(I \cdot k \cdot m \cdot n)$, kjer I predstavlja število iteracij algoritma.

Primer 5.12. Optimalna končna particija.
Spomnimo se primera 5.4. Iteracija algoritma 2 iz začetne particije $\Pi^{(0)} = \{\{0, 2\}, \{3\}\}$ generira particijo $\Pi^{(1)} = \{\pi_1^1, \pi_2^1\}$ z $\pi_1^{(1)} = \{0\}$ in $\pi_2^{(1)} = \{2, 3\}$, ki je grafično prikazana na sliki 5 in s tem doseže optimalen rezultat. ♢

SLIKA 5. Optimalna končna particija



5.4. **Ponavljajoči se elementi.**

Do zdaj smo predpostavili, da podatkovna množica vsebuje samo različne elemente, kar pa v praksi ni vedno res. Naj bo $A = \{\mathbf{a}^1, \ldots, \mathbf{a}^1, \ldots, \mathbf{a}^m, \ldots, \mathbf{a}^m\}$ podatkovna množica s ponavljajočimi se elementi, kjer z $w_i > 0$ označimo število kopij elementa $\mathbf{a}^i$. Optimizacijski problem je sedaj oblike

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sum_{i=1}^{m} w_i \|\mathbf{x} - \mathbf{a}^i\|^2, \quad w_i > 0. \tag{32}$$

Rešitev problema (32) dobimo na isti način kot rešitev problema (9) in je oblike

$$\mathbf{c}(A) = \frac{1}{w} \sum_{i=1}^{m} w_i \mathbf{a}^i,$$

kjer je $w = w_1 + \cdots + w_m$. Podobno kot v (14) za vsak $\mathbf{x}$ in $\mathbf{c} = \mathbf{c}(A)$ velja

$$\sum_{i=1}^{m} w_i \|\mathbf{x} - \mathbf{a}^i\|^2 = \sum_{i=1}^{m} w_i \|\mathbf{c} - \mathbf{a}^i\|^2 + w\|\mathbf{c} - \mathbf{x}\|^2.$$

Če sta $A = \{\mathbf{a}^1, \ldots, \mathbf{a}^1, \ldots, \mathbf{a}^p, \ldots, \mathbf{a}^p\}$ in $B = \{\mathbf{b}^1, \ldots, \mathbf{b}^1, \ldots, \mathbf{b}^q, \ldots, \mathbf{b}^q\}$ disjunktni množici iz $\mathbb{R}^n$ s ponavljajočimi se elementi, kjer $w_i > 0$ označuje število kopij elementa $\mathbf{a}^i$, $z_i > 0$ pa število kopij elementa $\mathbf{b}^i$, z uporabo leme 5.5 dobimo

$$Q\Big(A \bigcup B\Big) = Q(A) + Q(B) + w\|\mathbf{c} - \mathbf{c}(A)\|^2 + z\|\mathbf{c} - \mathbf{c}(B)\|^2, \tag{33}$$

kjer je

$$\mathbf{c} = \mathbf{c}\Big(A \bigcup B\Big) = \frac{w}{w+z}\mathbf{c}(A) + \frac{z}{w+z}\mathbf{c}(B).$$

Na isti način kot v (28) lahko sedaj izpeljemo spremembo objektne funkcije Q pri premiku w^* kopij vektorja $\mathbf{b}^q$. Najprej si poglejmo posebna primera (33). Naj bo $B = \{\mathbf{b}^q, \ldots, \mathbf{b}^q\}$ in $|B| = w^*$, množico

$$A \cup B = \{\mathbf{a}^1, \ldots, \mathbf{a}^1, \ldots, \mathbf{a}^p, \ldots, \mathbf{a}^p, \underbrace{\mathbf{b}^q, \ldots, \mathbf{b}^q}_{w^*}\}$$

pa označimo z A^+. Podobno kot v (23) dobimo

$$Q(A^+) = Q(A) + \frac{w^* w}{w + w^*}\|\mathbf{c}(A) - \mathbf{b}^q\|^2. \tag{34}$$

Naj bo sedaj $B = \{\mathbf{b}^1, \ldots, \mathbf{b}^1, \ldots, \mathbf{b}^q, \ldots, \mathbf{b}^q\}$. Iz B odstranimo w^* kopij vektorja $\mathbf{b}^q$ in množico $\{\mathbf{b}^1, \ldots, \mathbf{b}^1, \ldots, \mathbf{b}^{q-1}, \ldots, \mathbf{b}^{q-1}\}$ označimo z B^-. Na isti način kot v (27) dobimo

$$Q(B) = Q(B^-) + \frac{w^* z}{z - w^*}\|\mathbf{c}(B) - \mathbf{b}^q\|^2. \tag{35}$$

S pomočjo enakosti (34) in (35) dobimo formulo za izračun spremembe objektne funkcije, ki je oblike

$$[Q(A) - Q(A^+)] + [Q(B) - Q(B^-)] = \frac{w^* z}{z - w^*}\|\mathbf{c}(B) - \mathbf{b}^q\|^2 - \frac{w^* w}{w + w*}\|\mathbf{c}(A) - \mathbf{b}^q\|^2. \tag{36}$$

6. Izbira začetne particije in parametra k

6.1. **Metode za izbiro začetne particije.**

Izbira začetne particije je ključnega pomena pri dosegu efektivnega grupiranja, saj različne začetne particije doprinesejo k različnim rezultatom. Problem nastane, kadar se algoritem ustavi pri lokalnem minimumu, ki ni nujno globalni in zato ne generira optimalne končne particije. To lahko rešimo z izbiro druge začetne particije. Najbolj enostavni metodi za izbiro začetne particije, ki temeljita na naključni izbiri, sta naslednji.

(1) **Naključno seme (ang. random seed)** najprej izbere k naključnih točk iz podatkovne množice in jih označi za centre, ostale točke iz podatkovne množice pa dodeli najbljižjemu centru.
(2) **Naključno razčlenjevanje (ang. random partitioning)** vsako točko iz podatkovne množice naključno dodeli eni izmed k gruč.

Metodi sta enostavni za implementacijo, ker pa temeljita na naključni izbiri, ne zagotavljata dosega optimalne particije. Opisan problem lahko do neke mere rešimo tako, da r-krat generiramo naključno začetno particijo s pomočjo ene izmed zgoraj opisanih metod in izberemo particijo z minimalno vsoto razdalj v posameznih gručah oz. minimalno objektno funkcijo Q. Večji kot je r, večja je verjetnost, da bomo našli optimalno particijo, vendar se s ponavljanjem povečuje tudi časovna zahtevnost.

Obstajajo tudi številne deterministične metode, s pomočjo katerih lažje najdemo začetno particijo, ki vodi k bolj efektivnemu grupiranju. Ena izmed njih je metoda hierarhičnega razdvajanja na osnovi analize glavnih komponent (ang. PCA based divisive hierarchical approach), ki razdvaja podatkovno množico glede na smer dobljeno z analizo glavnih komponent, ki je omenjena že pri reševanju problema večdimenzionalnosti.

6.1.1. *Hierarhično razdvajanje.*

Za bolj efektivno grupiranje je dobro, da so začetni centri enakomerno porazdeljeni glede na vse atribute. To lahko storimo s pomočjo hierarhičnega razdvajanja na osnovi analize glavnih komponent. To pomeni, da začnemo z eno gručo, ki je ekvivalentna celotnemu podatkovnemu prostoru, jo razdelimo in izberemo eno izmed dobljenih gruč ter ponavljamo postopek dokler ni podatkovni prostor razdeljen v k gruč, kot zahteva algoritem. Pojavi se vprašanje, glede na katero smer bomo delili gručo.

Naj bo $\mathbf{c}$ aritmetična sredina gruče π in Q objektna funkcija kot v poglavju 5. Potem je

$$Q(\Pi_{stara}) = \sum_{\mathbf{a}\in\pi} \|\mathbf{a} - \mathbf{c}\|^2,$$

kjer je $\Pi_{stara} = \pi$. Po delitvi gruče π v gruči π_1 s centrom $\mathbf{c}_1$ in π_2 s centrom $\mathbf{c}_2$ je

$$Q(\Pi_{nova}) = \sum_{\mathbf{a}\in\pi_1} \|\mathbf{a} - \mathbf{c}_1\|^2 + \sum_{\mathbf{a}\in\pi_2} \|\mathbf{a} - \mathbf{c}_2\|^2,$$

kjer je $\Pi_{nova} = \pi_1 \cup \pi_2$. Vsak n-dimenzionalni vektor $\mathbf{a}$ lahko predstavimo kot uteženo vsoto n linearno neodvisnih ortonormiranih baznih vektorjev $\boldsymbol{\phi} = \{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n\}$:

$$\mathbf{a} = \sum_{s=1}^{n} y_s \boldsymbol{\phi}_s.$$

Podobno lahko zapišemo tudi

$$\mathbf{c} = \sum_{s=1}^{n} \alpha_s \boldsymbol{\phi}_s \quad \text{in} \quad \mathbf{c}_j = \sum_{s=1}^{n} \alpha_{js} \boldsymbol{\phi}_s, \ j = 1, 2.$$

Iščemo smer $\boldsymbol{\phi}_p$, glede na katero bomo projecirali podatke za razdelitev. Predpostavimo, da star center $\mathbf{c}$ in nova centra $\mathbf{c}_1$ in $\mathbf{c}_2$ ležijo na osi izbrani za projeciranje. Potem je $\boldsymbol{\phi}_p$, ki minimizira $Q(\Pi_{nova})$ enak $\boldsymbol{\phi}_p$, ki maksimizira

$$\sum_{\mathbf{a} \in \pi} (y_p \boldsymbol{\phi}_p - \alpha_p \boldsymbol{\phi}_p)^2 - \sum_{\mathbf{a} \in \pi_1} (y_p \boldsymbol{\phi}_p - \alpha_{1p} \boldsymbol{\phi}_p)^2 - \sum_{\mathbf{a} \in \pi_2} (y_p \boldsymbol{\phi}_p - \alpha_{2p} \boldsymbol{\phi}_p)^2, \tag{37}$$

kjer so y_p, α_p, α_{1p} in α_{2p} projecirane vrednosti $\mathbf{a}$, $\mathbf{c}$, $\mathbf{c}_1$ in $\mathbf{c}_2$ na $\boldsymbol{\phi}_p$. Dokaz se nahaja v [12]. Izraz (37) predstavlja razliko $Q(\Pi_{stara}) - Q(\Pi_{nova})$ glede na $\boldsymbol{\phi}_p$. Za določitev optimalne smeri je potrebno poznati centra $\mathbf{c}_1$ in $\mathbf{c}_2$, kar nas spet pripelje do problema grupiranja v dve gruči. Da se izognemo problemu grupiranja, se metoda zateče k suboptimalni smeri, posledično pa se rešuje poenostavljen optimizacijski problem

$$\max_{\boldsymbol{\phi}_p} \sum_{\mathbf{a} \in \pi} (y_p \boldsymbol{\phi}_p - \alpha_p \boldsymbol{\phi}_p)^2. \tag{38}$$

Rešitev problema (38) je smer $\boldsymbol{\phi}_{opt}$, ki največ prispeva k največji napaki oz. h $Q(\pi)$ in predstavlja prvo glavno smer. Podrobnejša izpeljava se nahaja v [11].

Merilo izbire gruče za deljenje je funkcija Q. Na vsakem koraku delitve izberemo gručo π, za katero je vrednost $Q(\pi)$ maksimalna. Kako sedaj razdelimo to gručo v izbrani smeri? Ker središče gruče predstavlja aritmetična sredina, je smiselno podatkovno množico razdvojiti glede na le to. Za izbrano gručo π projeciramo $\mathbf{a} \in \pi$ na prvo glavno smer in razdelimo π v dve podgruči π_1 in π_2 po naslednjem pravilu. Za vsak vektor $\mathbf{a}$ primerjamo projecirano vrednost y_{opt} s projecirano vrednostjo aritmetične sredine α_{opt} in če velja $y_{opt} \leq \alpha_{opt}$, pripišemo $\mathbf{a}$ podgruči π_1, v nasprotnem primeru pa podgruči π_2 ([11]).

6.2. **Izbira optimalnega parametra k.**

Določanje števila gruč v podatkovni množici je pogost problem pri particijskem grupiranju podatkov in je ločen od samega problema grupiranja. Izbira parametra k je odvisna od obsega in porazdelitve podatkovne množice, prav tako pa tudi od zastavljenega cilja. Le ta včasih točno določa število gruč, npr. če želimo razvrstiti paciente med zdrave ali bolne, je smiselno, da je $k = 2$. Velikokrat pa se nam zgodi, da s ciljem število gruč ni točno zastavljeno. Z večanjem parametra k brez kakršnihkoli omejitev, se bo vrednost $Q(\Pi)$, ki predstavlja končno napako, vedno manjšala. V ekstremnem primeru, kjer bo vsaka točka v svoji gruči, pa bo napaka ničelna. Izbira optimalnega števila gruč je tako iskanje ravnovesja med maksimalno kompresijo podatkovne množice z uporabo ene same gruče in maksimalno natančnostjo oz. minimalno napako z uporabo k, ki je enak moči podatkovne množice. V primeru, ko izbira k ni jasna iz predhodnega poznavanja podatkovne množice, ga izberemo s pomočjo ene izmed številnih metod za omenjen problem.

Ena izmed enostavnejših je komolčna metoda (ang. elbow method), ki deluje na naslednji način. Najprej izberemo množico parametrov k, npr. $K = \{1, 2, \ldots, 10\}$ in za vsak $k \in K$ uporabimo metodo k-voditeljev in s tem dobimo kvalitete končnih particij za vse možne $k \in K$. Narišemo odsekoma linearni graf objektne funkcije Q v odvisnosti od k in za k_{opt} vzamemo čim manjši $k \in K$, za katerega je $Q(k_{opt})$ dovolj nizka oz. $k \in K$, kjer ima graf $Q(k)$ komolec. Izbran k_{opt} seveda ni enolično določen in tako je izbira le tega odvisna od posameznikove interpretacije ter podatkovne množice. Uporaba metode in grafični prikaz izbire k se nahaja v podpoglavju 7.2.

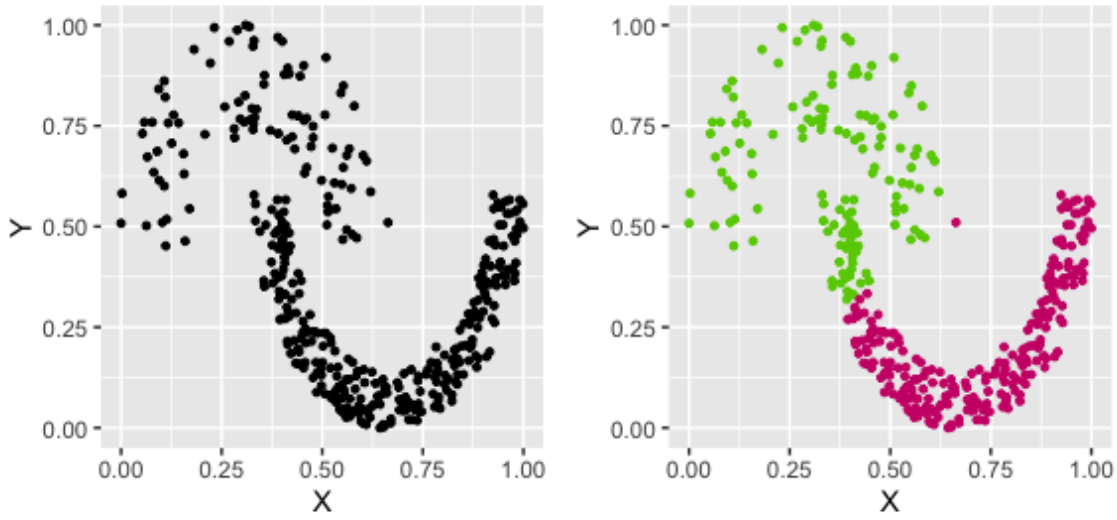
7. Uporaba kvadratične metode k-voditeljev

7.1. **Predpostavke.**

Kot že omenjeno, je uporaba kvadratične metode k-voditeljev, prav tako pa tudi ostalih algoritmov za grupiranje, zelo koristna za analiziranje raznih podatkovnih množic. Kljub temu, da kvadratična metoda k-voditeljev s pravilno podanimi vhodnimi podatki vedno vrne rezultat, pa le ta ni nujno vedno informativen. Na tem mestu je potrebno poudariti naslednje predpostavke, ki vodijo k bolj informativni končni particiji. Žal pa jih je v praksi težko v celoti upoštevati, še posebej, če se soočamo z večdimenzionalnimi podatkovnimi množicami, ki jih ne znamo grafično prikazati.

(1) Gruče v podatkovni množici morajo biti sferične oblike, da jih kvadratična metoda k-voditeljev ustrezno poišče. Natančneje, varianca porazdelitve mora biti sferična za vse spremenljivke. Na sliki 6 je primer uporabe algoritma na nesferični množici. Očitno je, da kvadratična metoda k-voditeljev ni ustrezno identificirala gruč.

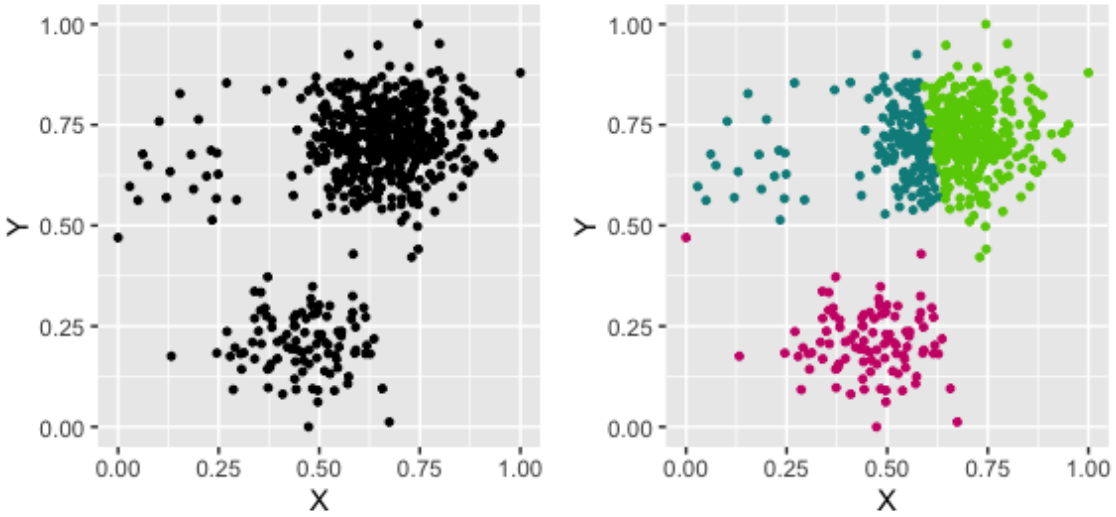
Slika 6. Uporaba kvadratične metode k-voditeljev na podatkovni množici z nesferičnimi gručami.



(2) Gruče morajo biti podobnih velikosti, kar pomeni, da se število podatkov od gruče do gruče ne sme preveč razlikovati. V nasprotnem primeru kvadratična

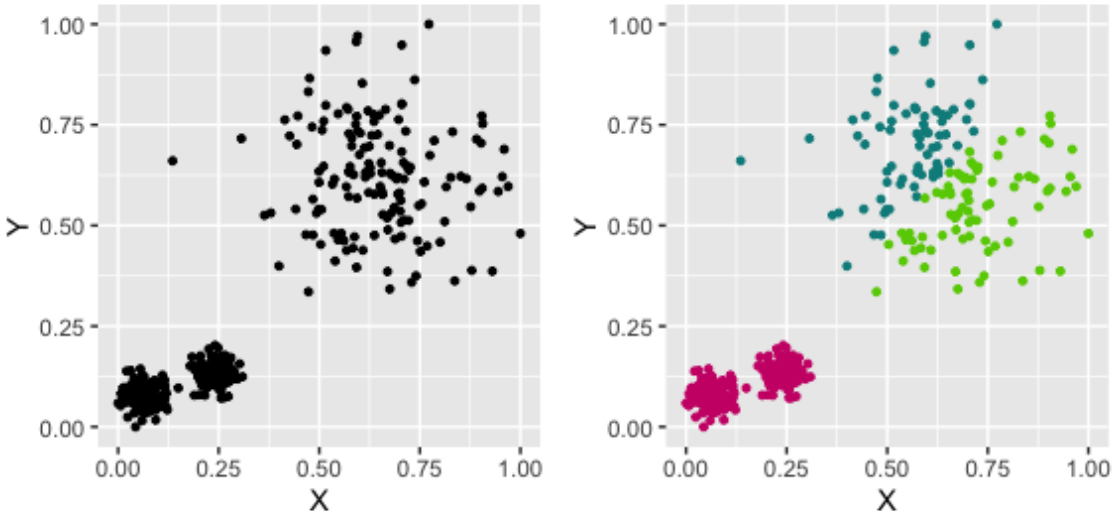
metoda k-voditeljev pripiše večjim gručam večjo težo kot manjšim. Na sliki 7 je prikazana uporaba algoritma na različno velikih gručah, iz koder je ponovno razvidno neustrezno identificiranje gruč.

SLIKA 7. Uporaba kvadratične metode k-voditeljev na podatkovni množici z različno velikimi gručami.



(3) Variance gruč se ne smejo preveč razlikovati, saj lahko to spet vodi v neefektivno grupiranje, kar je razvidno iz slike 8.

SLIKA 8. Uporaba kvadratične metode k-voditeljev na podatkovni množici z različnimi variancami gruč.



7.2. **Primer.**

Na spletni strani Svetovne banke [17] smo poiskali nekaj ekonomskih podatkov o državah za leto 2014, katere smo kasneje pogrupirali s kvadratično metodo k-voditeljev s pomočjo programa R. Najprej smo podatke pretvorili v ustrezno obliko, tj. matriko podatkov dimenzije 217×4, ki je delno prikazana v tabeli 1 (celotna matrika se nahaja v prilogi).

Tabela 1. Matrika podatkov.

	BDP na prebivalca ($)	rast BDP (%)	stopnja inflacije (%)	stopnja brezposelnosti (%)
Afganistan	633.948	1.313	4.604	9.1
Angola	5232.691	4.804	7.280	6.8
Albanija	4588.649	2.000	1.632	16.1
Armenija	3873.534	3.500	2.981	17.1
⋮	⋮	⋮	⋮	⋮

Kot večinokrat v praksi je podatkovna množica vsebovala nekaj manjkajočih vrednosti, katere smo izbrisali in tako zreducirali število vrstic iz 217 na 165. Vrednosti smo normalizirali s pomočjo min-max metode in s tem izenačili razpone vseh atributov.

Na ustrezno predprocesirani množici smo uporabili kvadratično metodo k-voditeljev, pri čemer smo začetno particijo izbrali s pomočjo metode naključno seme. To smo 100-krat ponovili, izbrali končno particijo z najmanjšo vrednostjo objektne funkcije Q ter s tem povečali verjetnost za doseg optimalne particije. Opisan postopek smo ponovili za različne parametre $k \in \{1, \ldots, 30\}$. Za doseg bolj optimalnega rezultata smo za izbiro parametra k uporabili komolčno metodo, ki je opisana v podpoglavju 6.2 in dobili graf objektne funkcije Q v odvisnosti od parametra k, ki je prikazan na sliki 9.

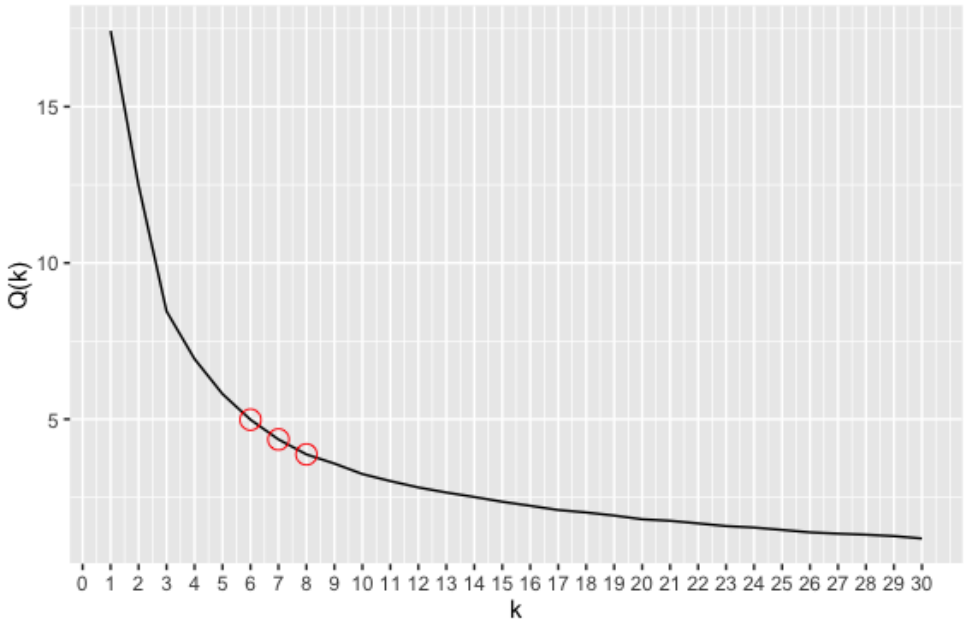
Ker rezultat omenjene metode ni natančno določen, smo se odločili, da zreduciramo množico parametrov k na $\{6, 7, 8\}$. Za lažjo odločitev smo natančno pregledali končne particije za omenjene k in za vsako državo izračunali razdaljo do centra gruče, v kateri se le ta nahaja. Izračunani podatki so nam omogočili boljši vpogled v porazdelitev držav znotraj gruč, saj smo le tako lahko videli, katere države so bolj izrazito oddaljene od svojega centra. Prav tako smo si pomagali s podatki o velikostih gruč in spreminjanju le teh pri večanju parametra k, ki so prikazani v tabeli 2.

Tabela 2. Velikosti gruč pri spreminjanju števila gruč.

$k = 6$	5	5	20	23	47	56		
$k = 7$	5	5	13	22	29	39	43	
$k = 8$	5	5	10	12	21	23	37	43

Za $k = 6$ je razlika med maksimalno in minimalno gručo bistveno večja kot za $k = 7$, medtem ko se iz $k = 7$ v $k = 8$ le ta ne spremeni. Če natančneje pogledamo še spremembo objektne funkcije $Q(k)$, vidimo, da se iz $k = 6$ v $k = 7$ zmanjša za

Slika 9. Komolčna metoda za izbiro optimalnega parametra k.



0.6316759, iz $k = 7$ v $k = 8$ pa za 0.4795225. Iz podrobnejše analize premikov držav iz gruče v gručo smo ocenili, da je optimalen k enak 7, saj je razmerje med manjšanjem objektne funkcije, spremembo velikosti gruč in odstranitvijo izrazito oddaljenih držav pri tem k najbolj optimalno.

Kot že omenjeno v poglavju 4 so pogost problem pri analizi podatkov odstopajoče vrednosti oz. osamelci. S pomočjo kvantilne funkcije smo za vsak atribut posebej pregledali porazdelitev vrednosti, pri čemer smo države prikazane v tabeli 3 označili za osamelce.

Tabela 3. Osamelci.

BDP na prebivalca ($)	rast BDP (%)	stopnja inflacije (%)
Švica	Ukrajina	Belorusija
Luksemburg	Etiopija	Gana
Macau (Kitajska)		Iran
Norveška		Malavi
Katar		Sudan

Zaradi izrazitih odstopanj vrednosti je skupina osamelcev glede na BDP na prebivalca tvorila svojo gručo v končni particiji, prav tako pa tudi skupina glede na stopnjo inflacije. Za ostali dve državi smo prav tako analizirali njuno umestitev v končni particiji, za doseg bolj optimalnega rezultata pa smo še enkrat grupirali države brez omenjenih dveh.

TABELA 4. Primerjava $Q(k)$ pred in po odstranitvi.

	$Q(6)$	$Q(7)$	$Q(8)$
pred odstranitvijo	4.985364	4.353689	3.874166
po odstranitvi Etiopije	5.308884	4.620608	4.097407
po odstranitvi Ukrajine	5.683028	4.945500	4.446749
po odstranitvi Ukrajine in Etiopije	6.223794	5.448372	4.862198

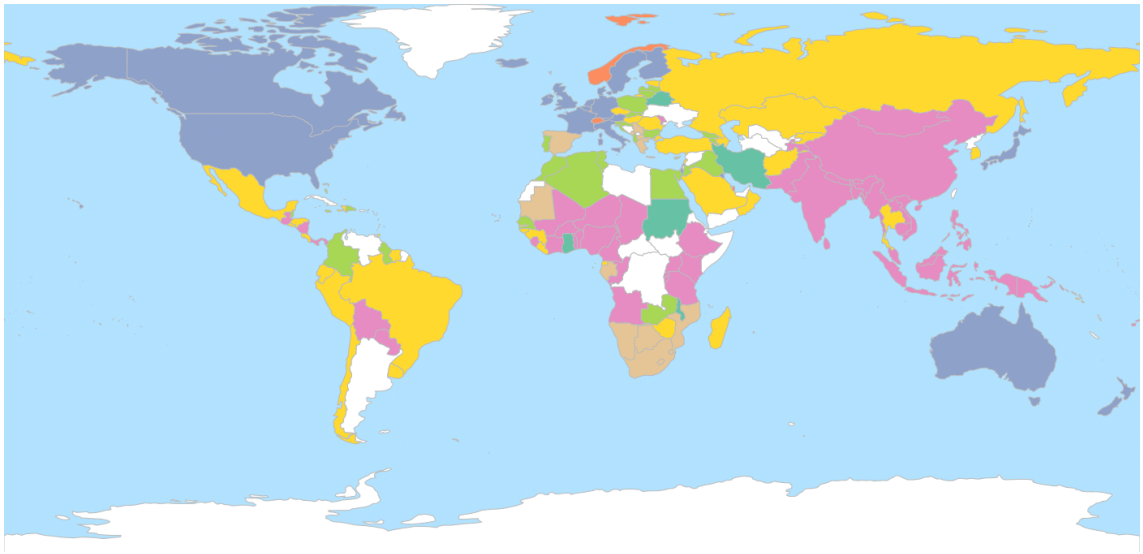
Očitno je, da je particija pred odstranitvijo držav bolj optimalna od ostalih, saj je vrednost funkcije Q manjša. Vhodna tabela je zato ostala enaka. Zaradi lege omenjenih držav v končni particiji smo se odločili, da iz končne particije izbrišemo Ukrajino, saj je po oddaljenosti od centra izrazito izstopala, kar je razvidno iz tabele 5. Z odstranitvijo se je zmanjšala varianca spremenjene gruče in s tem vrednost objektne funkcije Q, kar je pripomoglo k boljšemu rezultatu.

TABELA 5. Oddaljenost Ukrajine in Etiopije od centra pripadajoče gruče.

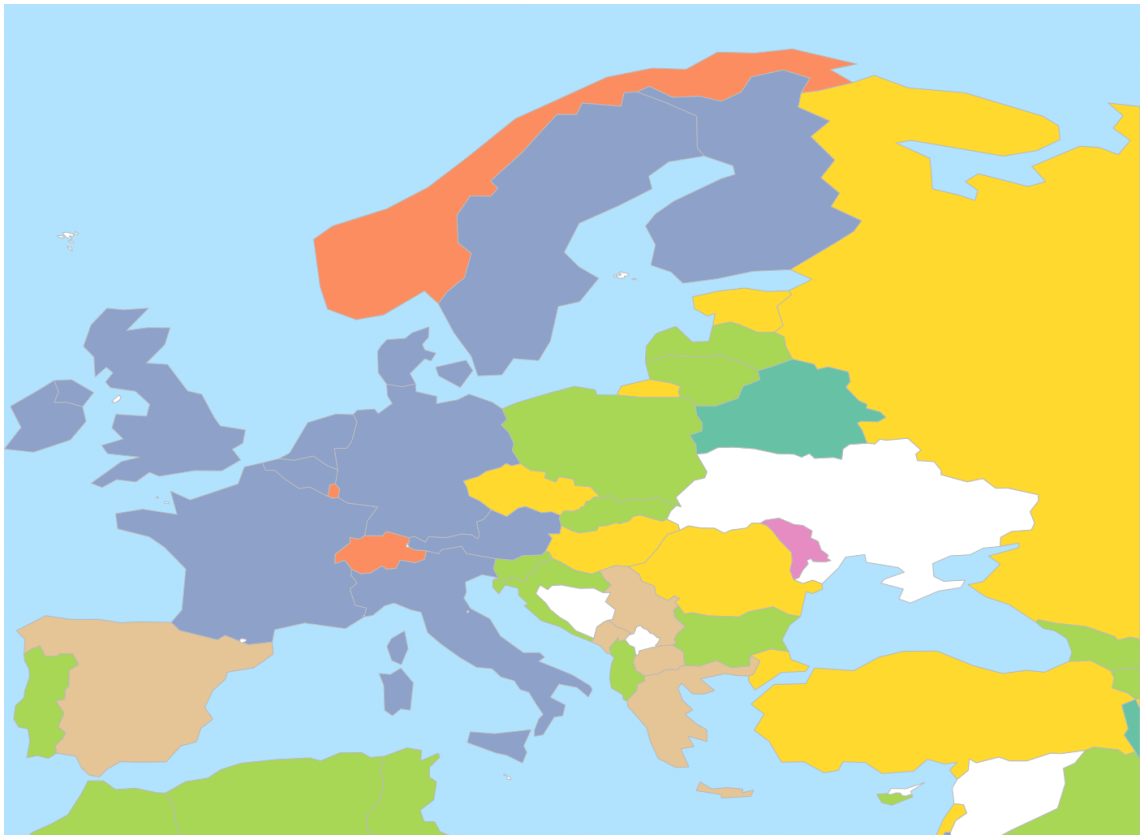
Gruča, ki vsebuje Ukrajino.		Gruča, ki vsebuje Etiopijo.	
Država	Oddaljenost od centra	Država	Oddaljenost od centra
⋮	⋮	⋮	⋮
Bahrajn	0.0422588206	Tadžikistan	0.045686996
Gvineja	0.0551151349	Mongolija	0.062146991
Ukrajina	**0.3177142047**	**Etiopija**	**0.065544818**

Končna particija metode k-voditeljev je prikazana na slikah 11 in 10, kjer vsaka barva predstavlja en del particije, tj. eno gručo, z izjemo bele barve, ki predstavlja države, ki niso bile obravnavane. Tabela s končno particijo se nahaja v prilogi.

SLIKA 10. Prikaz končne particije na zemljevidu sveta.



Slika 11. Prikaz končne particije na zemljevidu Evrope.



Literatura

[1] J. Kogan, *Introduction to Clustering Large and High-Dimensional Data*, Cambridge University Press, New York, 2007.

[2] J. Friedman, T. Hastie, R. Tibshirani, *The Elements of Statistical Learning, Data Mining, Inference and Prediction*, second edition, Springer, 2009.

[3] G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning with Applications in R*, Springer, 2013.

[4] *K-means clustering*, [ogled 10. 5. 2016], dostopno na `http://www.onmyphd.com/?p=k-means.clustering`.

[5] *Data mining*, [ogled 5. 5. 2016], dostopno na `https://en.wikipedia.org/wiki/Data_mining`.

[6] *Cluster analysis*, [ogled 5. 5. 2016], dostopno na `https://en.wikipedia.org/wiki/Cluster_analysis`.

[7] *An Introduction to Cluster Analysis for Data Mining*, [ogled 8. 6. 2016], dostopno na `http://www-users.cs.umn.edu/~han/dmclass/cluster_survey_10_02_00.pdf`.

[8] A.K. Jain, M.N. Murty, P.J. Flynn, *Data Clustering: A Review*, [ogled 10. 6. 2016], dostopno na `http://knight.cis.temple.edu/~vasilis/Courses/CIS750/Papers/jain99data.pdf`.

[9] D. Virmani, S. Taneja, G. Malhotra, *Normalization based K means Clustering Algorithm*, [ogled 18. 5. 2016], dostopno na `https://arxiv.org/pdf/1503.00900.pdf`.

[10] D. Napoleon, S. Pavalakodi, *A New Method for Dimensionality Reduction using K-Means Clustering Algorithm for High Dimensional Data Set*, [ogled 18. 5. 2016], dostopno na `http://www.ijcaonline.org/volume13/number7/pxc3872471.pdf`.

[11] T. Su, J. Dy, *A Deterministic Method for Initializing K-means Clustering*, [ogled 15. 7. 2016], dostopno na `http://www.ece.neu.edu/fac-ece/jdy/papers/init-kmeans.pdf`.

[12] T. Su, *Another look at non-random methods for initializing K-means clustering*, Master Project, June 2003, Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA.

[13] *Determining the number of clusters in a data set*, [ogled 17. 7. 2016], dostopno na`https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set`.

[14] *Using the elbow method to determine the optimal number of clusters for k-means clustering*, [ogled 17. 7. 2016], dostopno na `https://bl.ocks.org/rpgove/0060ff3b656618e9136b`.

[15] *Interacting with Data*, [ogled 17. 7. 2016], dostopno na `https://www.cs.princeton.edu/courses/archive/spring07/cos424/scribe_notes/0306.pdf`.

[16] D. Robinson, *K-means clustering is not a free lunch*, [ogled 25. 7. 2016], dostopno na `http://varianceexplained.org/r/kmeans-free-lunch/`.

[17] *World Bank Open Data*, [ogled 28. 7. 2016], dostopno na `http://data.worldbank.org/`.

[18] *How to label all the outliers in a boxplot*, [ogled 1. 8. 2016], dostopno na `https://www.r-statistics.com/tag/boxplot-outlier/`.