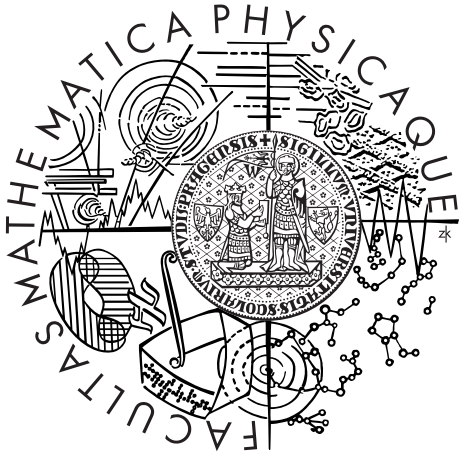
Univerzita Karlova v Praze Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Matěj Pacovský

Použití filtrovacích algoritmů ve shlukové analýze

Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce:	prof. RNDr. Jaromír Antoch, CSc.
Studijní program:	Matematika
Studijní obor:	Finanční matematika

Praha 2012

Dovolil bych si na tomto místě poděkovat vedoucímu práce, prof. RNDr. Jaromíru Antochovi, CSc., za cenné připomínky a úpravy, které přispěly k větší srozumitelnosti textu a také všem ostatním kteří mě u psaní bakalářské práce podporovali.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Použití filtrovacích algoritmů ve shlukové analýze

Autor: Matěj Pacovský

Katedra: katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: prof. RNDr. Jaromír Antoch, CSc., katedra pravděpodobnosti a matematické statistiky (305. 32-KPMS)

Abstrakt: Práce je rozdělena do pěti kapitol. V prvních dvou kapitolách shrnuji sebrané poznatky o shlukové analýze dat, uvádím definice pojmů použitých v práci a popisuji algoritmus k-průměrů. Ve třetí kapitole se zabývám filtrovacím algoritmem, který využívá filtrovací heuristiku během průchodu MRKD-stromem a tím urychluje algoritmus k-průměrů. Ve čtvrté kapitole popisuji algoritmus x-průměrů, který využívá všechny dosud zmíněné poznatky. V páté kapitole testuji všechny algoritmy na uměle vytvořených datech a na reálných datech z fyziky, přitom se v některých případech odkazuji na program WEKA, v němž je algoritmus x-průměrů naimplementován. Algoritmy o kterých pojednává tato práce jsou určeny pro objekty popsané pouze kvantitativními proměnnými. Jsou také vhodné k použití na velké datové soubory. Na přiloženém CD uvádím implementaci algoritmů v jazyku Matlab.

Klíčová slova: Algoritmus k-průměrů, algoritmus x-průměrů, filtrovací algoritmus, shluková analýza

Title: Use of filter algorithms in cluster analysis

Author: Matěj Pacovský

Department: Department of Probability and Mathematical Statistics

Supervisor: Prof. RNDr. Jaromír Antoch, CSc., Department of Probability and Mathematical Statistics

Abstract: The thesis is divided into five chapters. In the first two chapters I give the overview of clustering data analysis, I present definitions of terms used in the work and describe the k-means algorithm. Third chapter focuses on the filtering algorithm that uses heuristics when algorithm pass throught the MRKD-tree. The fourth chapter describes the x-means algorithm that uses all of the abovementioned findings. In the fifth chapter I test all algorithms both on artificial and real data from physics. In some cases I refer to the WEKA program where the x-means algorithm is implemented. Algoritms that are discussed in this thesis are intended only for objects described by quantitative variables. They are also suitable for large datasets. In the attached CD I present the implementation of algorithms in Matlab language.

Keywords: k-means algorithm, x-means algorithm, filtering algorithm, cluster analysis

Obsah

Úvod

Shluková analýza je zastřešující pojem pro rodinu metod matematické statistiky, jejichž cílem je nalezení skupin (shluků) podobných objektů. Algoritmy shlukové analýzy jsou implementované v řadě statistických programů (R, NCSS, Matlab, Statistica). Je tedy jednoduché na množinu datových objektů zavolat např. funkci, která provede třídění objektů pomocí algoritmu k-průměrů. Pokud ale potřebujeme tuto operaci zadávat víckrát a navíc na rozsáhlejších datech, zjistíme, že tento algoritmus je značně „pomalý“. Pro tyto případy je tedy výhodné algoritmus k-průměrů „vylepšit“, a snížit tak strojový čas potřebný k jeho běhu.

Snížení počtu operací potřebných k provedení algoritmu k-průměrů umožňuje tzv. filtrovací algoritmus. Tento algoritmus využívá geometrické vlastnosti shlukovaných objektů (bodů v prostoru). V této práci se zabývám jeho popisem a implementací v jazyku Matlab. Omezuji se přitom pouze na shlukování kvantitativních dat (body v $\mathbb{R}^n$). Při měření vzdáleností mezi body používám výhradně euklidovskou metriku.

Nevýhodou algoritmu k-průměrů je nutnost zadat číslo k udávající počet shluků. Algoritmus x-průměrů požaduje pouze zadání dolní a horní meze pro parametr x řídící počet shluků a hledá mezi různými počty shluků to rozdělení, které má nejmenší hodnotou Bayesova informačního kritéria. Algoritmus x-průměrů, jehož popisem se v této práci také zabývám, je tedy vylepšením původního algoritmu k-průměrů. Navíc efektivně využívá filtrovací algoritmus a jeho požadavky na strojový čas nejsou veliké.

V první kapitole definuji základní pojmy užité v práci. Ve druhé kapitole popisuji Lloydův algoritmus k-průměrů. Čerpám zde především z disertační práce Marty Žambochové (2010, [1]) a knihy Shluková analýza dat
(Řezanková, Húsek, Snášel, 2009, [2]) . Ve třetí kapitole uvádím filtrovací algoritmus tak, jak jej popsali autoři v článku An Efficient k-Means Clustering Algorithm. Analysis and Implementation (Kanungo a kol., 2002, [3]) . Ve čtvrté kapitole se zabývám popisem algoritmu x-průměrů a jeho zefektivněním pomocí filtrovacího algoritmu tak, jak jej popsali autoři v publikaci x-means: Extending k-means with Efficient Estimation of the Number of Clusters (Pelleg, Moore, 2000, [4])

V páté kapitole představuji implementaci algoritmů v jazyku Matlab a výsledky testování pokusných dat v programu Octave. Srovnávám výsledky podle kvality shlukování a v případě vlastních programů i podle počtu operací, které byly potřeba k běhu algoritmu. Kromě vlastních skriptů používám i externí statistický software Weka [6], v němž je algoritmus x-průměrů naimplementován.

1. Shluková analýza dat

1.1 Algoritmy shlukové analýzy a typy proměnných

Shluková analýza se zabývá dělením množiny datových objektů do více navzájem disjunktních podmnožin (dále jen shluků). A to tak, aby si objekty uvnitř jednotlivých shluků byly podobné co nejvíce a objekty z různých shluků si byly podobné co nejméně. Každý objekt v uvažovaném prostoru dat je charakterizován souborem znaků, typicky vektorem proměnných.

Algoritmy shlukové analýzy dělíme na:

a) Hierarchické

b) Nehierarchické (metody rozkladu)

Ad a) Hierarchické shlukovací metody vždy na výstupu nabídnou více možných rozkladů objektů do shluků. Výstupem hierarchického shlukování je *dendogram*, tj. graf který poskytuje informaci o tom, jaké je optimální sestavení shluků při různé hodnotě počtu shluků k.

Popišme si nejprve hierarichické shlukování shora dolů (od kořene k listům). Nechť máme dánu množinu objektů W, která je rozdělena na k shluků. Hierarchické algoritmy se snaží v každém kroku rozdělit nejméně homogenní shluk na dva a tím objekty roztřídit do $(k+1)$ shluků. Při tomto přístupu na počátku algoritmu množina všech objektů tvoří jeden shluk. V prvním kroku se tento shluk rozdělí na dva, ve druhém se nejméně homogenní ze shluků rozdělí na dva a takto se pokračuje, až je v $K-1$- tém kroku algoritmu splněna podmínka, že všech shluků je K. Získáváme tak postupné dělení množiny W na shluky $\{W_1^{(1)}, W_2^{(1)}\}, \{W_1^{(2)}, W_2^{(2)}, W_3^{(2)}\}, \ldots, \{W_1^{(K-1)}, \ldots, W_K^{(K-1)}\}$. Přitom všechny shluky v každém rozkladu jsou navzájem disjunktní. Tento způsob dělení shluků se v literatuře nazývá divizivní.

Opakem je shlukování zdola nahoru (od listům ke kořeni). Na začátku každý objekt tvoří jeden shluk a cílem každého kroku algoritmu je sloučit ty dva shluky, které by dohromady tvořily co nejhomogennější množinu. Tento přístup se nazývá algomerativní

Ad b) Nehierarchické metody shlukování mají typicky zadané číslo k udávající počet shluků, do kterých má být množina objektů rozdělena. Cílem těchto algoritmů je vytvořit rozklad množiny do návzájem disjunktních shluků tak, aby si každé dva objekty uvnitř jednoho shluku byly více podobné než každé dva objekty ležící v různých shlucích. Do rodiny těchto nehierarchických metod patří také algoritmus k-průměrů o němž pojednává tato práce. Získáme tak jednoznačný rozklad množiny objektů W do shluků $\{W_1, \ldots, W_k\}$.

Narozdíl od metod hierarchických neposkytují nehierarchické metody informace o celkové struktuře datového souboru, ale poskytují přímočarý rozklad objektů do předem zadaného počtu shluků.

Ve shlukové analýze dat můžeme porovnávat objekty, které jsou popsány různou sadou znaků. Znaky, neboli proměnné, dělíme na:

1. nominální - např. rod nebo druh rostliny

2. ordinární - např. průměrné hodnocení oblíbenosti na základě průzkumu
3. poměrové - např. hloubka kořene v závislosti na výšce rostliny
4. intervalové - např. interval ročních teplot, ve kterých rostlina může přežít
5. binární - např. uvedení, zda je rostlina jednoletá nebo víceletá.
6. kvantitativní - mezi tyto proměnné patří intervalové a poměrové proměnné.

1.2 Definice pojmů

V této práci se omezíme na práci s kvantitativími proměnnými. Budeme pracovat s reálnými čísly a reálnými vektory. Jako míru vzdálenosti použijeme euklidovskou metriku.

Definice 1.1. Nechť $(T, \|.\|)$ je metrický prostor o dimenzi m, $\boldsymbol{x} \in T$, $\boldsymbol{y} \in T$. Euklidovskou vzdáleností bodů $\boldsymbol{x}$ a $\boldsymbol{y}$ nazveme nezáporné číslo $d(\boldsymbol{x}, \boldsymbol{y})$, pro které platí:

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

□

Definice 1.2. Nechť $m \in \mathbb{N}$, $T \subseteq \mathbb{R}^m$ a $(T, \|.\|)$ je metrický prostor. *Objektem* rozumíme vícerozměrný bod $\boldsymbol{x}$, $\boldsymbol{x} \in T$.
□

Poznámka 1.1. Objekty jsou předměty nebo jevy určené ke klasifikaci. Každý je popsán $m-$tící znaků (proměnných). V této práci objekty reprezentujeme řádkovými vektory. Hodnotu i-té proměnné objektu $\boldsymbol{x}$ vyjadřujeme reálným číslem x_i.

Poznámka 1.2. Množina n objektů, kde každý je charakterizován m proměnnými, tvoří matici reálných čísel o dimenzi (n, m).

Definice 1.3. Nechť W je množina obsahující n objektů, které dohromady tvoří shluk.
Centroidem tohoto shluku nazveme bod $\boldsymbol{c} \in \mathbb{R}^m$, který je vektorem průměrných hodnot proměnných všech objektů ve shluku W, tj. vektor $(c_1, \ldots, c_m)$, kde

$$c_j = \frac{1}{n} \sum_{i=1}^{n} x_{i,j}$$

□

Ve dvourozměrném a trojrozděrném prostoru si lze každý objekt nakreslit v grafu a shluky si barevně nebo tvarově odlišit, aby bylo zřejmé, k jaké skupině objekty patří. Podobně si lze vyobrazit i rozdělení prostoru pomocí Voronojova diagramu. Ve shlukové analýze dat s tímto diagramem pracujeme i ve vícedimenzionálním prostoru. Vysvětlení tohoto termínu lze nalézt v následující definici a v připojeném obrázku.

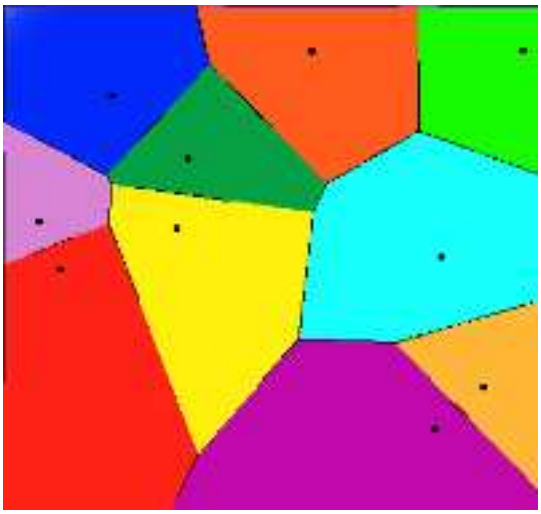
Definice 1.4. Nechť $m, l \in \mathbb{N}$, $T \subseteq \mathbb{R}^m$ a $(T, \|.\|)$ je metrický prostor. Nechť máme danou množinu center $\boldsymbol{y}_i \in T, i = 1, \ldots, l$. Voronojovým diagramem nazveme rozdělení prostoru T do Voronojových buněk $V(\boldsymbol{y}_i)$. Objekt $\boldsymbol{x}$ náleží Voronojově buňce centra $\boldsymbol{y}_i$ (píšeme $\boldsymbol{x} \in V(\boldsymbol{y}_i)$), pokud je splněno: $\forall j = 1, \ldots, l$, $j \neq i$

$$\|\boldsymbol{x} - \boldsymbol{y}_i\| \leq \|\boldsymbol{x} - \boldsymbol{y}_j\|.$$

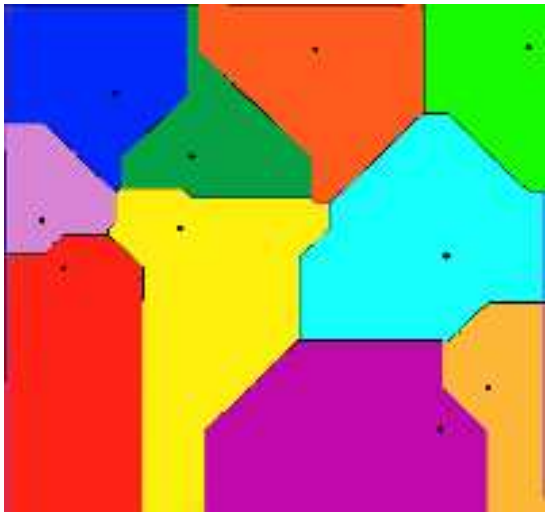
Hranice Voronojových buněk jsou body ze kterých je „nejblíže“ k více centrům. □

Poznámka 1.3.

- Všechny objekty nacházející se uvnitř Voronojovy buňky (tedy mimo hraničních objektů) $V(\boldsymbol{y}_i)$ „mají blíže“ k bodu $\boldsymbol{y}_i$ než ke kterémukoliv jinému bodu $\boldsymbol{y}_j$, $j \in \{1, \ldots, l\} \setminus \{i\}$
- Objekty nacházející se na hranici patří do více Voronojových buněk. Zde si postačíme s tímto zjednodušením. V algoritmu se pracuje s předpokladem, že vždy z jedné strany jsou Voronojovy buňky uzavřeny a z druhé otevřeny tak, aby objekty nenáležely najednou do více Voronojových buněk. Platí potom $V(\boldsymbol{y}_i) \cap V(\boldsymbol{y}_j) = \emptyset$ pro $i \neq j$ a $\cup_{i=1}^{l} V(\boldsymbol{y}_i) = T$.
- Pro konstrukci Voronojových buněk byla navržena řada algoritmů, z nichž nejpoužívanější je např. Holmesův
- Na obrázcích 1.1 a 1.2 jsou Voronojovy diagramy představeny jako spádové oblasti obchodů ve městě. Zákazník se v tomto městě rozhoduje navštívit obchod, ke kterému má nejblíže.



Obrázek 1.1: Voronojovy diagramy obchodů ve městě. Každému obchodu je přidělena Voronojova buňka. Nachází-li se turista v buňce nějakého obchodu, má to do tohoto obchodu nejblíže, existuje-li přímá cesta k obchodu (měříme euklidovskou vzdáleností). - *zdroj: en.wikipedia.org*



Obrázek 1.2: Voronojovy diagramy obchodů ve městě, použijeme-li manhattanskou metriku. Tato metrika v tomto modelu dává větší smysl.Lidé se totiž pohybují spíše po čtvercové síti (chodníky silnice). Zákazník nacházející se ve Voronojově buňce nějakého obchodu to má do tohoto obchodu nejblíže, pohybuje-li se po chodnících nebo po silnicích. - *zdroj: en.wikipedia.org*

1.3 Měření kvality shlukování

Abychom mohli porovnat dvě různá rozdělení n objektů do k shluků, potřebujeme způsob, jakým budeme měřit kvalitu shlukování. Jelikož se omezujeme pouze na kvantitativní data, pro vyjádření homogenity uvnitř shluků použijeme následující míry variability.

Definice 1.5. Nechť $W \subseteq T$ je n-prvková množina objektů a $\boldsymbol{c} \in \mathbb{R}^m$ je její centroid. W je shluk objektů a podle výše zmíněného jej lze reprezentovat maticí $\{x_{i,j}\}_{i=1,\ldots,n}^{j=1,\ldots,m}$ Rozptyl shluku W definujeme takto:

$$S_W^2 = \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{x_i} - \boldsymbol{c}\|^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\sum_{j=1}^{m}(x_{i,j} - c_j)^2\right]$$

□

Věta 1.1. *Rozptyl shluku W lze vypočítat dle vzorce*

$$S_W^2 = \frac{1}{n}\sum_{i=1}^{n}\langle \boldsymbol{x}_i, \boldsymbol{x}_i\rangle - \langle \boldsymbol{c}, \boldsymbol{c}\rangle$$

kde $\langle ., .\rangle$ je skalární součin

Důkaz.

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{i,j}-c_j)^2 &= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{i,j}^2 - 2x_{i,j}c_j + c_j^2)\\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}x_{i,j}^2 - \frac{2}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}x_{i,j}c_j + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}c_j^2\\
&= \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}x_{i,j}^2 - 2\sum_{j=1}^{m}\frac{\sum_{i=1}^{n}x_{i,j}}{n}c_j + \frac{1}{n}\sum_{i=1}^{n}\langle \boldsymbol{c}, \boldsymbol{c}\rangle\\
= \frac{1}{n}\sum_{i=1}^{n}\langle \boldsymbol{x}_i, \boldsymbol{x}_i\rangle - 2\langle \boldsymbol{c}, \boldsymbol{c}\rangle + \langle \boldsymbol{c}, \boldsymbol{c}\rangle &= \frac{1}{n}\sum_{i=1}^{n}\langle \boldsymbol{x}_i, \boldsymbol{x}_i\rangle - \langle \boldsymbol{c}, \boldsymbol{c}\rangle
\end{aligned}$$

❑

Poznámka 1.4. V případě, kdy jsou všechny objekty z prostoru reálných čísel, lze míru variability S_W^2 použít bez transformace dat. Pokud by se však v souboru znaků vyskytovaly proměnné různého typu, je třeba všechny proměnné standardizovat tak, aby měla každá v rozptylu shluku stejnou váhu. Tuto standardizaci v práci používat nebudeme.

Poznámka 1.5. Nechť jsou objekty rozděleny do k shluků a i-tému shluku jsou přiřazeny jeho charakteristiky S_i^2, $i = 1. \ldots, k$. Kvalitu aktuálního rozdělení objektů do shluků měříme pomocí součtu rozptylů jednotlivých shluků

$$Q = \sum\nolimits_{i=1}^{k} S_i^2$$

Poznámka 1.6. Chceme-li vyjádřit variabilitu shluku ve stejných jednotkách jako měříme jednotlivé proměnné, sčítáme směrodatné odchylky shluků $Q^* = \sum_{i=1}^{k}\sqrt{S_i^2}$

Má-li náš algoritmus rozhodnout také o počtu shluků, do kterých objekty umístí, je třeba definovat kritérium, podle kterého bude tyto různé varianty posuzovat. Výše uvedené míry variability proměnnou hodnotu k počtu shluků nebraly v úvahu. Existují však obecně dané postupy (pro všechny typy proměnných), jak tato kritéria sestavit. V našem případě se opět omezíme na posuzování kvantitativních proměnných.

Definice 1.6. Nechť $m \in \mathbb{N}$. Nechť $k \in \mathbb{N}$ je počet shluků a $n \in \mathbb{N}$ je počet shlukovaných objektů v m-dimenzionálním prostoru reálných čísel. Nechť dále n_h, pro $h = 1, \ldots, k$, je počet objektů ve shluku h a σ_l^2 a S_{hl}^2 pro $l = 1, \ldots, m$ jsou po

řadě výběrový rozptyl l-té proměnné a výběrový rozptyl l-té proměnné ve shluku h. Potom Schwarzovým bayesovským informačním kritériem, zkratkou BIC (Bayesian Information Criterion) nazveme reálné číslo definované podle Řezankové a kol. (2009) následujícími vztahy:

$$BIC = \underbrace{\sum_{h=1}^{k} \xi_h}_{I} + \underbrace{mk \log(n)}_{II},$$

kde $\xi_h = n_h\big(\sum_{l=1}^{m} \log \sqrt{(\sigma_l^2 + S_{hl}^2)}\big)$.
□

Poznámka 1.7. Pro centroid h-tého shluku $\boldsymbol{c}^h$ se složkami $(c_1^h, \ldots, c_m^h)$ a vektor průměrných hodnot $\overline{\boldsymbol{x}}$ jsou výběrový rozptyl ve shluku a výběrový rozptyl pro l-tou proměnnou definovány následujícími vzorci:

$$S_{hl}^2 = \frac{1}{n_h} \sum_{i=1}^{n_h} (x_{i,l} - c_{i,l})^2 \quad \text{a} \quad \sigma_l^2 = \frac{1}{n} \sum_{i=1}^{n} (x_{i,l} - \overline{x_{i,l}})^2,$$

kde objekty se nachází v matici $\{x_{i,l}\}_{i=1,\ldots,n}^{l=1,\ldots,m}$ a $\overline{x_{i,l}} = \frac{1}{n} \sum_{i=1}^{n} x_{i,l}$.

Poznámka 1.8.

- Rozptyl shluku h z definice 1.5 je součtem výběrových rozptylů přes všechny proměnné, $S_h^2 = \sum_{l=1}^{m} S_{hl}^2$
- Pomocí Bayesova informačního kritéria chceme najít kompromis mezi počty shluků a jejich variabilitou. Protože logaritmus je rostoucí spojitá funkce a ve výpočtu jsou sčítance I i II s kladným znaménkem, chceme kritérium BIC minimalizovat. Člen I měří variabilitu shluků, člen II „pokutuje“ hodnotu kritéria za vyšší počet shluků k. Na hodnotu kritéria BIC se tedy můžeme dívat podobně jako na hodnotu ztrátové funkce Q.

2. Algoritmus k-průměrů

Primárním cílem algoritmu k-průměrů je minimalizovat ztrátovou funkci, která měří variabilitu uvnitř shluků. Sekundárním cílem je maximalizovat funkci, která měří variabilitu různých shluků navzájem. Tvar funkce, jež může být použita k měření homogenity uvnitř shluků, jsem uvedl v předchozí kapitole v poznámce 1.5 a označil ji Q. Algoritmus k-průměrů ztrátovou funkci Q používá k rozhodování, jestli má na konci každého kroku pokračovat další iterací, nebo ukončit běh programu. Výstupem algoritmu je jediný rozklad množiny bodů do k shluků. Algoritmus tedy patří do rodiny nehierarchických metod shlukování.

2.1 Popis Lloydova algoritmu

Na vstupu je zadáno přirozené číslo k, jež udává počet shluků, do kterých má algoritmus objekty roztřídit. Dále je na vstupu zadána matice reálných čísel o dimenzi (n, m), která reprezentuje množinu objektů $\{\boldsymbol{x}_i \in \mathbb{R}^m, i = 1, \ldots, n\}$ a kladné číslo ε, udávající hodnotu prahové změny ztrátové funkce.

Na začátku první iterace algoritmus buď náhodně nebo podle nějaké metodiky vygeneruje množinu k center $\boldsymbol{C}^{(1)} = \{\boldsymbol{C}_1^{(1)}, \ldots, \boldsymbol{C}_k^{(1)}\}$. Dále definuje počáteční hodnotu ztrátové funkce $Q^{(0)} = \infty$ a položí $p = 1$.
[1]

Algoritmus nejprve přiřadí objekty centrům $\boldsymbol{C}^{(p)}$ a tím vytvoří shluky objektů. Objekt $\boldsymbol{x}_j$ ležící ve Voronojově buňce $V(\boldsymbol{C}_i)$ náleží shluku, který je reprezentován centrem $\boldsymbol{C}_i$, který získáme následující minimalizací:

$$\boldsymbol{C}_i = \underset{\{\boldsymbol{C}_h, h=1,\ldots,k\}}{\arg\min} \|\boldsymbol{C}_h - \boldsymbol{x}_j\| \tag{2.1}$$

Lloydův algoritmus (Voronojova iterace) využívá pozorování, že optimální umístění pro centrum se nachází v **centroidu** daného shluku. Dalším krokem algoritmu je vypočítání centroidů $\boldsymbol{c}^{(p)} = \{\boldsymbol{c}_1^{(p)}, \ldots, \boldsymbol{c}_k^{(p)}\}$. Centroid $\boldsymbol{c}_i^{(p)}$ náleží shluku, který je reprezentován centrem $\boldsymbol{C}_i^{(p)}$.

Poznámka 2.1. Centra shluků stejně jako centroidy reprezentují jednotlivé shluky. V popisu algoritmu používám oba termíny. Pro pochopení algoritmu k-průměrů (a dále i filtrovacího algoritmu) je potřeba vypořádat se s rozdíly které představují tato označení. **Centrum** shluku slouží jako označení bodu v prostoru, okolo něhož se tvoří shluk. **Centroid** shluku označuje „střed“ už vytvořeného shluku, jehož souřadnice jsou aritmetickými průměry souřadnic všech objektů nácházejících se ve shluku. Shluk s pořadovým číslem i je tvořen seskupováním objektů okolo centra $\boldsymbol{C}_i$. Na konci každé iterace Lloydova algoritmu je i-tý shluk reprezentován jak centroidem $\boldsymbol{c}_i$, tak centrem $\boldsymbol{C}_{i+1}$.

Třetím krokem algoritmu je vypočtení hodnoty ztrátové funkce $Q^{(p)}$ rozdělení množiny objektů k centroidům $\boldsymbol{c}^{(p)}$. Je-li $(Q^{(p-1)} - Q^{(p)}) \geq \varepsilon$, algoritmus pokračuje další iterací s množinou center $\boldsymbol{C}^{(p+1)} = \boldsymbol{c}^{(p)}$. V opačném případě dojde k ukončení algoritmu a výstupem bude přiřazení objektů centroidům $\boldsymbol{c}^{(p)}$.

[1] V programovém provedení je $Q^{(0)}$ velmi vysoké číslo, které by hodnota ztrátové funkce $Q^{(1)}$ neměla překročit, např 10^{10}.

Shrnutí Lloydova algoritmu

0. Počáteční zvolení center $\boldsymbol{C}^{(1)} = \{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k\},\ p = 1,\ Q^{(0)} = \infty$

1. Vytvoření shluků objektů okolo center $\boldsymbol{C}^{(p)}$ podle pravidla (2.1)

2. Vypočtení centroidů $\boldsymbol{c}^{(p)}$ shluků z kroku 1

3. Vypočtení ztrátové funkce $Q^{(p)}$.

- Když $Q^{(p-1)} - Q^{(p)} \geq \varepsilon$, pak $\boldsymbol{C}^{(p+1)} = \boldsymbol{c}^{(p)}$, $p = p+1$, návrat na krok 1
- Když $Q^{(p-1)} - Q^{(p)} < \varepsilon$, následuje ukončení algoritmu a výstupem je přiřazení objektů nejbližším centroidům z množiny $\boldsymbol{c}^{(p)}$

Poznámka 2.2.

- V implementaci algoritmu v kroku 1 dochází k porovnání vzdálenosti $d(\boldsymbol{C}_i, \boldsymbol{x}_j)$ pro všechna $i = 1, \ldots, k, j = 1, \ldots, n$, a přiřazení centra $\boldsymbol{C}_i$ objektu $\boldsymbol{x}_j$, ke kterému měl nejkratší vzdálenost. K tomuto procesu dochází i po ukončení algoritmu, kdy je potřeba každému objektu přiřadit „nejbližší“ centroid.
- Podmínku v kroku 3 lze nahradit jednodušší podmínkou bez nutnosti počítat ztrátovou funkci. Testuje se, jestli je množina center $\boldsymbol{C}^{(p)}$ shodná s množinou centroidů $\boldsymbol{c}^{(p)}$. Této podmínce by odpovídala volba $\varepsilon = 0$. Jedná se tedy o speciální a méně programově náročný případ popsaného algoritmu.

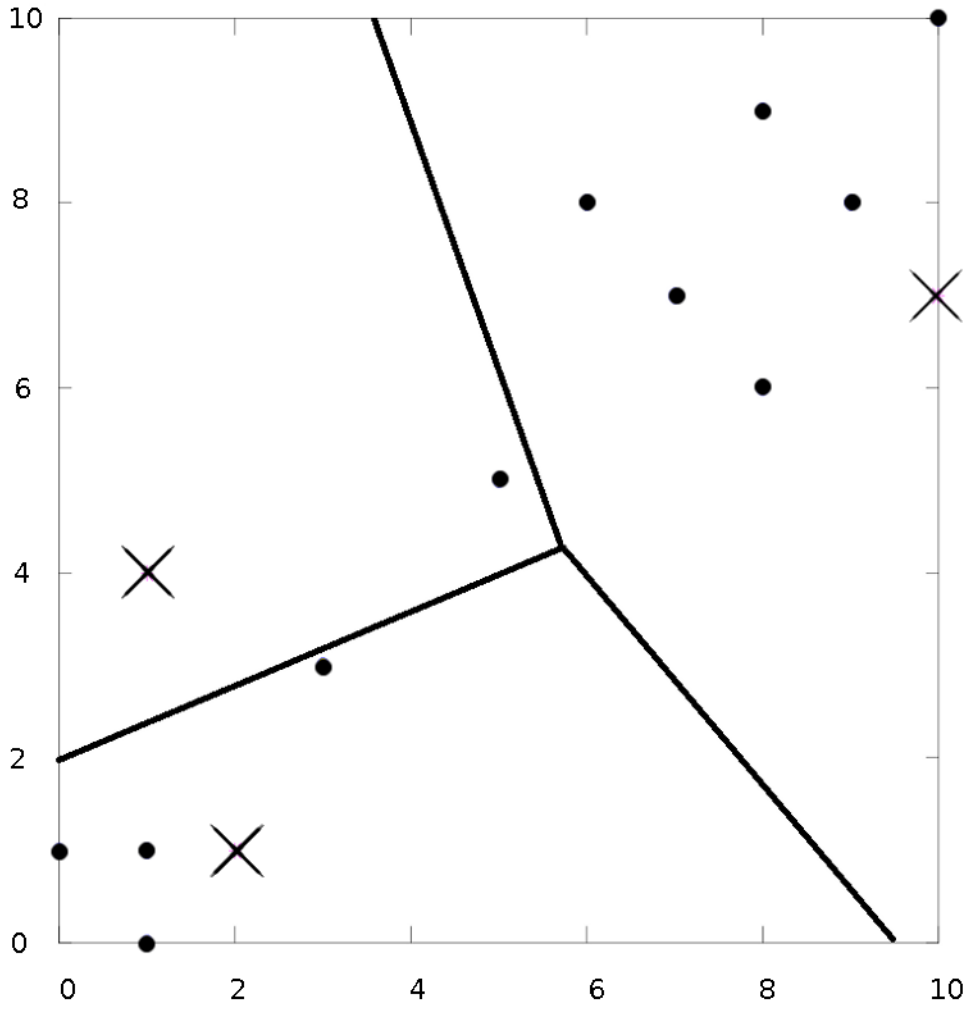
2.2 Příklad

Zadání: Nechť $m = 2$, $k = 3$ a máme zadanou množinu 11 objektů
$\{[0,1][1,0], [1,1], [3,3], [5,5], [7,7], [8,6], [6,8], [10,10], [9,8], [8,9]\}$
z metrického prostoru $(W = \langle 0, 10\rangle^2, \|.\|)$.
Rozdělte množinu objektů na 3 shluky tak, aby si objekty uvnitř shluků byly co nejvíc podobné.
Řešení:

Krok 0 Nejdříve náš algoritmus náhodně zvolí počáteční centra z prostoru W. Řekněme, že zvolil množinu $\boldsymbol{C}^{(1)} = \{[2,1], [1,4], [10,7]\}$.

Krok 1 Ke každému centru z $\boldsymbol{C}^{(1)}$ najde množinu bodů, které k němu „mají nejblíže“. Na obrázku číslo 2.1 vidíme Voronojovy buňky jednotlivých center.

Krok 2 Z bodů v jednotlivých Voronojových diagramech jsou vypočteny centroidy $\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3$.



Obrázek 2.1: Objekty z příkladu, počáteční centra a jejich Voronojovy buňky. Centra jsou znázorněna křížky.

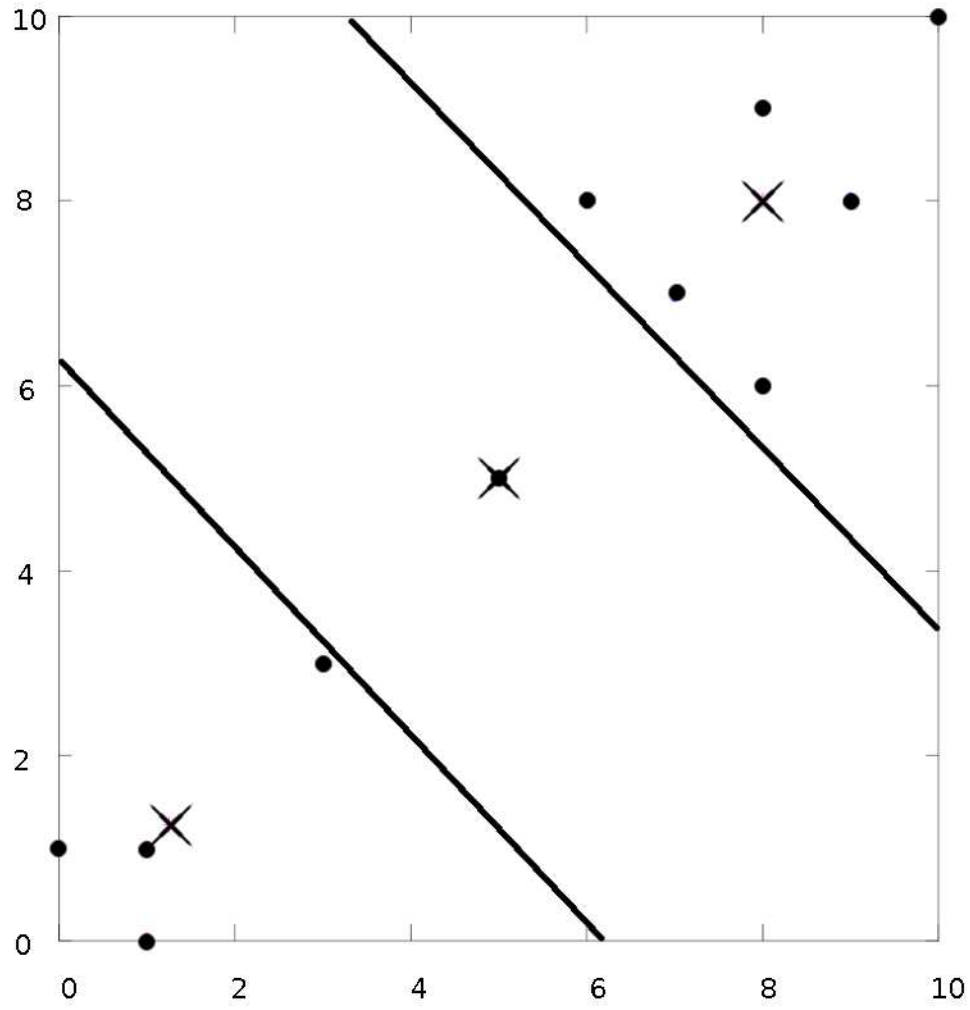
Krok 3 Jelikož se hodnota ztrátové funkce zmenšila ($Q^{(1)} = 29.5$), pokračuje algoritmus další iterací a do kroku 1 vstupuje s množinou $\boldsymbol{C}^{(2)} = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3\}$, $\boldsymbol{c}_1 = [1.25, 1.25], \boldsymbol{c}_2 = [5, 5], \boldsymbol{c}_3 = [8, 8]$.

Další iterace Algoritmus stejným způsobem jako výše přiřadí objekty nejbližším centrům, umístí centroidy a vypočte hodnotu ztrátové funkce, $Q^{(2)} = 29.5$. Jelikož $Q^{(1)} - Q^{(2)} = 0$, algoritmus ukončí běh a výstupem je rozdělení množiny objektů k centroidům $\boldsymbol{c}^{(2)}$.

K optimální hodnotě ztrátové funkce stačily pouze dvě iterace algoritmu. Rozdělení množiny bodů do tří shluků můžeme vidět na obrázku číslo 2.2.

Poznámka 2.3.

- Kdybychom za počáteční centra v příkladu 2.2 zvolili jiné body (např. všechny tři nacházející se okolo bodu $[0, 0]$), k optimálnímu řešení bychom



Obrázek 2.2: Finální pozice centroidů z příkladu 2.2 a jejich Voronojovy buňky. Centroidy shluků jsou znázorněny křížky.

došli až po 4 iteracích. Navíc bychom dokonvergovali k horšímu řešení a jeden shluk by zůstal prázdný.

- To, že některé shluky zůstanou na konci algoritmu prázdné, lze částečně ošetřit vložením okrajových podmínek. Pro některá data však neexistuje počáteční rozdělení centroidů tak, aby algoritmus k-průměrů rozdělil data do požadovaného počtu shluků.

- Výsledná kvalita shlukování podstatně závisí na počátečním rozložení centroidů. Lze tedy říci, že algoritmus k-průměrů nalezne vždy jen lokální minimum ztrátové funkce Q.

2.3 Nevýhody algoritmu k-průměrů

Lokální minimum. Z poznámky 2.4 a celkového charakteru algoritmu soudíme, že algoritmus k-průměrů hledá pouze lokální minimum ztrátové funkce Q.
Abychom našli uspokojující řešení, je třeba algoritmus spustit několikrát a vybrat rozdělení s nejmenší hodnotou ztrátové funkce.
Předem zadaný počet shluků k. Někdy se může stát, že data pocházejí z rozdělení představujícího k shluků v prostoru. Potom by hledání $(k-1)$ shluků těchto objektů nalezlo rozdělení s výrazně vyšší hodnotou ztrátové funkce v porovnání s hodnotou ztrátové funkce rozdělení do k shluků. Navíc se může stát, že nějaký shluk zůstane prázdný. Tento fakt závisí nejen na nevhodně zvoleném číslu k, ale také na počátečním rozdělení centroidů.
Časová složitost. Pro přiměřeně malé n a k algoritmus konverguje v přijatelném čase. Pokud je ale některé z těchto čísel výrazně větší, čas strávený na hledání nejbližšího centra v kroku 1 se zvýší. Operaci $d(\boldsymbol{x}, \boldsymbol{y})$ hodnotíme jako časově nejnáročnější část tohoto algoritmu.

V kroku 1 každé iterace algoritmu je třeba vypočítat vzdálenost od každého objektu ke každému centru. Pokud je objektů n, center k a dimenze prostoru je m, časová složitost algoritmu je $O(k * n * f(m))$. Funkce $f(m)$ vyjadřuje závislost dimenze objektů m na složitosti výpočtu. Závislost dimenze na složitosti totiž v jazyku Matlab nejspíše není lineární.

Poznámka 2.4. Jazyk Matlab umožňuje vypočtení vzdálenosti d dvou víceroměrných bodů $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ v jednom kroku pomocí Haddamardova umocnění jednotlivých složek vektoru. Je pravděpodobné, že časová náročnost výpočtu vzdálenosti dvou vektorů závisí i tak na jejich délce m. Píšeme tedy $O(k*n*f(m))$, jelikož složitost výpočtu nemusí být závislá na délce vektoru lineárně.

3. Filtrovací algoritmus

3.1 Popis filtrovacího algoritmu

Filtrovací algoritmus je heuristická nadstavba algoritmu k-průměrů. Používá datovou strukturu MRKD-strom, která umožňuje hierarchické dělení prostoru objektů. O její konstrukci pojednám v odstavci 3.2. Jedná se o binární strom, který rozděluje sekvenčně vstupní prostor na dvě části, každou část rozdělí znovu a dále rekurzivně až k listům. Každému uzlu ve stromu je přiřazena sada statistik bodů, které reprezentuje. O těchto statistikách pojednám v podkapitole 3.3.

MRKD-strom stačí vypočítat pouze jednou na začátku algoritmu. Pomocí tohoto binárního stromu algoritmus filtruje v prvním kroku Lloydova algoritmu centra shluků (kandidátská centra) a tím snižuje počet operací potřebných k dokončení prvního kroku algoritmu k-průměrů v každé jeho iteraci.

Průchod stromem v p-té iteraci algoritmu k-průměrů probíhá následovně: Množina center $\boldsymbol{C}^{(p)}$ se „prosévá“ skrz MRKD-strom od kořene směrem dolů. Nechť je množina kandidátských center uzlu u minimálně dvouprvková a označme ji $\boldsymbol{C}_u^{(p)}$. Množinu kandidátských center levého a pravého syna uzlu u označíme $\boldsymbol{C}_{synove}^{(p)}$. Ačkoliv se v MRKD-stromu levý a pravý syn liší, množinu kandidátských center sdílejí.

Podle uchovávaných statistik v uzlu u se algoritmus rozhodne, které centrum z množiny kandidátských center $\boldsymbol{C}_u^{(p)}$ zachová do množiny $\boldsymbol{C}_{synove}^{(p)}$. Platí tedy $\boldsymbol{C}_{synove}^{(p)} \subset \boldsymbol{C}_u^{(p)}$. Množina kandidátských center levého syna uzlu u je stejná jako množina kandidátských center pravého syna uzlu u, proto zde hovoříme pouze o jedné množině. Technice filtrování kandidátských center věnuji odstavec 3.4.

V každém uzlu jsou testovány následující podmínky, které ukončí průchod v celé větvi stromu a zároveň způsobí přiřazení objektů asociovaných s tímto uzlem některému ze zbývajících kandidátských center.

1. **Jedná-li se o list**, pak jsou asociované objekty uzlu přiřazeny ke kandidátskému centru, které je nejblíže „střednímu bodu“ asociovaných bodů uzlu. O významu středního bodu uzlu pojednám v dalších odstavcích.

2. **Je-li množina kandidátů jednoprvková**, pak se všechny asociované objekty uzlu přiřadí k tomuto jednomu kandidátskému centru bez nutnosti dalšího porovnávání a hledání nejbližšího centra pro objekty.

Poznámka 3.1.

- Listy MRKD-stromu reprezentují více objektů, pokud ve vstupním datovém souboru existuje více stejných objektů.

- Významné urychlení představuje právě podmínka č. 2. Díky ní se ukončí běh v celé větvi stromu a jednomu centru je přiřazena najednou celá množina objektů.

3.2 MRKD-stromy

MRKD-stromy slouží k reprezentaci souboru objektů. Objekty v tomto případě mohou být vícedimenzionální body v prostoru o dimenzi m. V KD-stromu (kvadrantový strom) obsahuje každý uzel ukazatele na k synů. V MRKD-stromu má každý uzel právě dva syny.

Jedná se tedy o binární stromovou strukturu, kde každý uzel reprezentuje množinu dat. Kořen stromu obsahuje všechny objekty. Každý z jeho dvou synů reprezentuje podmnožinu bodů kořene a tito dva synové jsou navzájem disjunktní a dohromady skládají nadřazený uzel (otce). Každý z dvou synů má další syny, pro něž platí stejná pravidla. Dělení množiny objektů pokračuje rekurzivně až k listům, které obsahují pouze jediný bod (viz poznámka 3.1). Rozhodování o tom, zda bude prvek uzlu patřit jeho levému nebo pravému synovi záleží na jaké straně $(m-1)$-dimenzionální nadroviny leží.

Definice 3.1. Nechť uzel u reprezentuje l-prvkovou množinu objektů B, $B = \{\boldsymbol{x}_i \in \mathbb{R}^m, i = 1, \ldots, l\}$. Množinu B nazveme **buňkou** tohoto uzlu. O všech objektech v buňce B řekneme, že jsou **asociované** s uzlem u.
□

Definice 3.2. **Ohraničující hyperkvádr** množiny B je nejmenší kvádr $H \subset \mathbb{R}^m$, obsahující všechny body z B. Platí tedy $\forall \boldsymbol{x} \in B, \boldsymbol{x} \in H$.
□

Nyní máme zadefinované základní prvky, které se vyskytují v MRKD-stromech a zbývá vyřešit, jak tento strom konstruovat. Jelikož dělení v každém uzlu probíhá podle stejného algoritmu, zabývejme se otázkou, jak rozdělit ohraničující hyperkvádr uzlu na dva tak, aby dělení bylo co nejefektivnější. Nechť im je index proměnné, která reprezentuje nejdelší stranu ohraničujícího hyperkvádru.

$$im = \arg\max_{j=1,\ldots,m}(\max_{i=1,\ldots,l} x_{i,j} - \min_{i=1,\ldots,l} x_{i,j})$$

Ohraničující hyperkvádr rozdělíme nadrovinou, která je kolmá k této nejdelší straně. Nadrovinu umístíme v bodě, kde proměnná s indexem im nabývá mediánu. Označíme $x^{(*)}$ medián proměnné im ze všech bodů množiny B. Do levého podstromu umístíme všechny objekty, které jsou v této proměnné menší nebo rovny mediánu, do pravého zbytek. Tvoříme tedy podmnožiny

$$B_{ls} = \{\boldsymbol{x}_i \in B, x_{i,im} \leq x^{(*)}\}$$

$$B_{ps} = \{\boldsymbol{x}_i \in B, x_{i,im} > x^{(*)}\}.$$

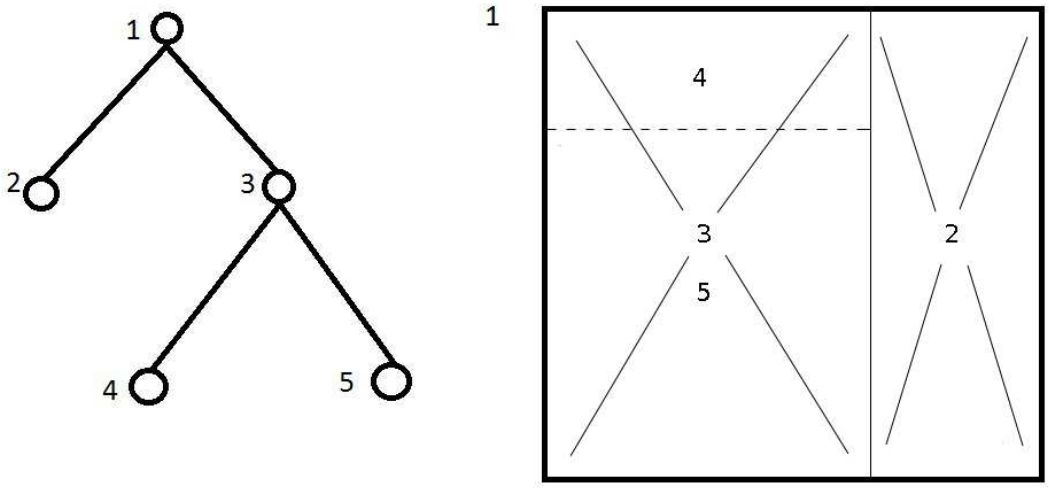
Platí $B_{ls} \cap B_{ps} = \emptyset$ a $B_{ls} \cup B_{ps} = B$. Pokud je buňka B_{ls} (resp. B_{ps}) nejméně jednoprvková, přiřadíme ji levému (resp. pravému) synu. Pokud je buňka B jednoprvková nebo obsahuje více stejných bodů, uzel u je list. MRKD-strom je tedy binární strom, který reprezentuje hierarchické dělení ohraničujících hyperkvádrů pomocí $(m-1)$-dimenzionálních nadrovin.

Poznámka 3.2.

- Ohraničující hyperkvádr je až na výjimky uzavřený.

- Pokud je počet objektů v buňce B sudý, neprochází dělící nadrovina žádným bodem a kvádry, které vzniknou přepůlením otcovského hyperkvádru jsou v těchto nově vzniklých stranách otevřené. Ohraničující hyperkvádry jsou potom uzavřené podmnožiny těchto kvádrů, obsahující stejné body. Abychom kvádr uzavřeli, je potřeba jej v zmenšit tak, aby byl podle definice nejmenší a tvořil ohraničující hyperkvádr.

- Pokud je počet objektů lichý, dělící nadrovina prochází nějakým bodem z množiny B_{ls}. Ohraničující hyperkvádr této množiny vznikne rovnou přepůlením nadrovinou. Ohraničující hyperkvádr množiny B_{ps} se vytvoří zkrácením nově vzniklého kvádru, tak aby byl uzavřený a obsahoval všechny objekty z množiny B_{ps}.



Obrázek 3.1: Ukázka dělení prostoru pomocí MRKD-stromu. Kořenová buňka (č.1) představuje celý prostor. Dělení objektů do synů s číslem 2 a 3 představuje svislá příčka. Dělení uzlu 3 představuje vodorovná čárkovaná příčka. Listové vrcholy jsou tedy uzly 2,4 a 5. Vrcholy hyperkvádrů uzlů č.2 a č.3 jsou zvýrazněny čárami vedoucími od čísel uprostřed.

3.3 Uchovávané hodnoty

3.3.1 Statistiky uchovávané v uzlech MRKD-stromu

Nechť objekty $\boldsymbol{x}_i$, $i = 1, \ldots, l$, $\boldsymbol{x}_i \in R^m$, jsou asociované s uzlem u. Potom uzel u uchovává následující informace o asociované množině objektů.

(0) Přirozené číslo udávající počet asociovaných objektů
$l^u \in \mathbb{N}$

(1) Vektor minimálních souřadnic $\boldsymbol{x}^u_{(1)}$ se složkami
$\boldsymbol{x}^u_{(1),j} = \min_{i=1,\ldots,l} x_{i,j},\ j = 1, \ldots, m$

(2) Vektor maximálních souřadnic $\boldsymbol{x}^u_{(l)}$ se složkami $\boldsymbol{x}^u_{(l),j} = \max_{i=1,\ldots,l} x_{i,j},\ j = 1, \ldots, m$

(3) Vektor součtu souřadnic SS^u se složkami $SS^u_j = \sum_{i=1}^n x_{i,j}, j = 1, \ldots, m$

(4) „Střední bod“ $\boldsymbol{x}^u_*$ asociovaných objektů (vektor mediánů po složkách). Kanugo a kol. (2002) použili ve své implementaci bod, jehož j-tá souřadnice je medián všech j-tých souřadnic asociovaných bodů uzlu. Při této volbě „středního bodu“ probíhá konstrukce MRKD-stromu přesně tak, jak jej popisuji v odstavci 3.2.

(5) Hodnotu čtverce Frobeniovy normy: $\|\mathbf{X}\|^{2\,u}_F = \sum_{j=1}^m \sum_{i=1}^l x^2_{i,j}$, kde $\mathbf{X}$ je matice obsahující všechna pozorování asociovaná s uzlem u.

3.3.2 Použití uchovávaných statistik

Máme-li v 1. kroku $(p)-$té iterace algoritmu k-průměrů množinu center $\boldsymbol{C}^{(p)} = \{\boldsymbol{C}^{(p)}_1, \ldots, \boldsymbol{C}^{(p)}_k\}$, budeme si při průchodu stromem udržovat v globální proměnné statistiky $\alpha_h, \boldsymbol{\beta}_h$ a γ_h pro každé centrum $\boldsymbol{C}^{(p)}_h$, $h = 1, ..., k$.

- Hodnota v proměnné α_h bude označovat počet dosud přiřazených objektů centru $\boldsymbol{C}^{(p)}_h$.
- Vektor $\boldsymbol{\beta}_h \in \mathbb{R}^m$ bude uchovávat součty hodnot proměnných všech objektů přiřazených centru $\boldsymbol{C}^{(p)}_h$.
- V proměnné γ_h budeme sčítat čtverce Frobeniovy normy objektů přiřazených centru $\boldsymbol{C}^{(p)}_h$.

Před průchodem stromu položíme

$$\alpha_h = 0,\ \boldsymbol{\beta}_h = (0, \ldots, 0),\ \gamma_h = 0\ \ \forall h = 1, \ldots, k.$$

Jestliže v uzlu u algoritmus dle ukončovacích podmínek přiřadí při průchodu MRKD-stromem množinu objektů centru $\boldsymbol{C}^{(p)}_h$, aktualizujeme hodnoty statistik $\alpha_h, \boldsymbol{\beta}_h$ a γ_h statistikami **(0), (3), (5)** definovanými v odstavci 3.3.1.

$$\alpha_h = \alpha_h + l^u,\ \boldsymbol{\beta}_h = \boldsymbol{\beta}_h + SS^u,\ \gamma_h = \gamma_h + \|\mathbf{X}\|^{2\,u}_F$$

Po ukončení průchodu stromem budeme moci jednoduše identifikovat centroidy shluků, ale také míry variability definované v odstavci 1.3. Centroid shluku objektů, které byly přiřazeny centru $\boldsymbol{C}^{(p)}_h$, vypočteme dle vzorce:

$$\boldsymbol{c}^{(p)}_h = \frac{1}{\alpha_h}\boldsymbol{\beta}_h$$

Rozptyl shluku objektů přiřazených centru $\boldsymbol{C}^{(p)}_h$ vzhledem k již vypočítanému centroidu $\boldsymbol{c}^{(p)}_h$ vyjádříme pomocí věty 1.1 pro $l = \alpha_h$.

$$S^2_h = \frac{1}{\alpha_h}\sum_{i=1}^{\alpha_h}(\boldsymbol{x}_i - \boldsymbol{c}_h)^2 = \frac{1}{\alpha_h}\sum_{i=1}^{\alpha_h}\langle \boldsymbol{x}_i, \boldsymbol{x}_i\rangle - \langle \boldsymbol{c}_h, \boldsymbol{c}_h\rangle$$

Pro libovolný m-prvkový vektor $\boldsymbol{x}$ lze psát $\langle \boldsymbol{x}, \boldsymbol{x} \rangle = \sum_{j=1}^{m} x_j^2$. Dosadíme-li dále za centroid

$$\langle \boldsymbol{c}_h, \boldsymbol{c}_h \rangle = \left\langle \frac{\boldsymbol{\beta}_h}{\alpha_h}, \frac{\boldsymbol{\beta}_h}{\alpha_h} \right\rangle.$$

a nahradíme-li součet druhých mocnin objektů uchovávanou hodnotu γ_h, tj.

$$\sum_{i=1}^{\alpha_h} \langle \boldsymbol{x}_i, \boldsymbol{x}_i \rangle = \sum_{i=1}^{l} \left[\sum_{j=1}^{m} x_{i,j}^2 \right] = \gamma_h,$$

dostaneme vyjádření rozptylu shluku objektů, které náleží centroidu $\boldsymbol{c}_h^{(p)}$ pomocí hodnot

$$S_h^2 = \frac{1}{\alpha_h} \gamma_h - \left\langle \frac{\boldsymbol{\beta}_h}{\alpha_h}, \frac{\boldsymbol{\beta}_h}{\alpha_h} \right\rangle$$

Rozptyl shluku může být vnímán jako míra kompaktnosti shluku. Sečteme-li v p-té iteraci algoritmu rozptyly všech shluků podle vzorce uvedeného v poznámce 1.5, dostaneme hodnotu ztrátové funkce $Q^{(p)}$, kterou chceme v p-tém kroku algoritmu porovnat s předchozí hodnotou a tím rozhodnout o dalším postupu algoritmu.

$$Q^{(p)} = \sum_{h=1}^{k} S_h^2 = \sum_{h=1}^{k} \left(\frac{1}{\alpha_h} \gamma_h - \left\langle \frac{\boldsymbol{\beta}_h}{\alpha_h}, \frac{\boldsymbol{\beta}_h}{\alpha_h} \right\rangle \right)$$

Poznámka 3.3. Fukce $Q^{(p)}$ se v průběhu algoritmu minimalizuje. Cílem algoritmu k-průměrů je tedy rozložit objekty do shluků tak, aby byla hodnota ztrátové funkce $Q^{(p)}$ co nejmenší.

3.4 Filtrování kandidátských center

Pro každý uzel MRKD-stromu udržujeme množinu **kandidátských center**. Každé z nich může v nějakém podstromu sloužit jako *nejbližší centrum* pro nějakou podmnožinu asociovaných bodů tohoto uzlu. Množina kandidátských center se ale zmenšuje s tím, jak procházíme MRKD-stromem směrem dolů. O úroveň níže postoupí pouze ta podmnožina kandidátských center, která projde filtrovací procedurou. Odstranění bodů, které jako centrum pro žádnou podmnožinu asociovaných objektů nemohou sloužit, probíhá následujícím způsobem.

Nechť u je uzel MRKD-stromu. Označme Z množinu kandidátských center uzlu u a B jeho buňku. Dále označme Z_{synove} kandidátská centra synů uzlu u (jedná se o levého i pravého syna uzlu u) a $\boldsymbol{x}_*^u$ střední bod objektů v buňce B, který jsme získali z uchovávané statistiky **(4)** z odstavce 3.3.1. Platí $Z_{synove} \subseteq Z$.

Při vytváření množiny Z_{synove}, neboli filtrování množiny Z, nejprve vypočítáme, který z kandidádů z množiny Z je nejblíže střednímu bodu $\boldsymbol{x}_*^u$. Označme jej $\boldsymbol{z}^*$, tj.

$$\boldsymbol{z}^* = \arg\min_{\boldsymbol{z} \in Z} \|\boldsymbol{z} - \boldsymbol{x}_*^u\|^2.$$

Přidejme kandidátské centrum $\boldsymbol{z}^*$ do množiny Z_{synove}. Poté pro všechny zbývající kandidáty $z \in Z \setminus \{\boldsymbol{z}^*\}$ rozhodneme, zda je odfiltrujeme podle následujícího pravidla.

a) Pokud má nějaká část buňky B blíže k $\boldsymbol{z}$ než k $\boldsymbol{z}^*$, přidáme $\boldsymbol{z}$ do množiny Z_{synove} - nefiltrujeme.

b) V opačném případě, kdy žádná část buňky B nemá blíže k $\boldsymbol{z}$ než k $\boldsymbol{z}^*$, kandidáta odfiltrujeme.[1]

- Nechť H je nadrovina, která ortogonálně půlí vektor $\overrightarrow{u} = \boldsymbol{z} - \boldsymbol{z}^*$. H rozděluje ohraničující hyperkvádr buňky B na dva podprostory. Jeden je blíže k $\boldsymbol{z}$ a druhý k $\boldsymbol{z}^*$. Pokud B leží celá v podprostoru, kde se nachází také kandidátské centrum $\boldsymbol{z}^*$, můžeme $\boldsymbol{z}$ odfiltrovat.
- Označme $v(H)$ vrchol buňky B, který maximalizuje skalární součin $\langle \overrightarrow{u}, v(H) \rangle$. Kandidáta $\boldsymbol{z}$ odfiltrujeme, pokud

$$\|\boldsymbol{z} - v(H)\| \geq \|\boldsymbol{z}^* - v(H)\|$$

.

Poznámka 3.4. Příslušný vrchol $v(H)$ buňky B vypočteme pomocí hodnot **(1), (2)** z odstavce 3.3.1, které máme uloženy v každém uzlu. Pokud je j-tá souřadnice vektoru $\overrightarrow{u}$ záporná, vezmeme za j-tou souřadnici bodu $v(H)$ hodnotu $x_{(1),j}$, v opačném případě vezmeme $x_{(l),j}$.

3.5 Filtrovací procedura v algoritmu k-průměrů

Algoritmus k-průměrů, který je vybaven filtrovací procedurou (dále jen filtrovací algoritmus), bude mít pozměněný průběh procesů. V kroku 1 každé iterace algoritmus pomocí MRKD-stromu a filtrovací podmínky „proseje“ centra skrz strom a každému centru přiřadí statistiky podle podkapitoly 3.3. V některých uzlech, které nejsou listy, zbyde jenom jedno centrum a tím se ukončí běh v celé větvi. Množiny asociovaných objektů těchto uzlů budou přiřazeny jednotlivým zbývajícím kandidátským centrům. Tím se výrazně sníží počet potřebných porovnání vzdáleností při tvoření shluků. Po splnění ukončovací podmínky ve 3. kroku ještě algoritmus provede závěrečný krok, kdy každému objektu přiřadí nejbilžší centrum. Tato operace totiž není zajišťována v průchodu MRKD-stromem.

Filtrovací algoritmus

0. Počáteční náhodné vygenerování center $\boldsymbol{C}^{(1)} = \{\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k\}$. Položíme $p = 1$ a $Q^{(p)} = \infty$

1. Průchod MRKD-stromem, filtrování kandidátských center popsané v předchozích odstavcích. Každému centru z množiny $\boldsymbol{C}^{(p)}$ jsou přiřazeny příslušné statistiky $\alpha, \boldsymbol{\beta}, \gamma$.

2. Vypočtení centroidů $\boldsymbol{c}^{(p)}$ ze statistik center z předchozího kroku

3. Vypočtení ztrátové funkce $Q^{(p+1)}$ podle vzorce uvedeného na konci odstavce 3.3.2.

[1]Odfiltrování kandidáta znamená, že jej vyřadíme z množiny kandidátských center, které postoupí o úroveň níže při procházení MRKD-stromu odshora dolů.

- Když $Q^{(p)} - Q^{(p+1)} \geq \varepsilon$, pak $\boldsymbol{C}^{(p+1)} = \boldsymbol{c}^{(p)}$, $p = p + 1$ a návrat na krok 1
- Když $Q^{(p)} - Q^{(p+1)} < \varepsilon$, skok na krok 4

4. Každému objektu je přiřazen nejbližší centroid z množiny $\boldsymbol{c}^{(p)}$

3.6 Příklad

Nechť máme k dispozici stejná data jako v příkladu 2.2. Úkol zůstává stejný: rozdělit uvažované objekty do 3 shluků. Začneme vytvořením MRKD-stromu z těchto dat.

[0,1] [1,0] [1,1] [3,3] [5,5] [7,7] [8,6] [6,8] [9,8] [8,9] [10,10]

- Z dat je patrno, že obě proměnné mají nejmenší hodnotu 0 a největší 10. Je tedy lhostejné, kterou zvolíme jako dělící. Algoritmus volí v tomto případě první souřadnici. Medián řady $(0, 1, 1, 3, 5, 6, 7, 8, 8, 9, 10)$ je číslo 6. Do levého podstromu tudíž umístíme objekty, které mají v první proměnné menší nebo stejnou hodnotu jako 6. Jedná se o o objekty:
  $\{[0, 1], [1, 0], [1, 1], [3, 3], [5, 5], [6, 8]\}$. Do pravého podstromu umístíme zbytek objektů. Dělení kvádrů takto pokračuje rekurzivně až k listům. Listové uzly reprezentují v tomto případě vždy jeden objekt.

- Máme-li vytvořený MRKD-strom a máme-li daná počáteční centra (nechť jsou stejná jako v příkladu 2.2, tedy body $\{[2, 1], [1, 4], [10, 7]\}$), můžeme spustit první iteraci. Fitrovací procedura se projeví v prvním kroku Lloydova algoritmu k-průměrů, jak je uvedeno v odstavci 3.5. Úkolem filtrovacího algoritmu je v každém uzlu MRKD-stromu odfiltrovat centra, která nemají šanci sloužit jako nejbližší centrum žádné podmnožiny bodů asociovaných s tímto uzlem. Heuristika užitá k tomuto filtrovaní byla vysvětlena v odstavci 3.4.

- Zabývejme se filtrováním kandidátských center $\{[2, 1], [1, 4], [10, 7]\}$ v uzlu u, který reprezentuje objekty $\{[8, 6], [7, 7], [9, 8], [8, 9], [10, 10]\}$. Situace je znázorněna na obrázku 3.2. Střed této množiny asociovaných bodů je objekt $[9, 8]$. Kandidátské centrum, které má nejblíže středu je bod $\boldsymbol{z}^* = [10, 7]$. Označme dále $\boldsymbol{z} = [1, 4]$. Vektor $u = \boldsymbol{z}^* - \boldsymbol{z} = [9, 3]$ má obě složky kladné. Vrchol buňky $V(H)$, který použijeme k rozhodnutí filtrování, má tedy podle poznámky 3.4 souřadnice $\boldsymbol{x}_{(2)} = [10, 10]$. Z obrázku je patrné, že $\|\boldsymbol{z} - v(H)\| \geq \|\boldsymbol{z}^* - v(H)\|$. Kandidátské centrum $\boldsymbol{z}$ tedy odfiltrujeme. Podobně by vypadalo filtrování kandidátského centra $[2, 1]$.

- Algoritmus v tomto uzlu odfiltroval obě dvě kandidátská centra a přiřadil objekty ke zbývajícímu centru. Rozhodl správně, že jako nejbližší centrum pro všechny asociované body musí sloužit právě bod $[10, 7]$. Ušetřil tak v Lloydově algoritmu několik dalších porovnávání vzdáleností. Časová úspora na takovém malém souboru dat samozřejmě není znát, na větších datových souborech by však podstatně urychlila běh programu.

- Po průchodu celým stromem byl ještě centru $[10, 7]$ přiřazen objekt $[6, 8]$. Statistiky $\alpha, \boldsymbol{\beta}, \gamma$ mají následující hodnoty:

$$\alpha = l^u + 1 = 5 + 1$$

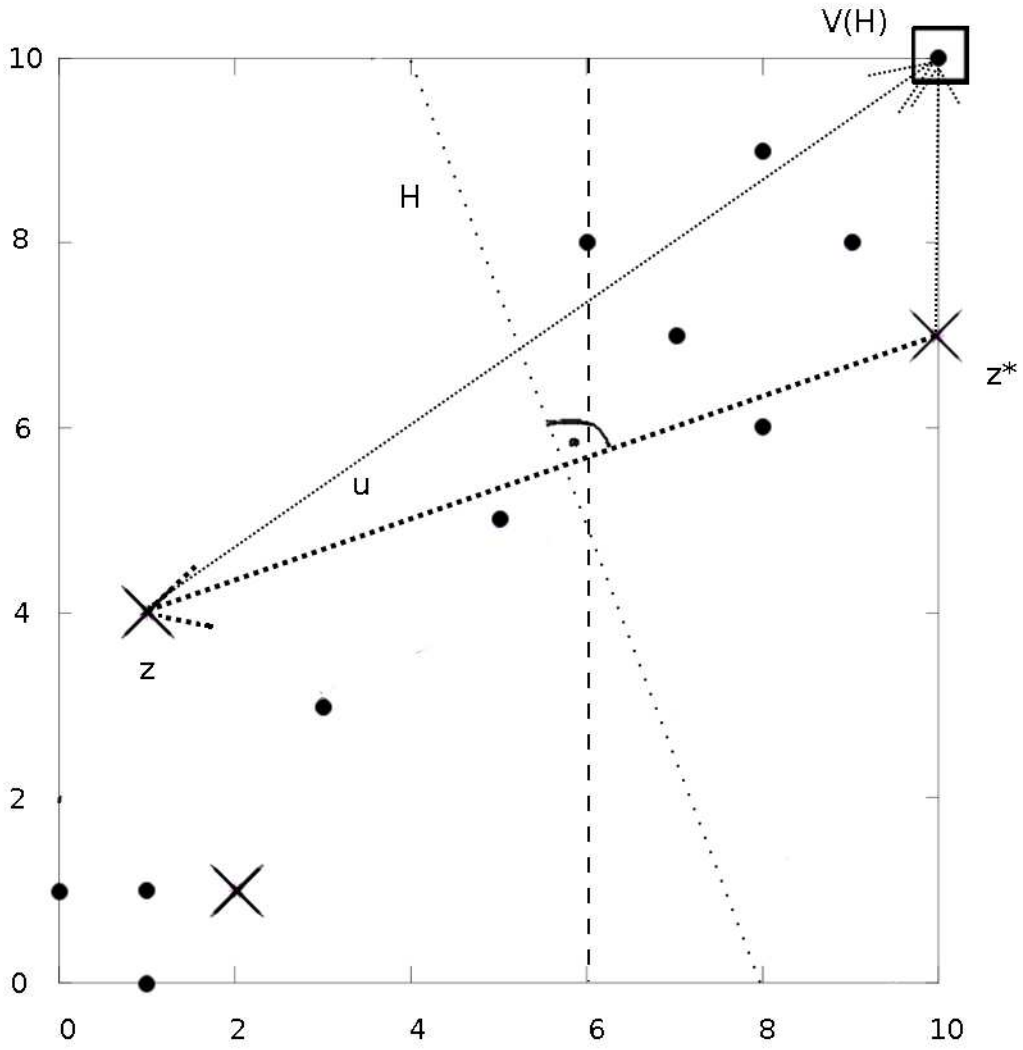
$$\boldsymbol{\beta} = [10, 7] + SS^u = [6, 8] + [8 + 7 + 9 + 8 + 10, 6 + 7 + 8 + 9 + 10] = [48, 48]$$

$$\gamma = 10^2 + 7^2 + \|\mathbf{X}\|_F^{2\,u} = 149 + 700 = 849$$

Centroid shluku objektů, které byly přiřazeny centru $[10, 7]$ je tedy $\boldsymbol{c}^1 = [48, 48]/6 = [8, 8]$. Pomocí uchovávaných statistik vypočteme ještě rozptyl tohoto shluku. Podle odstavce 3.3.2 je:

$$S_h^2 = \frac{1}{\alpha}\gamma - \left\langle \frac{\boldsymbol{\beta}}{\alpha}, \frac{\boldsymbol{\beta}}{\alpha} \right\rangle = \frac{1}{6} * 849 - \left\langle \frac{[48, 48]}{6}, \frac{[48, 48]}{6} \right\rangle = 13, 5$$

Program dále vypočte centroidy a rozptyly i ostatních shluků. Centroidy „pošle“ jako centra do další iterace a uloží si hodnotu ztrátové funkce Q. Konečné centroidy jsou shodné s centroidy na obrázku 2.2. Po druhé iteraci tedy algoritmus ukončí běh a výstup bude stejný jako výstup z Lloydova algoritmu bez filtrovací heuristiky.



Obrázek 3.2: Filtrování kandidátských center z příkladu 3.6. Na obrázku je čárkovaně označena levá příčka kvádru, v němž se nachází body asociované s uzlem u. Dále jsou tečkami nakreslené vektory $u = \boldsymbol{z}^* - \boldsymbol{z}$, $\boldsymbol{z}^* - V(H)$ a $\boldsymbol{z} - V(H)$, které posloužili k rozhodnutí o filtrování kandidátského centra $\boldsymbol{z}$. Na obrázku je dále tečkami nakreslena půlící nadrovina H. Je vidět, že všechny objekty asociované s uzlem u se nachází napravo od nadroviny H. Rozhodnutí o filtrovaní kandidátského centra $\boldsymbol{z}$ bylo tedy oprávněné. Ani jedno kandidátské centrum nemůže sloužit jako nejbližší centrum žádnému asociovanému objektu.

4. Algoritmus x-průměrů

4.1 Nedostatky algoritmu k-průměrů

Původní algoritmus k-průměrů (bez urychlení pomocí filtrovacího stromu) trpí třemi výraznými nedostatky. Prvním nedostatkem je fakt, že hodnota ztrátové funkce se především v pozdějších iteracích nezmenšuje natolik, aby odpovídala strojovému času, který je při těchto iteracích spotřebováván. Druhou nevýhodou je nutnost zadat uživatelem předem určenou hodnotu počtu shluků k. Třetí vadou algoritmu k-průměrů je nalezení pouze lokálně optimálního řešení rozdělení dat do shluků. Více o nevýhodách algoritmu se lze dočíst v podkapitole 2.3.

4.2 Algoritmus x-průměrů

Algoritmus x-průměrů se s těmito třemi nedostatky alespoň zčásti vypořádává. Pomocí Bayesova informačního kritéria definovaného v kapitole 1 porovnává rozklady množiny objektů pro různá přirozená k a přitom je relativně „rychlý“ díky filtrovací heuristice. Popíši nyní algoritmus x-průměrů tak, jak jej popsali autoři v článku (Pelleg, Moore, 2000,[4]) ve variantě s urychlením pomocí MRKD-stromů.

Algoritmus x-průměrů

0. Vstupem algoritmu je opět matice objektů $\{y_{i,j}\}_{i,j=1}^{n,m}$ (objekty tentokrát značíme pro přehlednost y), dvě přirozená čísla K_{max}, K_{min}, udávající meze pro počet shluků $K_{max} \geq K_{min}$. Dále přirozené číslo $maxiter$ udávající maximální počet iterací v globální i lokální proceduře k-průměrů a práh $\varepsilon \geq 0$ pro porovnávání hodnot BIC , který pro jednoduchost použijeme i pro porovnávání hodnot ztrátových funkcí v lokální i globální proceduře filtrovacího algoritmu. Na začátku se položí $x = K_{min}$, $BIC^{(0)} = \infty$ a $p = 1$, vygeneruje se počáteční x-prvková množina center $\boldsymbol{C}^{(1)}$ a vstoupí se do kroku 1.

1. Na data se aplikuje filtrovací algoritmus k-průměrů pro $k = x$, počáteční množinu center $\boldsymbol{C}^{(p)}$ a práh ε (průběh algoritmu je popsán v kapitole 3). Získá se x-prvková množina centroidů $\boldsymbol{c}^{(p)} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_x\}$.
Vypočteme hodnotu $BIC^{(p)}$ podle definice 1.6 a rozdělení shluků podle centroidů $\boldsymbol{c}^{(p)}$.
Vytvoříme si booleovskou proměnnou „prazdny“, kterou nastavíme na $false$. Pokud je nějaký shluk prázdný (to typicky znamená, že hodnota $BIC^{(p)}$ bude větší než hodnota $BIC^{(p-1)}$, čemuž chceme zabránit), změníme proměnnou „prazdny“ na $true$. Z množiny $\boldsymbol{c}^{(p)}$ odstraníme ty centroidy, kterým filtrovací algoritmus nepřiřadil žádné objekty a snížíme o příslušný počet prázdných shluků proměnnou x.
Je-li $((BIC^{(p-1)} - BIC^{(p)}) \leq \varepsilon$ a zároveň $prazdny = false)$ nebo $(p \geqslant maxiter)$, následuje ukončení algoritmu, s následujícím výstupem:

- Pokud se hodnota BIC oproti předchozí iteraci nezvětšila, platí $BIC^{(p-1)} \leq BIC^{(p)}$, je konečným rozdělením prostoru množina centroidů $\boldsymbol{c}^{(p)}$

- Pokud se hodnota BIC zvětšila, došlo ke globálnímu zhoršení rozdělení, vezmeme jako výstupní rozdělení množinu centroidů $\boldsymbol{c}^{(p-1)}$.

Všechny objekty přiřadíme nejbližším centroidům a na výstupu algoritmus uvede, kolik objektů se v jednotlivých shlucích nachází.
V opačném případě pokračujeme krokem 2.

2. Ve Voronojově buňce $V(\boldsymbol{c}_i)$ (jejich tvorba je vysvětlena v definici 1.4) se spustí modifikovaný lokální filtrovací algoritmus 2-průměrů pro počáteční volbu center $\boldsymbol{C}_i$, jejichž tvorbu stejně jako zmíněnou modifikaci vysvětlím níže a pro volbu prahové hodnoty ε. Získáme tak centroidy $\boldsymbol{c}_{i1}$ a $\boldsymbol{c}_{i2}$. Vše pro $i = 1, \ldots, x$. Pokračujeme následujícím krokem.

3. Položíme $k = x$. V i-té buňce se vypočte hodnota $BIC_{i1} = BIC_{\boldsymbol{c}_i}$ a $BIC_{i2} = BIC_{\boldsymbol{c}_{i1},\boldsymbol{c}_{i2}}$ pro variantu rozkladu Voronojovy buňky s jedním a dvěma centroidy a pro $i = 1, \ldots, k$. Seřadíme si buňky (centroidy) podle rozdílu $(BIC_{i2} - BIC_{i1})$ od nejmenšího po největší. První na řadě je tedy ta buňka, kde nastalo nejvýraznější zlepšení kritéria BIC. Pokračujeme následujícím krokem.

4. $\boldsymbol{C}^{(p+1)}$ je prázdná množina center. Položíme $j = 1$ a $x = 0$. Nyní přidáváme do množiny $\boldsymbol{C}^{(p+1)}$ centra podle následujícího pravidla:

- Je-li $BIC_{j2} < BIC_{j1}$ a zároveň $x \leq K_{max} + j - k - 2$, položíme $j = j + 1$, $x = x + 2$ a přidáme centroidy $\boldsymbol{c}_{j1}$ a $\boldsymbol{c}_{j2}$ do množiny $\boldsymbol{C}^{(p+1)}$.
- V opačném případě přidáme centroid $\boldsymbol{c}_j$ do množiny $\boldsymbol{C}^{(p+1)}$ a položíme $j = j + 1$, $x = x + 1$.

Vše pro $j = 1, \ldots, k$.
Množina $\boldsymbol{C}^{(p+1)}$ je x-prvková. Tímto postupem zaručíme, že v množině $\boldsymbol{C}^{(p+1)}$ budou zachovány centra, kde algoritmus nenavrhl rozdělení buňky na dvě a zároveň se „zdvojí“ centroidy s nejvýraznějším zlepšením lokální hodnoty BIC tak, aby hodnota x nepřekročila povolený počet shluků K_{max}. Algoritmus položí $p = p + 1$ a skočí na krok 1.

Poznámka 4.1.

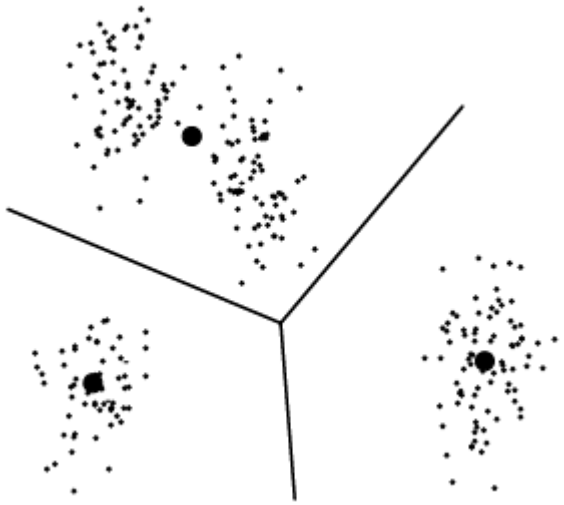
- **Modifikovaný lokální filtrovací algoritmus** je filtrovací algoritmus spuštěný na stejný MRKD-strom jako v kroku 1 na množinu center $\boldsymbol{C} = \boldsymbol{c}^{(p)}$. Modifikace spočívá v tom, že jakmile je v nějakém uzlu přiřazena množina objektů nějakému kandidátskému centru $\boldsymbol{c}_i, i = 1, \ldots, x$ (množina kandidátských center je jednoprvková nebo se nacházíme v listu), zaměníme toto centrum za dvě centra z množiny $\boldsymbol{C}_i$ nacházející se ve stejné Voronojově buňce. Algoritmus pustíme rekuzivně a na konci každé iterace upravíme centroidy $\boldsymbol{c}_{i1}$ a $\boldsymbol{c}_{i2}$. Otestujeme, zda není splněna podmínka konvergence pro všechny oblasti (ve všech buňkách teď probíhá algoritmus 2-průměrů) a pokud ano, pokračujeme v kroku 2 s množinou centroidů $\{\boldsymbol{c}_{i1}, \boldsymbol{c}_{i2}, i = 1, \ldots, x\}$

- **Počáteční volba center C_i** v kroku 2 spočívá v tom, že centra umístíme v každé Voronojově buňce $V(\boldsymbol{c}_i)$ co nejdál od sebe. Takovou volbu zajistíme tím způsobem, že si v 1. kroku ve filtrovacím algoritmu pro každé centrum budeme zaznamenávat vedle uchovávaných statistik, které jsem popsal v kapitole 3, ještě vektory minimálních a maximálních hodnot proměnných všech objektů. Tyto hodnoty si tak i tak uchováváme v MRKD-stromu pro realizaci filtrovací procedury, využijeme je tedy ještě k tomuto účelu.

- Autoři Pelleg a Moore popsali ve svém článku (2000) další případné urychlení algoritmu tím, že se ve 2. kroku algoritmu x-průměrů uchovávají v MRKD-stromu informace o tom, zda-li už byl v konkrétní Voronojově buňce algoritmus ukončen. Empirické testy totiž dokázaly, že zatímco některé shluky zůstávají klidné a stabilní po celou dobu algoritmu, některé mají stále tendenci se dělit. Pokud by šlo objekty v „klidných“ shlucích úplně odstranit z MRKD-stromu ve 2.kroku, byl by problém vyřešen. V tomto případě by se ale musel MRKD-strom v 2. kroku tvořit stále znovu, což je časově mnohem náročnější než případná úspora.

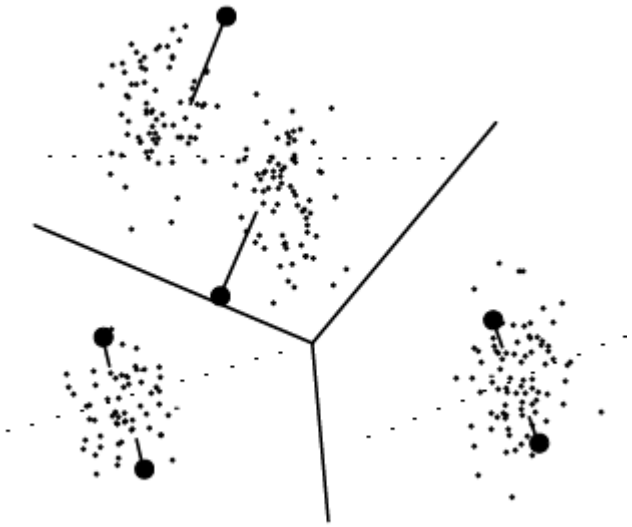
Algoritmus x-průměrů s použitím Bayesova informačního kritéria zaručuje, že centroidy shluků, které dobře modelují povahu dat, nebudou ovlivněny operacemi v oblastech, kde se na správném modelu ještě pracuje. Naopak, nové shluky budou vytvářeny prioritně v oblastech, kde špatná hodnota BIC ukazuje špatnou reprezentaci dat modelem. Proměnná hodnota počtu shluků je výhodná vzhledem k tomu, že někdy nevíme, z jakého rozdělení data pocházejí - kolik shluků bychom měli očekávat. Bylo by tedy třeba algoritmus k-průměrů spustit několikrát pro různé hodnoty k. Algoritmus x-průměrů stačí spustit jenom jednou.

Na následujících obrázcích je zřetelný postup algoritmu.



Obrázek 4.1: Po ukončení prvního kroku algoritmu x-průměrů pro $x = 3$ jsou objekty na obrázku rozděleny do shluků. Tlustou čarou jsou znázorněny hranice Voronojových buňek jednotlivých shluků. Zdroj [4]

.



Obrázek 4.2: Původní centroid byl v každé buňce rozdělen na dva. Na obrázku je patrný postup algoritmu 2-průměrů v každé Voronojově buňce. Čárkovaně je oblast Voronojových „podbuňek“ jednotlivých centroidů. Plnou čárou vedoucí od každého centroidu je znázorněn postup v jedné iteraci 2-průměrů. I když je tato iterace v každé buňce dělána nezávisle na ostatních objektech, je realizována jedním průchodem MRKD-stromu. Zdroj [4]

.

Obrázek 4.3: Po konvergenci 2-průměrů ve všech Voronojových buňkách byly vypočteny hodnoty BIC. Voronojova buňka, jež je na obrázku nejvýše, byla rozdělena na dvě oblasti, protože hodnota BIC_{i2} tohoto shluku rozdělení s dvěma centroidy byla nižší než hodnota BIC_{i1} tohoto shluku rozdělení s jedním centroidem. Algoritmus x-průměrů právě dokončil krok 4 a bude pokračovat krokem 1 pro x=4. Zdroj [4]

.

5. Testování algoritmů na datech

Filtrovací heuristiku v algoritmu k-průměrů (dále jen Filtrovací algoritmus), tak jak jsem ji popsal ve 3. kapitole, jsem implementoval v jazyku Matlab pomocí open-source programu Octave. MRKD-stromy jsem přitom reprezentoval cell strukturou, která umožňuje rekurzivní vkládání informací do matic přesně tak, jak se tomu děje při dělení ohraničujících hyperkvádrů při konstrukci MRKD-stromu. Dále jsem naprogramoval funkci, která pro rozklad množiny objektů do shluků vypočítá hodnotu Bayesova informačního kritéria definovaného v 1. kapitole. Algoritmus x-průměrů jsem naprogramoval taktéž v jazyku Matlab, za použití již sestaveného filtrovacího algoritmu. Pro srovnání jsem použil open-source program Weka [6], ve kterém je naimplementován algoritmus x-průměrů v jazyku Java. Tento software používám společně s Octave také ke grafickým výstupům. Pro spouštění skritptů z přiloženého CD je třeba použít program Octave. Ačkoliv je zápis kódu v Octave a Matlabu téměř stejný, v syntaxi se vyskytují nepatrné rozdíly, které by byly třeba před spuštěním mých skriptů v Matlabu upravit. Každý datový soubor testuji několika algoritmy. Část uvozená značkou **WEKA** označuje testování datového souboru pomocí algoritmu x-průměrů a programu WEKA. Část uvozená značkou **FILTALG** označuje testování datového souboru pomocí filtrovacího algoritmu v programu Octave. Část uvozená značkou **XMEANS** označuje testování datového souboru pomocí algoritmu x-průměrů a programu Octave.

Algoritmy testuji na několika různých sadách dat. Datový soubor **A** je uměle vygenerovaný soubor reálných objektů z rovnoměrného rozdělení. Datový soubor **B** je opět uměle vygenerovaný soubor reálných objektů, ale tentokrát ze čtyř různých normálních trojrozměrných rozdělení. Datový soubor **C** je dvojrozměrný uměle vygenerovaný soubor, kde jeden objekt je odlehlý od ostatních. Datový soubor **Nanočástice** je výstupem z fyzikálního měření polohy nanočástic v pevných látkách. Datový soubor **IRIS** je souborem měření korunních (petal) a okvětních (sepal) lístků.

Rozklady množin do shluků porovnávám pomocí ztrátové funkce Q popsané v první kapitole. Tuto funkci používá filtrovací algoritmus jako rozhodovací. Dále ke každému rozdělení uvádím hodnotu Bayesova informačního kritéria, které se snaží minimalizovat algoritmus x-průměrů. Důvod implementace filtrovacího algoritmu je urychlení procesu algoritmu k-průměrů. Uvádím tedy proto k některým testům filtrovacího algoritmu počet operací d, které byly potřebné k provedení každé iterace tohoto algoritmu. Operace d je porovnání vzdáleností vektorů $\boldsymbol{x}, \boldsymbol{y}$, tj. $d = d(\boldsymbol{x}, \boldsymbol{y})$. Je výpočetně nejnáročnější z celého algoritmu k-průměrů a pro ukončení každé iterace algoritmu k-průměrů z 2. kapitoly je potřeba fixní počet těchto operací roven $(k * n)$. Toto číslo uvádím zvášť k vybraným testům, aby bylo možné jej porovnat s náročností z filtrovacího algoritmu.

5.1 Programová implementace

5.1.1 Filtrovací algoritmus

Filtrovací algoritmus sepsaný v syntaxi jazyka Matlab lze najít na přiloženém CD. Program, který je implementací algoritmu vyloženého v odstavci 3.5 je rozdělen do pěti souborů. Hlavní a spouštěcí funkce je v souboru FILTALG_hlavni.m. Popis funkcí, vstupních a výstupních parametrů uvádím zde.

- **funkce FILTALG_hlavni**,
  volání [`centres`,`Q`,`cas`,`comp`]=FILTALG_hlavni(`data`,`k`,`maxiter`,`presnost`)
  vstupní parametry:

  - `data` - matice o rozměrech (n, m) v níž jsou uchovány informace o objektech. Každý objekt je na jednu řádku matice.
  - `k` - přirozené číslo udávající počet shluků, do kterých chceme objekty roztřídit.
  - `maxiter` - přirozené číslo udávající maximální možný počet iterací, které algoritmus provede
  - `presnost` $(= \varepsilon)$. Jedná se o číslo blízké nuly (např 0.01), které bude udávat maximální zmenšení ztrátové funkce mezi jednotlivými iteracemi, po kterém dojde k ukončení algoritmu.

  Popis funkce. Funkce obstarává správné spouštění algoritmu. Nejdříve spustí funkci, která vygeneruje počáteční centra. Poté spustí funkci, která vytvoří MRKD-strom a poté do počtu maximálních iterací nebo do ukončovací podmínky spouští funkci, která prochází MRKD-stromem a filtruje kandidátská centra. Na konci každé iterace funkce upraví umístění centroidů. Bližší informace o jednotlivých dílčích funkcích uvedu níže
  Výstup. Funkce uvede na výstupu umístění konečných centroidů, hodnoty ztrátové funkce Q v jednotlivých iteracích, dále čas, který byl spotřebován na každou iteraci a v proměnné comp počet operací d, které byly potřeba na provedení každé iterace.

- **funkce** [`cand`]= **FILTALG_initial_candidates**(`data`,`k`)
  Funkce vytvoří matici počátečních center. Centra vybere náhodně tak, aby jednotlivé souřadnice byly rovnoměrně rozděleny v celém vstupním prostoru.

- **funkce** [`knot`]= **FILTALG_vytvorMRKD**(`data`)
  Funkce vytvoří ze vstupních dat MRKD-strom a do každého uzlu uloží informace o asociovaných objektech, které jsou vyloženy v odstavci 3.3.1

- **funkce FILTALG_filtrujkandidaty**(`knot, candidates`)
  Funkce obstarává filtrování kandidátských center a přiřazování jednotlivým centrům statistiky vyložené v odstavci 3.3.2, pokud dojde k ukončovací podmínce průchodu stromem

- **funkce** `b`= **FILTALG_isfarther**(`zstar`,`z`,`minaxis`,`maxaxis`)
  Funkce vrátí logickou hodnotu udávající, zda je možné kandidátské centrum z odfiltrovat.

5.1.2 Algoritmus x-průměrů

Poněkud komplikovanější algoritmus x-průměrů, který je detailně popsán v odstavci 4.2 je implementován opět v jazyku Matlab. Využívá filtrovací algoritmus popsaný výše. Jednotlivé funkce jsou však v samostatných souborech, protože je dobré je odlišovat od funkcí filtrovacího algoritmu. Celkem se o běh programu stará 11 souborů. V každém souboru se nachází funkce, jejichž smysl vysvětlím níže. Program se spustí pomocí souboru XMEANS_hlavni.m.

- **funkce** [`vystup`,`bic`,`comp`,`Q`]= **XMEANS_hlavni(**`data`, `kmin`, `kmax`, `maxiter`, `epsilon`**)**,
  Vstupní parametry:
  - `data` - matice reálných čísel o rozměrech (n, m) s informacemi o n objektech.
  - `kmin` - dolní mez pro počet shluků
  - `kmax` - horní mez pro počet shluků
  - `maxiter` - maximální počet iterací algoritmu (v popisu algoritmu horní mez pro p)
  - `epsilon` - prahová hodnota pro změnu ztrátových funkcí BIC a Q. Používám ji pro zjednodušení jak v globálním cyklu pro BIC, tak ve všech voláních filtrovacího algoritmu, kde se pracuje se ztrátovou funkcí Q.

  Výstup:
  - `vystup` - matice o rozměrech $(x, m + 2)$. V prvních m sloupcích jsou hodnoty jednotlivých centroidů, které reprezentují shluky. V předposledním sloupci je počet objektů, které náležejí jednotlivým centroidům. V posledním sloupci je tento počet vyjádřený procentem z celkového počtu objektů. Proměnná x se určí v průběhu algoritmu. Platí $kmin \leq x \leq kmax$.
  - `bic` - vektor globálních hodnot Bayesova informačního kritéria (skončil-li algoritmus p-tou iterací, má vektor `bic` délku $p + 1$)
  - `comp` - vektor, jehož i-tá složka obsahuje informaci, kolik bylo potřeba v i-té iteraci potřeba provézt operaci d. Jedná se o vypočtení euklidovské vzdálenosti mezi vícedimenzionálními body v reálném prostoru.
  - `Q` - vektor, jehož i-tá složka obsahuje informaci, jaká byla v i-té iteraci hodnota ztrátové funkce Q (definice v 1. kapitole).

  Popis funkce. Hlavním tělem funkce je cyklus, který je ukončen, pokud se hodnota $BIC^{(p)}$ v p-té iteraci zmenší oproti $BIC^{(p-1)}$ méně než o ε nebo p dosáhne hodnoty $maxiter$. V každém běhu cyklu nejdříve proběhne globální konvergence parametrů. Jedná se o filtrovací algoritmus pro $k = x$. Potom se přejde k lokální úpravě struktury pomocí modifikovaného filtrovacího algoritmu. Průběh událostí lze shrnout do následujícího diagramu.
  Nejprve proběhne spuštění funkcí XMEANS_vytvorMRKD a XMEANS_initial_candidates, poté se přejde na cyklus, který rozdělím pro přehlednost na následující tři části:

1. *Globální* - Spuštěna funkce XMEANS_filtalg_globalne a XMEANS_BIC. Jedná se o upravení parametrů modelu s x shluky a vypočtení globální hodnoty Bayesova informačního kritéria
2. *Lokální* - Pokud $x < Kmax$, spuštěna funkce XMEANS_filtalg_lokalne a XMEANS_bic_lokalne. Podle výsledků lokálních hodnot Bayesova informačního kritéria se nyní každý shluk buď rozdělí na dva nebo zůstane nezměněný. Pokud se zvýší počet centroidů (shluků), začne se shluky, kde je změna nejvýraznější a s každým rozdělením se zvýší proměnná x o 1. Po ukončení lokální procedury není hodnota x vyšší než $Kmax$ a zároveň jsou zachovány všechny shluky, kde nedošlo k rozdělení.
3. *Ukončovací podmínka* - Pokud $((BIC^{(p-1)} - BIC^{(p)}) \leq \varepsilon$ nebo $(p \geqslant maxiter)$ dojde k ukončení algoritmu.

Algoritmus je ošetřen proti následujícím okrajovým případům:

- Je-li po kroku 1 nějaký shluk prázdný. Centroid tohoto shluku se vyřadí pro následující krok 2 a v této iteraci nemůže dojít k ukončení podmínkou $(BIC^{(p-1)} - BIC^{(p)}) \leq \varepsilon$. Velmi pravděpodobně se totiž hodnota BIC oproti původní iteraci zvýší a algoritmus by byl za normálních okolností ukončen.
- Je-li algoritmus ukončen při stavu $BIC^{(p-1)} \leq BIC^{(p)}$. V tomto případě se za výstupní vezme rozdělení z $(p-1)$ iterace. K tomuto případu dochází celkem často a je to způsobeno rozdílnou povahou lokálního a globálního výpočtu BIC. Pro kvalitu rozdělení je ale rozhodující globální hodnota BIC, a proto je v tomto případě rozhodnutí lokálních algoritmů potlačeno.

- **funkce XMEANS_vytvorMRKD**,
  Funkce je totožná s funkcí FILTALG_vytvorMRKD.

- **funkce XMEANS_initial_candidates**,
  Funkce je totožná s funkcí FILTALG_initial_candidates

- **funkce** [`centres`,`novacentra`,`ztrat`]= **XMEANS_filtalg_globalne**(`data`, `maxiter`, `presnost`, `centra`, `knot`),
  Funkce funguje na principu stejně jako FILTALG_hlavni. Rozdílem je pouze to, že funkce má jako parametr odkaz na kořen MRKD-stromu a také počáteční rozdělení centroidů. Výstupem této funkce jsou upravené centroidy, dále hodnota ztrátové funkce a kandidáti na nová dvě centra v každém shluku.

- **funkce XMEANS_filtrujkandidaty_globalne**,
  Funkce funguje jako FILTALG_filtrujkandidaty. Navíc předává funkci XMEANS_filtalg_globalne kandidáty na nová centra ve shlucích.

- **funkce** [`bic`,`podily`]=**XMEANS_BIC**(`cent`, `data`),
  Funkce vypočte hodnotu Bayesova informačního kritéria pro matici objektů, předanou proměnnou *data* a pro centroidy shluků, které předává proměnná

cent. Kromě hodnoty BIC funkce vypočítá i počet objektů, které náleží každému centroidu.

- **funkce XMEANS_filtalg_lokalne**,
  Funkce obstarává filtrovací algoritmus pro lokální algoritmus 2-průměrů v každé voronojově buňce.

- **funkce XMEANS_filtrujkandidaty_lokalne**,
  Jedná se o funkci, která stejně jako jí podobné provádí rekurzivní průchod stromem. Když se ale v této funkci dostaneme v nějaké úrovni stromu do fáze, kdy nám zbývá už pouze jediné kandidátské centrum, nahradíme toto jediné cetrum dvojicí nových center, která se nachází ve stejné Voronojově buňce. Průchod stromem pokračuje rekurzivně dál s touto dvojicí center. Nakonec je hodnota statistik konečného uzlu přičtena jednomu z „nových" center. Pokud průchod stromem skončí až v listu, statistiky obsažené v listu se přičtou bližšímu z dvou kandidátských center.

- **funkce** [`rozdelit,BIC`]=**XMEANS_bic_lokalne**(`data,centra, novacentra`),
  Funkce spočte pro všechny Voronojovy buňky okolo center z množiny *centra* hodnotu BIC pro aktuální rozdělení a rozdělení s novými centry, které jsou uloženy v proměnné *novacentra*. Pro každé centrum z množiny *centra* jsou dvě centra z množiny *novacentra*. Nacházejí se ve stejné Voronojově buňce. Na výstupu funkce ve vektoru *rozdelit* uvede i booleovskou informaci, zda je potřeba na základě hodnoty BIC příslušnou Voronojovu buňku rozdělit.

- **funkce XMEANS_isfarther**,
  Funkce je totožná s funkcí FILTALG_isfarther

5.1.3 Algoritmus x-průměrů s parametrem

Jedná se o algoritmus x-průměrů popsaný výše, který požaduje jeden vstupní parametr navíc. Tento parametr upravuje citlivost algoritmu na tvoření nových shluků v jeho lokální části. Na vstupu je zadáno číslo $par \geq 1$ udávající míru citlivosti BIC na tvorbu nových shluků. Pokud $par = 1$ jedná se o popsaný algoritmus x-průměrů. Čím vyšší bude par, tím méně bude algoritmus tvořit nové shluky. Hodnota modifikovaného BIC z definice 1.6 je v tomto skriptu změněna následovně:

$$BIC = 2\sum_{h=1}^{k} \xi_h + 2 * par * m * k \log(n)$$

5.2 Datový soubor A

Množina datových objektů A je zástupcem množiny objektů, které nejsou přirozeně rozděleny do shluků. Jedná se o 30000 pětirozměrných datových objektů, kde každá proměnná je náhodnou realizací spojitého rovnoměrného rozdělení z intervalu $\langle -10, 10 \rangle$.

WEKA Algoritmus x-průměrů byl spuštěn se vstupními parametry:

- $K_{min} = 4$, $K_{max} = 10$
- cutOffFactor= 0.5 (ekvivalent ε v popisu algoritmu x-průměrů)
- $maxIterations = 10, maxKMeans = 1000,$
  $MaxKMeansForChildren = 1000.$

Program rozložil množinu objektů na 4 shluky s počtem objektů 7332 (24%), 7670 (26%), 7544 (25%) a 7454 (25%).

Hodnocení kvality shlukování pomocí ztrátových funkcí:

$$Q = 459.86,\ BIC = 5.6068e + 005$$

Poprvé byl program spuštěn bez pomoci MRKD-stromů a celkový čas strávený v algoritmu byl 10.33 sekund. Podruhé jsem zaškrtl v programu Weka políčko „použít k výpoču MRKD-stromy“ a výpočet byl ukonček už po 7.95 sekundách se stejným výsledkem.

Každý ze čtyř počátečních shluků se algoritmus pokusil rozdělit na dva, ale hodnota BIC každého rozdělení byla v každém případě vyšší. Objekty tedy byly ponechány ve čtyřech shlucích a algoritmus byl ukončen po 1. iteraci.

FILTALG Filtrovací algoritmus byl pomocí Octave zavolán se vstupními parametry:

- $k = 4$
- $\varepsilon = 0.01$
- $maxiter = 10$

Na výstupu algoritmus uvedl centroidy, které reprezentují shluky s počty objektů 7311, 7739, 7649, 7301.

Hodnota ztrátových funkce dosahovaly $Q = 462.71$ a $BIC = 5.6227e+005$.

Filtrovací algoritmus tedy našel podle ztrátové funkce Q horší rozdělení než algoritmus x-průměrů spuštěný programem WEKA.

Zkusil jsem tedy zavolat filtrovací algoritmus ještě několikrát a nejlepší hodnota ztrátové funkce činila $Q = 459.96$, což je už jen o málo horší výsledek než poskytl algoritmus x-průměrů.

Jelikož i program WEKA skončil po první iteraci, výsledné rozdělení záviselo i zde pouze na počátečním rozdělení center, které bylo v případě volání filtrovacího algoritmu jiné. Prakticky jsme tedy ověřili, že dle očekávání kvalita shlukování závisí na počátečním rozložení shluků.

Počet provedených operací d podle jednotlivých iterací byl následující:

1	2	3	4	5	6	7
56586	54044	52294	50992	...	44442	43270

Vzhledem k tomu, že algoritmus k-průměrů bez heuristiky musí provést v každé iteraci $k * n = 4 * 30000 = 120000$ operací d, představuje filtrovací algoritmus v tomto případě přibližně dvojnásobné ulehčení výpočtu.

XMEANS Algoritmus x-průměrů byl pomocí Octave spuštěn se vstupními parametry:

- $K_{min} = 4,\ K_{max} = 10$
- $\varepsilon = 0.5$
- $maxiter = 10$

Algoritmus rozložil množinu objektů na 10 shluků s počtem objektů 2772, 2827, 2581, 2485, 3390, 3069, 3262, 3024, 3252, 3338.

Hodnocení kvality shlukování: $Q = 785.12, BIC = 5.1375e + 005$.

Program XMEANS spuštěný pomocí Octave se stejnými vstupními parametry našel tedy z hlediska Bayesova informačního kritéria lepší rozdělení než program WEKA. Z hlediska ztrátové funkce Q však nalezl horší rozdělení. Jelikož se hodnota ztrátové funkce Q v průběhu algoritmu zvyšovala s každou iterací (algoritmus ukončil dělení ve 3. iteraci, kdy dosáhl maximálního počtu shluků) bych řekl, že je odhad BIC podle definice 1.6 volen nevhodně. Výstupní vektor $Q = (477.59, 697.39, 785.12)$ potvrzuje, že nejmenší hodnotu Q mělo rozdělení po 1. iteraci algoritmu x-průměrů.

5.3 Datový soubor B

Množina datových objektů B je směsí bodů náhodných realizací trojrozměrného normálního rozdělení s různými směrodatnými odchylkami a s nezávislými složkami. Objekty byly generovány z $N([1,1,1],[0.3,0.3,0.5])$, dále $N([0,0,0],[0.1,0.01,0.4])$, dále $N([5,5,5],[0.5,0.5,0.5])$ a $N([2,2,2],[0.1,0.1,0.1])$. Množina obsahuje 2000 datových objektů, z nichž každých 500 objektů pochází z různého z uvedených trojrozměrných normálních rozdělení. Objekty tedy pochází z přirozených tříd v prostoru $\mathbb{R}^3$.

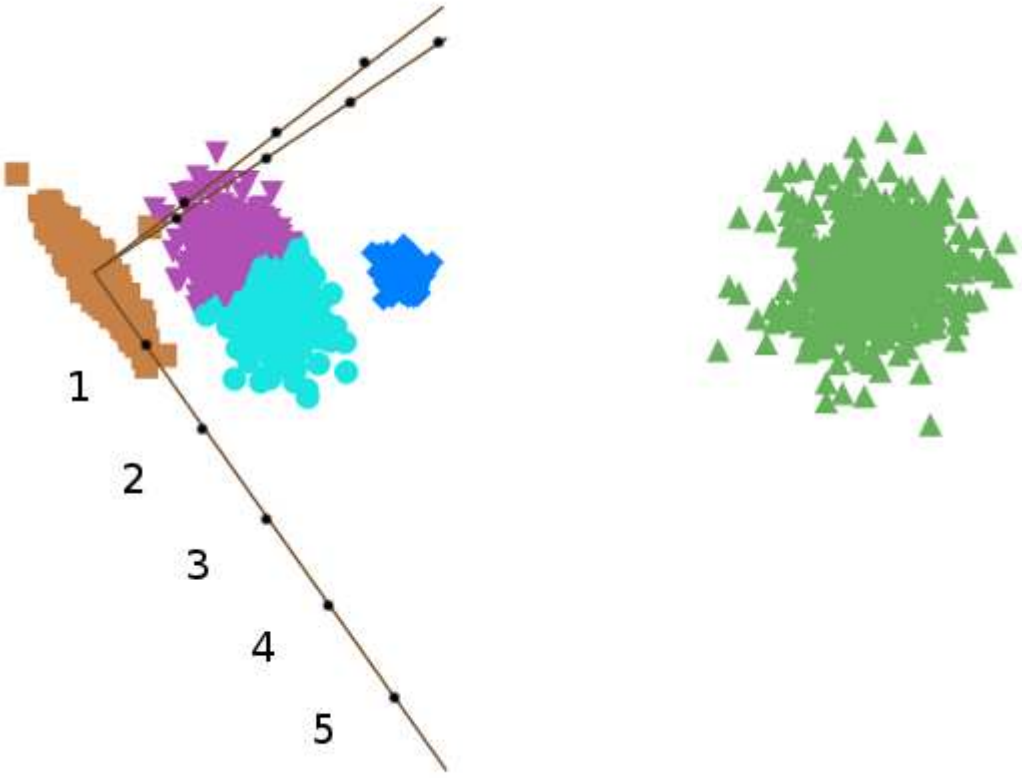
WEKA Algoritmus x-průměrů byl spuštěn s počátečními parametry:

- $K_{min} = 2,\ K_{max} = 8$
- cutOffFactor= 0.5
- $maxIterations = 10, maxKMeans = 1000,$
  $MaxKMeansForChildren = 1000.$

Algoritmus se zastavil po třech iteracích s rozdělením objektů do pěti shluků: 500 (25%), 500 (25%), 246 (12%),254 (13%), 500 (25%).

Hodnota ztrátových funkcí: $Q = 1.4757,\ BIC = -10159$.

Algoritmus správně rozdělil body podle tříd, ze kterých jsem původně data generoval, tedy do čtyř shluků a pak ještě jeden shluk rozdělil na dva. Centroidy těchto shluků jsou body $[1.034, 0.998, 1.467]$ a $[0.989, 0.963, 0.664]$. Rozhodnutí algoritmu odpovídá rozdělení třídy generovaných objektů z $N([1,1,1],[0.3,0.3,0.5])$ ve třetí proměnné, kde je největší směrodatná odchylka. Situaci lze shlédnout na obrázku 5.1.



Obrázek 5.1: Rozdělení objektů z množiny B do shluků pomocí algoritmu x-průměrů realizovaného programem WEKA. Je zde patrné velmi těsné rozdělení bodů reprezentovaných trojúhelníčky postavné na vrhol a bodů reprezentovaných vyplněnými kroužky podle nejdelší strany.

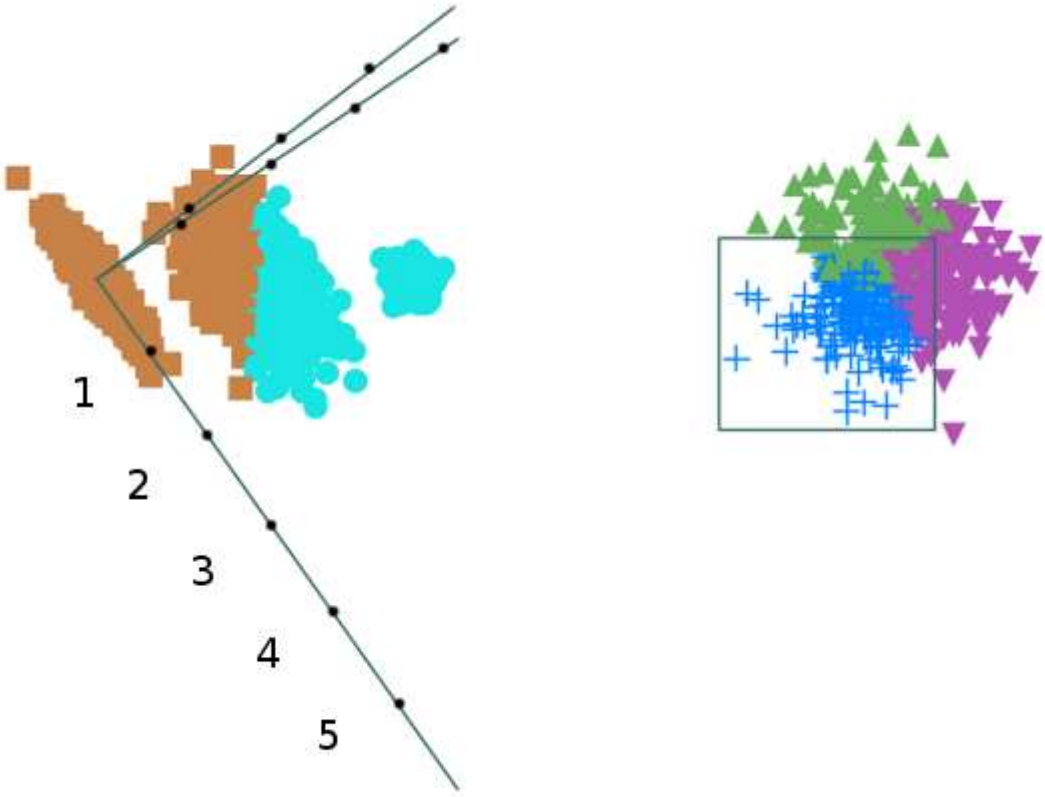
FILTALG Výstupem filtrovacího algoritmu (k=5) bylo úplně jiné rozložení do pěti shluků se zastoupením objektů $729, 177, 166, 771, 157$.

Hodnota ztrátových funkcí $Q = 2.6165$ a $BIC = -5497.1$ svědčí o horším rozdělení, než vypočítal program WEKA pomocí algoritmu x-průměrů. Tento fakt je dán odlišným počátečním rozdělením shluků a charakterem obou algoritmů. Situace je znázorněna na obrázku 5.2. Hodnota BIC je záporná protože vnitroshlukové rozptyly jsou menší než 1.

Algoritmus skončil po sedmi iteracích. Počty operací d, které byly potřeba k provedení každé iterace jsou patrné z následující tabulky.

1	2	3	4	...	6	7
775	809	979	1007	...	1465	1477

FILTALG Otestujeme ještě, zda-li je filtrovací algoritmus schopný data rozdělit do shluků odpovídající původním třídám. Uspokojující řešení vyšlo pro $k = 4$ až po několika pokusech, kdy bylo pokaždé na začátku algoritmu generováno jiné počáteční rozdělení shluků z rovnoměrného rozdělení.



Obrázek 5.2: Rozdělení objektů z množiny B do pěti shluků pomocí algoritmu k-průměrů.

Výstupem byly centroidy reprezetující shluky o velikostech $500, 504, 506, 490$.

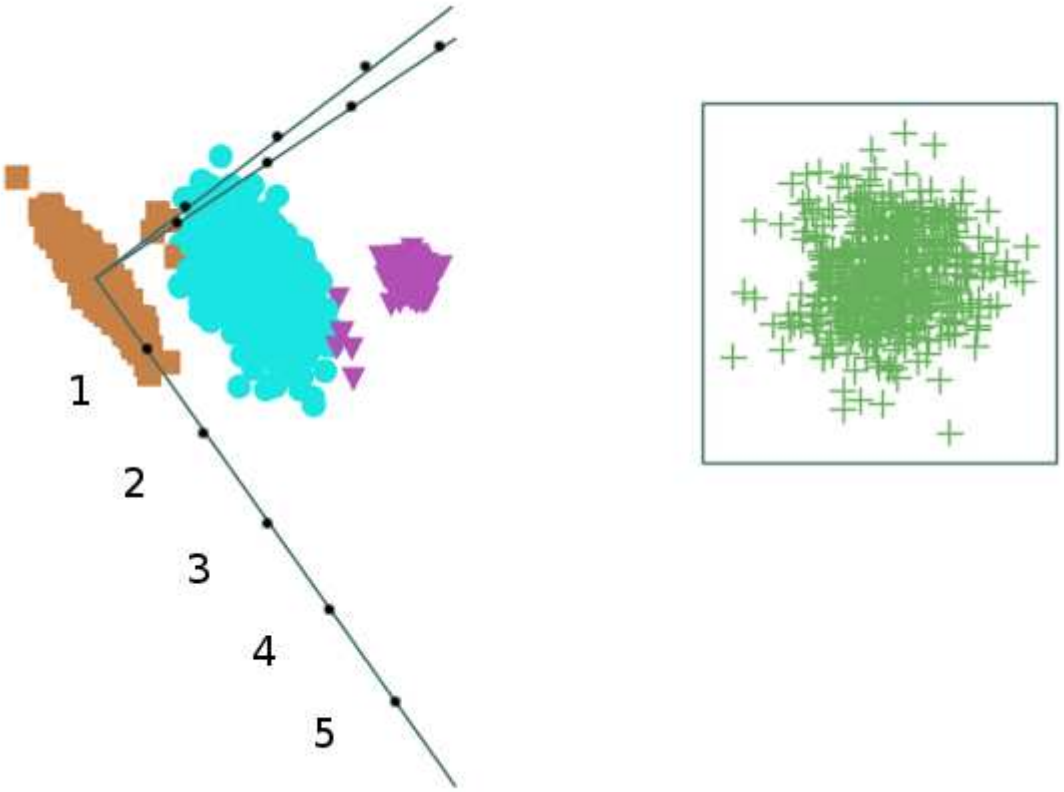
Hodnoty ztrátových funkcí $BIC = -9556.5$, $Q = 1.3724$. I když je tedy hodnota ztrátové funkce Q menší než u rozdělení vypočteného algoritmem x-průměrů, hodnota BIC je větší (což nevadí, filtrovací algoritmus se podle BIC neřídí). Situaci můžeme vidět na obrázku 5.3.

Provedené operace d v jednotlivých iteracích tohoto běhu algoritmu jsou zapsány v následující tabulce. Počet fixních operací d v každé iteraci algoritmu k-průměrů je roven $k * n = 4 * 2000 = 8000$. Filtrovací heuristika tedy v každé iteraci zamezila velkému množství zbytečně provedených operací d. Úspora těchto operací je oproti algoritmu k-průměrů více než desetinásobná. Filtrovací algoritmus spoří operace zejména ukončovací podmínkou č.2 popsanou v odstavci 3.1.

1	2	3	4	...	6	7
754	572	568	648	...	622	402

XMEANS Spustil jsem algoritmus x-průměrů s parametrem pomocí Octave na datový soubor B pro následující vstupní parametry.

- $K_{min} = 2$, $K_{max} = 50$



Obrázek 5.3: Rozdělení objektů z množiny B do čtyřech shluků pomocí filtrovacího algoritmu.

- $\varepsilon = 0.5$
- $maxiter = 10$
- $par = (1, 3, 5, 7, 9, 20)$

Účelem bylo otestovat, jak vytváření nových shluků v lokální části algoritmu x-průměrů závisí na parametru, jehož role je v kritériu BIC vysvětlena v odstavci 5.3.1. Hodnoty výsledného počtu shluků pro různé parametry *par* jsou uvedeny v následující tabulce.

par	1	3	5	7	9	20	100
x	50	46	21	13	10	5	4

Program WEKA se zastavil na hodnotě $x = 5$. Z tabulky je zřejmé, že abychom se dostali na stejnou hodnotu konečného počtu shluků, museli bychom zadat parametr $par = 20$. Algoritmus x-průměrů při této volbě našel rozdělení téměř shodné s rozdělením vypočítaným programem WEKA. Hodnota ztrátové funkce tohoto rozdělení činila $Q = 1.475$.

Při hodnotě parametru $par = 100$ algoritmus x-průměrů našel rozdělení s počtem shluků $x = 4$. Data jsem původně generoval ze 4 tříd, mohli

bychom tedy říci, že pro tuto hodnotu parametru *par* našel algoritmus model, který nejlépe vystihuje povahu dat. Skutečně, hodnota ztrátové funkce vyšla $Q = 1.3722$. Jedná se tedy o nejlepší rozdělení pro datový soubor B (hodnota Q pro filtrovací algoritmus s $k = 4$ činila 1.3724).

FILTALG Pro každou iteraci Lloydova algoritmu k-průměrů je při shlukování datového souboru B potřeba $k * n = 4 * 2000 = 8000$ operací. Počet operací d při použití filtrovacího algoritmu zavisí na tom, v kolikáté iteraci se algoritmus k-průměrů nachází. Výsledky můžeme vidět v následující tabulce, kde jsem testoval tři různá počáteční umístění center ve filtrovacím algoritmu pro $k = 4$ použitého na datový soubor B.

Iterace	1	2	3	4	5	6	7	8
Poč. rozd. č. 1	244	536	656	802	850	836	882	914
Poč. rozd. č. 2	520	814	882	886	852	860	878	882
Poč. rozd. č. 3	976	852	512	394	396	394		

Je tedy vidět, že počet provedených operací u filtrovacího algoritmu závisí nejenom na hodnotách k a n, ale také na počátečním rozdělení do shluků a fázi, v jaké se algoritmus nachází. V případě počátečního rozdělení č.1 počet provedených operací d s každou iterací stoupá, v případě počátečního rozdělení č. 3 naopak počet operací klesá. V každém případě je však zaznamenána podstatná úspora oproti algoritmu k-průměrů bez filtrovací heuristiky. Můžeme se dále domnívat, že počet operací filtrovacího algoritmu závisí také na přirozeném rozdělení objektů do shluků přesně tak, jak to představuje množina B.

5.4 Datový soubor C

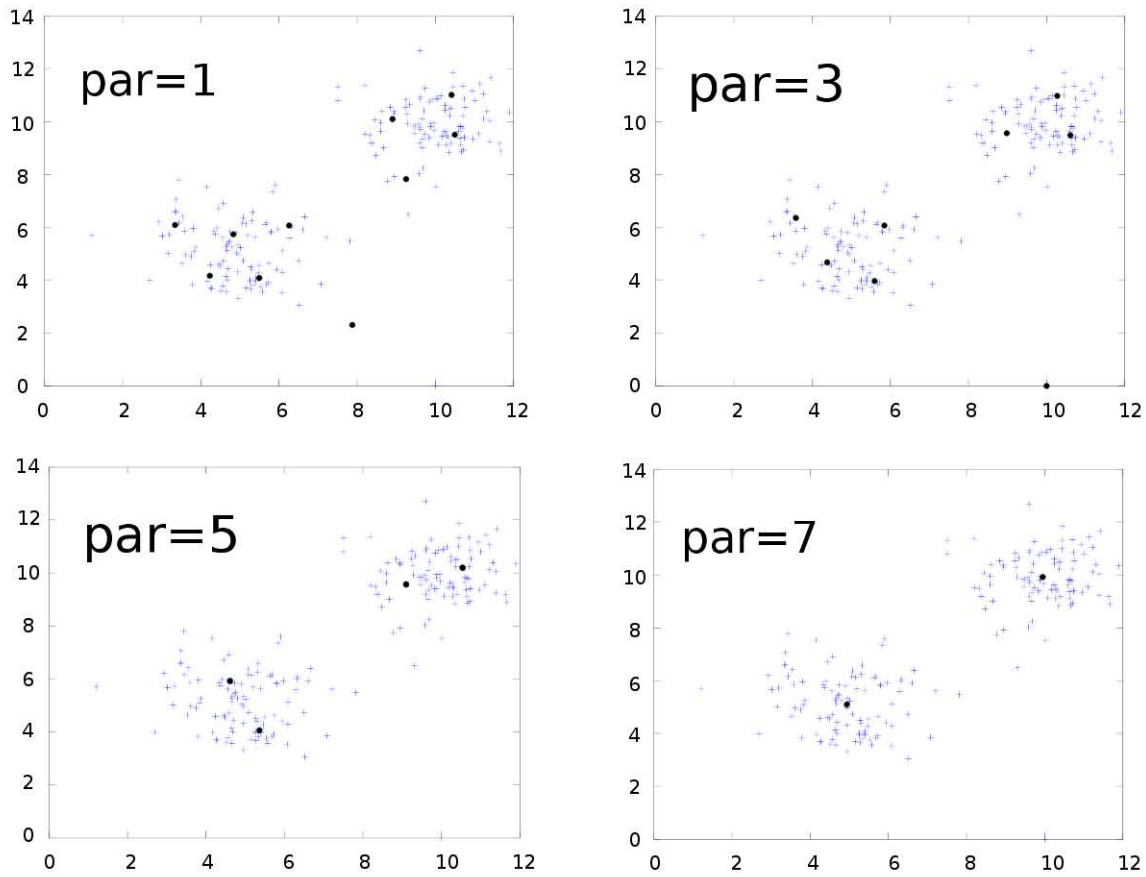
Datový soubor C je směsí 100 realizací dvojrozměrného normálního rozdělení se střední hodnotou [5,5] a směrodatnou odchylkou [1,1] s nezávislými složkami, 100 realizací dvojrozměrného normálního rozdělení se střední hodnotou [10,10] a směrodatnou odchylkou [1,1] s nezávislými složkami a odlehlého bodu se souřadnicemi [10,0].

Tento datový soubor posloužil k testování, do jaké míry je algoritmus x-průměrů s parametrem schopný modelovat povahu dat, vyskytuje-li se v datech odlehlý objekt.

XMEANS Algoritmus x-průměrů s parametrem byl na datový soubor C spuštěn s následujícími vstupními parametry, pro 4 různé hodnoty paramteru *par*.

- $K_{min} = 2$, $K_{max} = 50$
- $\varepsilon = 0.5$
- $maxiter = 10$
- $par = (1, 3, 5, 7)$

Jak je vidět na následujícím obrázku, pro $par = 3$ algoritmus vytvořil pro odlehlý objekt samostatný shluk. Pro $par = 7$ algoritmus umístil dva centroidy do středů tříd, ze kterých bylo původně generováno. Parametr *par* určuje váhu počtu shluků v lokálním výpočtu Bayesova informačního kritéria. Pro vyšší hodnoty parametru algoritmus spíše netvoří nové shluky.

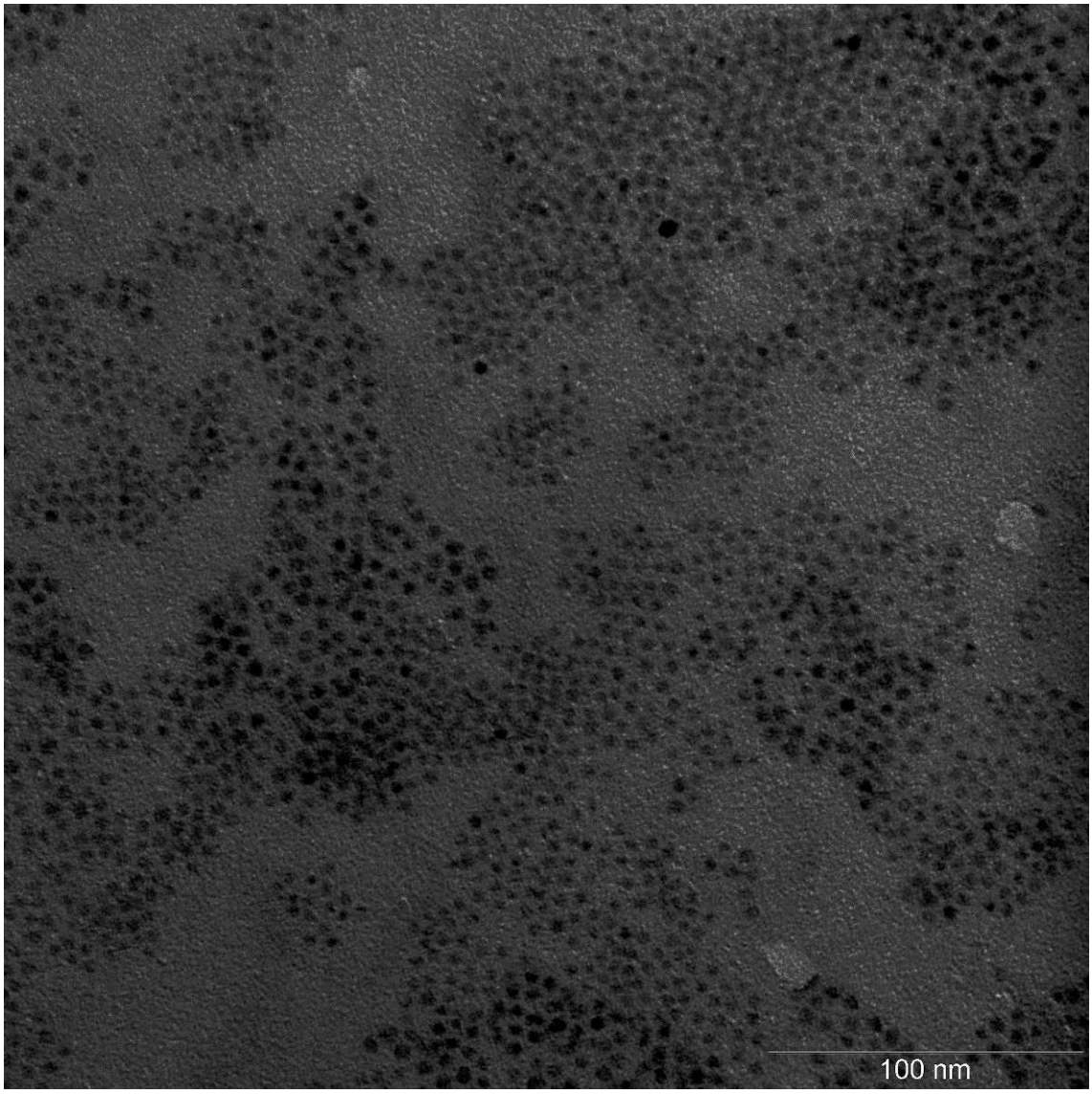


Obrázek 5.4: Datový soubor C a jeho rozdělení do shluků pomocí algoritmu x-průměrů. Značky "+"na obrázcích značí objekty a značky "o"centroidy jednotlivých shluků. Celkem jsou na obrázku 4 různé grafy pro 4 různé hodnoty parametru par.

5.5 Datový soubor Nanočástice

Data o rozměrech $(205, 3)$ pocházejí z měření velikosti a pozice nanočástic v zahřáté látce CDS prováděné na MFF v laboratořích pracoviště v Troji (obrázek 5.5). Každý objekt má zde tři proměnné. První proměnná určuje průměr částice v nanometrech, druhá pozici na vodorovné ose a třetí pozici na svislé ose taktéž v nanometrech. Zatímco průměry nanočástic jsou vesměs stejné, poloha jednotlivých částic se liší, jak je zjevné z následující tabulky.

č.proměnné	1	2	3
min	3	31	26
median	5	808	1096
max	7	2023	2007



Obrázek 5.5: Datový soubor Nanočástice ve zvětšení pod mikroskopem. Jsou zde zřetelné i nepatrné rozdíly v průměrech jednotlivých nanočástic. Průměr částit však zjevně nezávisí na jejich poloze.

Zadání zní: Rozdělte tyto nanočástice podle jejich parametrů do pěti shluků. K řešení použiji popsané algoritmy a program Octave. Algoritmy shlukují na základě všech tří proměnných, i když je zřejmé, že první proměnná se na rozložení shluků výrazně méně než druhá a třetí.

FILTALG V programu Octave spustím filtrovací algoritmus na data s následujícími vstupními parametry

- $k = 5$
- $\varepsilon = 0.1$
- $maxiter = 10$

Protože kvalita rozdělení závisí na počátečním rozdělení center, které je v naší interpretaci algoritmu voleno náhodně, spustím si algoritmus v jed-

noduchém cyklu desetkrát a zaznamenám si rozdělení do shluků, které vykázalo nejmenší hodnotu ztrátové funkce Q.

Nejmenší hodnotu konečné ztrátové funkce, $Q = 5.3070e5$, mělo sedmé spuštění. Algoritmus rozdělil objekty do shluků o zastoupení 30, 33, 38, 75 a 29. Toto rozdělení je znázorněno na obrázku 5.6. Průběh cyklu je popsán v tabulce pod hodnocením algoritmu x-průměrů.

XMEANS Spustím algoritmus x-průměrů na data s následujícími vstupními parametry.

- $K_{min} = 2$, $K_{max} = 5$
- $\varepsilon = 0.1$
- $maxiter = 10$

Jelikož kvalita výsledného rozdělení do shluků závisí podobně jako u algoritmu k-průměrů na počátečním rozdělení, spustím algoritmus v cyklu desetkrát a zaznamenám si rozdělení s nejmenší hodnotu BIC.

Nejmenší hodnotu Bayesova informačního kritéria mělo hned několik pokusů. Jednalo se o hodnotu $BIC = 4890.2$. V těchto případech byla hodnota ztrátové funkce $Q = 5.2961e5$. Algoritmus tedy nalezl dokonce lepší rozdělení než poskytl filtrovací algoritmus. Algoritmus x-průměrů rozdělil objekty do shluků o zastoupení 32, 76, 38, 29 a 30 objektů. Toto rozdělení je vidět na následujícím obrázku 5.6.

SROVNÁNÍ Algoritmus x-průměrů a filtrovací algoritmus jsem pro nalezení nejlepšího rozdělení pustil v cyklu desetkrát. V následující tabulce jsou výsledky ztrátových funkcí jednotlivých běhů algoritmu.

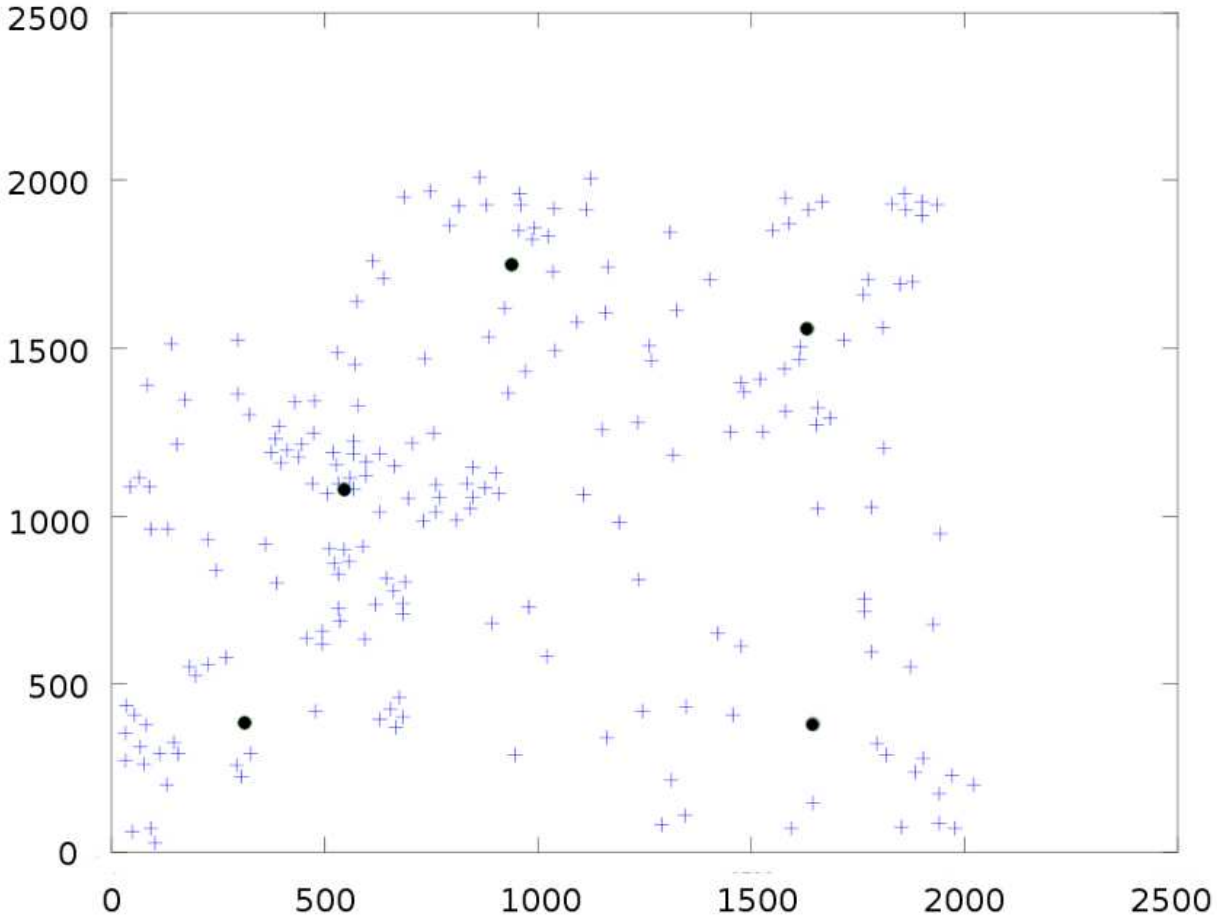
Běh	1	2	3	4	8	9	10
Q FILTALG	5.47e5	6.29e5	5.76e5	7.29e5	6.01e5	5.30e5	5.76e5
BIC XMEANS	4890.2	4890.2	4890.2	4890.2	4891.3	4891.3	4891.3
Q XMEANS	5.29e5	5.29e5	5.29e5	5.29e5	5.36e5	5.36e5	5.36e5

Jak z numerických hodnot, tak z grafického výstupu níže je zřejmé, že algoritmus x-průměrů je spolehlivější. Hodnoty ztrátových funkcí drží nízko při jakémkoliv počátečním rozdělení.

5.6 Datový soubor IRIS

Soubor pochází z datového uložiště [7]. V souboru se vyskytuje 150 objektů - květů, z nichž každý je popsán čtyřmi kvantitativními proměnnými

1. Délka okvětních lístků v centimetrech (sepal length)
2. Šířka okvětních lístků v centimetrech (sepal width)
3. Délka korunních lístků v centimetrech (petal length)



Obrázek 5.6: Datový soubor Nanočástice (pouze poloha částic) a jeho rozdělení do shluků. Pozice centroidů vypočítané pomocí algoritmu x-průměrů a filtrovacího algoritmu jsou okem nerozeznatelné, proto uvádím pouze jeden obrázek. V tomto grafu zobrazujeme polohu pouze podle druhé a třetí proměnné. Značky "+"na obrázcích značí objekty, značky "o"centroidy jednotlivých shluků.

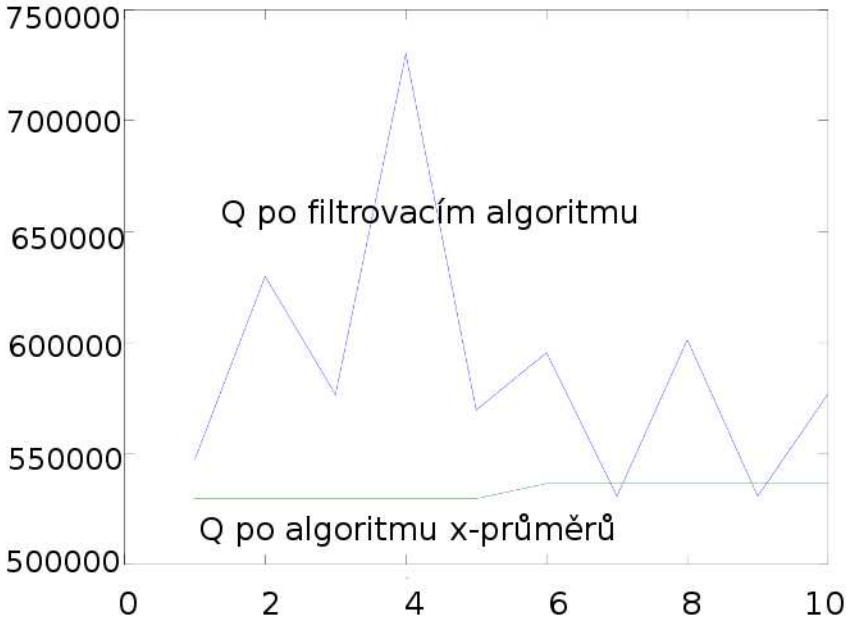
4. Šířka korunních lístků v centimetrech (petal width)

V souboru se vyskytuje 50 zástupců z každé ze tří tříd, Iris Setosa, Iris Versicolour, Iris Virginica. Algoritmy shlukové analýzy použijeme k úkolu přiřadit každý květ do třídy podle parametrů korunních a okvětních lístků.

FILTALG Pro nalezení nejlepšího rozdělení Algoritmus spustím opět desetkrát pro následující vstupní parametry.

- $k = 3$
- $\varepsilon = 0.1$
- $maxiter = 10$

Nejmenší hodnota ztrátové funkce napříč všemi běhy byla $Q = 1.5681$. Při tomto běhu algoritmus rozdělil objekty do třech shluků o velikostech 62, 38, 50. Do jednoho shluku tedy správně umístil všechny objekty z jedné třídy a do dvou shluků rozdělil (nikoliv libovolně) objekty z dvou tříd. Situaci lze shlédnout na obrázku 5.8.



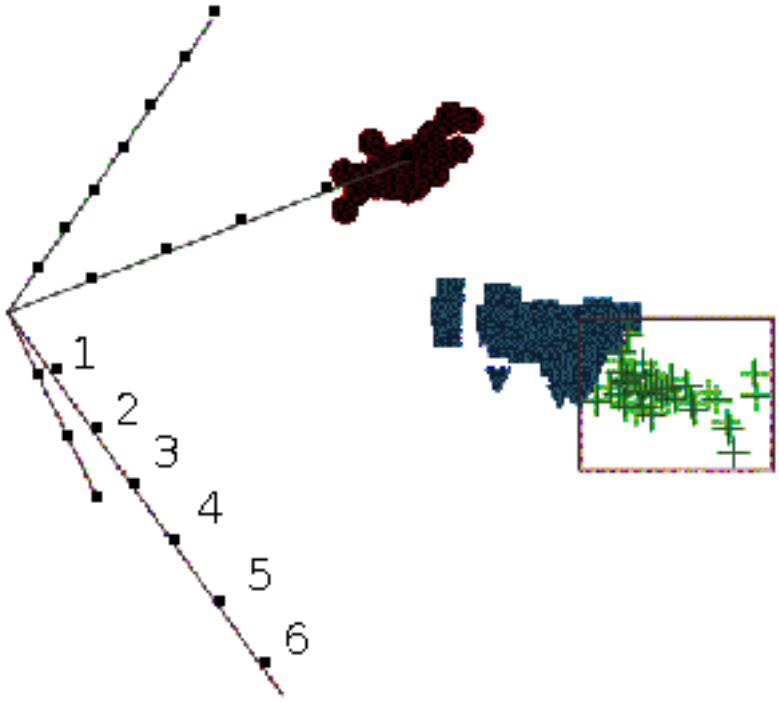
Obrázek 5.7: Srovnání hodnot ztrátových funkcí Q deseti běhů algoritmu x-průměrů a deseti běhů filtrovacího algoritmu. Z kolísání hodnot Q fitrovacího algoritmu je zřejmé, že kvalita rozdělení zde závisí na počátečním rozdělení. Hodnota Q algoritmu x-průměrů sice také závisí na počátečním rozdělení, díky charakteru algoritmu se však drží spolehhlivě nízko.

XMEANS Algoritmus jsem spustil desetkrát pro ověření stability rozdělení s následujícími vstupními parametry.

- $K_{min} = 2$, $K_{max} = 3$
- $\varepsilon = 0.1$
- $maxiter = 10$

Pokaždé algoritms x-průměrů ukončil běh s rozdělením do shluků o stejných velikostech jako filtrovací algoritmus, tedy 62, 38, 50. Nicméně umístění centroidů a tedy i rozložení shluků bylo odlišné. Odlišná byla vždy i hodnota ztrátové funkce, $Q = 1.5755$. I když tedy algoritmus stabilně ukončoval běh s menší hodnotou ztrátové funkce než filtrovací algoritmus, umístil vždy 12 objektů mimo původní třídy. Situace je na obrázku 5.9.

SROVNÁNÍ Stejně jako u datového souboru Nanočástice uvádím srovnání kvality shlukování filtrovacího algoritmu a algoritmu x-průměrů. Hodnoty ztrátové funkce Q pro 10 spuštění algoritmů jsou zřetelné z obrázku 5.10. Z tohoto srovnání vychází opět lépe algoritmus x-průměrů, jelikož prokazuje menší závislost na počátečním rozdělení center.



Obrázek 5.8: Graf datového souboru IRIS. Jednotlivé objekty jsou tvarově odlišené podle tříd. Několik trojúhelníčku pod čtverečky bylo filtrovacím algoritmem umístěno do shluku, který odpovídá jiné třídě.

5.7 Hodnocení algoritmů z hlediska praktické použitelnosti

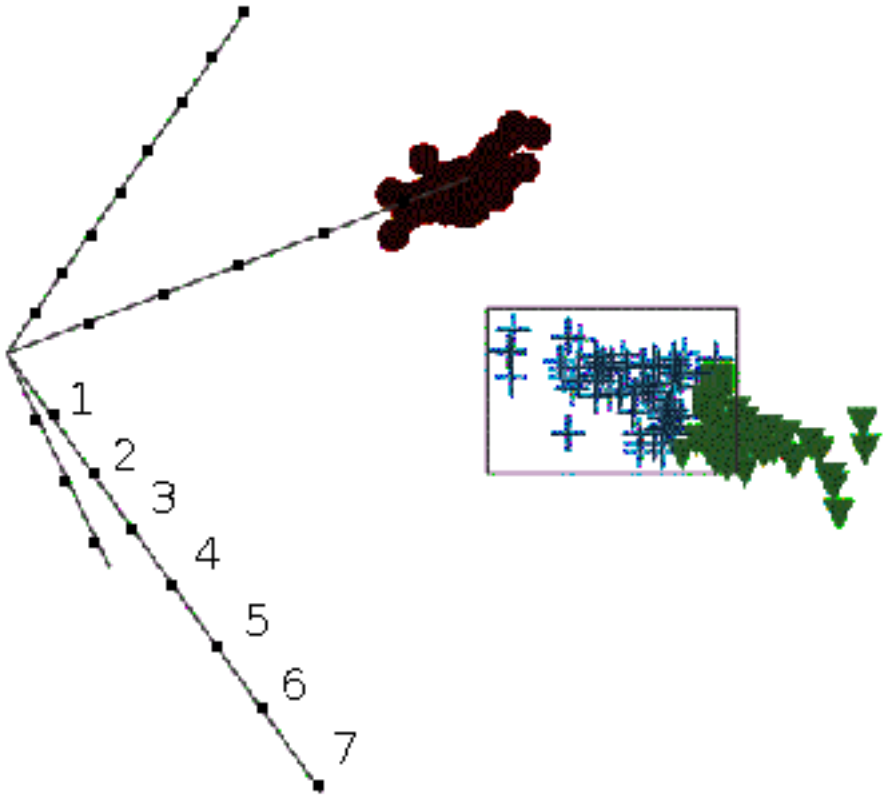
Jak filtrovací algoritmus, tak algoritmus x-průměrů vznikl z myšlenky algoritmu k-průměrů. Jejich výhody a nedostatky vztažené k algoritmu k-průměrů, které vzešly z praktického použití, shrnuji v následujícím odstavci.

Výhody filtrovacího algoritmu

- Filtrování kandidátských center - ušetření mnoha operací vypočtení vzdáleností mezi dvěma objekty.
- Shromažďování statistik urychluje rozhodování o dalším průběhu algoritmu.

Nedostatky filtrovacího algoritmu a jejich případné řešení

- Kvalita rozdělení podstatně závisí na počátečním rozdělení. Tento problém by mohl být vyřešen zavoláním nějakého inicializačního algoritmu, který navrhne vhodná počáteční centra. Takovým algoritmem je např. algoritmus K++. Pro spolehlivý výsledek filtrovacího algoritmu s náhodnou volbou počátečních center je třeba jej spustit několikrát a vybrat rozdělení s nejmenší hodnotou ztrátové funkce Q. Postup takového výběru je zřetelný na postupu shlukování datového souboru *Nanočástice* nebo *IRIS*.



Obrázek 5.9: Graf souboru IRIS. Jednotlivé objekty jsou tvarově odlišené podle tříd. Několik čtverců nad trojúhelníčky bylo algoritmem x-průměrů umístěno do shluku odpovídajícmu jiné třídě.

- Pro zjištění struktury dat je zapotřebí spustit algoritmus několikrát pro různé hodnoty počtu shluků k. Tento problém by šel vyřešit zakomponováním filtrovacího algoritmu do cyklu, který jej spustí několikrát pro různé hodnoty k s tím, že výstupem bude např. rozdělení, které mělo nejmenší hodnotu Bayesova informačního kritéria.

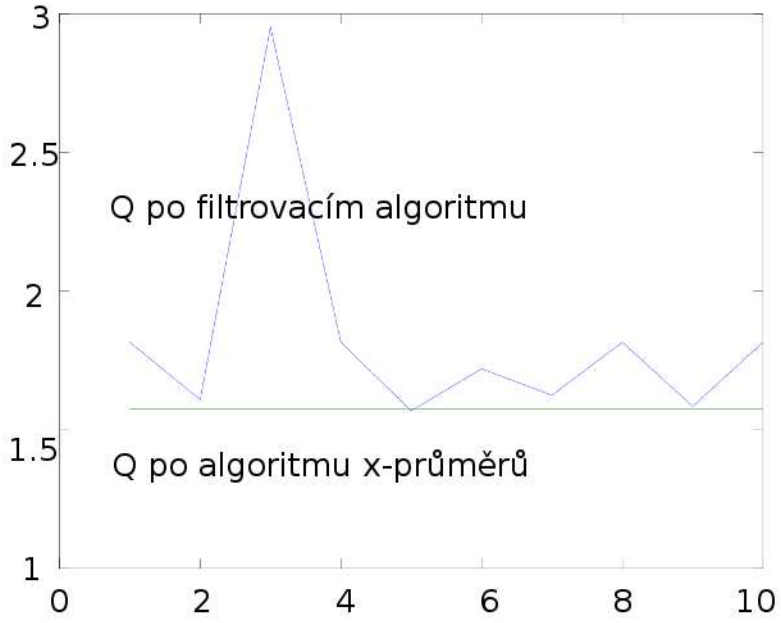
Výhody algoritmu x-průměrů

- Algoritmus dobře a spolehlivě modeluje povahu dat. V těch nejméně homogenních oblastech může navrhnout více shluků, přitom jej stačí spustit pouze jednou.

- Kvalita shlukování závisí méně na počátečním rozložení center než u algoritmu k-průměrů nebo filtrovacího algoritmu.

Nedostatky algoritmu x-průměrů a jejich případné řešení

- Bayesovo informační kritérium (definováno v 1. kapitole) v lokální části algoritmu je málo citlivé na proměnnou x. Pro praktické použití by nebylo špatné přidat do vstupu další proměnnou (podobně jak činí algoritmus x-průměrů s parametrem), která by určovala váhu aktuálního počtu shluků ve výpočtu BIC a zároveň by se vztahovala k danému souboru objektů. Na vstupu algoritmu by tak šlo přímo ovlivnit, zda má algoritmus spíše tvořit



Obrázek 5.10: Srovnání ztrátové hodnoty Q deseti běhů algoritmů x-průměrů a filtrovacího algoritmu.

nové shluky nebo zda chceme, aby se výsledná hodnota x blížila hodnotě $Kmin$.

- V některých oblastech vstupního prostoru algoritmus navrhne příliš mnoho shuků. Tento fakt závisí na počátečním rozdělení. Pokud se v nějaké iteraci algoritmu dva shluky, které jsou „blízko“ sebe, rozdělí na tři (v lokální části se jeden rozdělí na dva a v globální části tři centroidy dokonvergují v k-průměrech), bylo by vhodné za nějakých podmínek vyřadit jeden z centroidů, který už by se stal zbytečným.

5.8 Doporučené volby pro vstupní parametry algoritmů

5.8.1 Filtrovací algoritmus

Vstupní parametry filtrovacího algoritmu ovlivňůjí jeho běh. Algoritmus řeší problém roztřídit objekty do k shluků a většinou skončí při stavu, kdy hodnota ztrátové funkce předposlední iterace je shodná jako hodnota ztrátové funkce poslední iterace. U velkých souborů však konvergence nemusí být natolik zřejmá, a proto se vyplatí volit ukončovací parametry konvergence ε a $maxiter$. Doporučuji následující volbu vstupních parametrů:

- $k \in \mathbb{N}$, $k \geq 2$
- $\varepsilon \in [0, 1)$, algoritmus poběží i pro $\varepsilon \geq 1$, ale bude méně „přesný“
- $maxiter \in \mathbb{N}$, $10 \leq maxiter \leq 100$

Filtrovací algoritmus se vyplatí pustit víckrát pro stejné počáteční parametry a za řešení vzít to rozdělení s nejmenší hodnotou ztrátové funkce Q.

5.8.2 Algoritmus x-průměrů

Algoritmus x-průměrů může řešit dvě úlohy, podle kterých určíme vstupní paramtetry K_{max}, K_{min}.

1. Roztřídit objekty do k shluků. V tomto případě se vyplatí volit $K_{min} = 2$ a $K_{max} = k$. Jak je ukázáno na datových souborech Nanočástice a IRIS, algoritmus zde spolehllivě nalezne uspokojivé řešení i pro $K_{max} = 3$.

2. Najít takové x, aby si v něm shluky byly dostatečně podobné s ohledem na lokální hodnotu Bayesova informačního kritéria. V tomto případě je vhodné položit $K_{min} = 2$ a K_{max} zvolit veliké, např. $K_{max} = 50$. Dále je možné upravovat tendenci vytvářet nové shluky pomocí parametru v algoritmu x-průměrů. Tyto pokusy jsem ukazoval na příkladě datového souboru B a C. Ukázalo se, že parametr je vhodné volit v intervalu $(\log(n)m/16, 2\log(n)m)$, ale záleží na povaze dat a rozměru úlohy.

Podobně jako filtrovací algoritmus i algoritmus x-průměrů typicky dokončí běh v případě, kdy hodnota globálního Bayesova informačního kritéria v poslední iteraci je stejná nebo vyšší než hodnota tohoto kritéria v předposlední iteraci. Pro ostatní případy doporučuji následující vstupní parametry algoritmu.

- $\varepsilon \in [0, 1)$, algoritmus poběží i pro $\varepsilon \geq 1$, ale bude méně „přesný“
- $maxiter \in \mathbb{N}$, $10 \leq maxiter \leq 100$

Jak ukázaly pokusy na reálných datech, algoritmus x-průměrů stačí spustit pouze jednou a dospějeme k uspokojivému výsledku.

Závěr

V práci jsem teoreticky i prakticky popsal implementaci filtrovacího algoritmu a algoritmu x-průměrů, které oba vycházejí z urychlení algoritmu k-průměrů pomocí filtrovací heuristiky během průchodu MRKD-stromem.

Programy realizující oba algoritmy byly napsány v jazyku Matlab a testovány pomocí open-source programu Octave. Jelikož je volání funkcí (rekurzivní průchod MRKD-stromem) časově náročnější v tomto skriptovacím jazyku než např. v jazyku C nebo Java, nezaznamenal jsem časovou úsporu při provedení algoritmu k-průměrů s filtrovací heuristikou (v textu filtrovací algoritmus) oproti algoritmu k-průměrů bez heuristiky. Tuto časovou úsporu algoritmů hlavně u velkých souborů vyzdvihovali autoři Kannugo a kol. (2002), Pelleg a Moore (2000) ve svých článcích. Algoritmy měli napsané v jazyku C nebo v Javě, kde je rekurzivní průchod stromem časově nenáročný.

Jelikož smyslem této práce bylo především matematicky popsat oba algoritmy a vysvětlit jejich implementaci, ověření časové úspory jsem nahradil ověřením úspory potřebných operací vypočtení euklidovské vzdálenosti $d(\boldsymbol{x}, \boldsymbol{y})$. Při testování filtrovacího algoritmu jsem skutečně zaznamenal úsporu těchto operací oproti algoritmu k-průměrů, který potřebuje ke každé iteraci fixní počet operací d. Díky chytré myšlence filtrovací heuristiky nižší potřebný počet operací d filtrovacímu algoritmu nijak neubral na funkčnosti. Hodnotím jej tedy jako přínosný do skupiny metod shlukovací analýzy.

Bylo by zajímavé dále zkoumat funkčnost algoritmu, kdybychom místo absolutního prahu ε uvažovali relativní práh vzhledem k datům. Určitě by bylo také vhodné nahradit náhodnou volbu počátečních center nějakým promyšleným inicializačním algoritmem. Dále je možné experimentovat s metodami dělení MRKD-stromů a bylo by jistě přínosné popsat implementaci filtrovacího algorimtu na jiná než kvantitativní data při použití různých standardizací dat. Nicméně tyto úvahy už byly nad rámec textu.

Algoritmus x-průměrů je výhodný zejména pro situace, kdy si před spuštěním programu nejsme jisti, kolik shluků má být z dat vytvořeno. Nicméně i pro známé horní hranice $K_{max} = k$ dává tento algoritmus lepší a spolehlivější výsledky než filtrovací algoritmus k-průměrů, spustíme-li jej např. se vstupní hodnotou $K_{min} = 2$. V průběhu algoritmu totiž zůstávají „klidné“ shluky, kde je povaha dat modelována dostatečně, a naopak se mění oblasti, kde se na dobrém modelu z pohledu Bayesova informačního kritéria ještě pracuje.

Zatímco filtrovací algoritmus jsem naprogramoval přesně podle popisu v [3] a [1], pro algoritmus x-průměrů přesný návod k dispozici nebyl, a tak jsem pravděpodobně některé věci implementoval jinak než bylo navrženo v článku [4]. Konkrétně výpočet Bayesova informačního kritéria probíhá v mé interpretaci dle vzorce uvedeného v [2] a volba počátečních center v lokální části algoritmu x-průměrů probíhá taktéž pravděpodobně jinak. V testech své implementace algoritmu x-průměrů jsem skutečně zaznamenal menší citlivost lokálního Bayesova informačního kritéria na počet shluků x než program Weka, v němž je algoritmus naimplementován. Nicméně funkčnost tohoto algoritmu je velice uspokojivá, jak jsem dokázal v testech shlukování reálných dat z výzkumu.

Seznam použité literatury

[1] Žambochová, Marta: *Shluková analýza rozsáhlých souborů dat: nové postupy založené na metodě k-průměrů*. Disertační práce. Vysoká škola ekonomická v Praze, 2010.

[2] Řezanková, Hana, Húsek, Dušan, Snášel, Václav: *Shluková analýza dat*. Professional Publishing, 2. vydání, Praha 2009. ISBN 978-80-86946-81-8.jk

[3] Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, CH. D., Silverman, R., Wu, A. Y.: *An Efficient k-Means Clustering Algorithm. Analysis and Implementation*. IEEE Transactions on pattern analysis and machina inteligence, 24(7), 2002.

[4] Pelleg, D., Moore, A.: *x-means: Extending k-means with Efficient Estimation of the Number of Clusters*, Proc. 17th Int'l Conf. Machine Learning, July 2000.

[5] `http://en.wikipedia.org/wiki/Voronoi_diagram`

[6] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. http://www.cs.waikato.ac.nz/ml/weka

[7] `http://archive.ics.uci.edu/ml/datasets/`