ZÁPADOČESKÁ
UNIVERZITA
V PLZNI

Semestrální práce z předmětu KIV/SU

Shlukovací algoritmy

Datum:
13. února 2017

Vypracovali:
Matěj Kareš
Vojtěch Kinkor

Obsah

1 Zadání

Cílem práce je vytvoření aplikaci pro shlukovou analýzu dat různými metodami. Aplikace bude umožňovat pracovat s daty libovolné dimenze. Součástí bude vizualizace dat ve 2D prostoru (tedy v prostoru omezeném na jednu až dvě dimenze) v podobě bodového grafu. Data bude možné náhodně vygenerovat, případně načíst ze souboru. Výsledek shlukování bude možné exportovat do souboru.

2 Shluková analýza

Shluková analýza (případně též shlukování, clusterová analýza, clusterizace) se zabývá postupy, pomocí kterých lze v netříděných data nalézt shluky dat s podobnými vlastnostmi. Cílem je, aby podobnost dvou objektů v jednom shluku byla maximální, zatímco podobnost objektů v různých shlucích byla minimální. Výsledkem je nalezení vztahů mezi objekty bez vysvětlení, proč tyto vztahy existují.

Shlukem tedy nazýváme skupinu objektů, které jsou si navzájem podobné a rozdílné od objektů z jiných shluků.

Obvykle se mluví o tzv. „hard“ shlukování, které znamená, že shluky jsou vzájemně disjunktní a každý objekt patří právě do jednoho. V opačném případě se jedná o „soft“ nebo fuzzy shlukování – každý objekt může patřit do více shluků zároveň.

V rámci strojového učení se jedná o metodu učení bez učitele (unsupervised learning).

Shluková analýza má mnohé uplatnění, mj.:

- při porozumění datům a jejich struktuře
- použití v biologii (např. seskupení DNA sekvencí do genových rodin)
- průzkum trhu (rozdělení zákazníků do tržních segmentů)
- analýzy sociálních sítí
- zpracování obrazu (detekce hran, rozpoznávání objektů)

Algoritmy shlukové analýzy lze dělit na hierarchické a nehierarchické. První zmíněné využívají dříve nalezených shluků a na základě nich vytváří shluky nové. Naopak nehierarchické berou vždy vstupní množina dat jako celek, který se následně snaží rozdělit. Tato práce se dále zabývá právě nehierarchickými algoritmy.

2.1 Podobnost objektů v datech

Podobnost dvou objektů lze obvykle vyjádřit jako jejich vzájemnou vzdálenost. Pro mnoho typů dat lze použít různé metriky, mezi nejčastější patří následující dvě:

- Euklidovská vzdálenost:
$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{p} (x_{ik} - x_{jk})^2}$$
- Manhattanská vzdálenost:
$$d(x_i, x_j) = \sum_{k=1}^{p} |x_{ik} - x_{jk}|$$

V případě práce s textem lze například použít Levenshteinovovu vzdálenost, pro binární data Hammingovu vzdálenost, apod.

2.2 Formalizace problému

Obecně lze shlukování formalizovat na optimalizační problém, jehož vstupem jsou:

- trénovací data,
- vzdálenostní (cenová) funkce,
- optimalizační kritérium;

a neznámými veličinami:

- počet shluků,
- přiřazení jednotlivých objektů do shluků.

Cílem je pak minimalizace vzdálenostní funkce. Jedná se o tzv. NP-těžký problém, úlohy se tedy v praxi řeší heuristicky.

2.3 K-Means

Tato shlukovací metoda vychází z předpokladu, že data lze vnímat jako body v euklidovském prostoru. Mezi tyto body je zaneseno k fiktivních bodů, tzv. *centroidů*. Centroidy slouží jako geometrický střed jednotlivých shluků a příslušnost bodu ke shluku je tak dána nejbližším centroidem. Nejčastější metriky pro určení vzdálenosti u metody K-Means jsou euklidovská a manhattanská vzdálenost. Algoritmus funguje iteračně. Po přiřazení všech bodů nejbližším centroidům jsou centroidy posunuty do geometrického středu svého shluku a algoritmus je spuštěn znovu tolikrát, dokud není posun centroidů menší než zvolená hranice, případně se centroidy už nehýbou.

Algoritmus (často nazýván dle svého autora Lloydův algoritmus) pracuje následovně:

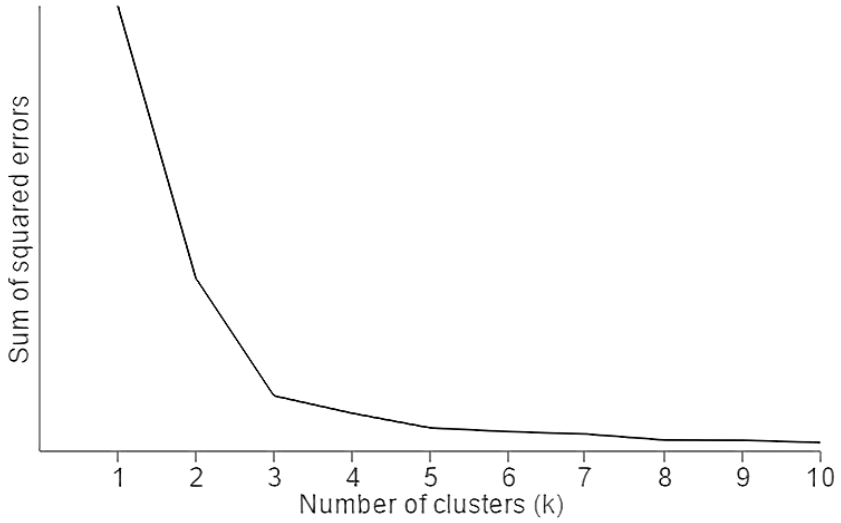
1. Vytvoří se k centroidů (na pozicích náhodně zvolených bodů z množiny vstupních dat).
2. Každému bodu se přiřadí nejbližší centroid (tvorba shluků).
3. Centroidy se posunou do geometrického středu jejich shluku.
4. Krok (2) dokud nedojde k ustálení centroidů.

Kvalita shlukování je určena funkcí J, jejíž výstupem je průměrná vzdálenost bodů od jejich centroidů. Algoritmus K-Means spouštíme vícekrát pro různé počáteční body a hledáme nastavení takové, aby hodnota funkce J byla co nejnižší. Pro n opakování algoritmu dostaneme n hodnot funkce J, z nichž nejmenší je lokální optimum. Globální optimum by bylo možné najít, pokud bychom vyzkoušeli všechny možné počáteční hodnoty pro inicializaci centroidů, avšak zlepšení výsledku funkce J už bývá nepatrné a časová náročnost je neúměrná zlepšení.

Počet shluků k je dán jako parametr. K-Means nainicializuje k náhodných počátečních bodů v prostoru, které jsou následně posouvány k nejbližším shlukům. U náhodně volených centroidů v prostoru může nastat situace, kdy bude jeden centroid umístěn tak daleko, že na něj nezbude žádný bod (jinými slovy každý bod je blíže jinému centroidu). V tomto případě je shluk prázdný a zaniká. Tato situace je nežádoucí vhledem k tomu, že uživatel zadává, kolik shluků očekává. Je tedy vhodné jako počáteční volit existující body z množiny vstupních dat, tímto způsobem budou počáteční pozice centroidů vždy u nějakého bodu a nestane se, že by clusteru nepřipadl žádný bod. Z této skutečnosti plyne také podmínka, že počet shluků k musí být menší než velikost vstupních dat.

Pro zjištění počtu shluků lze využít tzv. *metodu lokte (elbow method)*. Princip metody spočívá v inkrementálním volení počtu shluků a restartování algoritmu dokud nedojde k výraznému zlepšení hodnoty funkce J.

Na obrázku 1 je na ose x počet shluků a na ose y výstup funkce J. V grafu hledáme místo, ve kterém dochází k největšímu zlomu (zde ke zlomu dochází v případě 3 shluků). Metoda lokte není příliš spolehlivá u dat s velkým šumem, jelikož graf poměru počtu shluků a funkce J bude „hladší“ a nemůže tedy být spolehlivě řečeno, kde zlom nastal. Metoda lokte musí být shora omezena maximálním počtem shluků, který musí být menší než celkový počet bodů v množině dat, jinak dojde k situaci, kdy se každému bodu přiřadí právě jeden shluk.



Obrázek 1: Příklad závislosti hodnoty funkce J na počtu shluků.

Existuje mnoho modifikací této metody, jednou z nich je metoda K-Medians. Rozdíl spočívá v tvorbě nových centroidů pomocí mediánu všech bodů shluku (po jednotlivých souřadnicích) a používání Manhattanské metriky při přiřazování bodů ke shlukům. Smysl modifikace spočívá v toleranci výchylek v datech.

2.4 DBScan

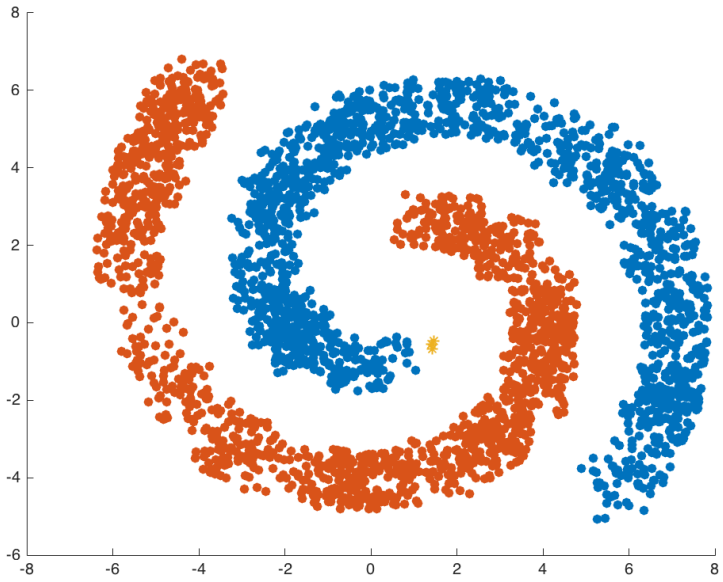
Principem algoritmu je nalezení nepřiřazeného bodu, kterým se začne tvořit nový shluk. Pokud se v ε okolí tohoto bodu nachází další body, jsou přidány do vznikajícího shluku a na všechny nově přidané je znovu aplikováno hledání dalších sousedů. Takto algoritmus najde x bodů, které jsou od sebe vzájemně vzdálené maximálně ε. Pokud je x větší než zvolené minimum, je shluk uznán za platný, v opačném případě je shluku rozpuštěn a pokračuje se dalším nepřiřazeným bodem. Jinými slovy – hledají se pouze shluky od určitého počtu bodů. Algoritmus lze zapsat následovně:

1. V množině bodů se najde první bod, který není přiřazen žádnému shluku a který ještě nebyl zvolen jako počáteční.
2. Prohledá se ε okolí tohoto bodu a je-li nalezen další bod, je přidán do shluku.
3. Pro každý nově přidaný bod do shluku se provede krok (2) dokud budou další body přibývat. Není-li přidán žádný nový bod pokračuje se na krok (4).
4. Pokud je v nově vzniklém shluku více bodů než je zadané minimum, je shluk uznán za platný, v opačném případě je rozpuštěn.
5. Krok (1), dokud existují „nenavštívené“ (nepřiřazené a nezvolené) body.

DBScan, na rozdíl od metody K-Means, dokáže shluky určit i když nejsou lineárně oddělitelné (viz obrázek 2) a navíc dokáže rozpoznat odlehlé body, které jsou pouze šumem (tzv. *outliery*). DBScan také sám určuje počet shluků a není potřeba počet zadávat nebo vypočítávat.

Pro svůj běh potřebuje znát dva parametry, jejichž volba vyžaduje určitou znalost dat.

1. Minimální počet bodů, které smí utvořit shluk.
2. Maximální vzdálenost dvou bodů ε. Body blíže než ε jsou považovány za sousedy.



Obrázek 2: Lineárně neoddělitelné shluky nalezené algoritmem DBScan

3 Popis implementace

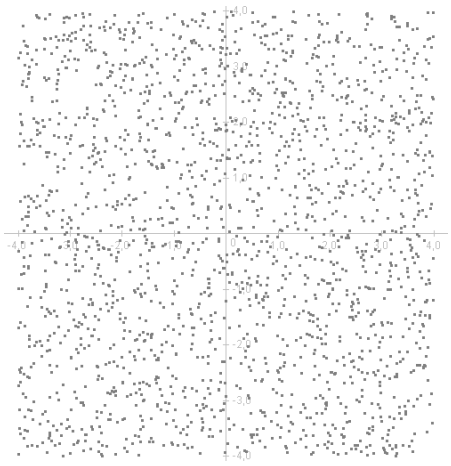
Aplikace byla naprogramována v jazyce Java ve verzi 8. Pro vytvoření grafického rozhraní byl použit designer obsažený v IDE IntelliJ IDEA 2016[1]. Při běhu jsou vypisovány na standardní výstup informace z průběhu shlukování.

3.1 Implementovaná funkcionalita

Aplikaci lze rozdělit do dvou oddělených částí – získání dat a zpracování dat.

Pro získání dat byly implementovány následující 3 možnosti:

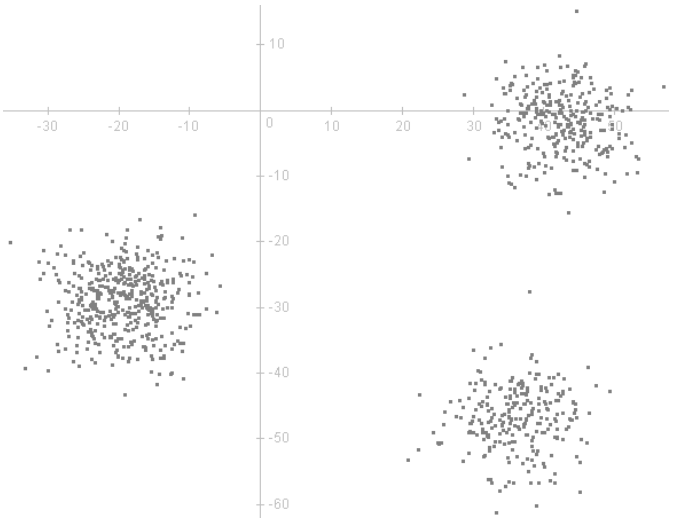
1. Náhodný generátor uniformních dat (rovnoměrně umístěných po prostoru). Výstup představuje nejhorší možný scénář pro provedení shlukování, neboť se v datech nevyskytuje žádný přirozený shluk. Za shluk lze považovat pouze data jako celek.



Obrázek 3: Náhodná uniformní data o velikosti 2000 bodů.

[1] Vývojové prostředí, viz `https://www.jetbrains.com/idea/`

2. Náhodný generátor shlukovaných dat. Výstup představuje (v optimálním případě) nejlepší možný scénář pro shlukování – v datech se vyskytují jednoznačně oddělitelné shluky. Ukázka:



Obrázek 4: Náhodná data o velikosti 1000 bodů rozdělených do 3 shluků.

3. Načtení dat z externího souboru.

Pro zpracování dat, v případě této aplikace tedy provedení shlukování, byly implementovány algoritmy K-Means, DBScan a K-Medians.

Bližší popis jednotlivých částí a jejich možností lze najít v kapitole 4.3.

3.2 Rozdělení kódu

Kód aplikace byl rozdělen dle vzoru MVC (model-view-controller), grafické rozhraní je tedy odděleno od aplikační/datové logiky. Jednotlivé části aplikace jsou odděleny do samostatných tříd s využitím technik OOP. Zde uvádíme pouze krátkých popis jednotlivých balíčků/tříd, konkrétnější informace jsou součástí kódové dokumentace (součást zdrojových kódu v javadoc formátu).

- Třída `App` – obsahuje vstupní bod aplikace (`main`); spouští aplikaci se standardním grafickým rozhraním.
- Třída `AppDev` – spouští aplikaci v režimu vhodném pro rychlé ladění/porovnání různých shlukovacích algoritmů nad shodnými daty. Ve výchozím stavu se pokouší načíst data ze souboru data.txt, v případě neúspěchu vygeneruje nová shlukovaná data a zpětně je uloží do zmíněného souboru.
- Třída `utils.ClusteringCanvas` – nástroj pro vygenerování grafu s daty. Používá se jako panel v rámci grafického rozhraní a též pro tvorbu grafu při exportu do externího souboru.
- Třída `utils.CsvDataProcessor` – obsahuje metody pro načtení/uložení dat z/do CSV souboru.
- Třída `utils.DataGenerator` – obsahuje metody pro generování náhodných dat.
- Rozhraní `utils.GUIController` – definuje rozhraní, pomocí kterého grafické rozhraní předává pokyny aplikační části.
- Třída `utils.GUIForm` – obsahuje samotné grafické rozhraní.

- Třída `structures.Point` – představuje instanci jednoho bodu libovolné dimenze. Obsahuje pomocné vlastnosti pro algoritmy a metody (například pro výpočet vzdálenosti od jiného bodu).
- Třída `structures.Cluster` – představuje shluk bodů (tedy instancí třídy Point). Je odděděno od třídy `HashSet`. Obsahuje metody pro operace nad shlukem (např. zjištění geometrického středu).
- Rozhraní `structures.ClusteringAlg` – definuje jednoduché rozhraní společné pro všechny shlukovací algoritmy.
- Rozhraní `structures.ClusteringAlgConf` – značkovací rozhraní tříd obsahujících konfigurace pro jednotlivé shlukovací algoritmy.

- Třídy `kmeans.KMeans`, `kmeans.KMeansConf` – třída provádějící shlukování algoritmem K-Means a třída s parametry pro provedení shlukování (tj. konfigurací).
- Třída `kmeans.KMedians` – oddděděna od třídy `KMeans`, provádí shlukování algoritmem K-Medians.
- Třídy `dbscan.DBScan`, `dbscan.DBScanConf` – třída provádějící shlukování algoritmem DB-Scan a třída s parametry pro provedení shlukování (tj. konfigurací).

4 Uživatelská příručka

4.1 Překlad aplikace

Překlad aplikace je možný nástrojem Ant, pomocí souboru build.xml v kořenovém adresáři. K překladu jsou nutné knihovny IDE IntelliJ IDEA (pro překlad GUI). Alternativně lze aplikaci přeložit po načtení projektu do zmíněného IDE.

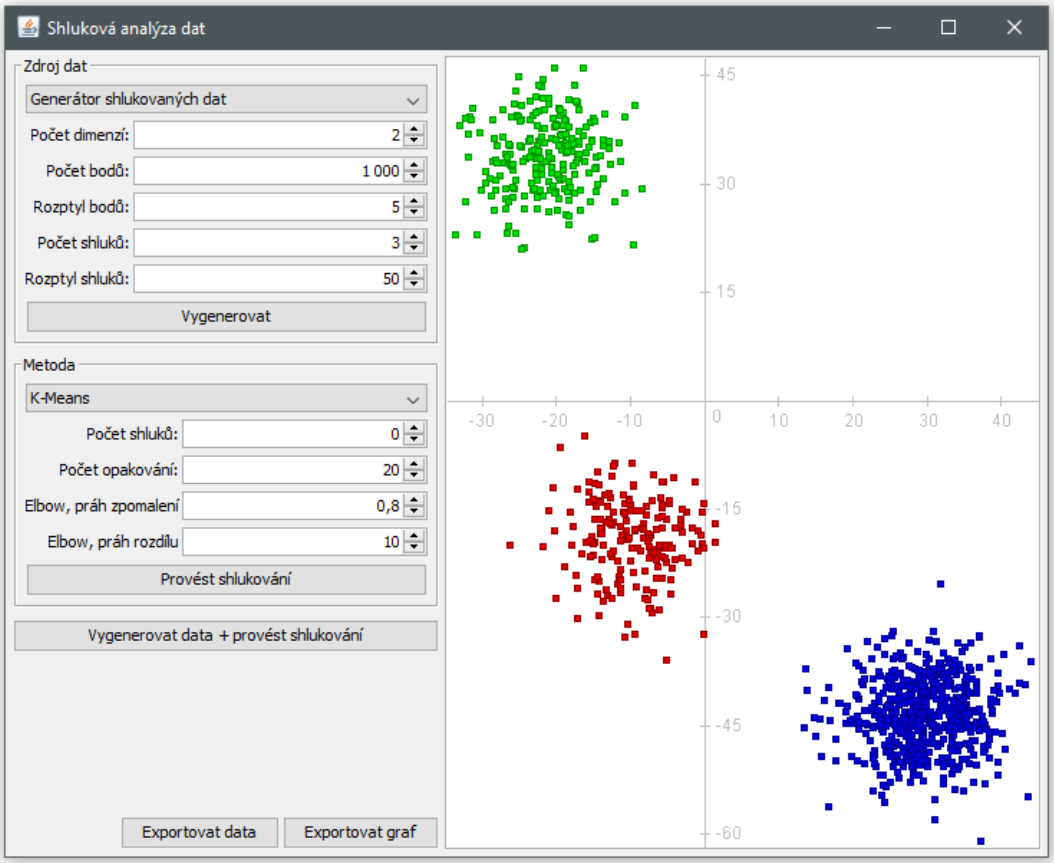
4.2 Spuštění aplikace

Aplikaci lze spustit příkazem:

```
> java -jar KMeans.jar
```

4.3 Práce s aplikací

Po spuštění se zobrazí grafické rozhraní, které se skládá z kontrolního panelu v levé části a oblasti pro vykreslení grafu v pravé části.



Obrázek 5: Grafické rozhraní aplikace

Kontrolní panel obsahuje dva základní bloky pro ovládání aplikace.

Zdroj dat

Zde je možné vybrat metodu, která se použije pro generování náhodný dat, případně zvolit externí CSV soubor, ze kterého budou data načtena. V panelu je možné dle aktuálně zvolené možnosti nastavit různé parametry. Po stisknutí tlačítka jsou data vygenerována/načtena a okamžitě zobrazena v grafu.

Metoda

Výběr jednoho ze tří implementovaných algoritmů, který bude použit pro provedení shlukování. Obdobně i zde je možné pro každou metodu nastavit různé parametry. Stisknutím tlačítka je poté shlukování zahájeno a výsledek zobrazen v grafu.

- Metoda K-Means – výchozí algoritmus. Je možné nastavit počet shluků a počet opakování jednotlivých iterací algoritmu – pro výstup je vybrána iterace s nejlepším (tj. nejmenším) ohodnocením. V případě, že je počet shluků nastaven na nulu, je pro nalezení optimálního počtu shluků použita metoda lokte. Současně s tím se zpřístupní dva parametry pro nastavení této metody. Jejich popis se zobrazí po najetí myší nad příslušná pole.
- Metoda DBScan. Jako parametry lze nastavit minimální počet bodů pro „akceptaci“ shluku a maximální vzdálenost (ε) pro přidání sousedních bodů do shluku.
- Metoda K-Medians. Jedná se pouze o modifikaci metody K-Means, používá shodné nastavení.

Kontrolní panel dále obsahuje tlačítko, které provede obě zmíněné činnosti během jednoho kroku – tedy vygenerování/načtení nových dat a následně shlukování, obojí dle aktuálně nastavených parametrů. Ve spodní části panelu se dále nachází dvě tlačítka sloužící pro export – první z nich exportuje data do CSV souboru, druhý exportuje graf jako obrázek do souboru ve formátu PNG.

Import/export dat

Aplikace umožňuje načíst data z externího souboru a rovněž data po shlukování exportovat. Pro tyto potřeby se používá jednoduchý tabulkový formát CSV, ve kterém je každý bod uložen na novém řádku, přičemž v jednotlivých buňkách (sloupcích) jsou souřadnice bodu (tedy každá „osa“ v novém sloupci). Jakmile se při načítání souřadnic narazí na nečíselnou hodnotu (tedy i prázdnou buňku), je zbytek řádku přeskočen. Očekává se, že všechny body budou mít stejný počet souřadnic; ten je zjištěn z prvního úspěšně načteného bodu. Soubor může mít například tuto podobu:

```
1;1;1;;První bod
2;2;2;;Druhý bod
−2.164;3.14;21.66666;;Třetí bod
```

Při exportování dat je za souřadnice bodů vložen prázdný sloupec a poté sloupec s označením shluku:

```
1;1;1;;Cluster 1
2;2;2;;Cluster 1
−2.164;3.14;21.66666;;Cluster 2
```

5 Závěr

Zadání bylo splněno v celém rozsahu. Vytvořená aplikace úspěšně demonstruje fungování vybraných shlukovacích algoritmů a umožňuje uživateli snadno vyzkoušet různé situace. Kód byl strukturován tak, aby bylo možné oddělit algoritmy od grafického rozhraní a použít samostatně. Aplikaci je možné snadno rozšířit o další funkcionalitu.