Fakulta informačních technologií Vysokého učení technického v Brně

Předmět: Soft Computing (SFC)

Autor: František Horázný

Login: xhoraz02

Ročník: 2-MIT

e-mail: xhoraz02@stud.fit.vutbr.cz

Varianta projektu - 36 Demonstrace činnosti algoritmu Fuzzy K-means

Algoritmus k-means

Tento algoritmus je popsán ve třetí přednášce na 5 straně. (https://www.fit.vutbr.cz/study/courses/SFC/private/20sfc_3.pdf).

Jedná se o algoritmus, který umí rozdělit data do K shluků. Vstupem do algoritmu jsou *N* rozměrná data, na kterých umíme změřit vzdálenost bodů, a počet shluků *K*, které chceme najít. Každý shluk je definován pouze svým těžištěm, proto se také nazývá *K-means*. Algoritmus má tyto kroky (zjednodušeně):

a. Inicializuj náhodně *K* středů shluků.
b. Pro každý datový záznam najdi nejbližší střed shluku a zapamatuj si ho.
c. Přepočti každý střed shluku podle k němu přiřazených bodů.
d. Pokud se žádný střed nezměnil nebo se dosáhlo maximální počtu kroků skonči, jinak pokračuj krokem *b*.

K-means však **nenajde** vždy stejné (a nemusí ani správné/požadované) shluky. Velmi závisí na počátečním nastavení středů.

Algoritmus fuzzy k-means

Tento algoritmus je popsán v deváté přednášce na 25. straně. (https://www.fit.vutbr.cz/study/courses/SFC/private/20sfc_9.pdf).

Fuzzy k-means, jak naznačuje název, je velmi podobný *k-means*. Vstup i výstup je stejný, ale navíc nepodléhá nalezení lokálního optima. Rozdíl je v tom, že každý bod při výpočtu středů nenáleží pouze jednomu shluku, ale všem zároveň určitou vahou, kde součet všech vah tohoto bodu se rovná 1. Při výpočtu středů shluků se tedy musí počítat se všemi body. Váha samotného bodu se pak počítá jako:

$$m_{ij}^{q} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|\vec{x}_i - \vec{c}_j\|}{\|\vec{x}_i - \vec{c}_k\|} \right)^{\frac{2}{q-1}}}$$

Kde *i* je index bodu, *j* index středu, *C* je počet středů.

Ze vzorce je vidět, že pokud vzdálenost počítaného středu bude větší než nějakého jiného, pak bude jeho váha menší než toho druhého. Pokud navíc snížíme hodnotu *q*. zvyšujeme tento rozdíl mezi váhami středů. Pokud bychom *q* nastavili na 1, pak by váha nejbližšího středu byla 1 a pro ostatní středy 0. Toto lze vypočítat analyticky, budeme mocnit nekonečnem a ve jmenovateli mít 0. Jak je známo dělení 0 v počítačích nelze, a proto je možné v mém programu vyzkoušet tento fakt pouze zadáním *q=1.00001*.

Nakonec bych chtěl vyjádřit nesouhlas s větou z přednášek ohledně *q*:

$$q \qquad \text{koeficient } 1 \leq q < \infty \quad (obvykle\ q = 2)$$

Tedy pokud *q=1*, potom se nejedná o fuzzy k-means a dále při použití vzorců z přednášky fuzzy k-means nefunguje obecně pro *q>2*. Oproti tomu při použití jiného vzorce (například z wikipedie) *fuzzy k-means* se stává pouze *k-means* již při *q=2*. O této problematice bych rád promluvil na obhajobě a ideálně i ukázal v praxi.

Popis implementace

Implementoval jsem program ve skriptovacím jazyce Python verze 3. Tento jazyk jsem vybral prioritně z důvodu procvičení počítání pomocí numpy polí. Dále jsou využity moduly tkinter pro grafické rozhraní, collections pro slovník s výchozí hodnotou (defaultdict) a random.

Program je rozdělen do dvou tříd (proti zvyklostem v jednom souboru z důvodu počtu řádků a subjektivní preference).

Třída *Data* obsahuje inicializaci dat a metody na přidání bodu do databáze, výpis databáze, vymazání celé databáze, náhodné obarvení bodů v databázi, resetování výpočtu do stavu před prvním krokem algoritmu, změny počtu shluků, krok algoritmu, deseti kroků algoritmu. Kromě samotného krokování algoritmu jsou metody dostatečně komentované v kódu a myslím, že není potřeba je blíže popisovat zde.

Metoda *step()* ve třídě *Data* má 3 různé fáze. První a třetí fáze počítají těžiště shluků a druhá fáze počítá náležitosti bodů do shluků.

e. **První fáze** je rozdílná od třetí tím, že počítá středy z barvy bodů, jak jim byla na záčátku přidělena (tlačítkem *randomize* nebo samotným nanesením kliknutím na okno). Tedy nepočítá s váhami, protože nemusí.
f. **Druhá fáze** vytvoří slovník, který jako klíče obsahuje těžiště (označené barvou) a hodnoty jsou listy obsahující váhu bodu. Po inicializaci tohoto slovníku následují 2

cykly, které pro každý bod a střed zvlášť vypočítá podle vzorce uvedeném výše váhy. Pro každý bod je také zaznamenávána nejvyšší hodnota. Nakonec se do třídy s grafickým rozhraním pošle informace o nových barvách bodů a jejich vykreslení.

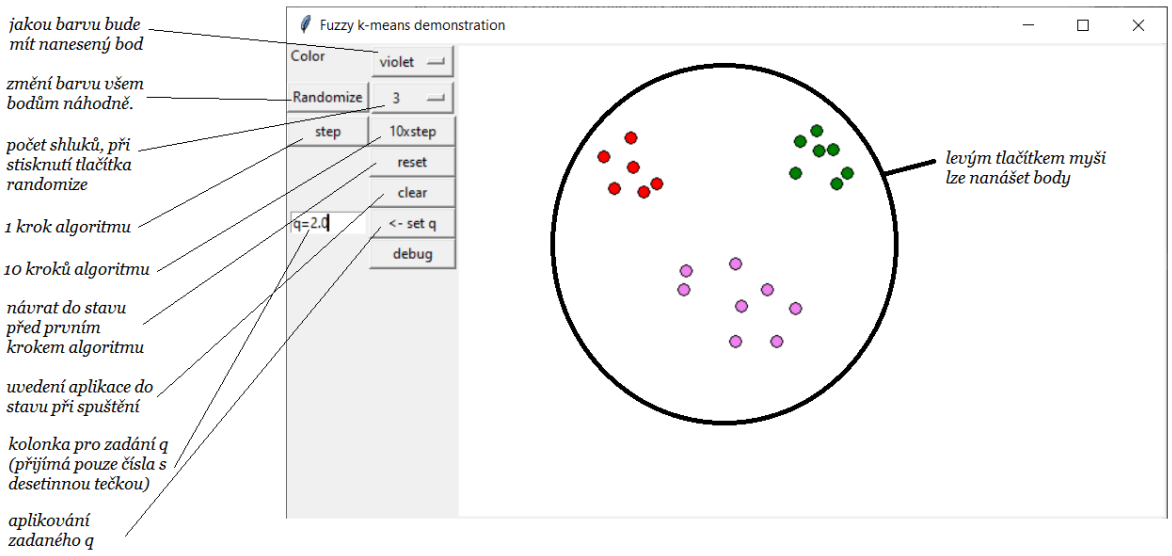
g. **Třetí fáze** pro každý střed vypočte jeho novou polohu vzorcem z přednášek a následně pošle informace o nových těžištích třídě s grafickým rozhraním.

Třída *Gui* inicializuje všechna tlačítka a *canvas* a naváže své metody na různé události, například kreslení do *canvas* pomocí myši. Při stisku některého tlačítka *Gui* mimo vizuální změny dává signály třídě Data o potřebě změnit i data. Tuto třídu dle mého názoru není nutné zde více popisovat, protože kód je dostatečně komentovaný.

Spuštění a ovládání programu

Pro spuštění je potřeba mít nainstalovaný python a moduly *tkinter* a *numpy*. Spuštění je testování na *Python* verzi 3.10.0, *numpy* verze 1.21.4 a *tk* verze 0.1.0.

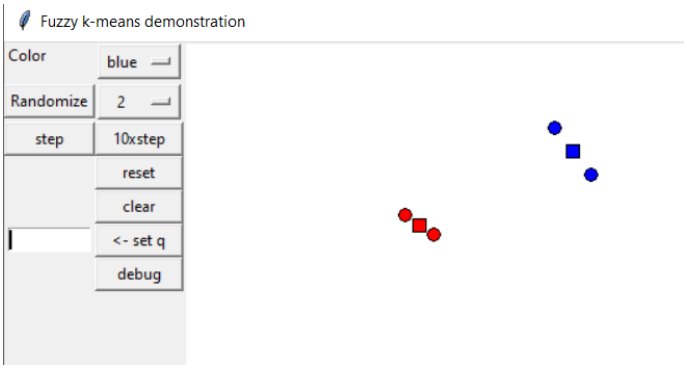
Samotné spuštění je bez parametrů: python fuzzykmeans.py



Kolonka pro zadání *q* musí obsahovat pouze číslo a musí být větší než 1 (například 1.001). *q* se aplikuje až při stisku tlačítka *„<- set q“*.

Ukázka práce s programem

- Klikněte do bílého prostoru 2x myší na různá místa.
- V menu barev si vyberte jinou barvu (například *„blue“*).
- Klikněte do bílého prostoru 2x myší na různá místa.
- Stiskněte tlačítko *„step“*
  Pravděpodobný stav, který vidíte:



- Stiskněte tlačítko reset
- Stiskněte tlačítko „*Randomize*“ (pozor na počet zobrazených barev, může jich být méně)
- Stiskněte desetkrát tlačítko „*step*“ nebo jednou tlačítko „*10xstep*“
  Měli byste vidět téměř to samé jako na obrázku výše.

Závěr

Aplikace funguje bez větších problémů. Díky robustnosti *pythonu* se mi nepodařilo nijak program shodit. Při návrhu aplikace jsem předpokládal využívání inteligentním a kompetentním tvorem. Rád jsem si vyzkoušel vytváření *GUI* v *pythonu* a jsem si vědom, že využití *numpy* polí mohlo být optimálnější. Z důvodu charakteru aplikace jsem ale nepředpokládal složitější výpočty, a proto využití funkce *append* do listu ve slovníku nepovažuji za větší zhoršení výkonu. Při obhajobě bych rád probral podmínky a vzorci pro q.

Dodatky

Již po vytvoření technické zprávy byla přidána funkčnost koláčových grafů pro ještě lepší demonstraci algoritmu. Tlačítko „max design“ (vzhled bodů podle jejich maximální váhy) při kliknutí změní vzhled bodů z jejich výsledné náležitosti na „pie design“ (vzhled bodů jako koláčový graf podle všech jeho vah). Je pak lépe viditelné, co se vlastně v algoritmu děje. Toto tlačítko a funkcionalita nejsou zahrnuty jinde v tomto dokumentu mimo tento odstavec.

Na začátku kódu byla přidána pomocná proměnná, která mění velikost nakreslených bodů (vizte kód a komentář).