Martin Kvietok
ID: 127214

Zadanie 2c – klastrovanie

ZS 2024/2025

Contents

Zadanie 3
Opis riešenia 3
Aglomeratívny algoritmus 3
Centroid 4
Medoid 4
Postup tvorby klastrov 5
1. Inicializácia klastrov 5
2. Tvorba matice vzdialeností 5
3. Zhlukovanie klastrov 5
Finálne testovanie 6
Akceptovateľná euklidovská vzdialenosť 6
Výpočet pomocou centroidov 7
Výpočet pomocou medoidov 8
Tabuľka časovej zložitosti 8
User manual 9
Záver 9

Martin Kvietok
ID: 127214

Zadanie

Cieľom tejto úlohy je implementovať algoritmus na klastrovanie bodov v dvojrozmernom priestore s rozsahom súradníc od -5000 do +5000 na oboch osiach X a Y. Na úvod sa v tomto priestore náhodne vygeneruje 20 bodov s jedinečnými súradnicami, ktoré budú slúžiť ako referenčné body pre následnú tvorbu ďalších bodov. Na základe týchto referenčných bodov sa potom vygeneruje ďalších 20 000 bodov, pričom každý z nich bude odvodený od jedného z počiatočných bodov a posunutý o náhodný offset v intervale od -100 do +100 na oboch súradniciach. Týmto spôsobom sa vytvorí priestor so zhlukmi bodov s prirodzenou štruktúrou.

Úlohou je naprogramovať klastrovač, ktorý dokáže analyzovať takto vzniknutý priestor a rozdeliť body do klastrov použitím aglomeratívneho klastrovania. Algoritmus má byť implementovaný v dvoch verziách: prvá verzia použije ako stred klastru centroid, zatiaľ čo druhá bude pracovať s medoidom. Na záver sa vytvorí grafická vizualizácia klastrov, kde budú jednotlivé zhluky bodov farebne odlíšené a ich centrá zvýraznené, čím sa dosiahne prehľadná prezentácia výsledkov klastrovania.

Opis riešenia

Aglomeratívny algoritmus

Na riešenie tohto zadania som implementoval aglomeratívny klastrovací algoritmus, ktorý patrí medzi hierarchické metódy klastrovania. Tento algoritmus funguje na princípe iteratívneho zlučovania jednotlivých objektov alebo existujúcich klastrov na základe ich vzájomnej podobnosti, pričom výsledkom je jeden celkový klaster zahŕňajúci všetky objekty. Algoritmus postupuje zdola nahor (tzv. bottom-up prístup), čo znamená, že začína s každým objektom ako samostatným klastrom a postupne spája klastre do väčších celkov podľa definovaného kritéria podobnosti.

V rámci môjho riešenia som použil ako kritérium zhlukovania minimálnu euklidovskú vzdialenosť medzi stredmi jednotlivých klastrov. V každom kroku som identifikoval dvojicu klastrov s najmenšou vzájomnou vzdialenosťou medzi ich stredmi a následne ich zlúčil do jedného väčšieho klastru, pričom som aktualizoval príslušné hodnoty v matici vzdialeností. Tento proces sa opakoval, až kým nevznikol jeden celkový klaster zahŕňajúci všetky objekty alebo kým vzdialenosť medzi stredmi všetkých klastrov neprekročila hodnotu 500, čím som zabezpečil, že vzdialenejšie klastre zostali oddelené.

Centroid

Centroid je vektorovým priemerom všetkých objektov (bodov) nachádzajúcich sa v danom klastri, čím reprezentuje priemernú polohu týchto bodov v priestore. Ide o abstraktný bod, ktorý nemusí korešpondovať so žiadnym konkrétnym bodom v datasete, no v tomto prípade slúži ako referenčný stred klastru. Tento prístup je výpočtovo efektívny a vhodný na reprezentáciu klastrov, avšak je citlivý na extrémne hodnoty (outliery), ktoré môžu skresliť priemer a teda aj samotnú pozíciu centroidu, čím môže dôjsť k narušeniu presnosti klastrovania v prípade výskytu výrazných odchýlok v rámci klastru.

$$\mathbf{c} = \left(\frac{1}{n} \sum_{i=1}^{n} x_{i1}, \frac{1}{n} \sum_{i=1}^{n} x_{i2}, \ldots, \frac{1}{n} \sum_{i=1}^{n} x_{id} \right)$$

Medoid

Medoid je skutočný objekt z datasetu, ktorý najpresnejšie reprezentuje príslušný klaster. Je definovaný ako bod, ktorý minimalizuje priemernú vzdialenosť k ostatným bodom v klastru, čím dosahuje najnižšiu kumulatívnu vzdialenosť k členom klastru. Tento prístup zvyšuje robustnosť reprezentácie klastrov, pretože medoid je menej ovplyvnený odľahlými hodnotami (outliermi) v porovnaní s centroidom, ktorý môže byť skreslený extrémami.

$$\mathbf{m} = \arg\min_{\mathbf{x}_i \in C} \sum_{\mathbf{x}_j \in C} d(\mathbf{x}_i, \mathbf{x}_j)$$

V rámci môjho riešenia som použil hodnoty centroidu na aproximáciu polohy medoidu. Na základe týchto hodnôt som identifikoval bod v datasete, ktorý bol k centroidu najbližšie, a tento bod som následne zvolil ako medoid daného klastru.

Postup tvorby klastrov

1. Inicializácia klastrov

Na uchovávanie potrebných dát som navrhol objektovo orientovaný prístup pomocou tried **Cluster**, **Point**, a **MatrixOfDistances** (matica vzdialeností).

- Trieda **Cluster** agreguje triedu **Point** prostredníctvom poľa Points, kde sa ukladajú všetky body v rámci klastra.
- Trieda **MatrixOfDistances** agreguje triedu **Cluster** a obsahuje všetky inicializované klastre.

Na začiatku som pridal do matice 20 inicializačných klastrov pomocou metódy *add_initial_clusters(self, number_of_clusters)* a následne som generoval ďalšie body pomocou *add_clusters(self, number_of_clusters)*. Každý bod sa pridáva do príslušného klastra prostredníctvom metódy *add_point(self, point, user_method)*, kde parameter user_method označuje, ktorá metóda klastrovania sa použije (0 pre centroid, 1 pre medoid).

Pre každý klaster som vypočítal centroid alebo medoid a priemernú vzdialenosť bodov od stredu klastru, ktorá nesmie presiahnuť hodnotu 500.

2. Tvorba matice vzdialeností

Matica vzdialeností sa vytvára prostredníctvom metódy *create_distance_matrix(self).* Táto matica vzniká na základe hodnôt uložených v poľach centroidov alebo medoidov klastrov.

Na výpočet vzdialeností medzi každým párom klastrov som použil euklidovskú vzdialenosť vypočítanú pomocou Pytagorovej vety, pričom som vytvoril maticu n×n. Pre efektívnosť som pracoval len s hodnotami v hornej trojuholníkovej matici, pričom ostatné hodnoty boli nastavené na nekonečno (inf).

3. Zhlukovanie klastrov

Zhlukovanie prebieha prostredníctvom metódy *merge_closest_clusters(self).* Tu som vytvoril usporiadaný slovník s najmenšími vzdialenosťami (limitovanými hodnotou 800) vo formáte (cluster1, cluster2, eculiidian_dist). Indexy klastrov sa následne posúvajú do funkcie *update_matrix(self, cluster1_index, cluster2_index)*, kde som využil dve polia:

- **clusters** – hlavné pole klastrov, ktoré je možné upravovať.
- **clusters_backup** – hlboká kópia poľa **clusters**, slúžiaca ako maska s nastaveným príznakom cluster_to_merge.

Pred každým spájaním klastrov som overoval, či oba klastre ešte neboli predtým spojené s inými, čo je kľúčové pre zachovanie podmienky aglomeratívneho klastrovania, kde sa v každom kroku môžu spojiť len dva jedinečné klastre (napr. klaster 1 s klastrom 3, ale už nie klaster 1 s klastrom 2).

V prípade, že boli klastre vhodné na spájanie, všetky body z **klastru B** som pridal do **klastru A** a vypočítal nový centroid/medoid a priemer bodov. Ak priemerná vzdialenosť presiahla 500, musel som sa vrátiť späť a odstrániť pridané body a skončiť proces. Následne som odstránil klaster B z hlavného poľa klastrov

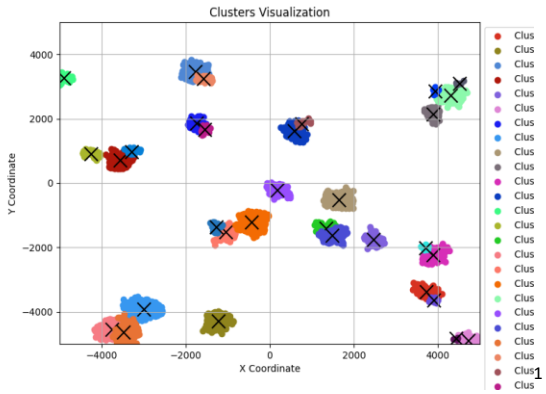
podľa jeho ID a v poli **clusters_backup** som nastavil príznak *cluster_to_merge* na True, aby som znemožnil ďalšie spájanie tohto klastru.

Po prechode všetkých minimálnych vzdialeností som vytvoril aktualizovanú maticu vzdialeností a opakoval celý proces, až kým neboli splnené kritériá pre všetky klastre.

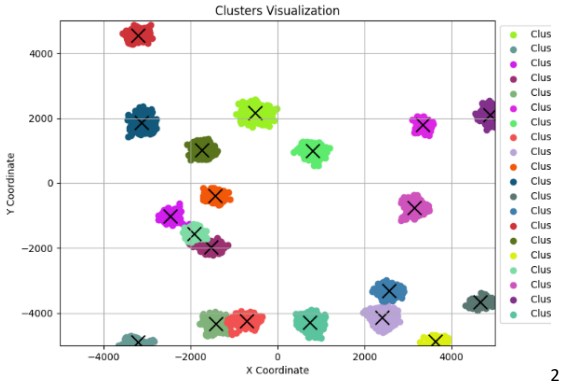
Finálne testovanie

Akceptovateľná euklidovská vzdialenosť

Maximálna akceptovateľná euklidovská vzdialenosť pri zhlukovaní zohrala hlavnú úlohu pri dosahovaní optimálnych výsledkov. Ako najlepšia hodnota sa osvedčila vzdialenosť 800, pri ktorej sa podarilo výrazne zredukovať počet klastrov a zároveň zlúčiť väčšie množstvo nezávislých skupín do jedného globálneho klastru. Na grafoch vidíme vývoj klastrov pri použití nižších hodnôt, konkrétne 250 a 500, kde nižšie hodnoty spôsobujú viac menších zhlukov, zatiaľ čo vyššia hodnota poskytuje kompaktnejšie zhluky s celistvejšou štruktúrou dát.



[1]



[2]

[1] maximálna vzdialenosť 250
[2] maximálna vzdialenosť 500

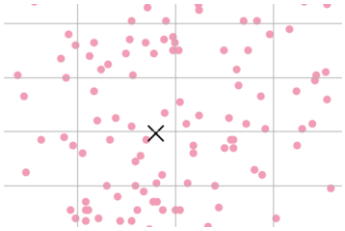
Martin Kvietok
ID: 127214

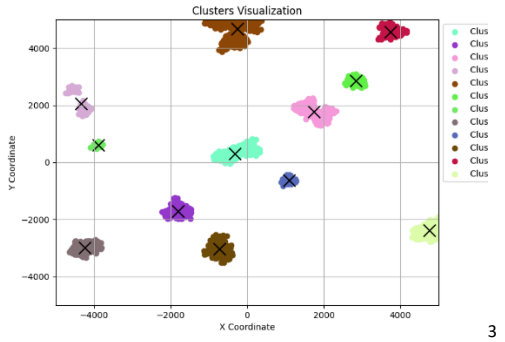
Výpočet pomocou centroidov

Hlavným problémom bola výpočtová zložitosť algoritmu, kvôli ktorej trvali výpočty pre 20 000 bodov niekoľko hodín. Najviac času zabralo generovanie a práca s maticou vzdialeností – pri matici 20 000 x 20 000 je až 199 990 000 unikátnych dvojíc (kombinácií), čo pri výpočtovej zložitosti n^3 znamená dosť veľkú záťaž. Každý cyklus si vyžadoval generovanie nových vzdialeností a vytvorenie novej matice.

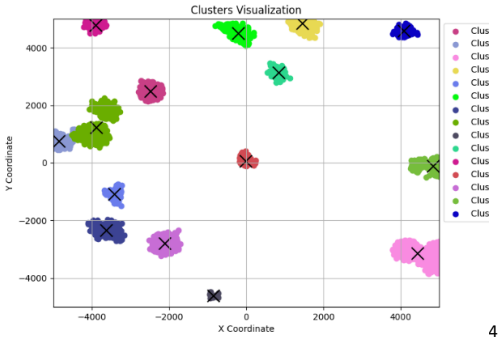
Riešenie som našiel zavedením maximálnej akceptovateľnej vzdialenosti, ktorá umožnila zlúčiť (mergovať) dvojice naraz, namiesto toho, aby sa matica aktualizovala pri každom novom kroku. Tento nápad bol načrtnutý aj v prednáške a významne znížil časovú zložitosť a skrátil výpočtový čas na desiatky sekúnd.

Vizualizácia centroidu v klastri

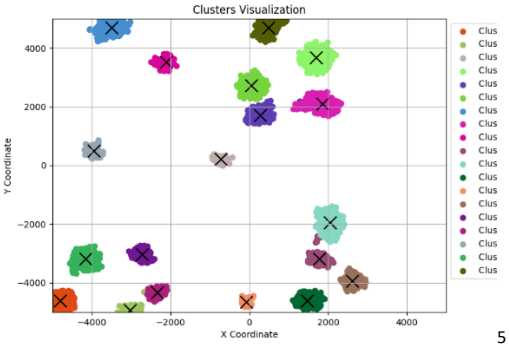




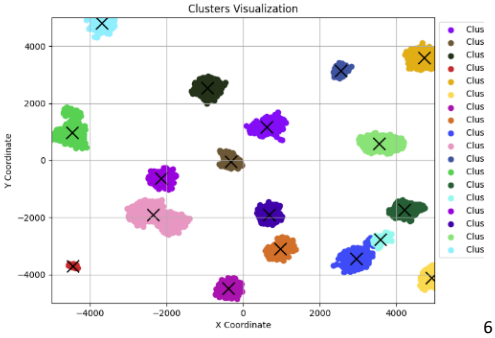
3



4



5



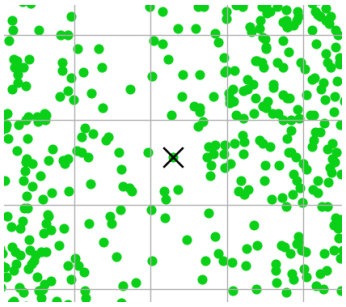
6

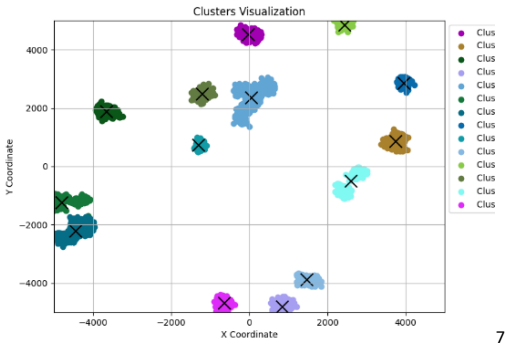
[3] 5000 bodov
[4] 10 000 bodov
[5] 15 000 bodov
[6] 20 000 bodov

Výpočet pomocou medoidov

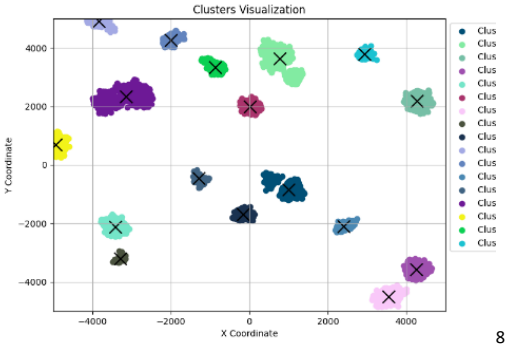
Výpočet pomocou medoidov sa z hľadiska dĺžky a zložitosti významne nelíšil od použitia centroidov. Hlavný rozdiel spočíval v tom, že zhlukované klastre vytvárali kompaktnejšie a homogénnejšie celky, pričom body na hraniciach klastrov (tzv. outliery) sa pri tejto metóde ľahšie priraďovali k najbližšiemu klastru. Výsledky však boli do značnej miery naďalej závislé od počiatočného výberu 20 bodov, ktoré výrazne ovplyvňovali celý priebeh a výsledky klastrovania.

Vizualizácia medoidu v klastri





7



8

Tabuľka časovej zložitosti

Počet bodov	Čas výpočtu
5000	do 10s
10000	do 30s
15000	do 60s
20000	2-3min

[7] 5000 bodov
[8] 10 000 bodov

User manual

Program bol odovzdaný ako riešenie v prostredí Visual Studio 2022 a obsahuje Python environment s pridanými knižnicami, konkrétne: random, numpy (efektívna práca s množstvom dát), scipy (výpočet euklidovskej vzdialenosti), copy (hlboké kopírovanie objektov) a matplotlib na vykresľovanie grafov. Spúšťacím súborom programu je Zadanie2c_klastrovanie.py, preto je potrebné overiť, či je tento súbor nastavený ako „Startup File“. Po kompilácii program vyžaduje zadanie vybranej metódy klastrovania a počtu generovaných bodov priamo do konzoly. Po dokončení výpočtov program vypíše čas potrebný na vykonanie operácií a zobrazí grafické znázornenie všetkých klastrov, kde sú klastre farebne odlíšené a ich stredy zvýraznené.

Záver

Navrhnutý klastrovací algoritmus efektívne rieši zadaný problém a dosahuje vysoký výkon, pričom jeho úspech závisí od vhodnej kombinácie vstupných parametrov. Optimalizácia výpočtovej rýchlosti výrazne zlepšila časovú náročnosť algoritmu a umožnila efektívne spracovanie veľkých množín dátových bodov. Hlavnou výzvou zostáva správne nastavenie maximálnej akceptovateľnej vzdialenosti, ktorá zabezpečí, aby klastre boli kompaktné a zároveň splnili požadovaný priemerný počet bodov. S vhodným prahom vzdialenosti dokáže algoritmus poskytnúť presné a stabilné výsledky, čím napĺňa stanovené ciele klastrovania.