Implementacja algorytmu Clarans

Jakub Ficek

7 czerwca 2022

1 Opis zadania

Celem zadania jest implementacja i zbadanie algorytmu Clarans [1].

Przyjęte założenia:

- Zaimplementowany zostanie algorytm Clarans
- Algorytm Clarans zostanie zbadany pod względem czasu wykonania dla danych o różnej liczbie punktów i wymiarów
- Porównany zostanie czas wykonania algorytmu Clarans z czasem wykonania algorytmu PAM
- Wykorzystana zostanie metryka euklidesowa do wyznaczania dystansów oraz wskaźnik sylwetkowy do oceny jakości grupowania

2 Opis implementacji

Projekt został zaimplementowany w języku Python. Do porównania algorytmów, użyta została implementacja algorytmu PAM z pakietu *pyclustering*. Dodatkowo, do wyznaczenia wskaźnika sylwetkowego i wizualizacji danych, zostały wykorzystane funkcje pomocnicze z pakietu *pyclustering*.

W katalogu data, znajdują się dane wykorzystane do eksperymentów. W pliku data.py, znajdują się metody, które pozwalają generować, wczytywać i wizualizować dane. W pliku clarans.py znajduje się implementacja algorytmu Clarans. W plikach measure_clarans.py i measure_pam.py znajdują znajdują się metody do zmierzenia czasu i wykonania algorytmów i jakości grupowania. W pliku requirements.txt, znajdują się pakiety koniecznie do uruchomienia projektu.

3 Opis formatu danych wejściowych i wyjściowych

Pliki z danymi zawierają punkty oddzielone znakiem nowej linii. Pojedyńcza linia zawiera kolejne współrzędne danego punktu, oddzielone przecinkami oraz numer klastra, do którego punkt należy.

Klasa Clarans przyjmuje w konstruktorze punkty w postaci listy list oraz parametry: liczba klastrów, numlocal i maxneighbor. Następnie, po wywołaniu metody cluster(), zwracana jest lista indeksów wyznaczonych medoid oraz klastry jako lista list, gdzie i-ta lista zawiera indeksy punktów przydzielonych do i-tego klastra.

4 Eksperymenty

Głównym celem eksperymentów było sprawdzenie skalowalności algorytmu Clarans.

Wykorzystane zostały, wygenerowane przeze mnie, dane w postaci grup o rozkładzie normalnym o jednostkowej macierzy kowariancji oraz sztuczne dane pochodzące z repozytorium *clustering-benchmark*. Parametry zbiorów przedstawione są w tabeli (1)

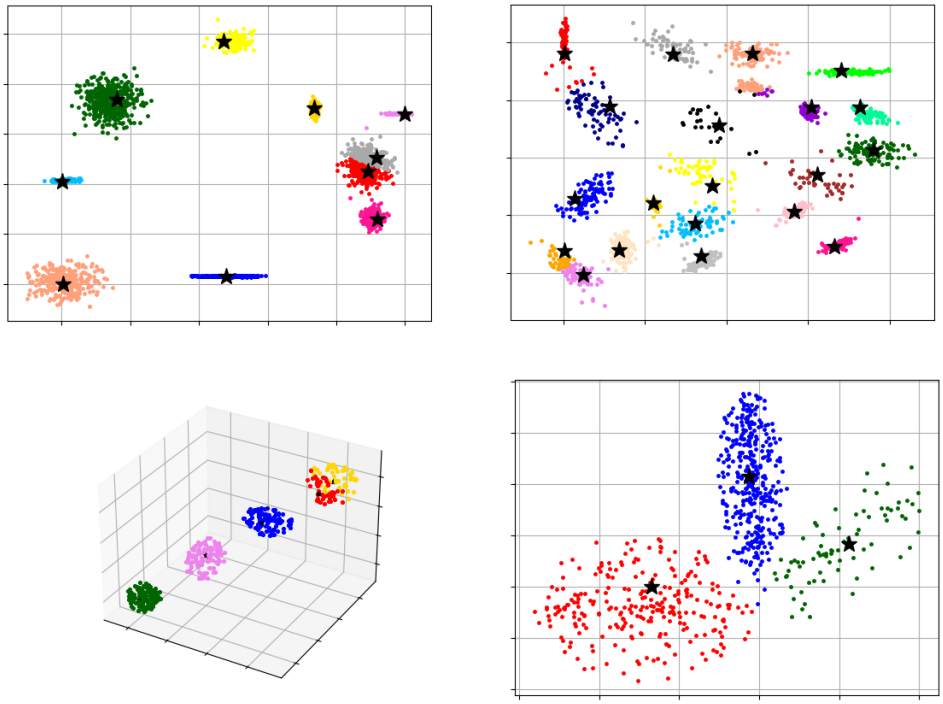
Czas wykonania or wskaźnik sylwetkowy był wyznaczany na podstawie wyników z 3 uruchomień programu. Parametr *numlocal* był ustawiany na 2, a *maxneighbor* na 100 dla wszystkich danych.

Dla niektórych danych, nie zostały wyznaczone parametry dla algorytmu PAM. Oznacza to, że czas wykonania algorytmu był zbyt długi i eksperyment nie został ukończony.

Tablica 1: Parametry wykorzystanych zbiorów danych.

Nazwa	**Liczba punktów**	**Liczba wymiarów**	**Liczba grup**
2d-20c-no0	1517	2	20
dpb	4000	2	6
disk-6000n	6000	2	2
cluto-t7-10k	10000	2	9
normal1	30000	10	3
normal2	30000	50	3
normal3	30000	100	3
normal4	60000	10	3
normal5	90000	10	3

5 Wyniki



Rysunek 1: Wizualizacja przykładowych wyników grupowania przy użyciu algorytmu Clarans.

Rysunek (1) przedstawia przykładowe wizualizacje grupowania dla małych danych. Wizualizacje pozwalają ocenić, że zaimplementowany algorytm działa właściwie.

Na podstawie wyników z tabeli (2), możemy zauważyć, że wskaźniki sylwetkowe, uzyskiwane przez oba algoryty, są bardzo podobne. Dla części małych zbiorów czasy wykonania są krótsze dla algorytmu PAM. Oznacza to, że algorytmu Clarans, raczej należy używać dla większych zbiorów danych, chociaż ta różnica może również wynikać z niekorzystnego doboru parametrów *numlocal* i *maxneighbor*.

Algorytm Clarans dobrze radzi sobie z dużymi zbiorami danych, w przeciwieństwie do algorytmu PAM, którego czasy wykonania dla dużych zbiorów rosną bardzo szybko. Czas wykonywania algorytmu Clarans

Tablica 2: Wyniki eksperymentów

Nazwa	Typ algorytmu	czas [s]	Wskaźnik sylwetkowy
2d-20c-no0	Clarans	32.25	0.81
2d-20c-no0	**PAM**	**7.89**	**0.81**
dpb	Clarans	18.59	0.64
dpb	**PAM**	**14.18**	**0.65**
disk-6000n	**Clarans**	**6.63**	**0.49**
disk-6000n	PAM	7.84	0.49
cluto-t7-10k	**Clarans**	**79.35**	**0.56**
cluto-t7-10k	PAM	467.25	0.56
normal1	**Clarans**	**165.19**	**0.28**
normal1	PAM	295.59	0.29
normal2	**Clarans**	**299.63**	**0.30**
normal2	PAM	848.08	0.30
normal3	**Clarans**	**1060.69**	**0.82**
normal3	PAM	NaN	NaN
normal4	**Clarans**	**266.86**	**0.80**
normal4	PAM	981.34	0.80
normal5	**Clarans**	**425.28**	**0.80**
normal5	PAM	NaN	NaN

mocno zależy od wartości parametrów *numlocal* i *maxneighbor*.

6 Instukcja użycia oprogramowania

Polecenia wykorzystywane przez użytkownika, aby korzystać z oprogramowania to:

- Utworzenie środowiska i instalacja oprogramowania:

```
$ virtualenv venv --python=3.9
$ source venv/bin/activate
$ pip install -r requirements
```

- Wygenerowanie sztucznych danych:

```
$ python data.py
```

- Uruchomienie algorytmu **Clarans**:

```
$ python measure_clarans.py <sciezka do pliku> <numlocal> <maxneighbor>
```

- Uruchomienie algorytmu **PAM**:

```
$ python measure_pam.py <sciezka do pliku>
```

Aby zwizualizować wyniki obu algorytmów, należy dodać na końcu komendy parametr *--visualize* (działa tylko dla liczby wymiarów mniejszej niż 4); na przykład:

```
$ python measure_pam.py data/2d-10c.arff --visualize
```

Bibliografia

[1] Raymond T. Ng, Jiawei Han *CLARANS: A method for clustering objects for spatial data mining* September 2002.