7. Implementacja wybranego algorytmu eksploracji danych

Algorytm AUTOCLUST

Kaczmarek Kacper

Listopad 2020

Plan prezentacji

1. Środowisko
   - Język programowania
   - Biblioteki
2. Implementacja algorytmu
   - Krótki opis algorytmu
   - Przygotowanie
   - Faza pierwsza
   - Faza druga
   - Faza trzecia
3. Dane
   - Wykorzystane dane

Język programowania

Językiem implementacji algorytmu do projektu będzie Python. Jest to popularny język i łatwy w zrozumieniu oraz nie potrzebuje dużej ilości konfiguracji. Pozwala on na realizowanie skomplikowanych operacji matematycznych oraz ich proste obrazowanie.

Biblioteki

Algorytm opiera się na wielu operacjach matematycznych oraz wykorzystuje wektory czy macierze.

- *math* - do skomplikowanych operacji matematycznych
- *NumPy* - do operacji na macierzach oraz wektorach
- *scipy* - do triangulacji Delaunay
- *Matplotlib* - do wizualizacji wyników
- *UnitTest* - do testów

Opis algorytmu

Wersja implementacji algorytmu opisana jest w tym artykule. Przedstawię poniżej ideę algorytmu, który będę implementował podczas pracy nad projektem.

Przygotowanie

Pierwszą rzeczą, która rozpoczyna działanie algorytmu jest zastosowanie triangulacji Delaunay'a, która dzieli przestrzeń na trójkąty, którymi wierzchołkami są punkty sprawdzane w algorytmie. Taką triangulację można osiągnąć korzystając z algorytmu w bibliotece *SciPy*.

Sortowanie krawędzi

Do posortowania krawędzi potrzeba najpierw policzyć kilka wielkości, które pomogą nam je podzielić na grupy. Najpierw localną średnią długość krawędzi (dla każdej krawędzi dotyczącej punktu).

$$\text{Local\_Mean}(p_i) = \sum_{e \in N(_i)} \frac{|e|}{||N(p_i)||} \tag{1}$$

$N(p_i)$ oznacza zbiór krawędzi wychodzących z punktu p_i
$|e|$ jest długością poszczególnej krawędzi
Następnie lokalną średnie standardowe odchylenie długości krawędzi (dla każdej krawędzi dotyczącej punktu).

$$\text{Local\_St\_Dev}(p_i) = \sqrt{\sum_{j=1}^{d(p_i)} \frac{\text{Local\_Mean}(p_i) - |e_j|)^2}{d(p_i)}} \tag{2}$$

Sortowanie krawędzi c.d.

Następnie możemy wyliczyć średnie odchylenie standardowe dla całego zbioru

$$\text{Mean\_St\_Dev(P)} = \sum_{i=1}^{n} \frac{\text{Local\_Dt\_Dev}(p_i)}{n} \tag{3}$$

Oraz ostatnią wartością przypisaną do konkretnego punktu jest relacyjne standardowe odchylenie.

$$\text{Relative\_St\_Dev}(p_i) = \frac{\text{Local\_St\_Dev}(p_i)}{\text{Mean\_St\_Dev}(P)} \tag{4}$$

Sortowanie c.d.

Teraz można dokonać podziału na krawędzie Krótkie, Długie i Inne:

$$\text{Short}(p_i) = \{e_j | \ |e_j| < \text{Local\_Mean}(p_i) - \text{Mean\_St\_Dev}(P) \ \& \ e_j \in N(p_i)\} \tag{5}$$

$$\text{Long}(p_i) = \{e_j \ | \ |e_j| > \text{Local\_Mean}(p_i) + \text{Mean\_St\_Dev}(P) \ \& \ e_j \in N(p_i)\} \tag{6}$$

$$\text{Other}(p_i) = N(p_i) - (\text{Short}(p_i) \cup \text{Long}(p_i)) \tag{7}$$

Faza pierwsza

W fazie pierwszej należy usunąć Długie krawędzie, ponieważ są za długie, aby łączyły punkty w tej samej grupie. Chwilowo nie brane są pod uwagę również krawędzie Krótkie, ponieważ będą one rozważane w fazie drugiej.

Faza druga - krawędzie Inne

Tę fazę można podzielić na dwa etapy. Pierwszym jest wykorzystanie krawędzi innych na wstępny podział punktów na grupy. Podział ten jest oparty na twierdzeniu, że każda z Innych krawędzi łączy ten sam nietrywialny komponent. Tutaj brane są pod uwagę takie punkty, które mają jedynie więcej niż jedną krawędź Inną. Oznacza to, że mogą zostać pojedyncze punkty, które nie będą miały dopasowania. Zostaną one rozważone w kolejnym etapie

Faza druga - krawędzie krótkie

1. Dla każdego punktu p_i, który ma krawędzie Krótkie, wykonujemy dla niego poniższe kroki:
   a. Rozstrzygnij klasę C dla punktu p_i:
      i. Jeśli są dwie krawędzie Krótkie, których klasa nie jest ta sama to:
         A. Policz wielkość klasy punktu końcowego krawędzi Krótkiej i wyznacz M (największą klasę)
         B. Niech C zostanie M (jeśli M nie jest jednoznaczne - niech C zostanie ta klasa, z najkrótszą krawędzią do p_i)
      iii. Jeśli wszystkie klasy są sobie równe to niech C będzie ta, na którą wskazują wszystkie punkty.
   b. Jeśli punkt p_i posiada krawędzie Inne, które łączą punkty z różnych klas - usuń je.
   c. Przywróć wszystkie krawędzie Krótkie, która łączą ten punkt z klasą C.
2. Jeszcze raz przelicz klastry.
3. Dla każdego singletona (których klasa zawiera jeden element) przywrócić wszystkie krawędzie Krótkie, jeśli punkty na końcach tych krawędzi należą do tej samej klasy.

Faza trzecia

Faza ta polega na wykryciu niezgodności drugiego stopnia. W tym celu należy policzyć lokalną średnią długość krawędzi dla drugiego stopnia (wszystkie krawędzie wychodzące z punktu o długości dwóch krawędzi lub mniej). Wyraża się ona wzorem

$$\text{Local\_Mean}_{2,G}(p_i) = \sum_{e \in N_{2,G}(p_i)} \frac{|e|}{||N_{2,G}(p_i)||} \tag{8}$$

Dla każdego punktu p_i usuwa wszystkie krawędzie w $N_{2,G}(p_i)$, które są długimi krawędziami, czyli które spełniają warunek:

$$|es| > \text{Local\_Mean}_{2,G}(p_i) + \text{Mean\_St\_Dev}(P) \tag{9}$$

Wykorzystane dane

Dane wykorzystywane przy realizacji tego projektu pochodzą ze strony kaggle. Jest to osiem różnych zbiorów sklasyfikowanych po dwie grupy na każdy zbiór o różnej gęstości rozłożenia danych oraz różnym rozkładzie klas na przestrzeni dwuwymiarowej - 2D dataset. Dane te można w prosty sposób zobrazować i przedstawić wyniki działania algorytmu AUTOCLUST. Dodatkowo dane są etykietowane, co pozwoli zweryfikować poprawność rozwiązania.

Przykładowe dane