KMeans Inicjalizacja

23 listopada 2017

Grzegorz Borkowski

1. WSTĘP

Zadanie zostało zaimplementowane w Pythonie 3.6.1
Do raportu został dołączony Jupyter Notebook z kodem źródłowym programu.

2. ZADANIE

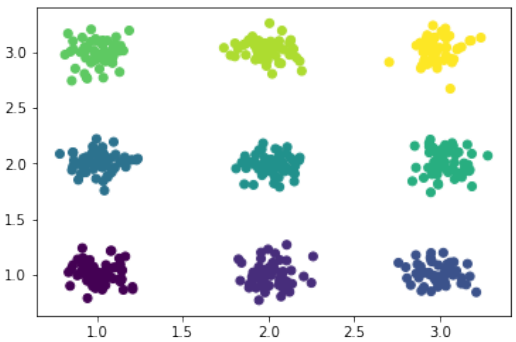
2.1 Treść zadania

Wygenerujmy zbiór danych wyglądający +- tak jak na załączonym rysunku. Uruchamiamy na nim algorytm k-means z k równym 9 i następującymi metodami inicjowania środków klastrów: * Random - z rozkładem jednostajnym po całym zakresie wartości; * Forgy - wybieramy k elementów ze zbioru jako początkowe środki; * Random Partition - losowo dzielimy zbiór na k klastrów, początkowy środek klastra to średnia z elementów które w ten sposób w nim się znalazły; * k-means++ - wybieramy początkowe środki w sposób opisany w paperze z załącznika.

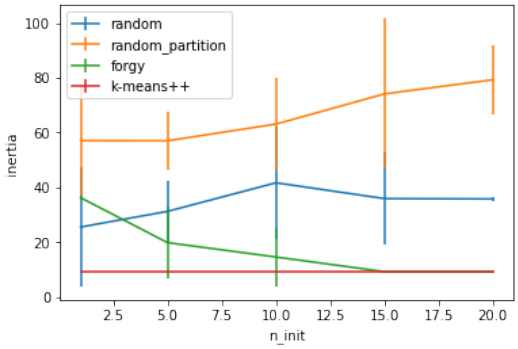
Naszym celem jest uzyskanie wykresu jakości klastryzacji Q w zależności od numeru iteracji n dla wszystkich powyższych metod (wszystkie wyniki na jednym wykresie). Jakość Q rozumiemy jako wybraną metrykę jakości (np. Davies-Bouldin index czy Dunn index, może być dowolna rozsądna inna - ale nie Silhouette). Proces k-means jest silnie stochastyczny, więc eksperyment powtarzamy wielokrotnie, a na wykresie pokazujemy średni wynik i jego odchylenie standardowe jako errorbary.

Dodatkowo należy zwizualizować miarę Silhouette dla każdej obserwacji ze zbioru wraz z ich podziałem na klastry na 3 etapach jednego przykładowego przebiegu: na początku, w trakcie i na końcu.

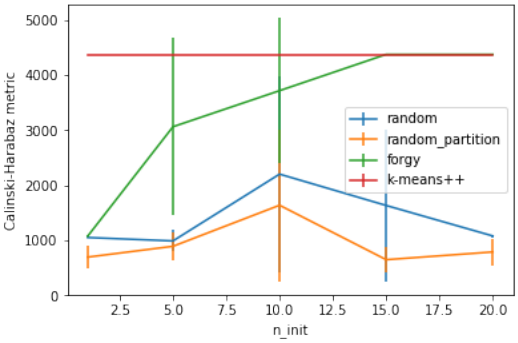
2.2 Rozwiązanie



Rysunek 1. Wygenerowany zadany zbiór 500 elementów.



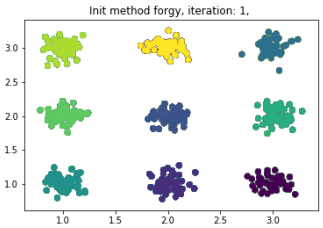
Rysunek 2. Suma kwadratów dystansu próbek do środka najbliższego klastra, pięć pomiarów dla każdego ninit (liczba inicjalizacji k-means z różnymi centroidami - wynik zwrócony - najmniejszy błąd ze wszystkich prób)



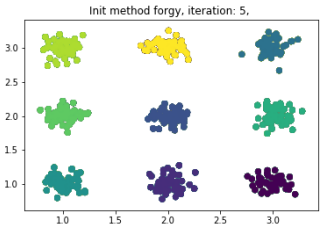
Rysunek 3. Metryka Callinski-Harabaz, średnia i odchylenie standardowe z pięciu pomiarów dla każdego ninit

Tabela 1. Średnia miara Silhouette dla każdej obserwacji po różnych przebiegach

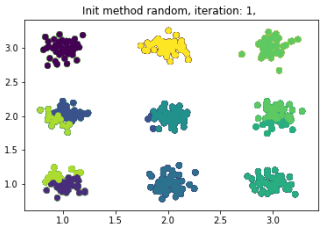
	1 Iteracja	5 iteracji	15 Iteracji
Random	0.543	0.539	0.673
Random Partition	0.346	0.474	0.561
Forgy	0.810	0.810	0.810
K-Means++	0.810	0.810	0.810



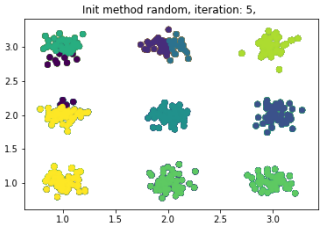
Rysunek 4. Forgy, po pierwszej iteracji



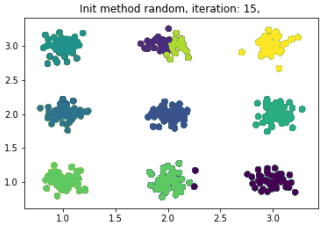
Rysunek 5. Forgy po pięciu iteracjach



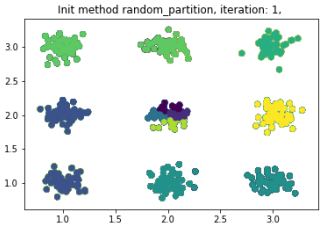
Rysunek 6. Random po pierwszej iteracji



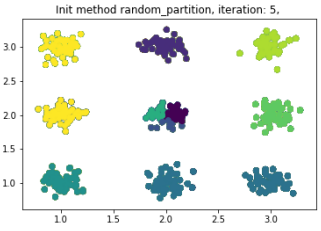
Rysunek 7. Random po pięciu iteracjach



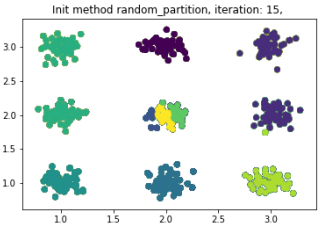
Rysunek 8. Random po piętnastu iteracjach



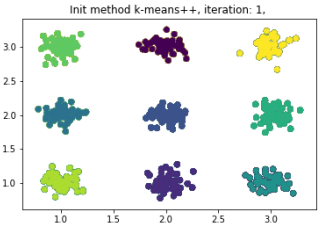
Rysunek 9. Random partition po pierwszej iteracji



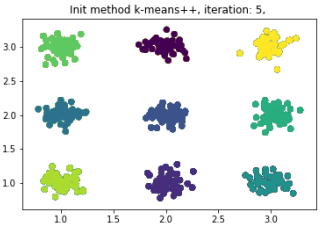
Rysunek 10. Random partition po pięciu iteracjach



Rysunek 11. Random partition po piętnastu iteracjach

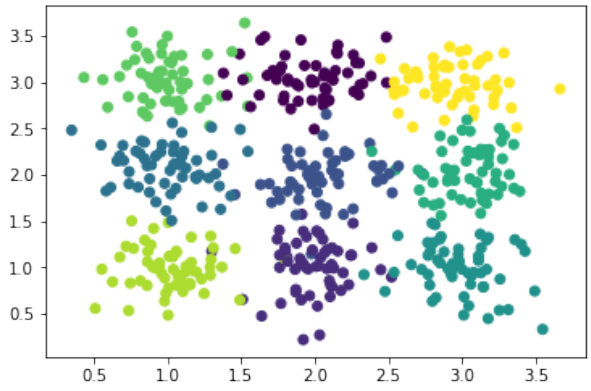


Rysunek 12. KMeans++ po pierwszej iteracji

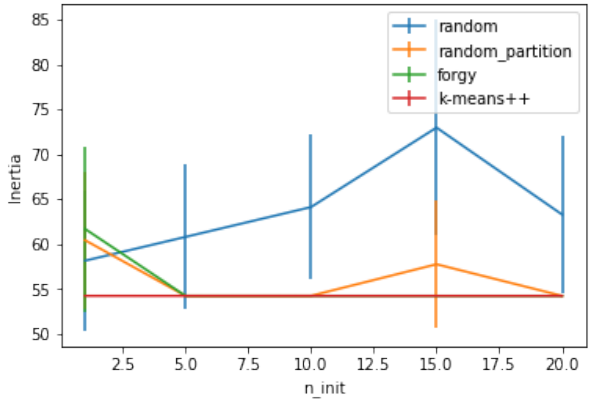


Rysunek 13. KMeans++ po pięciu iteracjach

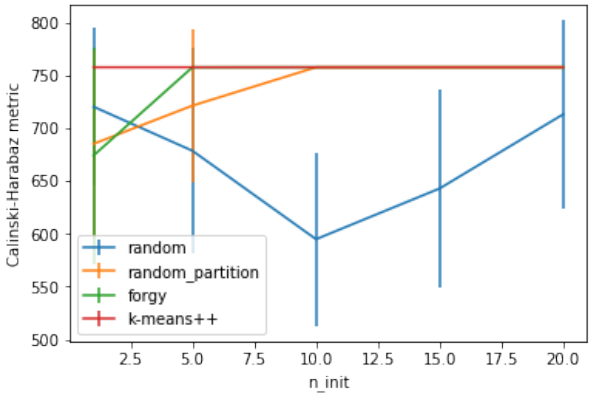
Powtórzyłem eksperyment dla innego zbioru wejściowego, który wyglądał następująco:



Rysunek 14. Drugi zbiór testowy



Rysunek 15. Suma kwadratów odległości próbek od najbliższego środka klastra



Rysunek 16. Metryka Callinski-Harabaz dla drugiego zbioru

3. WNIOSKI

(1) Metoda inicjalizacji k-means++ dla badanych zbiorów testowych zapewnia najlepsze klastrowanie według metryki "inertia" i Callinski-Harabaz
(2) Metoda inicjalizacji Forgy zapewnia lepsze rezultaty niż metoda Random Partition i Random
(3) Metoda Random Partition nie daje sobie dobrze rady w przypadku gdy środki klastrów są mniej więcej równomiernie rozmieszczone - wtedy random partition wybiera jako początkowe środki klastrów punkty mniej więcej w połowie odległości między najdalej oddalonymi punktami na pewnej osi
(4) Metryka Silbouette w badanym przypadku dobrze korelowała ze skutecznością metody inicjalizacji - metoda K-Means++ i Forgy osiągały lepsze rezultaty niż metody Random
(5) Pomimo, że klasteryzacja jest dużo trudniejsza niż klasyfikacja biorąc jako kryterium łatwość w ewaluacji wyznaczonego modelu, to istnieją metryki(Silbouette, Callinski-Harabaz), które mogą określić mniej więcej jak dobrze dane zostały podzielone na klastry