

PADR 2020/2021

Praca domowa nr 1 (max. = 15 p.)

W ramach niniejszego projektu zaimplementujesz i przetestujesz algorytm spektralny analizy skupień (*spectral clustering*) oparty na grafie kilku najbliższych sąsiadów punktów z wejściowego zbioru danych.

Termin oddania pracy: 13.11.2020, godz. 10:00.

Prace domowe należy przesłać za pośrednictwem platformy Moodle – **jedno archiwum .zip**¹ o nazwie typu `Nazwisko_Imie_NrAlbumu_Nick_pd1.zip`. W archiwum znajdować się powinien jeden katalog, `Nazwisko_Imie_NrAlbumu_Nick_pd1`, dopiero w którym umieszczone zostaną następujące pliki:

- plik `spectral.R` zawierający implementacje funkcji `Mnn()`, `Mnn_graph()`, `Laplacian_eigen()`, `spectral_clustering()` itd.; [12 p.]
- plik `testy.Rmd` i `testy.pdf` – testy poprawności zaimplementowanych metod na przynajmniej trzech *własnych* zbiorach danych z \mathbb{R}^2 lub \mathbb{R}^3 (z ilustracjami m.in. w postaci wykresów); [3 p.]

Nazwy plików nie powinny zawierać polskich liter diakrytyzowanych (przekształć $q \rightarrow a$ itd.).

W nazwach plików wynikowych, `Nazwisko_Imie_NrAlbumu_Nick_pd1.(Rmd pdf R)`, Nick oznacza wybrany przez Państwa pseudonim, którego będziemy używać do publikowania wyników (inny niż nazwa użytkownika na platformie Github).

1 Zadanie analizy skupień

Niech dana będzie macierz $\mathbf{X} \in \mathbb{R}^{n \times d}$ reprezentująca n punktów $\{x_1, \dots, x_n\}$ w \mathbb{R}^d . Zadanie analizy skupień² (ang. *cluster analysis*) jest przykładem uczenia bez nadzoru. W dużym uproszczeniu, jego celem jest *automatyczne* znalezienie takiego *podziału* zbioru danych na $k > 1$ (dane z góry) parami rozłącznych i niepustych podzbiorów – zwanych *skupieniami* – tak by obserwacje należące do tego samego skupienia były do siebie jak najbardziej *podobne* (np. leżały „blisko” siebie), zaś obserwacje z dwóch różnych skupień były możliwie jak najbardziej od siebie *odmienne*.

2 Ocena jakości podziału

Wynikiem działania wszystkich rozpatrywanych tutaj algorytmów analizy skupień będzie ciąg $\mathbf{z} \in \{1, \dots, k\}^n$, taki że z_i określa, do którego z k skupień należy punkt x_i . Zachodzi oczywiście $(\forall j = 1, \dots, k) (\exists i) z_i = j$.

Zakładamy tutaj, że algorytm analizy skupień jest *dobry*, jeśli generuje podziały podobne do referencyjnych etykiet. Do oceny podobieństwa dwóch k -podziałów mogą Państwo użyć następujących miar:

¹A więc nie: .rar, .7z itp.

² Zob. np. [Koronacki J., Ćwik J., *Statystyczne systemy uczące się*, EXIT, 2008, rozdz. 9] lub [Hastie T., Tibshirani R., Friedman J., *The Elements of Statistical Learning*, Springer, 2017, rozdz. 14.3] – <http://web.stanford.edu/~hastie/ElemStatLearn/>

- indeks Fowlkesa–Mallowsa (FM)³, zob. `dendextend::FM_index`;
- skorygowany indeks Randa (AR)⁴, zob. `mclust::adjustedRandIndex`.

Każdy z powyższych indeksów zwraca wartość równą 1, jeśli dwa dane k -podziały są równoważne. Im ich wartość jest dalej od 1, tym bardziej są one od siebie różne.

Uwaga 1: Brana będzie pod uwagę jakość kodu. Na przykład kod należy zamknąć w dobrze udokumentowane, wyspecjalizowane funkcje, tak by uniknąć powtórzeń itp. Kod powinien być dobrze udokumentowany (np. przez komentarze). Algorytm, który Państwo implementują będzie wykorzystywany przy pracy domowej nr 4. Dlatego warto zadbać o jakość kodu i jego czytelność tak by za kilka miesięcy mogli się Państwo nim bez problemu posłużyć.

Uwaga 2: Mogą Państwo korzystać z pętli gdy jest to niezbędne.

Uwaga 3: Jeśli nie potrafisz czegoś zaimplementować samodzielnie, posłuż się gotowcem (w szczególności metoda spektralna jest już gdzieś zaimplementowana...) – uzyskasz przynajmniej choć kilka punktów (oraz poćwiczysz pisanie raportu). Własne trzy zbiory benchmarkowe też możesz wygenerować bez implementacji poniższych.

3 Algorytm spektralny i jego implementacja

Algorytm spektralny w wersji, którą tutaj zaimplementujesz, polega na zastosowaniu „zwykłej” procedury k średnich na odpowiednio zmodyfikowanej (poddanej różnym przekształceniom określonym przez widmo macierzy „bliskości” analizowanych punktów) macierzy \mathbf{X} .

Napisz funkcję `spectral_clustering(X, k, M)`, która dla $\mathbf{X} \in \mathbb{R}^{n \times d}$, $k \geq 2$ oraz $M \in \mathbb{N}$ zwraca k -podział zbioru danych \mathbf{X} wyznaczony przy użyciu opisanych niżej podprocedur:

1. znajdowanie M najbliższych sąsiadów wszystkich punktów;
2. stworzenie grafu „sąsiedztwa” i uspoźnienie go;
3. wyznaczenie odpowiednich k wektorów własnych jego laplasjanu;
4. zastosowanie algorytmu k średnich w nowej przestrzeni danych.

3.1 Macierz najbliższych sąsiadów

Napisz funkcję `Mnn(X, M)` (*M-nearest neighbors*), która dla $\mathbf{X} \in \mathbb{R}^{n \times d}$ oraz $M \in \mathbb{N}$ wyznacza macierz $\mathbf{S} \in \mathbb{N}^{n \times M}$, taką że $s_{i,j}$ jest indeksem j -tego najbliższego sąsiada x_i względem metryki euklidesowej.

W szczególności ma zachodzić $(\forall i) s_{i,1} = \arg \min_{j \neq i} \|x_i - x_j\|$ (przy założeniu, że odległości się nie powtarzają).

3.2 Macierz sąsiedztwa

Napisz funkcję `Mnn_graph(S)`, która jako argument przyjmuje macierz $\mathbf{S} \in \mathbb{N}^{n \times M}$ wygenerowaną przy użyciu powyższej funkcji.

Funkcja ta generuje symetryczną macierz $\mathbf{G} \in \{0, 1\}^{n \times n}$, taką że $g_{i,j} = 1$, jeśli $(\exists u) s_{i,u} = j$ lub $s_{j,u} = i$.

\mathbf{G} jest więc macierzą sąsiedztwa reprezentującą graf nieskierowany \tilde{G} o n wierzchołkach, taki że i -ty wierzchołek jest połączony z j -tym, jeśli x_i jest wśród M najbliższych sąsiadów x_j lub x_j jest wśród M najbliższych sąsiadów x_i .

³ [Fowlkes E.B., Mallows C.L., A Method for Comparing Two Hierarchical Clusterings, *Journal of the American Statistical Association* **78**(383), 1983, 553–569]

⁴[por. Hubert L., Arabie P., Comparing Partitions, *Journal of the Classification* **2**, 1985, 193–218]

Z oczywistych względów n nie może być zbyt duże (powiedzmy większe niż 50,000). W praktyce funkcja `Mnn_graph(S)` powinna zwracać macierz rzadką, zob. `scipy.sparse` w Pythonie lub pakiet `Matrix` (klasa `dsrMatrix`) w R. W niniejszym projekcie nie jest to wymogiem, ale zachęcam do poszerzenia swojej wiedzy i rozwoju nowych umiejętności.

Należy wykryć wszystkie składowe spójne (na przykład przy użyciu algorytmu przeszukiwania wszerz (BFS) lub w głąb (DFS)). Jeśli graf \tilde{G} jest spójny, zwracamy \mathbf{G} bez dalszych modyfikacji.

Mogą Państwo wykorzystać gotowe implementacje tych algorytmów z pakietów R-a.

W przeciwnym przypadku, zakładając, że w \tilde{G} jest p składowych spójnych, należy dodać do \tilde{G} dokładnie $p - 1$ (nieskierowanych) krawędzi (w dowolny poprawny sposób), tak by \tilde{G} uspojnić. Dopiero tak zmodyfikowaną macierz sąsiedztwa zwracamy w wyniku działania funkcji.

3.3 Laplasjan i jego wektory własne

Funkcja `Laplacian_eigen(G, k)` dla $k > 1$ i macierzy \mathbf{G} jak wyżej:

1. wyznacza laplasjan grafu \tilde{G} , tj. $\mathbf{L} = \mathbf{D} - \mathbf{G}$, gdzie \mathbf{D} jest macierzą diagonalną taką, że $d_{i,i}$ jest stopniem i -tego wierzchołka w \tilde{G} ;
2. wyznacza macierz $\mathbf{E} \in \mathbb{R}^{n \times k}$, której kolumny składają się z wektorów własnych macierzy \mathbf{L} odpowiadających 2., 3., ..., $(k+1)$ najmniejszej wartości własnej;
3. zwraca \mathbf{E} jako wynik.

Do wyznaczania wektorów własnych używamy oczywiście funkcji „wbudowanej”.

3.4 Algorytm k -średnich

Na tak wyznaczonej macierzy \mathbf{E} , należy uruchomić algorytm k -średnich. Jego gotową implementację znajdziesz w jednej z bibliotek.