PDRPy - Praca domowa nr 2

Wawrzeńczyk Adam

4 maja 2019

Wstęp

Zadanie polegało na zaimplementowaniu algorytmu spektralnego analizy skupień z użyciem Rcpp oraz porównaniu go z innymi algorytmami zapewnianymi przez R oraz dostępnymi na CRAN. Zbiór przetestowanych algorytmów obejmuje:

- **Algorytm spektralny** z parametrem M:
  - M = 2 - *spectral_M2*
  - M = 5 - *spectral_M5*
  - M = 8 - *spectral_M8*
  - M = 10 - *spectral_M10*
  - M = 12 - *spectral_M12*
  - M = 15 - *spectral_M15*
  - M = 20 - *spectral_M20*
  - M = 30 - *spectral_M30*
  - M = k, gdzie k jest oczekiwaną liczbą skupieć - *spectral_Mk*
- **Algorytm k-średnich** z pakietu wbudowanego stats - *kmeans*
- **Algorytm hierarchiczny hclust** z pakietu wbudowanego stats:
  - różne sposoby tworzenie dendrogramu - *hclust_ward.D, hclust_ward.D2, hclust_single, hclust_complete, hclust_average,hclust_mcquitty, hclust_median, hclust_centroid* oraz *hclust_centroid2*. Druga część nazwy odpowiada zastosowanemu wariantowi algorytmu. Dwa sposoby wywołania algorytmu *hclust centroid* wynikają z faktu, że w dokumentacji sugerowaną metodą obliczania odległości dla algorytmu jest nie zwyczajna odległość Euklidesowa realizowana przez funkcję *hclust_centroid*, a odległość Euklidesowa podniesiona do kwadratu (*hclust_centroid2*)
- **Algorytm hclust2** z pakietu CRAN genie z parametrem gini (0 - 1):
  - gini = 0.1 - *genie_0.1*
  - gini = 0.2 - *genie_0.2*
  - gini = 0.3 (wartość domyślna) - *genie_0.3*
  - gini = 0.5 - *genie_0.5*
  - gini = 0.8 - *genie_0.8*
- **Algorytm Fuzzy C-Means** z pakietu CRAN advclust - z parametrem fuzzyfier (> 1):
  - fuzzyfier = 1.5 (wartość domyślna) - *fuzzy_default*
  - fuzzyfier = 2 - *fuzzy_2*
  - fuzzyfier = 5 - *fuzzy_5*
  - fuzzyfier = 10 - *fuzzy_10*
- Podjąłem również próbę zastosowania **algorytmu DBSCAN**, jednak ze względu na nadzwyczajną wrażliwość na parametry nie nadawał się on do wykorzystania w tak ogólnych testach.

Skrótowe nazwy pojawiające się w powyższym będą używane do identyfikacji algorytmów również dalej.

Algorytm spektralny

Implementacja algorytmu spektralnego została zrealizowana jako pakiet R. Aby móc go używać, należy zbudować oraz zainstalować projekt zawarty w folderze **MySpectralClustering**. W implementacji funkcji `Mnn()` z uwagi na jak największą wydajność algorytmu oraz zapewnienie poprawności rozwiązania i optymalności algorytmu została wykorzystana zewnętrzna implementacja kd-drzewa dostępna pod licencją MIT na

githubie. Kd-drzewo pozwala na znalezienie k najbliższych sąsiadów wierzchołka w grafie z n wierzchołkami w czasie $O(k\sqrt{n})$, szybciej niż algorytm naiwny. Z tego samego względu w Rcpp zostały zaimplementowane fragmenty pozostałych funkcji. Szczegółowe testy tego algorytmu zostały opisane w dokumencie **testy.pdf**. Wymagane w poleceniu pliki znajdują się pod ścieżkami *MySpectralClustering/src/spectral_aux.cpp* oraz *MySpectralClustering/R/spectral.R*

Pozostałe elementy rozwiązania

Foldery **benchmark** oraz **my_benchmark** zawierają odpowiednio dołączone do zadania zbiory benchmarkowe oraz zbiory wygenerowane jako część rozwiązania zadania (zostały one wygenerowane za pomocą plików **graph.R, labirynth.R oraz windows.R**). Uruchamianie zaimplementowanych algorytmów jest realizowane w pliku **benchmark_executor.R**. Uzyskane wyniki znajdują się odpowiednio w folderach **output** oraz **my_benchmark_output**, podzielone względem algorytmów. Plik **helper.R** jest plikiem pomocniczym do generowaniu raportów. W dalszej części tego dokumentu skupię się na opracowaniu wyników otrzymanych dla załączonych danych benchmarkowych.

Opracowanie wyników

Porównanie otrzymanych wyników zostanie wykonane za pomocą wyznaczonych dla każdego rozwiązania skorygowanego indeksu Randa **(AR)** oraz indeksu Fowlkesa-Mallowsa **(FM)**.

Zagregowane indeksy Randa i FM

Dla dostarczonych danych benchmarkowych obliczone zostały średnie wartości indeksów AR i FM. Porównane zostały one również z wartościami tych indeksów dla danych ustandaryzowanych, z uwzględnieniem zmiany spowodowanej przez standaryzację. Zestawienie otrzymanych wyników znajduje się w poniższych tabelach. Zostały one posortowane poczynając od największych wartości dla indeksu AR i zaokrąglone do trzech cyfr po przecinku.

Algorithm	RA	RA - st. data	RA - difference	FM	FM - st. data	FM - difference
genie_0.3	0.809	0.795	-0.014	0.867	0.856	-0.011
genie_0.2	0.798	0.784	-0.014	0.865	0.853	-0.012
genie_0.5	0.798	0.783	-0.015	0.860	0.853	-0.007
genie_0.1	0.770	0.756	-0.014	0.851	0.839	-0.012
hclust_wardD	0.634	0.619	-0.015	0.782	0.773	-0.009
fuzzy_default	0.599	0.559	-0.04	0.760	0.730	-0.03
hclust_wardD2	0.587	0.584	-0.003	0.763	0.761	-0.002
fuzzy_2	0.586	0.557	-0.029	0.750	0.728	-0.022
hclust_average	0.582	0.545	-0.037	0.777	0.759	-0.018
genie_0.8	0.570	0.540	-0.03	0.764	0.753	-0.011
hclust_centroid2	0.566	0.516	-0.05	0.775	0.759	-0.016
kmeans	0.553	0.513	-0.04	0.724	0.704	-0.02
spectral_M12	0.534	0.518	-0.016	0.708	0.692	-0.016
hclust_centroid	0.531	0.427	-0.104	0.767	0.725	-0.042
spectral_M30	0.526	0.446	-0.08	0.697	0.646	-0.051
hclust_complete	0.521	0.484	-0.037	0.734	0.705	-0.029
hclust_mcquitty	0.510	0.490	-0.02	0.720	0.706	-0.014
spectral_Mk	0.491	0.517	0.026	0.690	0.694	0.004
spectral_M15	0.489	0.523	0.034	0.676	0.688	0.012

Algorithm	RA	RA - st. data	RA - difference	FM	FM - st. data	FM - difference
spectral_M10	0.487	0.483	-0.004	0.667	0.678	0.011
hclust_median	0.483	0.376	-0.107	0.702	0.672	-0.03
hclust_single	0.475	0.449	-0.026	0.729	0.719	-0.01
spectral_M20	0.472	0.482	0.01	0.664	0.668	0.004
spectral_M5	0.469	0.492	0.023	0.663	0.680	0.017
spectral_M8	0.463	0.456	-0.007	0.659	0.655	-0.004
fuzzy_5	0.462	0.428	-0.034	0.647	0.623	-0.024
fuzzy_10	0.448	0.427	-0.021	0.636	0.623	-0.013
spectral_M2	0.318	0.365	0.047	0.576	0.587	0.011

Na podstawie otrzymanych wyników można poczynić kilka obserwacji.

- Wartości obu indeksów, pomimo, że odbiegają od siebie, zachowują się podobnie. Pomimo, że wartości są posortowane względem indeksu AR, możemy zaobserwować (z nieznacznymi odstępstwami) wyraźny malejący trend również dla kolumn odpowiadających indeksowi FM. W dalszej części dokumentu skupimy się na "rankingu" tworzonym przez indeks AR.
- Wpływ standaryzacji zmiennych na zdecydowaną większość algorytmów analizy skupień jest znikomy. Wartości różnicy d dla danych ustandaryzowanych zostały oznaczone kolorami:
  - $d \leq -0.1$ - czerwony,
  - $-0.1 \leq d \leq -0.05$ - pomarańczowy,
  - $-0.05 \leq d \leq 0.05$ - czarny,
  - $0.05 \leq d \leq 0.1$ - jasnoniebieski,
  - $0.1 \leq d$ - ciemnoniebieski

  dla indeksu AR, natomiast połowa tych wartości dla indeksu FM.

  Różnice znaczące (oznaczone kolorami innymi niż czarny) występują bardzo rzadko. Łatwo też zauważyć, że zdarzają się one zwykle w przypadku algorytmów "gorszych", o niższych wartościach indeksów - najczęściej źle skalibrowanych (złe parametry) lub po prostu działających jedynie w szczególnych przypadkach. Niemniej jednak, sytuacje, gdy standaryzacja przynosi zły efekt praktycznie nie występują - przeważająca większość różnic ma znak dodatni.
- Dobrze (uniwersalnie) skalibrowany algorytm genie osiąga o wiele lepsze wyniki niż którykolwiek z innych testowanych algorytmów. Proponowany przez twórców parametr gini równy 0.3 rzeczywiście osiąga uniwersalnie najlepsze wyniki. Delikatne odchylenia od tej wartości nadal osiągają uniwersalnie zadowalające wyniki. Jednak nawet z paremetrem gini tak dużym jak 0.8 algorytm ten jest uniwersalnie lepszy od większej części innych algorytmów, natomiast ma swoją niszę - są przypadki, dla których generuje on rozwiązanie najlepsze.
- Tradycyjne algorytmy hierarchiczne osiągają bardzo rozbieżne wyniki w zależności od zastosowanej metody. Najlepsze z nich okazują się te oparte na algorytmie Warda, natomiast zdecydowanie najgorszy - algorytm medianowy. Znajdujący się ponad nim algorytm centroidowy bardzo dużo zyskuje na przekazaniu mu kwadratów odległości - algorytm *hclust_centroid2* osiąga znacznie lepsze wyniki niż *hclust_centroid*.
- Algorytm spektralny osiąga przeciętne wyniki. Wartość parametru M (liczby sąsiadów) ma dla niego duże znaczenie - wartości rzędu kilkunastu okazały się działać najlepiej, okazując się lepsze od tradycyjnego algorytmu k-średnich. Dobrym pomysłem okazało się również uzależnienie parametru M od liczby poszukiwanych skupień. Bardzo małe oraz bardzo duże wartości M powodują spadek ogólnej jakości algorytmu, ale mają swoją niszę. Wyjątkiem jest M równe (oraz mniejsze niż) 2 - wyniki w ten sposób otrzymane okazują się być bardzo złej jakości. Dużą wadą algorytmu jest też długi czas wykonania w przypadku dużych zbiorów danych, związany głównie z czasochłonnym procesem znajdowania wektorów własnych.
- Algorytm Fuzzy C-Means dla wartości fuzzyfiera zbliżonych do domyślnej (1.5) osiąga bardzo dobre wyniki, chociaż nadal bardzo daleko mu do algorytmu gini. Zbytni wzrost fuzzyfiera powoduje jednak znaczny spadek jakości algorytmu. Podobnie jak algorytm spektralny, czas wykonania dla dużych zbiorów danych może być długi (zależnie od prędkości zbieżności).

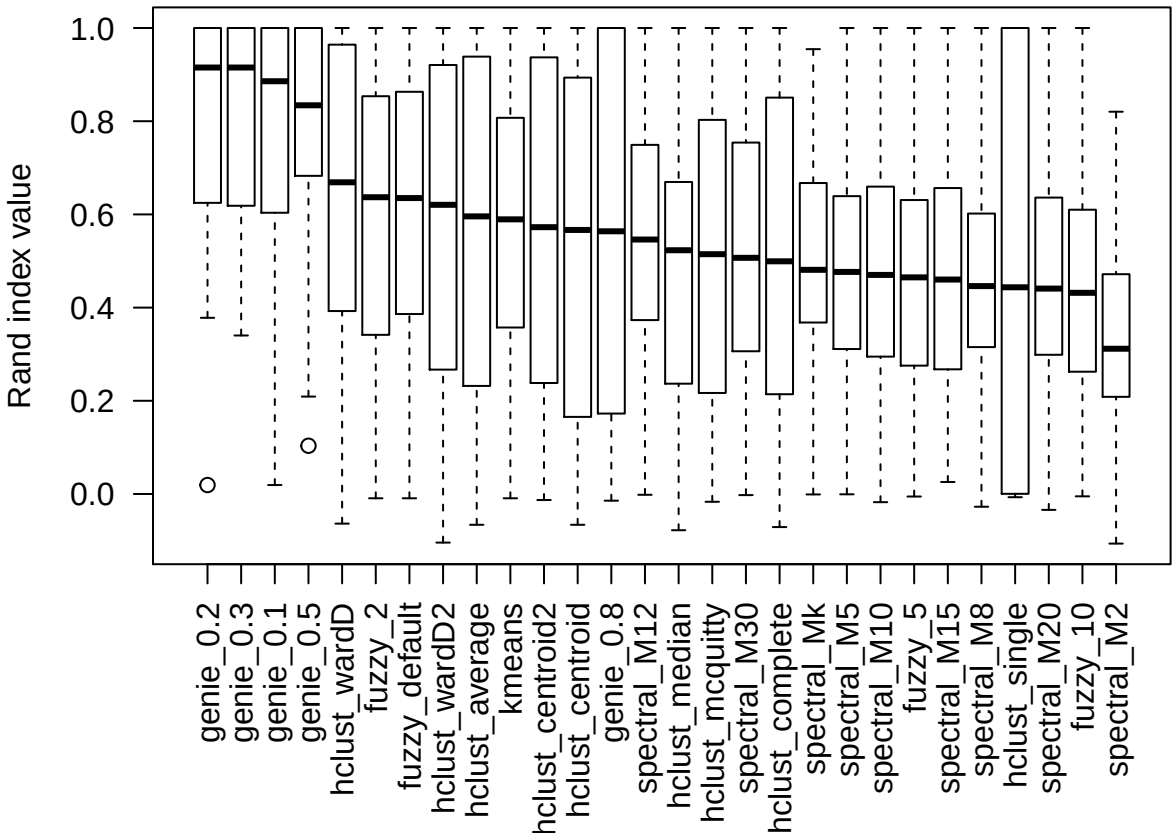
Obliczyłem również wartości odchylenia standardowego dla wszystkich indeksów.

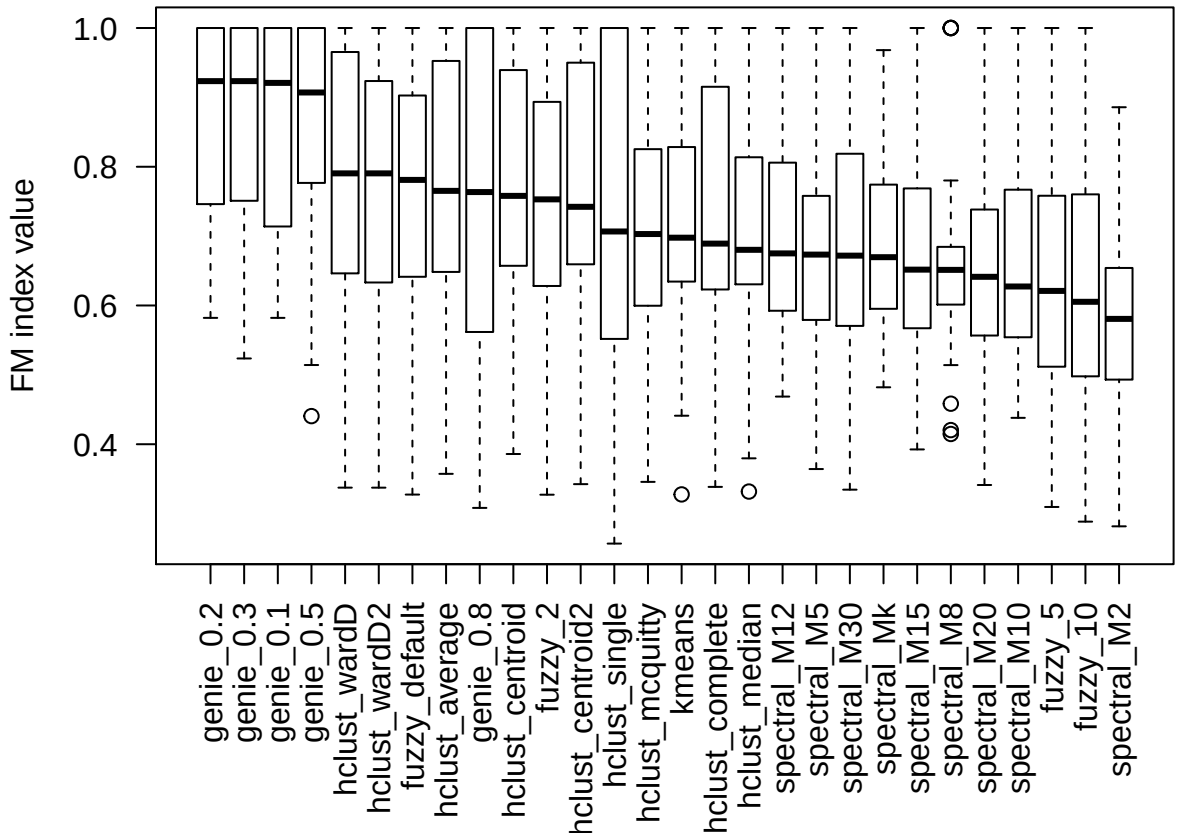
Algorithm	RA	RA - st. data	FM	FM - st. data
hclust_single	0.436	0.438	0.240	0.243
genie_0.8	0.405	0.417	0.225	0.225
hclust_centroid	0.386	0.375	0.174	0.161
hclust_centroid2	0.378	0.394	0.179	0.180
hclust_average	0.370	0.371	0.181	0.180
hclust_wardD2	0.340	0.357	0.181	0.190
hclust_complete	0.338	0.326	0.171	0.164
hclust_wardD	0.334	0.340	0.186	0.188
fuzzy_default	0.332	0.322	0.182	0.175
hclust_mcquitty	0.332	0.318	0.172	0.163
fuzzy_2	0.327	0.328	0.180	0.179
spectral_M30	0.321	0.282	0.186	0.168
hclust_median	0.316	0.327	0.167	0.147
kmeans	0.304	0.318	0.159	0.166
spectral_M12	0.291	0.291	0.158	0.165
fuzzy_10	0.282	0.280	0.181	0.181
spectral_M10	0.282	0.310	0.167	0.169
spectral_M20	0.280	0.313	0.168	0.186
fuzzy_5	0.278	0.277	0.169	0.175
spectral_M15	0.269	0.277	0.155	0.171
genie_0.1	0.263	0.290	0.155	0.173
spectral_M8	0.250	0.265	0.138	0.146
genie_0.5	0.239	0.287	0.161	0.185
spectral_M5	0.238	0.253	0.146	0.147
genie_0.2	0.235	0.263	0.145	0.164
spectral_Mk	0.233	0.207	0.122	0.126
genie_0.3	0.216	0.247	0.149	0.169
spectral_M2	0.209	0.218	0.125	0.143

Również stąd możemy wyciągnąć użyteczne informacje:

- Wartości odchylenia standardowego dla większości algorytmów są stosunkowo duże. Pokazuje to, że są one wrażliwe na wybranie dobrego przypadku użycia dla danego algorytmu.
- Górna część tabeli obejmuje algorytmy "wyspecjalizowane". Algorytm *hclust_single*, pomimo osiągania ogólnie złych wyników, okazuje się jednak mieć potencjał w bardzo wielu przypadkach - podobnie jak większość algorytmów hierarchicznych. Zgodnie ze wcześniejszym stwierdzeniem, do grupy "specjalistów" należy również algorytm *genie_0.8*.
- Dolna część tabeli wskazuje na osiąganie w miarę zbliżonych wyników - jeśli szukamy algorytmu uniwersalnego, chcemy wybrać jeden z nich. Należy jednak uważać - o ile algorytm *spectral_k* i algorytmy z rodziny *genie* są świetnymi algorytmami uniwersalnymi, *spectral_M2* jest uniwersalnie beznadziejny.

Następnym wykonanym przeze mnie krokiem było wykonanie wykresów skrzynkowych dla obu indeksów, tym razem uporządkowane względem mediany.


Rand index value
1.0
0.8
0.6
0.4
0.2
0.0
genie_0.2
genie_0.3
genie_0.1
genie_0.5
hclust_wardD
fuzzy_2
fuzzy_default
hclust_wardD2
hclust_average
kmeans
hclust_centroid2
hclust_centroid
genie_0.8
spectral_M12
hclust_median
hclust_mcquitty
spectral_M30
hclust_complete
spectral_Mk
spectral_M5
spectral_M10
fuzzy_5
spectral_M15
spectral_M8
hclust_single
spectral_M20
fuzzy_10
spectral_M2



Płyną z nich kolejne wnioski:

- Po raz kolejny algorytmy *gini* wyraźnie wyróżniają się spośród reszty. Charakteryzuje je również bardzo wysoka mediana - przeciętny znaleziony wynik jest wyjątkowo dobry i często okaże się lepszy od algorytmu, który na danym zbiorze danych działa dużo lepiej niż zwykle.
- Najdłuższe słupki na wykresie odpowiadają wcześniej zidentyfikowanym "specjalistom", a najkrótsze - algorytmom uniwersalnym.
- Algorytm *spectral_Mk*, pomimo generalnie dobrego działania, nigdy nie zwraca rozwiązać poprawnych lub bardzo zbliżonych według indeksu Randa.