Karl M. Heuer / David E. Roberson

Autumn 2024
Exercise class:
in 302 on October 22nd

Exercises for Computational Tools for Data Science (02807)

Week 7: Mining social network graphs

For this exercise sheet we recommend installing and using the following Python libraries: *networkx*, *scikit-learn* and *matplotlib*.

For more information about *scikit-learn* and *matplotlib* see Exercise Sheet 6.

For information about *networkx* see: https://networkx.org/.

Exercise 1: Divisive clustering via Girvan–Newman

Implement a divisive hierarchical clustering algorithm via the Girvan–Newman method and compute modularity scores. Hence, you should implement the following parts:

1. Compute the betweenness centrality for every edge of a graph.
2. Implement a divisive hierarchical clustering algorithm based on removing edges with highest betweenness centrality and considering connected components as clusters.
3. Compute modularity for a clustering of a graph.

Test your implementations on the Karate-Club graph, found in *networkx.karate_club_graph*.

Compare your implementations with the ones from *networkx*:

- *networkx.algorithms.centrality.edge_betweenness_centrality*
- *networkx.algorithms.community.girvan_newman*
- *networkx.algorithms.community.modularity*

Use *matplotlib* to visualise the communities.

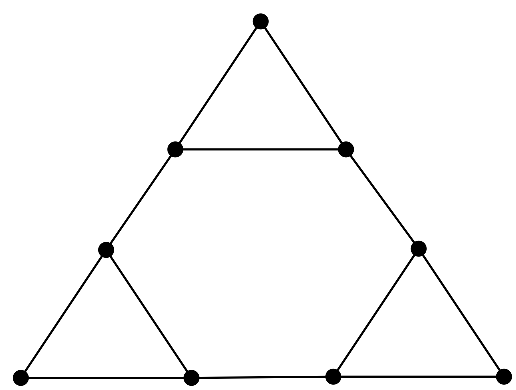
Exercise 2: Comparing Girvan–Newman with Louvain

For the following exercise you may use implementations of the Louvain algorithm from *scikit-network*, see https://scikit-network.readthedocs.io/en/latest/index.html.

1. Run the Louvain algorithm on the Karate-Club graph as well.
2. Compare the results that the Louvain algorithm and the Girvan–Newman algorithm yield. For this, compute the modularity of the found clusterings.
3. Visualise a hierarchy of clusterings that can be found by the Louvain algorithm, e.g. via a dendrogram.

[Optional] Exercise 3:

Consider the following graph G.



Compute the eigenvalues of the Laplacian matrix $L(G)$ of G.

(To compute eigenvalues you could use *numpy.linalg.eigvals* (you should import *numpy*).)

(If you use *networkx*, you could use *networkx.linalg.laplacianmatrix* to obtain the Laplacian matrix of G.)

Derive a bipartition of G via the second smallest eigenvalue of $L(G)$.

Compare your result with *SpectralClustering* from *sklearn.cluster* for 2 clusters. Experiment with *SpectralClustering*, e.g. by using more clusters or running it on the Karate-club graph from *networkx.karate_club_graph*.