

ELU 501

Data science, graph theory and social network studies

Yannis Haralambous (IMT Atlantique)

November 9, 2018

Part

Lecture 3

Community detection

Dividing graphs into clusters

- Our goal is to subdivide a graph into clusters representing some feature among vertices of a cluster and some feature among vertices of different clusters.
- This can reveal structure and allow focusing on specific clusters.
- *Graph partitioning* is “supervised” in the sense where we give in advance the number and/or the size of clusters. Purpose: e.g., numerical calculations.
- *Community detection* is “unsupervised” since we simply seek groups of vertices, called “communities” where members of the same community are “close to each other” and members of different communities are “far from each other”. Purpose: e.g., better understanding of the network.

Graph partitioning

- Partitioning in two parts: *graph bisection*.
- FORMAL STATEMENT OF THE PROBLEM: Divide the graph into two partitions ($P_1 \cup P_2 = G$, $P_1 \cap P_2 = \emptyset$) of given sizes such that the number of edges between the partitions (also called *cut size*) is minimal.
- The number of possible 2-partitions of n vertices is $\approx \frac{2^{n+1}}{\sqrt{n}}$, that's a lot!
- We need a heuristic.

The Kernighan-Lin algorithm

- Let $G = (V, E)$ be the graph. Take a random partition V_1, V_2 of the appropriate sizes. This partition has a given cut size $c = CS(V_1, V_2)$.
- Take all edges $v_i v_j$ such that $v_i \in V_1$ and $v_j \in V_2$. Calculate for each one the cut size c_{ij} of partitions $V_1 \cup \{v_j\} \setminus \{v_i\}$ and $V_2 \cup \{v_i\} \setminus \{v_j\}$. Choose the minimum c_{ij} and apply that switch of vertices to partitions.
- Restart the process by choosing vertices which have not been moved already.
- When no unmoved vertices remain, go back and choose the partition with minimal cut size.
- Complexity $O(mn^2)$ where $n = |V|$ and $m = |E|$.

Assortativity/Homophily

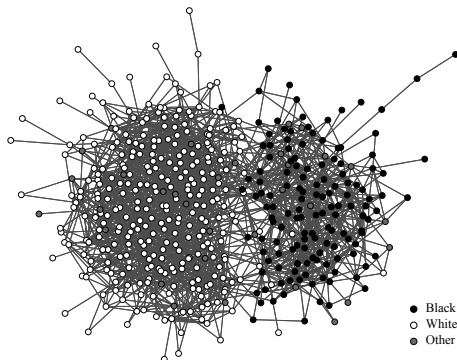
- *Assortativity* (🇫🇷 *assortativité*) is the property of a network in which edges connect preferably vertices with similar characteristics. In social networks assortativity is also called *homophily* (🇫🇷 *homophilie*).
- Let us consider that each vertex v_i belongs to a type c_i described by a finite number of values (modalities) $1 \dots n_c$.
- The total number of edges connecting vertices of the same type is $\sum_{\text{edges } v_i v_j} \delta_{c_i c_j} = \frac{1}{2} \sum_{i,j} A_{ij} \delta_{c_i c_j}$ where δ is the Kronecker symbol.
- Counting the total number of such edges is not interesting if we have nothing to compare it with.
- We will compare it to the *expected* number of edges.


Assortativity/Homophily

- Let k_* be the degree of v_* and $m := |E|$. Then there are $2m$ “ends of edges” and for each edge of the k_i ones leaving v_i the probability that it is one that enters v_j is $k_j/2m$.
- Therefore the expected number of assortative edges is $\frac{1}{2} \sum_{i,j} \frac{k_i k_j}{2m} \delta_{c_i c_j}$.
- Let us take the difference between existing and expected, divided by the number of edges, we call it the *modularity* $\langle \text{modularité} \rangle$ of the graph:

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta_{c_i c_j}.$$

Assortativity/Homophily



- The friendship network of 470 students of a US high school (ages 14–18) (taken from NEWMAN, 2010). The value of Q for this example is 0.305.
- We will call $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$ the *modularity matrix*  *matrice de modularité* of the graph.

Assortativity by scalar values of type

- Let x_* be a scalar type for vertices.
- Let $\mu = \frac{1}{2m} \sum_i k_i x_i$ be the mean of x_* at edge ends.
- Let us define

$$\frac{\sum_{i,j} A_{ij} (x_i - \mu)(x_j - \mu)}{2m} = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) x_i x_j.$$

This quantity behaves like a covariance: if x_i and x_j tend to take the same values, it is increasingly positive, if they tend to take values on different sides of the mean, it is increasingly negative.

- If we consider that the graph is *perfectly assortative* ($A_{ij} = 1 \Rightarrow x_i = x_j$) then the formula becomes

$$\frac{1}{2m} \sum_{ij} (A_{ij} x_i^2 - \frac{k_i k_j}{2m} x_i x_j) = \frac{1}{2m} \sum_{ij} (k_i \delta_{ij} - \frac{k_i k_j}{2m}) x_i x_j.$$



Assortativity by scalar values of type

- We define the *assortativity coefficient*  *coefficient d'assortativité* as the ratio of the two:

$$r = \frac{\sum_{i,j} (A_{ij} - k_i k_j / 2m) x_i x_j}{\sum_{i,j} (k_i \delta_{ij} - k_i k_j / 2m) x_i x_j}.$$

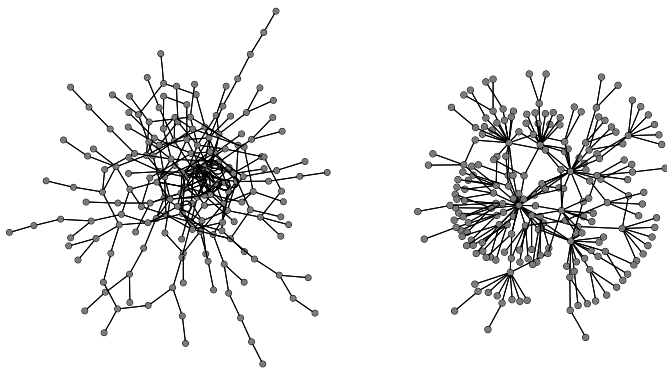
- It is a correlation coefficient in the sense that it takes values between -1 (perfectly disassortative) and 1 (perfectly assortative).

Assortativity by degree

- *Assortativity by degree*  *assortativité par le degré* means that hermits hang out with hermits and highly social people with highly social people.
- We then observe a *core/periphery structure*  *structure de noyau/périphérie*.
- The index formula becomes

$$r = \frac{\sum_{i,j} (A_{ij} - k_i k_j / 2m) k_i k_j}{\sum_{i,j} (k_i \delta_{ij} - k_i k_j / 2m) k_i k_j}.$$

Assortativity by degree



To the left an assortative (by degree) graph, to the right a disassortative one (NEWMAN, 2010).

Community detection based on modularity

- We can attempt to subdivide the graph in subgraphs of maximal modularity.
- The equivalent of Kernighan-Lin:
 - ① take a random division into two groups,
 - ② consider for each vertex the change in modularity which would occur if it were moved to the other group,
 - ③ apply the best one (or the least bad one),
 - ④ repeat the process without processing that vertex again,
 - ⑤ when all iterations are finished go back and find the state with maximum modularity,
 - ⑥ run the algorithm again using that state as the initial one, until modularity no more improves.
- Complexity: $O(mn)$, where $n = |V|$, $m = |E|$.

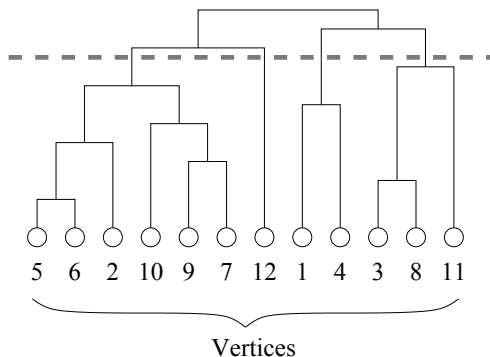
Division in more than one groups

- One can use methods such as *simulated annealing* (🇫🇷 *recuit simulé*) or *genetic algorithms* (🇫🇷 *algorithme génétique*) or a *greedy algorithm* (🇫🇷 *algorithme gourmand*) such as the following:
 - 1 start with one vertex per group,
 - 2 form pairs of groups by choosing at each time the pair that increases the most (or decreases the least) modularity,
 - 3 when all vertices are united in one group, go back and choose the configuration with highest modularity.
- A radically different approach is to search for edges between communities. When removing them we end up with connected components representing communities.

Betweenness-based approach

- We define *edge betweenness* $\langle \text{synexité d'arête} \rangle$ in a way similar to vertex betweenness. Here is an algorithm based on this notion:
 - ① search for the edge with highest betweenness and remove it,
 - ② recalculate betweenness and remove again the edge of highest value,
 - ③ repeat the process until there is no edge with positive betweenness,
 - ④ represent all steps in a *dendrogram* $\langle \text{dendrogramme} \rangle$,
 - ⑤ select, depending on the number and size of communities you want, a horizontal cut of the dendrogram.

Betweenness-based approach



This algorithm provides a *hierarchical decomposition* of the communities in the graph.

The Radicchi approach

- Instead of calculating the global betweenness again and again, (RADICCHI *et al.*, 2004) proposed to remove edges belonging to loops of length 3 or 4 (think of weak links in the sense of Challenge 1).

Similarity-based approaches

- There are many *similarity measures* (🇫🇷 *mesure de similarité*) that can be defined between vertices, based on external criteria (metadata, etc.).
- We will concentrate on two types of similarity measures based solely on the data provided by the graph itself (vertices and edges): *structural equivalence* and *regular equivalence*.
- The first, *structural equivalence* (🇫🇷 *équivalence structurelle*) considers the number of common neighbors of two vertices. We will examine three such measures: *cosine similarity*, *Pearson coefficients* and *Euclidean distance*.

Cosine similarity


- As already seen, the number of common neighbors between v_i and v_j is $n_{ij} = \sum_k A_{ik}A_{kj}$, n_{ij} being a cell of A^2 .
- If we consider A_{ik} and A_{kj} as vectors then we can take their cosine, this gives us *cosine similarity* (🇫🇷 *mesure cosinus*):

$$\sigma_{ij} = \frac{\sum_k A_{ik}A_{kj}}{\sqrt{\sum_k A_{ik}^2} \sqrt{\sum_k A_{kj}^2}} = \frac{n_{ij}}{\sqrt{k_i k_j}}$$

(the latter equality when A has binary values).

- Cosine similarity takes values between 0 and 1.

Pearson correlation coefficients

- We can, once again, reason by comparing the existing with the expected.
- If v_i and v_j have resp. k_i and k_j neighbors, how many common neighbors should we expect them to have?
- If we simplify, we get $k_i k_j / n$.
- We this we can write the number of common neighbors minus the expected one as $\sum_k A_{ik} A_{kj} - \frac{k_i k_j}{n} = \sum_k (A_{ik} - \langle A_i \rangle)(A_{jk} - \langle A_j \rangle)$ where $\langle A_i \rangle$ is the mean of elements of the i th row of the matrix.
- This is n times the covariance of the two rows of A .
- We can normalize it to get *Pearson correlation coefficients*  *coefficient de correlation de Pearson*:

$$r_{ij} = \frac{\sum_k (A_{ik} - \langle A_i \rangle)(A_{jk} - \langle A_j \rangle)}{\sqrt{\sum_k (A_{ik} - \langle A_i \rangle)^2} \sqrt{\sum_k (A_{jk} - \langle A_j \rangle)^2}}.$$

- Pearson correlation takes values between -1 and +1.

Euclidean distance

- A third way of measuring structural equivalence is the *Euclidean distance* (distance euclidienne): the number of neighbors that differ between two vertices:

$$d_{ij} = \sum_k (A_{ik} - A_{jk})^2,$$

which can be normalized by dividing by the maximum $(k_i + k_j)$:

$$d'_{ij} = \frac{\sum_k (A_{ik} - A_{jk})^2}{k_i + k_j} = 1 - 2 \frac{n_{ij}}{k_i + k_j}.$$

Regular equivalence

- The notion of structural equivalence is quite local: to be similar two vertices must be quite close in the graph.
- We define *regular equivalence* (français: *équivalence régulière*) in a recursive way: *similar (in the sens of regular equivalence) vertices are those that have neighbors which are themselves similar.*
- To do this we follow the same argumentation as for Katz and PageRank: similarity σ_{ij} can be defined as a function of the similarity of neighbors $\alpha \sum_{kl} A_{ik} A_{kl} \sigma_{kl}$, to which we add *self-similarity* δ_{ij} .
- This can be simplified by $\sigma_{ij} = \alpha \sum_k A_{ik} \sigma_{kj} + \delta_{ij}$ (Katz) and $\sigma_{ij} = \frac{\alpha}{k_i} \sum_k A_{ik} \sigma_{kj} + \delta_{ij}$ (PageRank).
- In algebraic terms, the latter is $\sigma = (D - \alpha A)^{-1} D$, where D is the diagonal matrix of degrees.

Hierarchical clustering

- To return to community detection, we need to extend vertex similarity by *vertex group similarity*.
- There are (at least) three ways to achieve this:
 - ① *single-linkage clustering* (🇫🇷 *clustering à lien simple*) where we take as similarity of two groups the maximum of similarities of vertices,
 - ② *complete-linkage clustering* (🇫🇷 *clustering à lien complet*) where we take as similarity of two groups the minimum of similarities of vertices,
 - ③ *average-linkage clustering* (🇫🇷 *clustering à lien moyen*) where we take as similarity of two groups the average of similarities of pairs of vertices.

Hierarchical clustering

- Here is the *hierarchical clustering algorithm* (🇫🇷 *algorithme de clustering hiérarchique*):
 - 1 choose a similarity measure and calculate it on all vertex pairs,
 - 2 consider each vertex as a singleton group,
 - 3 find the pair of groups of highest similarity and merge them,
 - 4 recalculate similarity between groups and restart step 3, until only one group is left,
 - 5 represent all steps in a dendrogram,
 - 6 select, depending on the number and size of communities you want, a horizontal cut of the dendrogram.
- Hierarchical clustering is good at finding cores of groups but not very good in attaching peripheral vertices to groups.

The Louvain method

- (BLONDEL *et al.*, 2008) describe an interesting scaling-down community detection method.
- This method operates on edge-weighted graphs: the definition of modularity is still

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta_{c_i c_j}$$

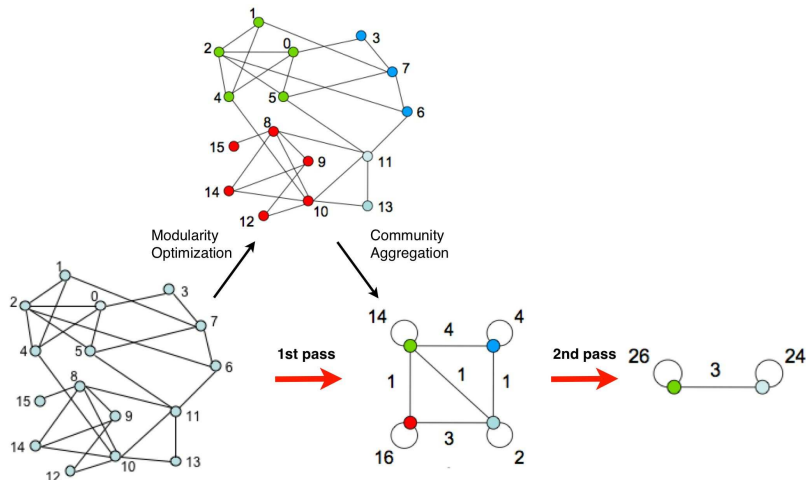
except that now A is the matrix of weights of edges and k_i is the sum of weights of edges attached to v_i .

- The algorithm proceeds in two steps.
- Step 1.
 - 1 start by considering vertices as being singleton groups,
 - 2 for every v_i take each neighbor v_j and calculate the gain of modularity of attaching v_j to the group of v_i , make the move only if modularity is > 0 ,
 - 3 stop when a local modularity maximum is achieved, i.e., when no vertex move can improve modularity.

The Louvain method

- Step 2. Merge all vertices of a group in a single vertex. Weights of edges between groups become weights of edges between the new vertices. Edges between vertices of a given group become self-loops weighted by the sum of weight of intra-group edges.
- Continue until a single group only remains. Represent the different steps in a dendrogram.
- This algorithm is very fast and can be applied to very large networks.
- It has been implemented for the Python `networkX` package:
<http://perso.crans.org/aynaud/communities/>

The Louvain method



(BLONDEL *et al.*, 2008)