Social network analysis: community detection

Donglei Du
(ddu@unb.edu)

Faculty of Business Administration, University of New Brunswick, NB Canada Fredericton
E3B 9Y2

October 20, 2016

Table of contents

Section 1

An introduction

Introduction

- Community structure: a cohesive group of nodes that are connected "more densely" to each other than to the nodes in other communities.
- Community structures are quite common in real networks.
- Being able to identify these sub-structures within a network can provide insight into how network function and topology affect each other
- Finding communities within an arbitrary network can be a computationally difficult task (it is related to clustering in data mining or machine learning with a key difference though).
- Despite these difficulties, however, several methods for community finding have been developed and employed with varying levels of success.
- The evaluation of algorithms, to detect which are better at detecting community structure, is still an open question.

Quantify modularity of a community [Newman and Girvan, 2004] I

partition = division

c = partition ids.

c_i = com/group of v_i

- Modularity is the fraction of the edges that fall within the given groups minus the expected such fraction if edges were distributed at random.

$\delta_{ij} = \delta(c_i, c_j)$

- Formally, given a particular division of a network into ℓ communities $\{c_1, \ldots, c_\ell\}$, the modularity of this division is

modularity quality

Avg

B_{ij}

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j) = \frac{1}{2m}\sum_{ij} B_{ij}\delta(c_i, c_j), \qquad (1)$$

=1 same com

=0 diff com

all edges in community

1 = edge

0 = not

E[] edges in-com random graph (unif edge dist) wit m edges total

k_i = deg

k_j deg

prob of edge ij = $\frac{k_i \cdot k_j}{2m}$

where

- m is the number of edges;
- A is the adjacency matrix;
- ? k_i is the degree of node i;
- c_i is the type (community);

$= \sum_{ij} \frac{k_i \cdot k_j}{2m}\delta_{ij}$

$2m = \sum_i k_i$

Quantify modularity of a community [Newman and Girvan, 2004] II

$$E[\#\text{edges in random graph}] = \sum_{ij} \text{prob}(\text{edge } ij) = \sum \frac{k_i k_j}{2m}$$

Same com   diff com

- δ is the ~~Kronecker delta~~ such that $\boxed{\delta(x, y) = 1}$ if $x = y$ and $\boxed{0 \text{ otherwise}}$;
- the modularity matrix B has elements

$$\boxed{B_{ij} = A_{ij} - \frac{k_i k_j}{2m}.}$$

- Note that

$$\sum_j B_{ij} = 0, \ \forall i$$

- Q is strictly less than 1, takes positive values if there are more edges between vertices of the same type than we would expect by chance, and negative ones if there are less.

$$\sum_j B_{ij} = \sum_j \left[A_{ij} - \frac{k_i k_j}{2m}\right] = \sum_j A_{ij} - k_i \underbrace{\sum_j \frac{k_j}{2m}}_{=1} = k_i - k_i \cdot \underbrace{\frac{\sum_j k_j}{2m}}_{=1} = 0$$

Why (1)? I

- The modularity (1) is the difference between two terms:
  - The fraction of number of edges of the same type for the given division is equal to
  $$\frac{1}{m}\sum_{i,j}\delta(c_i,c_j) = \frac{1}{2m}\sum_{ij}A_{ij}\delta(c_i,c_j)$$
  - The total number of edges of the same type for a random network is equal to
  $$\frac{1}{2m}\sum_{ij}\frac{k_ik_j}{2m}\delta(c_i,c_j)$$
    - Consider a particular edge attached to vertex i with degree k_i.
    - There are by definition $2m$ ends of edges in the entire network, and the chances that the other end of our particular edge is one of the k_j ends attached to vertex j is thus $k_j/2m$ if connections are made purely at random.

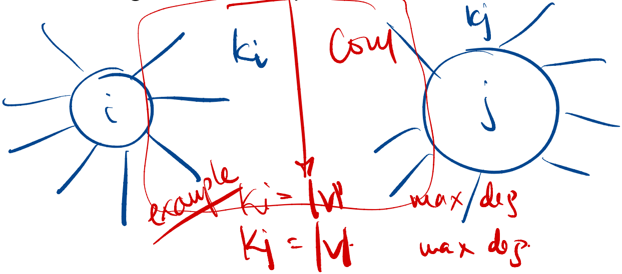
Why (1)? II

$m = \#$ edges in graph.

prob(edge) $\cdot 1$ + prob(no edge) $\cdot 0$

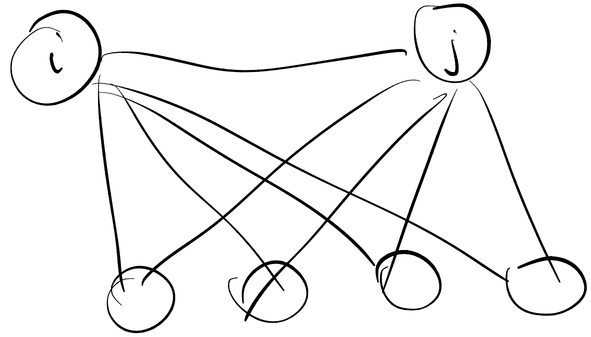
?

- Counting all k_i edges attached to i, the total expected number of edges between vertices i and j is then $k_i k_j / 2m$, leading to the desired expected number of edges between all pairs of vertices of the same type.



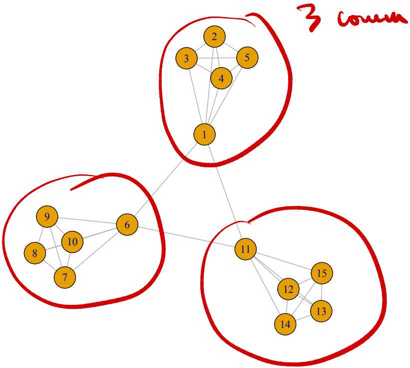
$$\frac{k_i \cdot k_j}{2m} = \frac{V^2}{2m}$$

incorrect?!?


i
j

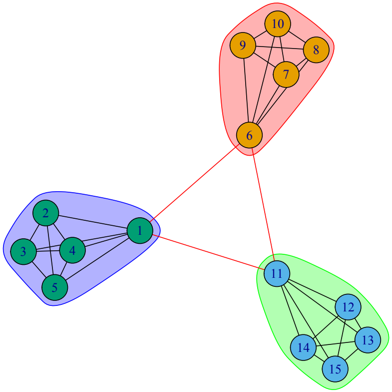
Modularity: an example

```
##
## Attaching package:  'igraph'
## The following objects are masked from 'package:stats':
##
##    decompose, spectrum
## The following object is masked from 'package:base':
##
##    union
```



Modularity: an example

```
## [1] 0.5757576
## [1] 0.5757576
```



Section 2

Community-detection algorithms

Community-detection algorithms I

- Edge-betweeness: The Girvan-Newman method slow
  - Divisive method: starts with the full graph and breaks it up to find communities.
  - Too slow for many large networks (unless they are very sparse), and it tends to give relatively poor results for dense networks.
- Some other methods (See igraph manual)
  - fastgreedy.community: modularity optimization algorithm for finding community structure
  - label.propagation.community: near linear time algorithm
  - leading.eigenvector.community: calculating the eigenvector of the modularity matrix for the largest positive eigenvalue and then separating vertices into two community based on the sign of the corresponding element in the eigenvector.
  - multilevel.community: modularity measure and a hierarchial approach
  - optimal.community: Graphs with up to fifty vertices should be fine, graphs with a couple of hundred vertices might be possible.
  - spinglass.community: spin-glass model and simulated annealing.

fast approx

hierc. clustering

Community-detection algorithms II

- walktrap.community: random walks

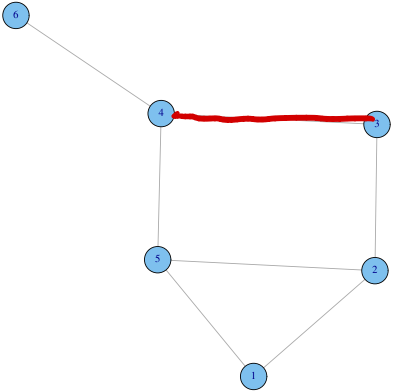
Subsection 1

The Girvan-Newman betweeness method for graph partition [Newman and Girvan, 2004]

Edge Betweeness

= importance of edge = proportion of SP that pass through edge

"Suez canal"



- The edge betweenness for edge e:

$$\sum_{s,t \neq v} \frac{\sigma_{st}(e)}{\sigma_{st}},$$

where

- σ_{st} is total number of shortest paths from node s to node t and $\sigma_{st}(e)$ is the number of those paths that pass through e.
- The summation is $n(n-1)$ pairs for directed graphs and $n(n-1)/2$ for undirected graphs.

- The edge betweenness for the graph on the left:

Edge	Betwenness
12	2
15	3
23	3.5
25	2.5
34	3.5
45	5.5
46	5

prop of SP that pass through edge 34

pair 62: 2 SP

1 pass through edge 34

How to find the edge betweeness in the example?

- For example: for edge 23, the $n(n-1)/2 = 6(6-1)/2 = 15$ terms in the summation in the order of 12, 13, 14, 15, 16, 23, 24, 25, 26, 34, 35, 36, 45, 46, 56 are

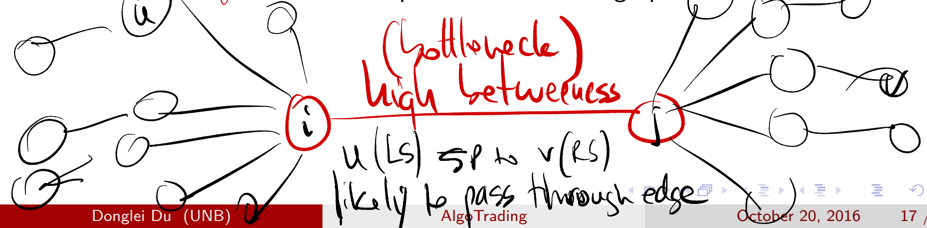
$$\frac{0}{1}+\frac{1}{1}+\frac{0}{1}+\frac{0}{1}+\frac{0}{1}+\frac{1}{1}+\frac{1}{2}+\frac{0}{1}+\frac{1}{2}+\frac{0}{1}+\frac{1}{2}+\frac{0}{1}+\frac{0}{1}+\frac{0}{1}+\frac{0}{1}.$$

- Here the denominators are the number of shortest paths between pair of edges in the above order and the numerators are the number of shortest paths passing through edge 23 between pair of edges in the above order.

	1	2	3	4	5	6
1		0/1	1/1	0/1	0/1	0/1
2			1/1	1/2	0/1	1/2
3				0/1	1/2	0/1
4					0/1	0/1
5						0/1
6						

Edge betweenness based community structure detection

- The idea is that it is likely that edges connecting separate modules have high edge betweenness as all the shortest paths from one module to another must traverse through them.
- So if we gradually remove the edge with the highest edge betweenness score we will get a hierarchical map, a rooted tree, called a dendrogram of the graph.
    - The leafs of the tree are the individual vertices
    - The root of the tree represents the whole graph



The Girvan-Newman method for graph partition [Newman and Girvan, 2004]

Step 1. Find the edge of highest betweenness - or multiple edges of highest betweenness, if there is a tie - and remove these edges from the graph.

- This may cause the graph to separate into multiple components. If so, this is the first level of regions in the partitioning of the graph.

Step 2. Recalculate all betweennesses, and again remove the edge or edges of highest betweenness.

repeat

- This may break some of the existing components into smaller components; if so, these are regions nested within the larger regions.

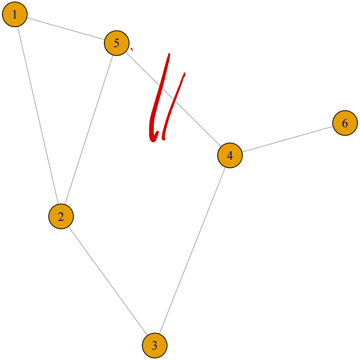
high betweeness recomputed!

Step 3 Proceed in this way as long as edges remain in graph, in each step recalculating all betweennesses and removing the edge or edges of highest betweenness.

The Girvan-Newman method for graph partition [Newman and Girvan, 2004]

- The method gives us only a succession of splits of the network into smaller and smaller communities, but it gives no indication of which splits are best.
- One way to find the best split is via the the modularity concept (1).

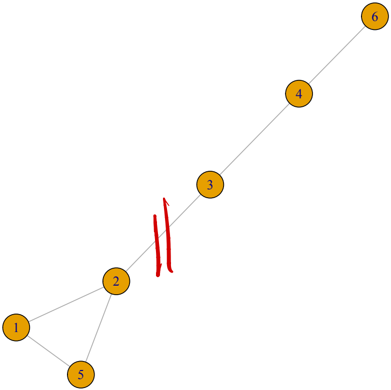
An example via package igraph: Illustration of the Girvan-Newman method




```
## [1] 2.0 3.0 3.5 2.5 3.5 5.5 5.0
```

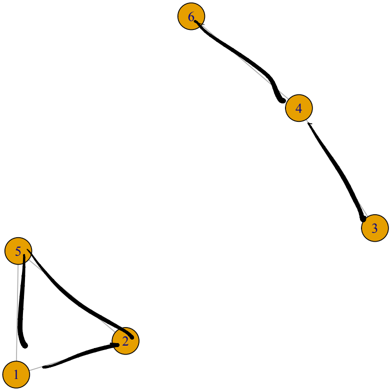
Delete edge (5,4) high bet.

```
## [1] 4 1 9 4 8 5
```



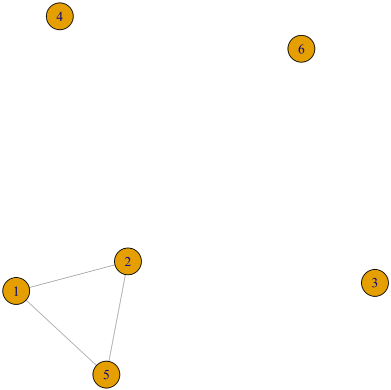
Delete edge (3,2) high between

```
## [1] 1 1 1 2 2
```



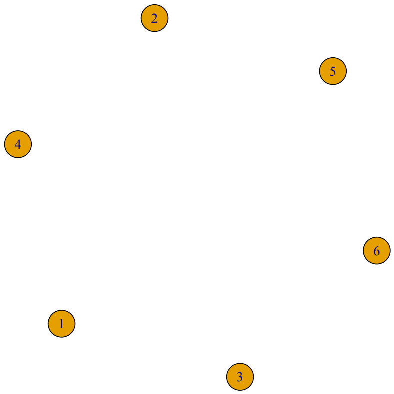
Delete edge (4,3) and (6,4)

```
## [1] 1 1 1
```

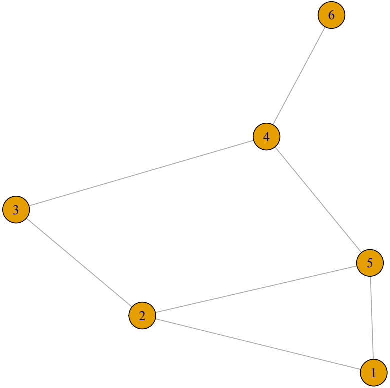


Delete edge (2,1), (5,1) and (5,2)

```
## numeric(0)
```

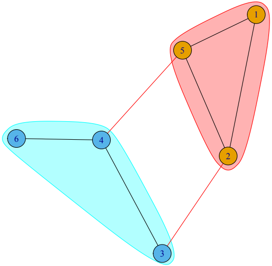


An example via package igraph: the edge.betweenness.community() function



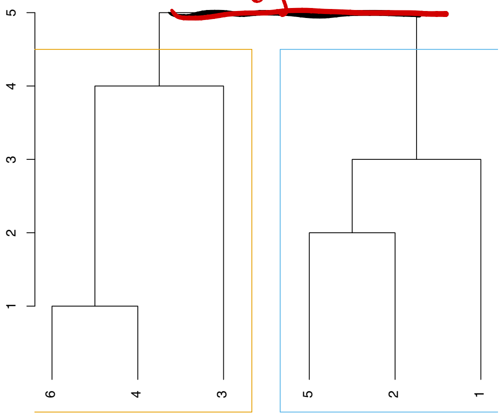
Find community via igraph function edge.betweenness.community()

```
## IGRAPH clustering edge betweenness, groups: 2, mod: 0.2
## + groups:
##   $`1`
##   [1] 1 2 5
##
##   $`2`
##   [1] 3 4 6
##
## Community sizes
## 1 2
## 3 3
## [1] 0.2040816
```



Plot dendrogram for hierarchical methods via igraph function dendPlot()

high betweeness



fast approx
bottom up
hierc. clustering

Subsection 2

The spectral modularity maximization community detection algorithm

The spectral modularity maximization algorithm [Newman, 2006a, Newman, 2006b]: division of a network into just two parts

decompose δ
pairwise $\rightarrow$ unitwise

- Recall (1)

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j) = \frac{1}{2m}\sum_{ij} B_{ij}\delta(c_i, c_j)$$

split

- Introduce a decision variable

$$s_i = \begin{cases} 1, & \text{if node } i \text{ belongs to group 1} \\ -1, & \text{if node } i \text{ belongs to group 2} \end{cases}$$

- Note that

$$\boxed{\delta(c_i, c_j) = \frac{1}{2}(1 + s_i s_j)}$$

$s_i = s_j \Rightarrow \delta = 1$
$s_i \neq s_j \Rightarrow \delta = 0$

- We now have, noting that $\sum_j B_{ij} = 0$, $\forall i$ for the modularity matrix:

$$\boxed{Q = \frac{1}{4m}\sum_{ij} B_{ij} s_i s_j = \frac{1}{4m} s^T B s}$$

semipos def $\Rightarrow$ spectral decmp.

The spectral modularity maximization problem

- Now our problems becomes

all possible "splits" subsets

$$\boxed{\max \left[\frac{1}{4m} s^T B s : s \in \{-1, 1\}^n \right]}.$$

NP-hard.

- This problem in general is NP-hard.
- [Newman, 2006a, Newman, 2006b] propose a heuristic, called the leading.eigenvector algorithm.

of B

Approx

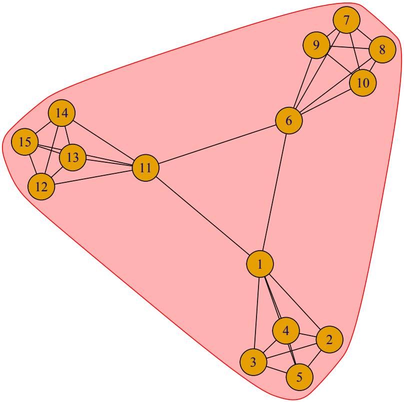
How are about unknown number of communities?

- Simulated annealing: slow and only works for a few hundreds of nodes
- Genetic algorithm: slow and only works for a few hundreds of nodes
- Greedy algorithm: fast and can work for millions of nodes
- ...

Approximation algorithms for modularity maximization

- [Agarwal and Kempe, 2008, Dinh and Thai, 2013]
- More work to be done on approximation algorithms...

An example using package igraph function: leading.eigenvector.community()



References I

Agarwal, G. and Kempe, D. (2008).
Modularity-maximizing graph communities via mathematical programming.
The European Physical Journal B, 66(3):409–418.

Dinh, T. N. and Thai, M. T. (2013).
Community detection in scale-free networks: Approximation algorithms for maximizing modularity.
Selected Areas in Communications, IEEE Journal on, 31(6):997–1006.

Newman, M. E. (2006a).
Finding community structure in networks using the eigenvectors of matrices.
Physical review E, 74(3):036104.

References II

Newman, M. E. (2006b).
Modularity and community structure in networks.
Proceedings of the National Academy of Sciences, 103(23):8577–8582.

Newman, M. E. and Girvan, M. (2004).
Finding and evaluating community structure in networks.
Physical review E, 69(2):026113.