Clustering Algorithms on Graphs
Community Detection

6CCS3WSN-7CCSMWAL
Contents

- Zachary’s famous example
- Community structure
- Modularity
- The Girvan-Newman edge betweenness algorithm
In the beginning: Zachary’s Karate Club

A social network of a karate club studied by Wayne Zachary from 1970 to 1972. The club had 34 members. Based on interactions between members outside the club, Zachary found 78 social links.

During the study a conflict arose between the administrator ”John A” and instructor ”Mr. Hi”, which led to the split of the club into two. Half of the members formed a new club around Mr. Hi, members from the other part found a new instructor or gave up karate.

Before the split each side tried to recruit adherents of the other party. Thus, communication flow had a special importance and the initial group would likely split at the ”borders” of the network.

Zachary used an algorithm (min-cut, max-flow) to correctly predict each member’s decision except member 9, who went with Mr. Hi instead of John A.
Network of the Zachary Karate Club. Node 1 stands for the instructor, (Mr. Hi) node 34 for the administrator (John A). The colours show the true final two groups.

Figure and text from https://en.wikipedia.org/wiki/Zachary’s_karate_club
A network is said to have community structure if the nodes of the network can be meaningfully grouped into (potentially overlapping) sets of nodes such that each set of nodes is densely connected internally.

In the case of non-overlapping communities, the principle is that the network divides naturally into groups of nodes with dense connections internally and sparser connections between groups.

The term *Community* is based on the idea that pairs of nodes are more likely to be connected by an edge if they are both members of the same community, and less likely to be connected if they are not.
See https://en.wikipedia.org/wiki/Community_structure from which this summary and figure is taken.
When we do clustering on graphs we call it community detection.

The problem is more complex than clustering vector data because the distance between data points (vertices) is not measured by Euclidean distance but by how far away they are in graph distance.

Graph distance $d(a, b)$ length of the shortest path from vertex $a$ to vertex $b$.

The algorithms have to reveal and exploit the implicit structure of the graph.

Typical algorithms include:

1. Minimum–cut method (as used by Zachary)
2. Modularity maximization (optimal or heuristic)
3. Girvan-Newman algorithm (edge betweenness)

Typically, the community-ness of a group of vertices is measured by the density of edges between them. But how much density makes a community? If we generate a graph randomly some groups of vertices will have higher edge density than others. Are they communities?
Karate club

Karate graph

Optimal clusters

Edge betweenness
Some explanation:

- The Karate graph. The club president and the instructor had an argument. An edge shows who talked to who outside the club.
- The algorithm decides how many communities is the correct number for this graph. The number of communities was not part of the input (unlike $k$-means).
- The red edges are between-clusters and the black ones within cluster.
How to measure communities?

- How can we (try to) measure the quality of the clusters (communities) found by the algorithm. Why should we think/believe that these answers are meaningful.
- The graph itself is a community, and the individual vertices are (personal) communities. At what point do we stop dividing the graph up into smaller and smaller clusters, and say ’that’s enough’?
- A community is something which has an inside and an outside, and the linkage between the two of them differs. Inside the community things are more connected.
- We need a numeric criteria of inside-ness and outside-ness. Something we can try to maximize.
- A quantity called *modularity* is often used to decide the best number of communities.
- **Modularity** is often used to decide the best number of communities.
- The figure 'Optimal clusters' was obtained by maximizing the modularity over all possible clusters of all sizes to arrive at the 5 communities shown. Maximum modularity was 0.42.
- The figure 'Edge betweenness' was obtained by recursively removing the edge with the maximum remaining edge betweenness and checking the modularity at each step. The maximum modularity was about 0.40. The index is the number of broken edges.
- The betweenness algorithm divides the graph by deleting edges. The dendrogram is formed from the top down. The first pair of clusters split along the cut between the yellow and blue (3,9,31) and the pink (8, 14,20). This is nearer the 'correct' (2 group) answer for Zachary’s Karate club.
require(igraph)
kарате <- make_graph("Zachary")
plot(kарате, main="Karate graph")

ebc<-cluster_edge_betweenness(kарате, bridges=TRUE)
plot(ebc,kарате, main="Edge betweenness")

yyy=as.hclust(ebc, hang = -1, use.modularity = FALSE)
plot(yyy,main="Edge betweenness dendrogram")

modularity <- sapply(0:ecount(kарате), function(i){
  g2 <- delete.edges(kарате, ebc$removed.edges[seq(length=i)])
  cl <- clusters(g2)$membership
  modularity(kарате,cl)
})

# we can now plot all modularities
plot(modularity, pch=20)
Inside and outside edge density
Modularity

The concept of modularity is fundamental to community detection.

- Given a group, the modularity $Q$ of the group is the difference between the number of edges within the group and the expected number of edges within the group. This is then scaled to give a number between minus one and one ($-1 \leq Q \leq 1$).

- The more positive the value of $Q$ the more significant the grouping. The entire graph (as one community) has $Q = 0$.

- The expected number of edges within a group. This is calculated based on the null hypothesis that the edges were formed randomly. i.e. No community structure. Random edges? e.g. $G(n, p)$ each edge is inserted with probability $p$. 
Modularity

- We need to take into account the existing degrees of each vertex. For shorthand we use $d_u$ for the degree of vertex $u$ (other notations include $d(u)$, $k_u$).

Let $m$ be the number of edges in the graph. For any pair of vertices, the probability $P_{uv}$ of edge $uv$ is defined as

$$P_{uv} = \frac{d_u d_v}{2m}.$$  

- Let $G$ be a graph with vertex set $V$, $|V| = n$ and edge set $E$, where $|E| = m$. $A = [A_{uv}]$ be the $n \times n$ adjacency matrix of $G$, so that $A_{uv} = 1$ if $uv$ is an edge of $G$ and $A_{uv} = 0$ otherwise. Suppose we have $k$ groups $S = \{S_1, \ldots, S_k\}$ then $Q$ is defined by

$$Q(S) = \frac{1}{2|E|} \sum_{i=1}^{k} \sum_{u,v \in S_i} (A_{uv} - P_{uv}).$$

- In other words we add up all the edges in the group, and subtract off the number of edges we expect to see.
Remember that $\sum_u d_u = 2m$ (sum of the degrees is twice the number of edges). So

$$(2m)^2 = \left(\sum_u d_u\right)^2 = \sum_{u \in V} \sum_{v \in V} d_u d_v = \sum_{(u,v) \in E} d_u d_v.$$  

Rephrasing this: The total degree $2m$ can be written as

$$2m = \sum_{(u,v) \in E} \frac{d_u d_v}{2m}$$

In order to make probability $P_{uv}$ add up to the total degree $2m$, the sum is taken over all ordered pairs e.g. $(u, u), (u, v), (v, u)\ldots$
Example

```r
G = make_empty_graph(n=4, directed=FALSE)
G = add.edges(G, c(2, 3, 2, 4))
plot(G)
modularity_matrix(G, V(G))
modularity(G, c(1, 1, 1, 1))
```

# The output

```r
> modularity_matrix(G, V(G))
[1,]  0  0.0  0.00  0.00
[2,]  0 -1.0  0.50  0.50
[3,]  0  0.5 -0.25 -0.25
[4,]  0  0.5 -0.25 -0.25
```

#

```r
> modularity(G, c(1, 1, 2, 2))
[1] -0.5
```
Example. \( V = \{1, 2, 3, 4\}, \; E = \{23, 24\}, \; d_1 = 0, \; d_2 = 2, \; d_3 = 1, \; d_4 = 1. \) The \( P(G) \) matrix etc. Calculate once for the whole graph.

\[
P_{1,2} = \frac{0 \cdot 2}{4} = 0, \quad P_{2,3} = P_{2,4} = \frac{2 \cdot 1}{4} = \frac{1}{2}, \quad P_{3,4} = \frac{1 \cdot 1}{4} = \frac{1}{4}
\]

\[
P_{2,2} = \frac{2 \cdot 2}{4}, \quad P_{3,3} = \frac{1 \cdot 1}{4}.
\]

\[
A = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0
\end{pmatrix}, \quad P = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 1 & 1/2 & 1/2 \\
0 & 1/2 & 1/4 & 1/4 \\
0 & 1/2 & 1/4 & 1/4
\end{pmatrix},
\]

The modularity matrix (see previous output)

\[
A - P = \begin{pmatrix}
1 & 2 & 3 & 4 \\
0 & 0 & 0 & 0 \\
0 & -1 & 1/2 & 1/2 \\
0 & 1/2 & -1/4 & -1/4 \\
0 & 1/2 & -1/4 & -1/4
\end{pmatrix}
\]
Try a few clusters

Let's try \( S = \{1, 2\}, \{3, 4\} \).

What are we adding up? The parts of \( A - P \) that fit our partition, i.e.

\[
\begin{pmatrix}
1 & 2 & 3 & 4 \\
1 & 0 & 0 & 0 \\
2 & 0 & -1 & 0 \\
3 & -1/4 & -1/4 & 0 \\
4 & -1/4 & -1/4 & 0
\end{pmatrix}
\]

So \( Q(S) = \frac{1}{4}(-2) = -0.5 \). (see previous slide)

Best answer? \( S = \{1\}, \{2, 3, 4\} \)

\[
\begin{pmatrix}
1 & 2 & 3 & 4 \\
1 & 0 & 0 & 0 \\
2 & -1 & 1/2 & 1/2 \\
3 & 1/2 & -1/4 & -1/4 \\
4 & 1/2 & -1/4 & -1/4
\end{pmatrix}
\]

\( Q(S) = 0 \)

Exercise. Try \( S = \{1, 3\}, \{2, 4\} \). Ans = -0.125.
Answer to exercise

Ques. Modularity of partition $S = \{1, 3\}, \{2, 4\}$
Ans. $Q(S) = -0.125$

Explanation. First of all lets check what R thinks the answer is.
Use the code a few pages back with cluster vector $(1,2,1,2)$ i.e. the clusters for vertices $(1,2,3,4)$

modularity(G,c(1,2,1,2))
[1] -0.125

Rearrange the relevant parts of $A - P$ to fit our partition, i.e.

\[
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & -1/4 & 0 \\
0 & -1/4 & 1/2 & 1/2 \\
0 & 1/2 & -1/4 & 0
\end{pmatrix}
\]

Total of entries $-1/4 - 1 + 1/2 + 1/2 - 1/4 = -1/2$
So $Q(S) = \frac{1}{4}(-1/2) = -1/8 = -0.125$. 

The Girvan-Newman Algorithm (GNA)

Summary:

- Uses edge betweenness (see lecture on vertex centrality)
- Deleting edges one by one causes the graph to disconnect
- GNA: Delete the most central remaining edge, checking the modularity as we go
- Stop clustering at the modularity maximum
- At any step the residual graph is a collection of connected components
Edge betweenness

For a given graph $G = (V, E)$, carry out the following steps for each pair of vertices in the same component:

(i) For a given pair of vertices $u, v$ assign one unit of flow in total.

(ii) Find the number, $k(u, v)$, of shortest paths from $u$ to $v$.

(iii) Assign $1/k(u, v)$ units of the flow to each shortest path from $u$ to $v$.

(iv) For each shortest $(u, v)$-path, record the edges in the path.

After all this is finished:
For each edge $e \in E$ count up how much flow goes through the edge $e$ adding over all shortest paths between all pairs of vertices $u, v \in V$ which use the edge $e$. 
Girvan-Newman algorithm for community detection

Repeat steps (1), (2), (3) until no edges left:

(1) Calculate the betweenness of all existing edges in the graph
(2) Remove the edge(s) with the highest betweenness
(3) If new components formed these are potential communities

Apart from doing all this, the problem is how to make the process efficient if possible. This is not given here. Crudely, the complexity is $O(m^2n^2)$. It takes $O(m)$ for BFS to find shortest path from a given vertex $u$ to a vertex $v$, this is repeated for $O(n^2)$ pairs, and $m$ edge deletions.
Example 1

(a) Step 1

(b) Step 2

(c) Step 3

For more detail see
https://www.ismll.uni-hildesheim.de/lehre/cmie-11w/script/lecture5.pdf
• Girvan-Newman partitions correctly
  – exception: node 9 assigned to region of 34 (left part)
  – at the time of conflict, node 9 was completing a four-year quest to obtain a black belt, which he could only do with the instructor (node 1)
Shortest paths: e.g. \( k(1, 2) = 2, \ k(1, 5) = 1 \). 

The betweenness of the edges:

<table>
<thead>
<tr>
<th>Edge</th>
<th>17</th>
<th>13</th>
<th>72</th>
<th>23</th>
<th>34</th>
<th>46</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Betweenness</td>
<td>3.5</td>
<td>6.5</td>
<td>3.5</td>
<td>6.5</td>
<td>12</td>
<td>6</td>
<td>6</td>
</tr>
</tbody>
</table>

Edge 13. The betweenness is 6.5. Shortest paths from vertex 1 to 3, 4, 5, 6 give 1 each. Paths from vertex 7 to 3, 4, 5, 6 via 13 give and \( 1/2 \) each. The path 132 from vertex 1 to vertex 2 gives \( 1/2 \).

After deletion of edge (3, 4) betweenness of the edges is

<table>
<thead>
<tr>
<th>Edge</th>
<th>17</th>
<th>13</th>
<th>72</th>
<th>23</th>
<th>46</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>Betweenness</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
The best modularity was 0.32. Two communities
The height is the number of partition–into–components moves.
Delete edge 17 and 32 in one 'step' to break 7123 into 71, 32
There are some non-explicit decisions in the R-code