14: Clique Finding

Machine Learning and Real-world Data (MLRD)

Andreas Vlachos
(based on slides by Simone Teufel)

Last session: betweenness centrality

- You implemented betweenness centrality.
- This let you find “gatekeeper” nodes in the Facebook network.
- We will now turn to the task of finding clusters in networks.

Clustering and Classification

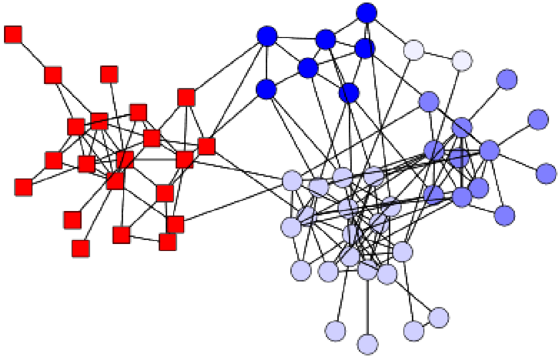
- Clustering: automatically grouping data according to some notion of closeness or similarity.
- Classification (e.g., sentiment classification): assigning data items to predefined classes.
- Clustering: groupings can emerge from data, unsupervised, not pre-defined.
- Clustering for documents, images etc: anything where there's a notion of similarity between items.
- Most famous technique for hard clustering is k-means: very general (also variant for graphs).
- Also soft clustering: clusters have graded membership

Agglomerative vs. divisive clustering

- agglomerative clustering works bottom-up.
- divisive clustering works top-down, by splitting.
- Newman-Girvan method — a form of divisive clustering.
- Criterion for breaking links is edge betweenness centrality.

Dolphin data: different clustering layers

- squares vs circles: first split
- shades of blue: further splits

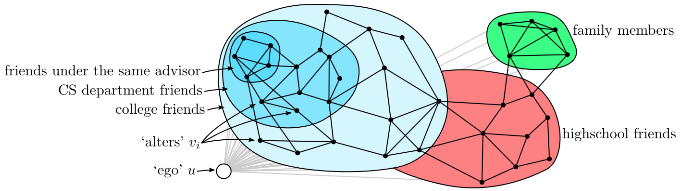


Newman and Girvan (2004)

Facebook circles dataset: McAuley and Leskovec (2012)

- Profile and network data from 10 Facebook ego-networks.
- An ego network is a network emanating from one person.
- Circles are defined as Facebook friends in a particular social group.
- Gold-standard circles are manually identified by the egos themselves.

Facebook Circles task



- Complete network consists of 4,039 nodes in 193 circles.
- Average: 19 circles per ego, each circle with average of 22 alters.
- You will cluster only a small network derived from one ego.

Doing the full Facebook Circles task

25% of circles are contained completely within another circle
50% overlap with another circle
25% have no members in common with any other circle

Requires more sophisticated methods than Newman-Girvan:

- Nodes may be in multiple circles, so we need soft clustering.
- Use sociological/demographic data from outside the network data.

Evaluating simple clustering

- Assume data sets with gold standard or ground truth clusters.
- But: unlike classification, we don't have labels for clusters, number of clusters found may not equal to the number of true classes.
- purity: assign label corresponding to majority class found in each cluster, then count correct assignments, divide by total elements (cf accuracy).
  `http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html`
- But best evaluation (if possible) is extrinsic: use the system to do a task and evaluate that.
- More on evaluation in the next lecture

Newman-Girvan method

while number of connected subgraphs $<$ specified number of clusters (and there are still edges):

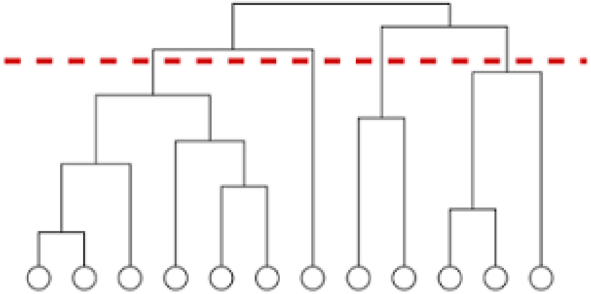
1. calculate edge betweenness for every edge in the graph
2. remove edge(s) with highest betweenness
3. recalculate number of connected components

Note:

- Treatment of tied edges: either remove all (today) or choose one randomly.

Newman-Girvan Method: Stopping Criterion

- The image below is called a dendrogram.
- Either: stop at prespecified level (tick).
- Or: complete process and choose best level by ‘modularity’ (Newman, 2004; starred tick).



Newman and Girvan (2004)

Edge betweenness centrality

- Previously: $\sigma(s,t|v)$ — the number of shortest paths between s and t going through node v.
- Now: $\sigma(s,t|e)$ — the number of shortest paths between s and t going through edge e.
- Algorithm only changes in the bottom-up (accumulation) phase.

Final Task

Task 12:

- Determine connected components
- Change code for betweenness centrality (from node to edge)
- Implement the Newman-Girvan method to discover clusters in the network provided.

Code for determining connected components

- Today’s graph is disconnected: there are five connected components.
- Finding connected components: depth-first search, start at an arbitrary node and mark the other nodes you reach.
- Repeat with unvisited nodes, until all are visited.
- Implementation hint: depth-first, so use recursion (the program stack stores the search state).