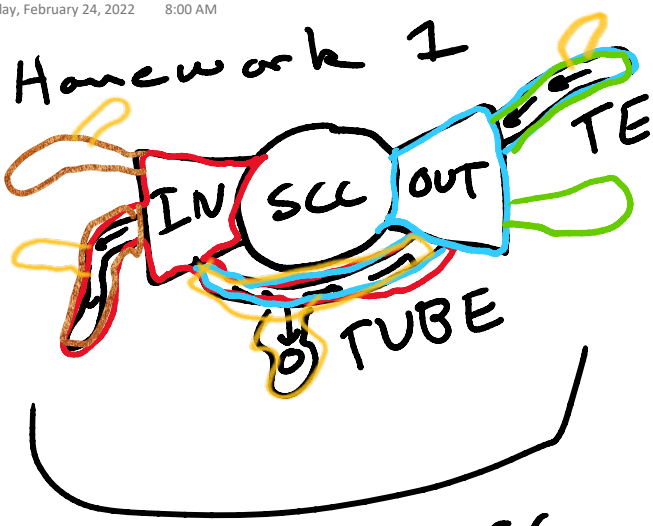
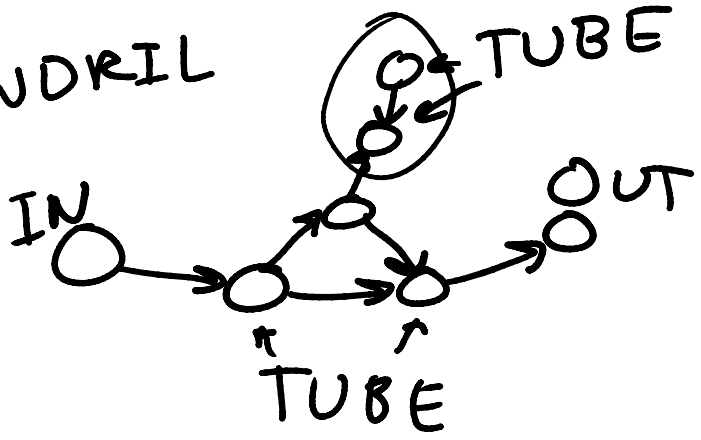
Lecture 10 - Community Detection

Thursday, February 24, 2022 8:00 AM

Homework 1



WCC



IN: 10k
OUT: 3k
SCC: 67k
TUBE: ~50
TENDRIL: ~1100

Community Detection

Basic definition: identify relatively dense regions in a network

We consider communities in various ways:

→ Friend groups

→ Groups w/ shared interest

→ Basic interactions

Classic dataset:

Zachary Karate club

- Back in the day there was a Karate Club
- Network defined based on interactions outside of the club
- At one point in time:
  * The club split between two factions
  * One was club president and other was instructor
  * Interesting aspect: the split followed almost perfectly along the topological boundary defined on the network above

defined on the network above

→ Fundamental hypothesis of community detection:

Communities can be defined or described fully via network topology

What we'll discussing:

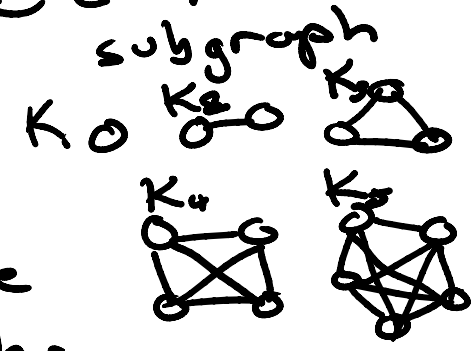
- How to define "density"
- Algorithms to solve C.D.
- How to evaluate C.D. outputs
- Related problems (partitioning)

Density definitions for communities

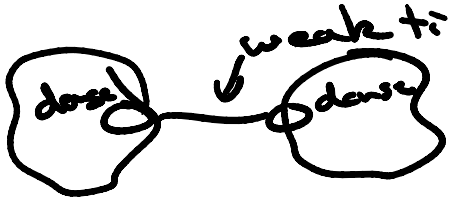
Not clear cut in terms of how we might define "density"

What about cliques?

≈ complete subgraph



- Can't get more dense than a clique
- Issue: large cliques are rare in most networks
- Issue: computationally difficult
  - * NP-complete in general
  - * Equivalent to subgraph match
  - * $O(1.19^n)$ for cliques
- However: triangles can be quite useful (K_3)
  - * density ~ clustering coeffient
  - * complexity not bad $O(|E|^{3/2})$
  - * However, extropolating local clustering to a region is tough



Connectivity

- connected components → not useful
- k-connectivity: more useful
  ↑ but tough to compute (poly. time)

how many vertices to remove to disconnected

- However, connectivity doesn't necessarily directly translate to our notion of density

Better definitions: relative edge density

↳ ratio of internal to external edges of a subgraph

Strong community: $\forall v \in C : d_{int}(v) \geq d_{ext}(v)$

weak community: $\sum_{v \in C} d_{int}(v) \geq \sum_{v \in C} d_{ext}(v)$

Both give explicit measures of density

↳ can also give overall quality measure

↳ but, trivial optimal solution would just have all vertices in same community

just have all vertices in same community

Other better measures: Modularity and conductance (ϕ)

Number of communities

Number of communities will vary based on our density criteria and latent network properties

Challenges:

- Usually unknown - we can only consider topology and not ground truth
- Possible per-vertex groupings given some groups → exponential
- Possible groups → super-exponential

→ solution space is "quite large"

C.D. algorithms have to consider both of the above

* Exact solutions generally infeasible

* Exact solutions generally infeasible
* Heuristics or greedy algorithms used in practice

Community detection algorithms

Types:
- Agglomerative: we combine communities in some way to reach a local maximum for some optimization
- Divisive: we cut communities in some way to reach local maximum for some optimation
- Hierarchical: we create a hierarchical structure of communities by doing the above

Note: communities are often hierarchical in practice

Example:

Note: communities can also overlap
→ adds additional complexity

Label Propagation

- Agglomerative
- Iterate:

  For all $v \in V(G)$:

  $C(v)$ = max over neighbors with ties broken randomly

Pros: simple to implent

O(n) complexity

good results in practice

Cons: Can have bad results

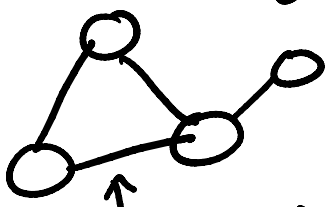
optimal solution is single comm.

hierarchical structure not explicity captures

Louvain Algorithm

Ravasz Algorithm

- Agglomerative
- Iterate until only a single community
  select some (C_i, C_j) pair to merge
  Based on similarity:



$$S_{ij} = \frac{|N(i) \cap N(j)| + a_{ij}}{\min(d(i), d(j)) + 1 - a_{ij}}$$

a_{ij} ← value in adj matrix

↑ merge/contract

$$0 \leq S_{ij} \leq 1$$

Pros: captures full hierarchy
can change our similarity measure

Cons: complexity not great $O(n^3)$
not optimizing a global metric

Girvan-Newman

- Divisive

- Universe
- Iterate:
  Select edge e with highest betweeness centrality and cut it

  communities are connected comps.

Note: we're cutting weak ties

Pros: good results since we use a lot of inherent social network properties / phenomena

Cons: sloooowwwwwww $O(n^3)$