Lecture 10: Networks (Counting on steroids)
Modeling Social Data, Spring 2017
Columbia University

Chris Lam

April 7, 2017

1 On Networks

1.1 History

- '30s: Breaking news: Networks are a thing!
- '60s: Random graph theory: Erdos + Rengi ('59)
  - thought of graphs as math, as objects to be studied
  - high probability: more clustered in one component
  - low probability: more scattered across multiple components
- '70s: Granovetter ('73): Clustering and weak ties
  - The friends of my friends are often friends.

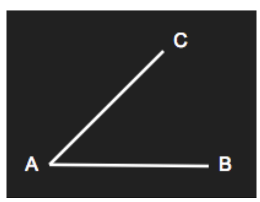

Figure 1: Granovetter: this is forbidden; it's impossible for A-C and A-B to be the case without B-C being a thing as well!

  - Ties can be strong (triadic closure) or weak (bridges).
  - I may not know too well the people who bridge me to other communities.
- '70s - relatively recently: Cross-platform data outside of surveys for social networks isn't lying around, making it hard to study.
- '70s: de Solla Price ('65,'76): Cumulative advantage in citation networks – in many other words:
  - Uneven distribution of attention
  - Popularity begets itself
  - There are a few celebrities and a bunch of nobodies.
- '90s: Watts + Strogatz ('98): Small-world networks
  - Randomly rewired edges of a regular network

  - Bridged the gap between IRL and the completely random graph
  - Featured short path lengths (ie. just a few hops from A to B), triadic closure, and bridging

- '00s: Newman, Barabusi, Watts ('06): Empirical structure from actual data, ie. hairballs
  - Adamic + Glance ('05): Homophily
  - Warning: location of nodes (blogs) may be contrived.
  - Favors the lowest-energy configuration: force-directed, springlike edges that collapse close-together nodes more densely together in parameter space.

1.2 Types of networks

Networks are abstractions of different types of data. We can be handed social (think: Facebook), informational (think: the web, political blogs, citations), activity (think: email), biological, and even geographical (think: roads) networks. It's important not to lose sight of what's being abstracted to a network.

Representations, ie. levels of abstraction

- Undirected

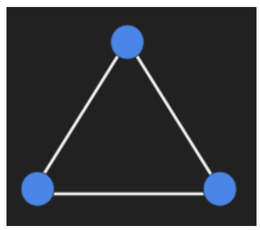

Figure 2: Bidirectional friendship (one would hope)

- Directed

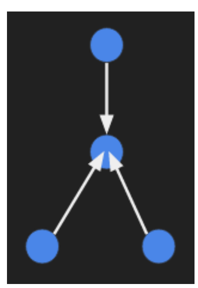

Figure 3: Directed network, eg. followers of a FB page

- Weighted (the old ARPAnet, the OG Internet, whose edge costs varied)
- Metadata: attributes of the nodes and edges themselves
- Ego networks: by changing the threshold for what constitutes a 'meaningful' interaction or relationship, we change what the network looks like. 'All my FB friends', for example, will be much denser than 'my carefully maintained relationships'.

1.3 Data Structures of Networks

- Edge list: storage :) compute :(
  - Compute time $\propto$ number of edges
  - To check if edge is present, requires a big scan, linear through number of edges
- Adjacency matrix: checking edges :) linear algebra :)
  - Storage in a sparse matrix is more efficient
  - The not as big scan: run down the row or column; but this gets less easy for directed graphs because the matrix for these aren't symmetric
  - Compute time $\propto$ number of nodes
- Adjacency list: graph traversal :)
  - Compute time $\propto$ average number of neighbors for all nodes

Descriptive Stats of Networks:

Stat	Definition	Associated algorithm
Degree	# connections a node has	Degree distributions (counting)
Path length	Shortest path between 2 nodes	BFS
Clustering	How many friends of friends are also friends?	Triangle counting
Components	# disconnected parts	Connected components

2 Coding up Networks

- Computing degree distribution (ie. How many nodes have 1,2,etc. neighbors?)

  group by source

  count # destination nodes $\rightarrow$ (source,degree)

  group by degree

  count # source nodes

- Computing path length - sometimes we'll need to logscale to handle distributions for which magnitude comparisons make more sense (think celebrities again)

  BFS: every newly discovered node is at distance, or path length, of one more than the current maximum distance. All pairs' path length $\propto$ # nodes x # edges

  init nodes at ∞

    source dist 0

    curr boundary is source

    new boundary is empty

  explore non-empty boundary

  loop over all nodes in curr boundary

  explore each undiscovered neighbor

    dist $\leftarrow$ curr dist + 1

    add neighbor to boundary

  curr boundary $\leftarrow$ next boundary

  terminate when no more 'next boundary'

- Computing connectedness

  init nodes at ∞

  pick random unreached node (until no unreached nodes left)

  run BFS from that node

    label everything that's reached as one component

- Counting number of mutual friends for every pair $\propto$ d^2; that is, the few celebrities among us will kill our computers.

  We can use an adjacency list - here, it's: i, j1, j2, and j3

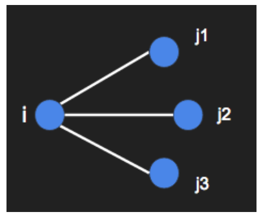

Figure 4: Counting mutual friends

  for each node:

    run over all pairs of its neighbors

    increment count by 1

- Counting triangles: checking if the j's themselves are connected

  Now it makes more sense to use an adjacency matrix, which provides a better memory footprint - else, computation is hell.

  for each node:

    run over all its neighbors

  increment node's count if neighbors are connected

  We can measure how clustered a network is, or how connected a person is, by taking the ratio of number of actual triangles over possible triangles.