Network Analysis

Nishanth Nair

March 30,2020

Motivation

- A **network** is a collection of **entities**, together with a set of **relations** on those entities. Similar to Graphs.
  - Entities: nodes or vertices. Properties could inlude
    - from immediate connections
      - Degree: Number of edges on the node
      - Indegree: How many directed edges are incident on the node
      - Outdegree:How many directed edges originate from the node
    - from entire graph
      - Centrality (betweenness, closeness)
  - Relationships: Ties or edges
    - Directed or undirected
    - Properties could include
      - weight
      - Ranking
      - Type
- Formal notation for a graph, G = (V,E)
  - Vertex set, V = $v_1, v_2 ... v_n$
  - Edge set, $E \subseteq v_i, v_j : v_i, v_j \in V$

Examples of network analysis

- **Terrorist Networks**:
  - Uncloaking Terrorist Networks, written by Valdis E. Krebbs
  - Reconstructed terrorist network of 9-11 attack based on newspaper articles and other media
  - Key takeaways:
    - Very sparse network
    - Hijackers on same team were 2 or more steps away from each other. A strategy for keeping cell members distant from each other, and from other cells, minimizes damage to the network if a cell member is captured or otherwise compromised
    - How do they organize and execute?
      - use of transitory shortcuts in network

        - Held meetings connecting distant parts of network
        - Reduced average path length in network by over 40%, dramatically improving information flow
- **Sex in high school**
  - Chains of Affection: The Structure of Adolescent Romantic and Sexual Networks, written by Peter S. Bearman, James Moody, and Katherine Stovel
  - Asked high school students about "special romantic relationships" (last three plus current)
  - Found actual network to be very different from simulated structure
  - Taakeaway:
    * Greater implications for disease transmission
    * Lots of triads
- **Adopting facebook**
  - Structural Diversity in Social Contagion, written by J. Ugander, L. Backstrom, C. Marlow, and J. Kleinberg
  - Questioned how to get people to sign up for Facebook
    * Mary hasn't signed up for Facebook.
    * Josh sends Mary an invitation.
    * Dan sends Mary an invitation.
    * Is Mary more likely to sign up if Dan and Josh know each other or don't know each other?
  - Key takeaway:
    * Determined likelihood that person signs up for Facebook depends on number of unique connected components rather than size of network.
    * More likely to signup if invitations come from people who don't know each other.(from disconnected components)

Representing Networks

Data Collection

- Data for networks or graphs can be collected as follows:
  - **Passive**
    * From data sources like facebook, twitter, linkedin etc.
    * Automatically capture relationships between people
  - **Active**
    * Own-tie lists: Give free-form questions to name connections to people.
    * Network roster: Provide census with possible responses.

Data Persistence

- Vertices can be stored as onjects having multiple attributes (eg. if vertex is a city, attributes can be name,population etc)
- If each vertex can be given a unique id like 0,1,2..., the reference to the object ccanbe stored in an array.
- If the unique key is a name or string, a hash table can be used to store vertices.

Edge List

- Edges are stored in an un-ordered list
- Each edge object can have a unique id, references to the IDs of the vertices it connects and any additional attribute information (like weight or distance etc)
- In this representation,
  - If we have to find neighbors of a specific vertex, we have to go through entire edge list which takes O(|E|) = O(m) time
  - If we have to find neighbors of all vertices, it would take O(m.n) time
  - Space complexity:
    - O(n) to store vertices
    - O(m) to store eddges
    - So total of O(m+n)

Adjacency List

- Here, edges are stored as a list against each vertex.
- **Approach 1**: Vertex and edges stored seperately
  - Vertices are stored in an array or hash table
  - Edges are stored in an array or hash table of linked lists (adjacency lists)
    - Each element(array or hashtable) represents a unique vertex id and points to the list of vertices its connected with.
    - Each element in linked list has a vertex id (and optionally edge info like weight stored as a tuple)
- **Approach 2**: Vertex and edges stored together(**Object and pointer** approach)
  - Pointers to a vertex object are stored in an array or hash table of linked lists
  - Each vertex object has an adjacency list as an attribute within it.
  - Not useful when same set of vertices can have different adjacency lists.
    - Example, cities havinf train routes, air routes, bus routes etc
- In a undirected graph, say u-v, v is found in adjacency list of u and vice versa
- In a directed graph, say u->v, v is found in adjacency list of u but not the other way around.

- In this approach, neighbors of vertices can be found in O(degree of vertex) time.
- Space complexity:
  - To store vertices = O(n)
  - to store edges = O(m)
- Adjacency list is preferred in the following:
  - In a social network of n individuals, there are n x (n-1) possible friendships (where n may be on the order of billions).
  - However, even the most gregarious will have relatively few friends compared to the size of the network (thousands, perhaps, but still far smaller than hundreds of millions).
  - The same is true for the hyperlink structure of the web
  - Each individual web page links to a minuscule portion of all the pages on the web.

Adjacency Matrix

- Using a matrix to represent edges
- If there are n vertices, use a n x n matrix
- If element is 1, vertices are connected, otherwise they are not
- In a simple graph with no self loops, the diagonal will have all 0.
- The matrix will be symmetric for **undirected graphs**. i.e element at i,j will be same as element at j,i
- The matrix is not symmetric in a **directed graph**.
- To find neighbors of a vertex i, go to row i and scan entire row to find its neighbors.
  - Time complexity = O(n), since we scan entire row
  - Space complexity = O(n^2), irrespective of number of edges
- Since space complexity is fixed, this representation is good only if the graph is **dense** i.e has many edges ($\approx O(n^2)$).
  - Here, graph would be filled with ones.
- In a **sparse graph**, |E| << $|V^2|$
  - Example facebook:
    - Around 2 billion people.
    - Each person might havve a few hundred friends
    - This is a sparse graph.
    - using an adjacency matrix will need huge amount of space. Addjacency list might be better here.
- Most real-world graphs are sparse, so most algorithms will use adjacency lists.

- **Is there an edge (u,v) in a graph?**
- Adjacency matrix: lookup matrix[u][v] = $O(1)$ time

- Adjacency List: Lookup vertex u in O(1) and then scan list in O(degree of u) time
- **How do we store weight info?**
  - Instead of 1 or 0, we can store weight of edge as each element andd use null or 0 or -1 to represent no path

Adjacency Maps

- Vertices are stored in an array or hash table
- Each element above points to a hash table.
  - The hash table has vertex as key and edge weights or object as values
- The question, **is the edge (u,v) in the graph** can be looked up in constant time
- Gets us advantage of adjacency list (space) and adjacency matrix (time to look up)

Grids

- Graphs can be represented in the form of a grid where each element is a vertex.
- In this case, edges are implicit.
- For example, if a vertex is at element (i,j), its neighbors would be (i,j-1),(i,j+1),(i+1,j),(i-1,j)

Multi modal networks

- **One-mode network**: connections among one sort of entity.
  - E.g., romantic relationships
- **Two-mode network**: connections among two sorts of entities
  - Typically Bipartite graphs
    - Two types of nodes.
    - Edges join nodes of different types.Eg, People and products
    - Mostly used in recommendation engines.

Descriptive statistics for networks

- Quantify properties of whole structure and properties of individual positions.
  - Node-level indices
  - Graph-level indices

Node Level Indices

- Real-valued function of graph and vertex.
- Quantify properties of nodes in network.
- Describe and measure properties of particular position in network.
- Common families of properties include:
  - Centrality
  - Ego-net Structure
  - Alter covariate indices.

Centrality

- Extent to which node is central to network.
- Many distinct centrality concepts (no one way to be central).
- Different types of centrality aid and hinder different kinds of actions.
- Measures correlate, though being highly central in one respect doesn't always mean being central in other respects.
- Some measure include:
  - Degree
  - Betweenness
  - Closeness
  - Eigenvector Centrality
  - Katz centrality
  - Alpha centrality

Degree

- Number of direct ties for a given node.
  - **Outdegree**
    - Number of directed connections from given node
    - Interpreted as a form of gregariousness.
  - **Indegree**:
    - Number of directed connections to given node
    - Interpreted as a form of popularity.
- This measure denotes the overall activity or extent of involvement of the node in relations.
- A node in high degree positions are influential but also subject to influence of others
- **Undirected Graph**
  - Sum of degree's of all vertices = 2 * number of edges
- **Directed Graph**

  - Sum of indegree = Sum of out degree = number of edges

- Formally, given a graph as an addjacency matrix Y, the degree of a node i is computed as:

$$d(i, Y) = \sum_{j=1}^{N} Y_{ij}$$

- where,
  - $d(i, Y)$ is the degree of node i in graph Y represented as an adjacency matrix.
  - N Total nnumber of nodes in the graph
  - Y_{ij} is an edge from node i to node j. Or, entry in row i, column j of the adjacency matrix.
- From an adjacency matrix, for a directed graph
  - **Outdegree**: sum of corresponding row
  - **Indegree**: sum of corresponding column

Betweenness

- Measures tendency of a node to reside on the shortest paths between other nodes.
  - or, how important a node is to the shortest paths through the network.
- Quantifies extent to which the node serves as bridge
- High betweenness positions associated with:
  - "broker" or "gatekeeper" roles
  - Control information flow
- In directed networks, betweenness can have several meanings.
  - A user with high betweenness may be followed by many others who don't follow the same people as the user.
    - This would indicate that the user is well-followed.
  - Alternatively, the user may have fewer followers, but connect them to many accounts that are otherwise distant.
    - This would indicate that the user is a reader of many people.
  - Understanding the direction of the edges for a node is important to understand the meaning of centrality.
- Computing betweenness of a node i in a graph Y:

$$b(i, Y) = \sum_{j \neq i} \sum_{k \neq i} \frac{f(j, k, i)}{f(j, k)}$$

- where,
  - $f(j, k, i)$ - Number of shortest paths between j and k that involve i.

  - $f(j, k)$ - Total number of shortest paths between j and k.

- The process is described:
  - To compute betweenness for a node N, we select a pair of nodes and find all the shortest paths between those nodes.
  - Then compute the fraction of those shortest paths that include node N.
  - Example:
    * If there were five shortest paths between a pair of nodes, and three of them went through node N, then the fraction would be 3/5=0.6.
  - We repeat this process for every pair of nodes in the network.
  - We then add up the fractions we computed, and this is the betweenness centrality for node N.

Closeness

- Indicates how close a node is to all other nodes in the network.
- High closeness positions quickly distribute information.
- Ratio of minimum distance to other nodes to observed distance to other nodes.
  - Extent to which position has short paths to other positions
- Computing closeness of a node i in a graph Y:
  - Inverse of sum of distance to all other nodes normalized by number of nodes

$$c(i, Y) = \frac{(N-1)}{\sum_{j=1}^{N} D(i, j)}$$

where, - N - Number of nodes in the graph. - $D(i, j)$ - Distance between nodes i and j.

- Example: Compare closeness of D and A below.

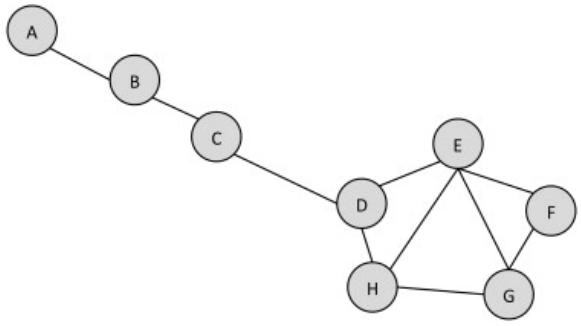

Figure 1: Example graph

First, compute the closeness of node D

Node	Shortest Path from D
A	3 (D–C–B–A)
B	2
C	1
E	1
F	2
G	2
H	1

Figure 2: Shortest distance from D

- $\sum_{j=1}^{N} D(i,j) = 3+2+1+1+2+2+1 = 12$
- $c(D,Y) = \frac{7}{12} = 0.58$
- Next, compute the closeness of node A

Node	Shortest Path from A
B	1
C	2
D	3
E	4
F	5
G	5
H	4

Figure 3: Shortest distance from A

- $\sum_{j=1}^{N} D(i,j) = 1+2+3+4+5+5+5 = 25$

- $c(A, Y) = \frac{7}{25} = 0.28$
- Thus, since node D's closeness centrality is 0.58 and node A's is 0.28, node D is more central by this measure.

Example of various measures of centrality in a network

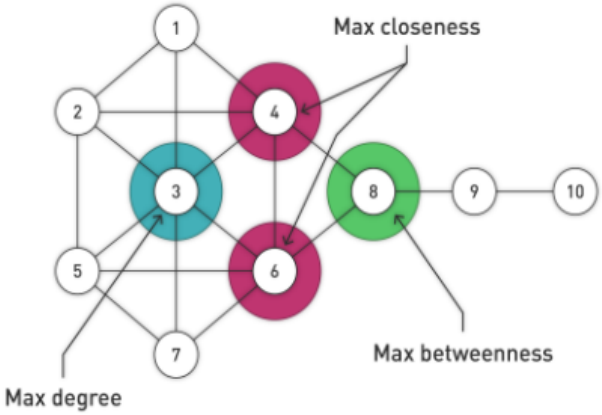

Figure 4: Measures of centrality in a network

Eigenvector Centrality

- Measures "coreness" within network.
- Centrality of node depends on neighbor's centrality
  - A person with few connections could have a very high eigenvector centrality if those few connections were to very well-connected others.
- Interpreted as arising from reciprocal process in which centrality of each actor is proportional to sum of centralities of actors to which it is connected.

- Eigenvector centrality allows for connections to have a variable value, so that connecting to some vertices has more benefit than connecting to others.
- The **PageRank algorithm** used by Google's search engine is a variant of Eigenvector Centrality, primarily used for directed networks.

Graph Level Indices

- Real-valued function of graph
- Method of understanding social networks or graph in entirety
- Aggregate features of structure as a whole
- Some properties include:
  - Degree distribution
  - Connected components

  - Giant components
  - Density

Degree distribution

- This is a histogram of the degrees of all nodes in network.
- Simplest networks have similar degrees.
- Real-world networks have very different degree distributions.
  - Long-tail phenomenon
    * Very few hubs with high degree
    * Many low degree nodes

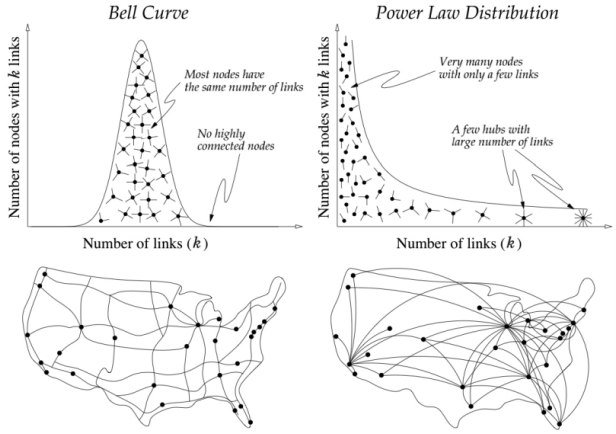

Figure 5: Degree distributions

Connected components

- Connectivity in an undirected graph means that every vertex can reach every other vertex via any path.
- If the graph is not connected the graph can be broken down into Connected Components.
- Directed network: edges directional but not necessarily connected
  - Strongly connected component: each node within component can be reached from every other node in component by following directed links
  - Weakly connected component: each node reached from every other node by following links in either direction

Giant components

- Largest component encompassing significant fraction of graph
- Examples:
  - Structure of "special romantic relationships" at Jefferson High School

– Mobile phone networks in the United States

Density

- Fraction of possible edges that are present
- Probability that given edge is in graph
- Overall measure of activity
- For **undirected graphs**, density is given by:

$$\delta = 2 \times \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} Y_{i,j}}{N.(N-1)}$$

- Example 1:

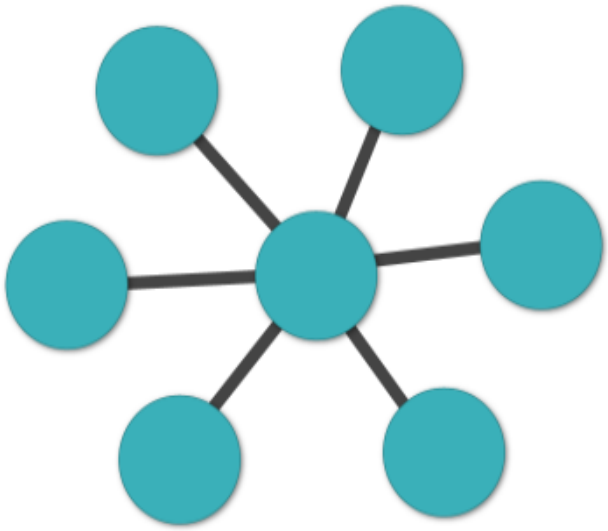

Figure 6: example 1

$$\delta = 2 \times \frac{\text{number of edges}}{N.(N-1)} = 2 \times \frac{6}{7.(7-1)} = \frac{2}{7}$$

- Example 2:

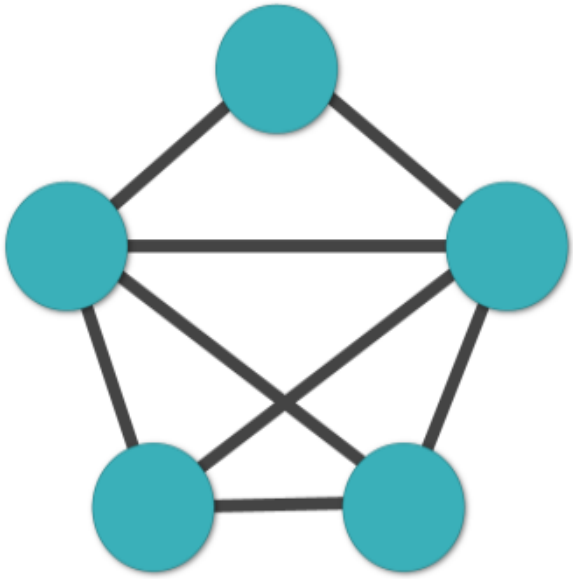

Figure 7: example 2

$$\delta = 2 \times \frac{\text{number of edges}}{N.(N-1)} = 2 \times \frac{8}{5.(5-1)} = \frac{4}{5}$$

- For **directed graphs**, density is given by:

$$\delta = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} Y_{i,j}}{N.(N-1)}$$

Distance Measures

- Shortest path
  - Also called geodesic path
  - Shortest sequence of links connecting two nodes
  - Not always unique
- Diameter
  - Largest geodesic distance in graph

Modeling Network Formation

- Modeling network formation allows us to analyze the following questions:
  - Why do graphs exist
  - What processes led to the observed structures
- Implications for why observed characteristics exist
- Practical applications

  - Network robustness
  - Disease spread
  - Web search
  - Opinion formation
- Fundamental understanding of networks

Random Networks

- Attributed to Erdos and Renyi
- Formed when nodes connected at random
- Differentiated from real-world networks in two ways
- No power law; number of edges incident on each node is Poisson distribution.
  - No hubs
- No local clustering and triadic closures.
  - Constant, random, independent probability of two nodes being connected
  - Low clustering coefficient

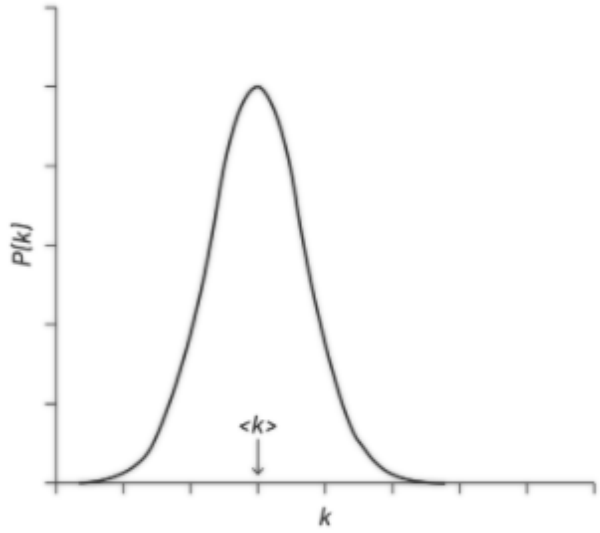

Figure 8: Poisson Distribution of degree

Small-World Networks

- Network in which most nodes are not neighbors of one another, but most nodes can be reached from every other by small number of steps
- Network in which typical distance L between two randomly chosen nodes (number of steps required) grows proportionately to logarithm of number of nodes N in network.

$$L \propto log(N)$$

Random Rewiring: Watts-Strogatz Model

- Makes network into small world using few random links in otherwise structured graph (average shortest path is short)
- Explains small-world clustering problem but not degree distribution

Preferential Attachment:Barabasi-Albert Model

- New node picks from existing nodes according to features of existing nodes (degree).
- Solves degree distribution but not clustering.

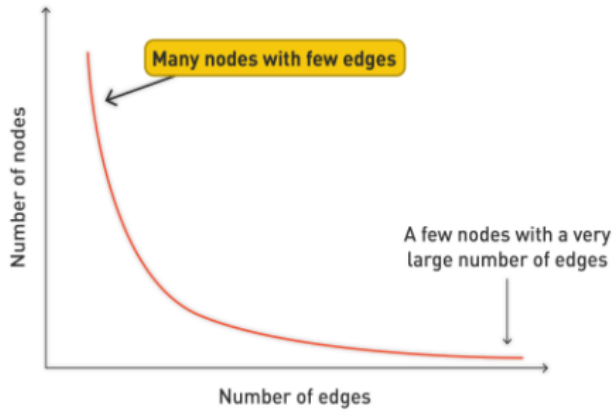

Figure 9: Barabasi-Albert Model

Power-law network

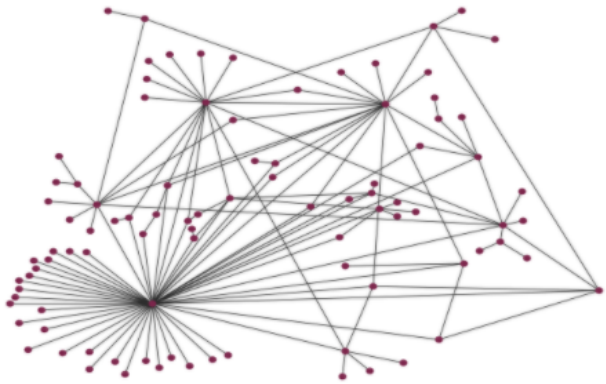

Figure 10: Power-law network

- Robust to random breakdown
- Vulnerable to targeted attacks

  - Breaks up when hubs removed
- Examples of real-World Power-Law Networks
  - Needle sharing
  - Sexual contact
  - E-mail networks
  - Viruses (persist no matter how low their infectiousness)