Lectures 10 & 11: Networks
Modeling Social Data, Spring 2017
Columbia University

April 14, 2017

Notes from aj2708

1 Introduction and Brief History

1.1 Erdos-Renyi model

In graph theory, the Erdos Renyi model is a model using for generating random graphs. They are named after Paul Erdos and Alfred Renyi, In the model introduced by Erdos and Renyi, all graphs on a fixed vertex set with a fixed number of edges are equally likely; in the model introduced by Gilbert, each edge has a fixed probability of being present or absent, independently of the other edges. These random graphical models can be used in the probabilistic method to prove the existence of graphs satisfying various properties, or to provide a rigorous definition of what it means for a property to hold for almost all graphs

A graph generated by the binomial model of Erdos and Renyi (p = 0.01) Here are two variants of this model described below.

1. In the G(n, M) model, a graph is chosen uniformly at random from the collection of all graphs which have n nodes and M edges. For example, in the G(3, 2) model, each of the three possible graphs on three vertices and two edges are included with probability 1/3.

2. In the G(n, p) model, a graph is constructed by connecting nodes randomly. Each edge is included in the graph with probability p independent from every other edge. Equivalently, all graphs with n nodes and M edges have equal probability of being connected.

The parameter p in this model can be thought of as a weighting function; as p increases from 0 to 1, the model becomes more and more likely to include graphs with more edges and less

1.2 Modeling Social Networks:

Then we moved on to computing the number of friends that any two nodes have in common, motivated by the problem of friend recommendations on social networks. The underlying idea can be traced back to Granovetter: two people are likely to know each other if they have many mutual friends. To compute the number of mutual friends between all pairs of nodes, we exploit the fact that the neighbors of every node share that node as a common friend. To count all mutual friends we simply loop over each node and increment a counter for every pair of its neighbors. For each node this scales as the square of its degree, so the whole algorithm scales as the sum of the squared degrees of all nodes. This can quickly become expensive if we have even a few high-degree nodes, which are quite common in practice.

1.3 Modeling the web

The most widely known generative model for a subset of scale-free Albert's rich get richer generative model in which each new Web page creates links to existing Web pages with a probability distribution which is not uniform, but proportional to the current in-degree of Web pages. This model was originally invented by Derek J. de Solla Price in the year 1965 under the term cumulative advantage) According to this process, a page with many in-links will attract more in-links than a regular page. This generates a power law but the resulting graph differs from the actual Web graph in other properties such as the presence of small tightly connected communities.

1.4 Watts-Strogatz Model

The Watts–Strogatz model is described as a random graph generation model that produces graphs with small-world like properties, including short average path lengths and high clustering coefficient. It was proposed by Duncan J. Watts and Steven Strogatz in 1998.They overcome the following deficiencies of the ER graph models properties observed in many real-world networks:

1. They do not generate local clustering and triadic closures. Instead because they have a constant, random, and independent probability of two nodes being connected, ER graphs have a low clustering coefficient.

2. They do not account for the formation of hubs. Formally, the degree distribution of ER graphs converges to a Poisson distribution, rather than a power law observed in many real-world, scale-free networks.[3]

   The Watts and Strogatz model was designed as the simplest possible model that addresses the first of the two limitations. It accounts for clustering while retaining the short average path lengths of the ER model.

1.5 Homophily, Contagion

Homophily, or the formation of social ties due to matching individual traits Social Contagion, also known as social influence; The causal effect of an individual's covariates on their behavior or other measurable responses

1.6 Modeling political Communities

The political blogosphere and the 2004 U.S. election: divided they blog Lada A. Adamic from HP Labs and Natalie Glance from Intelliseek Applied Research Center studied the linking patterns and discussion topics of political bloggers and in particular, the degree of interaction between liberal and conservative blogs and to discover any difference in the structure of the two communities by analyzing top 40 A-list blogs, over the period of 2 months preceding US 2004 elections and to quantify how often they referred to each other, to understand the intersections of the topics discussed, in a both intra-community and inter-community manner.

1.7 Echochambers

Jonathan Bright from Oxford University in his paper Explaining the emergence of echo chambers on social media: the role of ideology and extremism explains the concept of echo chambers which refers to the idea that online conversations about politics are typically divided into a variety of subgroups, and that this division takes place along ideological lines with people only talking to others with which they are already in agreement. Therefore, it can be said there is subgroup or subcommunity formulation wheremost of the interaction takes place and intercommunity interaction is a lot lesser.

2 Types of networks

:

1. Social Networks like Facebook, Google Plus, Twitter
2. Information Networks, World Wide Web
3. Activity Networks like email
4. Biological Networks like protein interactions
5. Geographical Networks like roads, railway tracks etc.

The professor also mentioned in the class that Facebook is slowly moving towards becoming an information network rather than a social network.

3 Network Representation

Networks can be represented in many ways.

Networks can be directed or undirected. Directed graphs are those graphs where the edges between two nodes has a direction property associated with it such that an edge from node A to node B doesn't necessarily imply an edge from node B to node A. Undirected graphs are graphs where edges don't have the direction property such that an edge form node A to B implies that there is also an edge from node B to node A.

Unweighted graphs are those graphs where the edges have no weights associated with them, whereas weighted graphs have edges that have a certain weight associated with them which usually denotes the strength of connection between two nodes.

There are other kinds of abstractions for representing networks such as metadata networks (where there is certain metadata associated with the nodes of the graph).

While creating networks, it is also suggested to do a lot of preprocessing to ensure that we are able to create a good network representation.

4 Data Structures:

Networks can be represented by various data structures. Mainly two types of data structures are used for representing graphs. In an adjacency matrix representation, we have a matrix having N rows and N columns for a graph having N nodes and each entry in the matrix representing the presence or absence of an edge (or the weight of the edge in a weighted graph). In this representation, edge lookup is constant time but it requires a lot of storage space. It is possible to represent the adjacency matrix in a sparse fashion.

Another possible representation for a graph is an adjacency list representation such that for each node in the graph, we have a list of nodes it is connected to (basically there is an edge emanating from the source node connecting the nodes in the adjacency list). In an adjacency list representation, it takes O(E) time to check for the presence or absence of edge between two nodes. Adjacency list is convenient for certain graph traversals like BFS and DFS.

5 Describing Networks

There are many kind of descriptive statistics to describe networks such as Indegree is the number of incoming connections for a particular node.

Outdegree is the number of outgoing connections for a particular node.

Using the degree statistics, we can create degree distributions which can be used as modeling tools in many network analysis tasks.

Clustering tries to ascertain how many friends of friends are also friends. This can be done using cycle counting (like triangle counting).

Measures like clustering coefficient can be used as descriptive statistics for discovering subcommunities in the graph, the size of the subcommunities, the density of interactions within the subcommunities and deriving the bounds on interactions between different subcommunities.

Number of connected components is a measure of how many disconnected parts does the network have. The number of connected components can be counted using a graph traversal algorithm like depth first search or breadth first search.

Path Length basically measures the length of the shortest path between two nodes. We can use breath first search for finding the shortest path between two nodes. To find the shortest path between two nodes we can use breadth first search starting from the source node and terminating the breadth first traversal when we first encounter the destination node.

Triangle Counting is basically counting the number of cycles in the graph of length 3. This can basically be done using done several algorithms. It is also a measure of the number of such nodes, A, B and C in a graph such that A and C, and B and C are connected to each other and also A and B are connected to each other.

We learnt in the class that we should experiment with toy networks initially while creating visualizations. In the class, a road network of a particular city was shown and it was found that a lot of interesting inferences could be drawn from the visualization. One such inference was that were some private gated communities at the edge of the city. Also, from the density of the edges, one could reason about the density of the road network for that part of the city. We also discussed the Wikipedia voting network, the degree distributions of number of voters. And we inferred that very few people actually vote on Wikipedia.

```
breadth first search:
    unmark all vertices
    choose some starting vertex x
    mark x
    list L = x
    tree T = x
    while L nonempty
    choose some vertex v from front of list
    visit v
    for each unmarked neighbor w
        mark w
        add it to end of list
        add edge vw to T


Depth first search:
dfs(vertex v)
    {
    visit(v);
    for each neighbor w of v
        if w is unvisited
        {
        dfs(w);
        add edge vw to tree T
        }
    }


Finding Components:
DFS(G)
    {
    make a new vertex x with edges x->v for all v
    initialize a counter N to zero
    initialize list L to empty
    build directed tree T, initially a single vertex {x}
    visit(x)
    }

    visit(p)
    {
    add p to L
    dfsnum(p) = N
    increment N
    low(p) = dfsnum(p)
    for each edge p->q
        if q is not already in T
        {
        add p->q to T
        visit(q)
        low(p) = min(low(p), low(q))
        } else low(p) = min(low(p), dfsnum(q))
    if low(p)=dfsnum(p)
    {
        output "component:"
        repeat
        remove last element v from L
        output v
        remove v from G
        until v=p
    }
    }
```

6 Algorithm to find mutual friends

: For each node in the graph, run over all the pairs of its neighbors using an adjacency list. All these pairs of nodes, have at least 1 mutual friend.
This has a time complexity which is sum for all nodes the square of the length of their adjacency list. In a densely connected graph, this algorithm has a very bad time complexity. Also, in case a node is like a celebrity, then it may be connected with majority of the other nodes in the graph. In such a situation, the runtime of the algorithm becomes quadratic or more than quadratic overall in the number of number of the graph which can be very bad for large networks.

Notes from cl3425

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.

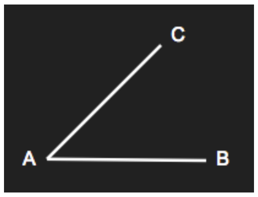

Figure 1: Granovetter: this is forbidden; it's impossible for A-C and A-B to be the case without B-C being a thing as well!

  - 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

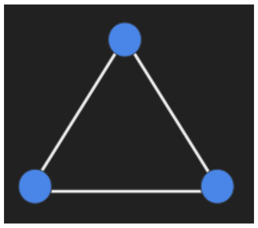

Figure 2: Bidirectional friendship (one would hope)

- Directed

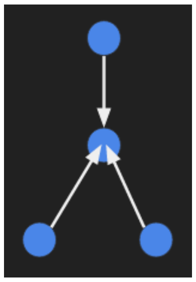

Figure 3: Directed network, eg. followers of a FB page

- 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 → (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 ← curr dist + 1
  add neighbor to boundary
curr boundary ← 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

```
for each node:
  run over all pairs of its neighbors
  increment count by 1
```

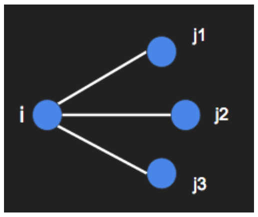

Figure 4: Counting mutual friends

- 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.

Notes from nd2506

1 Networks:

1.1 Types of Networks

:

We learned about the different types of networks. Some of them are:

1. Social Network: A social network is a social construct which is made up of social actors (which are individuals or organizations), a set of dyadic ties and other social interactions b/w the actors. Facebook is the simplest example of such a construct.
2. Information Network: An example of the information network is the world wide web
3. Activity network : A graphical method for showing dependencies between tasks (activities) in a project. The network consists of nodes connected by arcs. Nodes denote events and represent the culmination of one or more activities. Arcs represent activities and are labeled with the name of the activity and have an estimated time to complete the activity.
4. Biological Networks: A biological network is any network that applies to biological systems. eg. protein interactions, disease spreading models, etc.
5. Geographical Networks: These are networks which model geographical entities. eg Airport connectivity, roads, underwater cave systems.

1.2 Network Representations:

A network has several representations. It can be directed or undirected, weighted or unweighted etc. When looking at network data it is very important to correctly preprocess the data. It's important to understand which data to throw out.

1.3 Data Structures:

From a computational point of view the following data structures are used:

1. Array of tuples : It is simple for storage but difficult for computing stuff. eg to check if an given pair of nodes has an edge takes $\mathcal{O}(E)$ time.
2. Adjacency matrix : It is quick for calculations. I takes $\mathcal{O}(1)$ time for edge check. It's good for linear algebra. We can also use a sparse representation for computation.
3. Adjacency list: It is good for graph traversals.

1.4 Describing Networks:

A network is described using nodes and arcs. It has several attributes like degree distributions, path length , triangles (In a social network this might mean checking if friends of friends are friends), counting the number of connected components.

1.5 Path Length:

Given a network if we want to find a least path length b/w any two vertices we can apply the breadth first search algorithm. The algorithm is given below:

```
Breadth-First-Search(Graph, root):

    create empty set S
    create empty queue Q

    add root to S
    Q.enqueue(root)

    while Q is not empty:
        current = Q.dequeue()
        if current is the goal:
            return current
        for each node n that is adjacent to current:
            if n is not in S:
                add n to S
                n.parent = current
                Q.enqueue(n)
```

Similarly one can use BFS to find the connected components of a graph.

1. Given an algorithm for BFS we can run BFS randomly from any node. When the algorithm ends we have one of the connected components.
2. Choose another node which hasn't been visited before and run BFS from that node.
3. Repeat until every node has been visited.

1.6 Friend Lists:

As an exercise we calculated mutual friends for all pairs of nodes. The algorithm is simple. For all nodes simply loop over all pairs of neighbours and increment the counter for that pair by 1. However, this can get ugly pretty quickly. The runtime is $\mathcal{O}(\text{outdegree}^2|V|)$. So this is not suitable for graphs with long adjacency lists like star graphs.

Next we calculated the number of triangles in a graph. The algorithm is similar to to the mutual friends algorithm except we also need to check if the pairs have an edge b/w them. The total number of triangles is half this value as all pairs have been counted twice.

Using this we can find the clustering coefficient which is a measure of how clustered the network is. The number of triangles divided by the number of possible triangles gives the clustering coefficient.

Notes from sh3589

1 Network Analysis Briefing

1.1 History

Network analysis has a history starting from 1930. The hot topic today is Contagious effect through social networks.

1.2 Types of Networks

Determining which kind of network it is is often the first step towards a social network analysis, since each network type has its unique properties.

1. Social Networks e.g. Facebook, Twitter networks
2. Information Networks e.g. political blogs (instead of relationship of friends, the interactions may only be referring)
3. Activity Networks e.g. emails
4. Biological Networks e.g. protein interaction clusters
5. Geographical Networks e.g. city roads

1.3 Levels of Networks

1. direction
2. weighted e.g. cost of edge movement, cost of communication
3. metadata e.g. every other data associated

1.4 Which Network

Networks can be very different in terms of what you want to convey with it. We can define how strict we are about the what counts as an edge.
For example, whether to include one-way communication or reciprocated communication in an analysis of facebook relationships; In an organization, whether we care more about the hierarchy or the collaboration of the individuals.

1.5 Data Structure

1. Simple Storage.
   Pro: Simple for storage;
   Con: Complex checking
2. Adjacency Matrix.
   Pro: Easy checking, constant times; When network is directed, matrix is symmetric.
   Con: Matrix is often sparse and hard for storage.
3. Adjacency List.
   Pro: Good for graphing;
   Con: Storage.

1.6 Descriptive Statistics

1. degree
2. path length
3. clustering
4. component

2 Application in R

package: igraph

3 Algorithms for Counting on Networks

3.1 Degree Distribution

group by source nodes
count the number of destination nodes to get source degree
group by degree
count the number of source nodes

3.2 Shortest Path Length

Basic idea: start with source nodes and explore the neighbors.

Algorithm:

```
init nodes at infinity
    source dist.0
    current boundary
    new boundary is empty
explore non-empty boundaries
loop over all nodes in current boundary
explore each undiscovered neighbors
iterate until no undiscovered neighbors
```

3.3 Connected Components

```
init nodes at infinity
pick random unreached nodes
run BFS from that node
    label everything reached as one component
repeat until there is no reachable nodes
```

3.4 Mutual Friends

Application: Friend recommendation on social media

Algorithm: (need adjacency list)

```
for each node:
    run over each pairs of its neighbours, increment by 1
```

This algorithm can get expensive when the adjacency list is long, for instance, celebrity nodes.

3.5 Counting Triangles

```
(keep adjacency matrix and adjacency list)
for each node:
    run over each pairs of its neighbors, increment by 1
    (check adjacency matrix) if neighbors connected
```