Lecture 10: Networks

Nishchal Dethe

April 7, 2017

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.