CS224W - Social and Information Network Analysis

Fall 2010
Assignment 1

Due 11:59pm October 14, 2010

General Instructions

You are required to write the name of your collaborators for this assignment on your solution report. You are also required to submit the source code used to obtain your solutions along with your report.

The data set you would need for this assignment can be found here:
http://www-personal.umich.edu/ mejn/netdata/astro-ph.zip .
This data set is a collection of scientific collaborations across researchers from the Astrophysics domain. Each node is a researcher, and each edge represents co-authorship of the two researchers together on a paper. The weight of each edge signifies indirectly, the strength and extent of collaboration between two such authors. For the assignment below, you are expected to first implement a framework that represents this co-authorship network in the form of an undirected weighted graph. We encourage you to use any graph analysis tools or packages you might want to or feel comfortable in using (SNAP for C++, NetworkX for Python, JUNG for Java etc). You can also find some here : http://www.stanford.edu/class/cs224w/resources.html

Submission Instructions

We expect you to be able to access your Dropbox folder on:

http://coursework.stanford.edu (within your own space).

Whichever files you want us to see are to be archived (or zipped) together into a single file and placed into this Dropbox folder. This file should be named of the format: <SUNetId>_ <Last-Name>_<First-Name>_HW1 : where these are your last and first names respectively. Within this file, we require the following at least :

a. Your solution report. You can submit it as a .pdf or .doc. This report should be named as : <SUNetId>_<Last-Name>_<First-Name>_HW1_Ans (.doc or .pdf)
b. Any source code that you use towards obtaining your results stated in your solution report. Also include the mention of any tools whichever you use towards your solution(s).

Questions

1. Network Characteristics (30 points - Sudarshan)

Here, we would be dealing with not just the original network (henceforth called real world network), but also a configuration model (rewired network) and a G(n,m) Erdos-Renyi random network.

Rewired network - For each node n_i in the random network, create d_i spokes (half-edges), where d_i is the degree of the node i in the original network. Now randomly connect endpoints of these spokes - one possible way to do this is to create a random ordering of the spokes, and then simply connect those spokes that are adjacent in the ordering - connect spokes at positions $2i + 1$ and $2i$, for $i = 1...$. Notice that this procedure will likely create a multi-graph a graph with multiple edges (pairs of connected spokes) between a pair of nodes, and self-loops on nodes. Now, simplify the resulting multi-graph by collapsing multiple edges between a pair of nodes into a single edge and remove self loops.

G(n,m) Random network - Pick n nodes (where n = the number of nodes in the real world network) and m edges at random (where m = the number of edges in real world network) to construct this network.

Note - Parts (a) and (b) are to be done for all the three networks.

(a) Plot and compare the linear-linear and log-log degree distribution of all three networks. Superimpose the plots from the three networks for each of the linear-linear and log-log distributions respectively.

(b) Recall that the local clustering co-efficient for an undirected graph G(V,E) is defined as

$$C_i = 2|e_jk|/[k_i \cdot (k_i - 1)]$$

where $v_j, v_k \in N_i, e_jk \in E, k_i$ is the degree of node i, and N_i is a set of network neighbors of node i. Find the *average clustering coefficient* for all the three networks – it is defined as

$$C = (1/|V|) * [\sum_i c_i, i \in V].$$

Also, compute the *average path length* of the three networks. To do this, pick at random, any densely connected node in the network and compute the *shortest path lengths* between it and every other node (with a non-zero degree) in the network. An average of these path lengths is considered as the average path length for the network. Compare the average path lengths and average clustering coefficient across the three network types. Refer to Slide 24 in the lecture slides from class on September 27. (In other words, does the real world network follow a small world model ? Discuss.)

Complete parts (c) and (d) for the real-world network only:

(c) The *neighborhood overlap* of an edge connecting nodes A and B is defined as follows :

number of nodes who are neighbors of both A and B/number of nodes who are neighbors of at least one of A or B

Draw a plot of the neighborhood overlap of edges as a function of their percentile in the sorted order of all edges by tie strength. (Refer Figure 3.7 from our textbook)

(d) *k-core decomposition* of the co-authorship network. For $k = 1...$ max(degree), remove all nodes of degree less than k. The connected components that are left after all nodes of degree less than k have been removed are called the *k-cores* of the graph. Plot a distribution of k vs. number of nodes in the *k-core*, and discuss what this kind of plot might reveal about the structure of the network.

2. Exploring Robustness of Networks (30 Points - Sonali)

Consider the original network, the re-wired configuration network and G(n,m) network model (as computed in Question 1). For these three networks, do the following :

(a) You will be deleting nodes randomly from these networks and computing the average path length of the network after each deletion phase. (Since it would be computationally intensive to compute the average path length after the deletion of every node, delete nodes in batches of X, say $X = |N|/100$, where $|N|$ is the total number of nodes in the network). First delete X nodes, compute average path length of the network, delete additional X nodes, compute average path length and so on. Plot the average path length of the network versus the percentage of the deleted nodes of all three networks in the same plot. What do you infer from the plot? Note: Refer Question 1.(b) about computing the average path length of the network.

(b) For this part, we want you to delete the nodes in the sequence of decreasing degree (instead of doing it randomly). Draw a similar plot, of the average path length of the network versus the percentage of the nodes deleted for all three networks. Determine the effect of deletions of nodes of high degrees across these three networks by observing the plot.

(c) Comment on the robustness of these networks using observations from part a) and b). What kind of attacks are these networks susceptible to?

3. Open-Ended Question (10 points - Sudarshan and Sonali)
Describe a further interesting property or pattern of this network that you identify by experimenting with the data , similar to in problem 1 and 2. You can explore how the network behaves if nodes having high

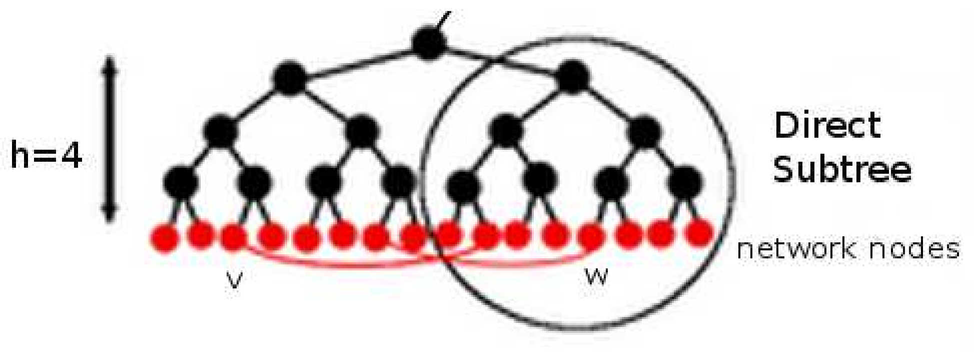

Figure 1: Illustration for Question 4 of the hierarchical graph where virtual nodes are black and leaf nodes with random links are red

weight edges are deleted or anything else you like. You dont need to hand in any code, just a write-up of what you were looking for and what you found.

4. Decentralized Search (30 points - Nadine: part 1 and 2 , Jennifer: part 3)

Let's consider the original small world experiment where letters advanced through the social network one person at a time. To find the true shortest path, one would instead need to perform a *breadth-first search*, that is, send letters to all friends and not just one. Although this flooding of the network was not feasible, the experiment was still largely a success, despite the apparent fragility of advancing a single letter.

[Watts-Dodds-Newmann, 2002] Structured social relationships in terms of nodes residing in a hierarchy, where the hierarchy can encode different groupings by activity or profession, etc. In this problem, we explore the properties of the hierarchical network that allow this type of decentralized search to be effective.

Part 1

Let T be a complete k-ary tree (each node has k children). Consider a network whose nodes reside in the leaves, and the tree above them models the hierarchy. We define the tree distance between two leaves v and w as: $h(v, w)=$ height of the least common ancestor of nodes v and w.

(a) Fix a network node v. How many network nodes are at a distance h from v?

Part 2

We now connect the nodes of the network so that each node has degree d. Assume that the network contains n nodes. Each outgoing edge e of a node v has the following probability of being connected to a node w: $P_e(v \to w) = ck^{-h(v,w)}$ where c is a normalizing constant (Note that this is different from the overall probability of v and w being connected.).

Claim 1: For any node v, any height h and any direct sub-tree T' of v's ancestor at height h (see figure), one of v's d links goes into T' with high probability.

Claim 2: The property in Claim 1 guarantees efficient search: We can reach a node w from any node v in $O(\log(n))$ steps.

(b) What is the normalizing factor c equal to? (Note that c can be a function of n. For a fixed node v, write the total mass $1 = \sum_{i=1}^{log(n)}$ P(choosing a w such that $h(v, w) = i$))

(c) Let $E_{v,T'}$ denote the bad event that v has no links that go into T' and define $E = \cup_{v,T'} E_{v,T'}$. Show that $P(E_{v,T'}) \leq e^{\frac{-cd}{k}}$ and find an upper bound on $P(E)$ (using the union bound). Find an appropriate value of d to prove Claim 1 using this result (d can be a function of n and we prove Claim1 in the limit as

$n \to \infty$).

(d) We now use the property that for a height h, any node and direct sub-tree are linked with high probability in order to improve searching. Prove Claim 2 (hint: use recursion).

Part 3

Using what you have proved above, we will now examine the effect of changing the random linking probability. First construct a complete binary tree of height 10. Treat all nodes except leaf nodes as virtual nodes that simply encode the h(v,w) proximity relationship, as discussed above, between leaf nodes. For this question, we will also introduce another parameter, a scale factor α. For all the leaf nodes, we want to set the average degree to be 5; i.e. create 5 unconnected spokes leaving each node. Now, connect the spokes, where the probability of an edge being connected to another node is proportional to $c2^{-\alpha h(v,w)}$. Make sure that your normalizing constant changes correctly with respect to α. Once the graph has been constructed, randomly choose a starting point s and end point t. Specifically, perform a search where, for each hop, a current node only sees its outgoing links, and always pick the the link that gets you closest to t. If you reach a node where no outgoing links get you closer to t, classify the search as a failure. Repeat this for 1000 random (s, t) pairs.

(e). Evaluate the impact of changing α by plotting α vs. average decentralized search time for getting from s to t, where average search time is defined by the average number of hops traversed between the pairs of nodes. Comment on what an ideal value of α might be.

(f). Comment on how changing the value of α changes the lengths of paths in the graph. What do you observe when edges are short? What about when edges are long? How can you explain the search times as a function of α?

(g). Evaluate the impact of changing the degree of each node. Set the degrees equal to 2 and 10. Report the search failure rate for each degree 2, 5, and 10 and α=1.