COMS 4995-1: Intro Social Networks — Due on Thursday, November 8th, 2012

Assignment #3 – Division and Temporal Evolution

Chris J. Riederer, Zhikun Ma (TAs) — A. Chaintreau (instructor)

Why there are three parts in this assignment: Each part fulfills one of the objectives of the class:

- **Manipulate concepts:** Getting Familiar with the technical concepts used in class, by reproducing similar arguments. Being proficient by manipulating the object to answer some small-size problem.
  You are expected to answer this question rigorously, the answer can be quite short as long as it contains all the required argument to justify your answer.
- **Experience the concepts :** Being able to reproduce these concepts in real or synthetic data. Study their properties in real examples.
- **Connect the concepts to real-life:** Interpret a problem you find in light of the concepts or principles you have learned. Develop a critical eye to determine how the concepts introduced are useful in practice.

How to read this assignment : Exercise levels are indicated as follows

($\rightarrowtail$) "elementary": the answer is not strictly speaking obvious, but it fits in a single sentence, and it is an immediate application of results covered in the lectures.
Use them as a checkpoint: it is strongly advised to go back to your notes if the answer to one of these questions does not come to you in a few minutes.

($\curvearrowright$) "intermediary": The answer to this question is not an immediate translation of results covered in class, it can be deduced from them with a reasonable effort.
Use them as practice: how far are you from the answer? Do you still feel uncomfortable with some of the concepts and definitions? which part could you complete quickly?

($\looparrowright$) "tortuous": this question either requires an advanced concept, a proof that is long or inventive, or it is still open.
Use them as an inspiration: can you answer any of them? does it bring you to another problem that you can answer or study further? It is recommended to work on this question only AFTER you are done with the rest!

Part A — Manipulating the Concepts

Exercise 1: Joining an unbalanced world (6 pt)

In this exercise, we will learn whether a node can easily join an unbalanced world. We consider a graph $G = (V, E^+, E^-)$ that contains positive and negative edge and is complete (*i.e.*, all pairs of nodes have one edge between them). It is said *balanced* if all triangle contains either 1 or 3 friendship edges.

Imagine a new person X joins the graph G and has a complete freedom in deciding whether she maintains a positive or negative relationship with anybody else in the graph. The question is whether she can do so *while not being involved in any unbalanced triangle herself.* That means the following, if G is balanced, then X wants to join the graph while making sure she does not break this property. Otherwise, if G is already not balanced, some "forbidden triangles" exist, but at least X would like to make sure that after joining she is not part of any "forbidden triangles" herself.

1. ($\rightarrowtail$) Show by hand that if the graph G contains up to three nodes and is balanced, X can join the graph and not create any unbalanced triangle.

2. ($\curvearrowright$) Using a result from the class, prove that for a balanced graph of any size, then it is possible for X to always do so.

3. ($\curvearrowright$) Assuming now that G contains 3 nodes and is not balanced, what is the best thing that X can do to join the graph and not be part of any unbalanced triangle?

   For all cases of G, you should either prove that it is possible by providing a solution, or give a proof that it is impossible. We encourage that you provide a concise explanation instead of the result of an exhaustive search.

4. ($\rightarrowtail$) Provide a general conclusion describing in which cases X can successfully join any graph G and not be part of unbalanced triangle?

   Briefly discuss how this result generalizes when you use another definition of "balanced", generally called weakly balanced, in which triangles with three enemies are allowed, and the only triangles to avoid are the one with exactly two edges with positive signs.

Exercise 2: A variant of the community guided attachment model (6+0.5 pt)

In class, we have analyzed a first community guided attachment model (CGA) which connect a set of nodes in a way that relates to a hierarchy. More precisely we have assumed that the set of nodes are *leaves* of a tree and that the tree allows to define a natural distance between them. As an example two leaves from the exact same parent are close whereas leaves of branch of the tree that met only at the root are far. More generally we can always define the distance between two nodes u and v, (*i.e.*, two leaves of the tree) as the number of edges in the path that connects them in the tree, which we denote $d(u, v)$.

We now consider a model where the set of nodes are not only leaves, but all nodes in the tree, including interior nodes. To simplify the model we assume, as made in class, that this tree representing the hierarchy is a complete b_ary tree. For a node u we denote its height by h_u (which is 0 if u is a leaf, 1 if u is the parent of a leaf, and so on). The total height of the tree is $h_{\max}$, so that there are $b^{h_{\max}}$ leaves in the tree, $b^{h_{\max}-1}$ nodes of height $h = 1$ and so on.

We will make the same assumptions as in the model seen in class:

- We are interested in large set of nodes, so we will assume that $h_{\max}$ is going to infinity. This implies that the number of nodes N goes to infinity as well, as we have:

$$N = \sum_{h=0}^{h_{\max}} b^{h_{\max}-h} = \frac{b^{h_{\max}+1} - 1}{b - 1} \,.$$

- The graph between the nodes is created recursively. It initially contains a single node, which is also a leaf. A new "generation of leaves" is then added to the graph by creating b children for each leaf. Each new leaf u creates a directed edge to another node v (a leaf or an interior node), with probability

$$\mathbb{P}\left[u \rightsquigarrow v \in E\right] = \frac{1}{c^{d(u,v)/2}} \,.$$

  These events are independent among all pairs (u, v). Note that this probability decreases with the distance $d(u, v)$ which is also natural since there are more and more nodes at a given distance d as d grows. Note also that the edge is directed and that by definition v has height at least the same as u.

The goal of this exercice is to understand under which condition this model creates edge densification. In other words, as $h_{\max}$ increases, we wish to determine for which choices of b and c the number of edges in the graph (or equivalently the average degree of a nodes) grows as a power function of N.

We remind that, when f and g are two functions of N, we denote by $f(N) = \Theta(g(N))$ the fact that two constants $M > 0, M' > 0$ exist such that $M \cdot g(N) \leq f(N) \leq M' g(N)$.

1. ($\curvearrowright$) Prove that for a leaf (*i.e.*, $h = 0$), the number of nodes at distance $d \leq 2h_{\max}$ is,

$$\texttt{NeighIn}(h_{\max}, 0, d) = \sum_{j=\lceil \frac{d}{2} \rceil}^{\min(d-1, h_{\max})} (b-1) b^{d-j-1} + \mathbb{I}_{\{d \leq h_{\max} - h\}} \,.$$

   Deduce that there exist $A > 0, A' > 0$ such that for any $d \leq 2h_{\max}$:

$$A \cdot b^{\frac{d}{2}} \leq \texttt{NeighIn}(h_{\max}, 0, d) \leq A' \cdot b^{\frac{d}{2}} \,.$$

2. ($\curvearrowright$) Deduce for $c < b$ that there exist $B > 0.B' > 0$ such that the average out-degree of a leaf, denoted $\texttt{DegOutLeaf}(h_{\max}, 0)$, satisfies:

$$B \cdot \left(\frac{b}{c}\right)^{h_{\max}} \leq \texttt{DegOutLeaf}(h_{\max}, 0) \leq B' \cdot \left(\frac{b}{c}\right)^{h_{\max}} \,.$$

   We assume that $h_{\max} = \frac{\ln(N)}{\ln(b)}$, which is a good approximation as N becomes large. Deduce that, as a function of the number of nodes N, the average out degree $\texttt{DegOutLeaf}(N)$ of a leaf satisfies $\texttt{DegOutLeaf}(N) = \Theta\left(N^{1 - \frac{\ln c}{\ln b}}\right)$.

3. ($\curvearrowright$) Conclude that the same result holds for the average out-degree of any node in the tree.

   (Hint: use the fact that (1) leaves have maximum average out-degree among the nodes, and (2) there is a constant fraction of leaves in the graph for any $h_{\max}$.)

4. ($\curvearrowright$) What can you say about the average degree of a node as a function of N when $b < c$?

The rest of this exercice, a good training for a most advanced proof, is only half a point. Using similar arguments, you will learn how to prove that this model creates a power-law distribution for in-degree.

5. ($\looparrowright$) For a node u with height h, show that the number of nodes v at a distance d and with height $h_v \leq h$, when we assume that $d \leq h$, is equal to:

$$\texttt{NeighIn}(h_{\max}, h, d) = b^d + \sum_{j=\max\left(1, \lceil \frac{d-h}{2} \rceil\right)}^{\min\left(\lfloor \frac{d}{2} \rfloor, h_{\max} - h\right)} (b-1) b^{d-1-j} \,.$$

   Remember that nodes at distance d may be in the subtree rooted in u as well as in other parts of the tree.

   Deduce that there exists $C > 0, C' > 0$ such that for all $h_{\max}, h, d$, such that $d \leq h \leq h_{\max}$ we have

$$C \cdot b^d \leq \texttt{NeighIn}(h_{\max}, h, d) \leq C' \cdot b^d \,.$$

6. ($\curvearrowright$) If we now assume $d > h$, how is the above expression modified? Deduce that there exists $D > 0, D' > 0$ such that for all $h_{\max}, h, d$, such that $h < d \leq 2h_{\max} - h$ we have

$$D \cdot b^{\frac{d+h}{2}} \leq \texttt{NeighIn}(h_{\max}, h, d) \leq D' \cdot b^{\frac{d+h}{2}} .$$

7. ($\curvearrowright$) Deduce that there exists $E > 0, E' > 0$ such that:

$$E \cdot \left(\frac{b}{c}\right)^{h_{\max}} c^{h/2} \leq \texttt{DegIn}(h_{\max}, h) \leq E' \cdot \left(\frac{b}{c}\right)^{h_{\max}} c^{h/2}$$

8. ($\curvearrowright$) Show that, when $h = h_{\max}$ we obtain the node with largest average degree, and by the previous inequality we have $\texttt{DegIn}(N, h = h_{\max}) = \Theta\left(N^{1-\frac{1}{2}\frac{\ln c}{\ln b}}\right)$.

   N.B.: More generally, we have $\texttt{DegIn}(N, h) = \Theta\left(N^{1-\frac{1}{2}\frac{\ln c}{\ln b}} c^{-\frac{h_{\max}-h}{2}}\right)$. This implies that the node with "depth" $h_{\max} - h$ (and, hence, which has rank i, where $i = b^{h_{\max}-h}$, in the decreasing sequence of average degree) has a degree $(\sqrt{c})^{h_{\max}-h}$ smaller than the node with larger average degree. In other words, the node with rank i has degree smaller by factor $(\sqrt{c})^{\frac{\ln i}{\ln b}} = i^{\frac{1}{2}\frac{\ln c}{\ln b}}$. Hence, the sequence of average degrees follows a power-law with coefficient $\frac{1}{2}\frac{\ln c}{\ln b}$.

Part B — Experiencing the Concepts

Coding Assignment Submission All programing should be written in one file. The name of this file should be {UNI}_homework3.py, with {UNI} replaced with your UNI. For example, if my UNI is cjr2149, I would name my assignment cjr2149_homework3.py. This file should be uploaded into your Drop Box on Courseworks before the deadline. Please name the file correctly, paying attention to the extension, and do not compress your file before uploading. Points may be subtracted if you do not follow these procedures.

Exercise 3: Communities (4pt)

1. (↷)Write a function that finds the most "between" edge of a graph. Recall that "edge-betweeness" is defined as the number of shortest paths that go through e. Your function should take a NetworkX graph as input, and return the edge with highest betweeness. Your function does not have to be highly efficient. For graphs of several hundred nodes, it may take a few seconds to compute.

   Please name your function "most_between".

2. (↷)Now that we're able to find the most between edge, we can try to break a graph into communities. Under "Detailed Content" on the course website, you will find a link to download the graph "Sub-Wiki.txt". This graph is derived from a Wikipedia talk graph. The nodes represent the "talk" page of a user. An edge exists if a user has edited a talk page or had their talk page edited by another user.

   For this problem, write a function that takes no input. It should read the graph, then use your most_between function to break the graph into communities. The graph starts as one connected component. Repeatedly remove the most between edge until the graph has two connected components. Return these two components in a list (preferably using NetworkX's `connected_components` function).

   Please name your function "partition"

Exercise 4: Balance (8pt + 0.5pt for (↬))

1. (↷)In class, we considered "stable" and "unstable" triangles in social graphs. Recall that a stable triangle has exactly one or three edges corresponding to friendships, and that a triangle is unstable if it is not stable. Let us consider the "unbalance" or a graph to be the fraction of triangles that are not stable. In other words, a triangle is unbalanced if it contains two positive signs or no positive signs. A graph's unbalance is the number of unbalanced triangles over the number of total triangles.

   With NetworkX, you can add attributes to nodes and edges. To learn more, please look at the tutorial: `http://networkx.lanl.gov/tutorial/tutorial.html#edge-attributes`

   The `k_clique_communities` function from NetworkX may also be helpful for this exercise.

   Write a function that takes a NetworkX graph as input, and returns the graph's unbalance. Assume that each edge in the graph has a property "relationship" that maps to an int. The int should be "1" if the two nodes are friends and "-1" if they are enemies. Name your function "unbalance".

2. (↣)On the website slashdot.org, users can tag each other as friends or foes. This gives us excellent data to look at balance. Download the February 21st, 2009 Slashdot graph from SNAP at http://snap.stanford.edu/data/soc-sign-Slashdot090221.html.

Write a function that takes no inputs and returns the unbalance of the slashdot graph. It should read the graph inside the function. To read in the file properly, please use the following code:

```
G = nx.read_edgelist('soc-sign-Slashdot090221.txt', nodetype=int, data=(('relationship',int)
```

The name of your function should be "slashdot_unbalance".

3. (↷)We will now look at the unbalance of random graphs. Write a function that takes a float, p, as an input. Generate a random graph using the NetworkX function `gnm_random_graph`. To make it the same size as the Slashdot graph, set $n = 82144$ and $m = 549202$. Please also use a seed of 6772827383. For each edge, set the "relationship" attribute to be 1 with probability p, and -1 with probability 1-p. Then, return the graph's unbalance.

   The name of your function should be "random_unbalance".

4. (↣)Let's now compare the Slashdot graph to the synthetic graph.

   Write a function "simulate_slashdot" that takes no inputs. The function should calculate the empirical probability of an edge being a friendship, based on the Slashdot data set. That is, it should calculate the number of edges with relationship equal to one over the number of total edges. It should input this number into "random_unbalance" and return the result.

5. (↛)The answer to this question should be turned in with the written portion of the assignment. One of the two types of unstable triangles has two friendship edges and one enemy edge. Let us define a "referee" to be a node that is friends with two nodes that are enemies. In the provided data set, take a look at all of the referees. Do you notice anything special about them? For example, do they a higher or lower degree than most other nodes? Based on the graph structure, do you think they are powerful, or weak? Write down whatever you observe, and what an explanation might be.

Exercise 5: Evolution (8pt) In this problem, we'll study the evolution of graphs over time. Thanks to Jure Leskovec for inspiring this problem.

1. (↷)We've explored the "rich get richer" phenomenon of preferential attachment. In this question, we'll create a different model of graph creation.

   Write a function "preferential_edge". This function will generate a graph in the following way: Begin with a complete graph of three nodes. At each "timestep", randomly select an edge from the graph. Introduce a new node to the graph. Add an edge between your new node, and one of the end points of the edge you selected. Of the two possible nodes, attach your new node to one of them uniformly at random.

   The function will take an int as input. Run your simulation for that number of timesteps.

2. (↷)A criticism of preferential attachment models is that they put too much weight on old nodes. Nodes that arrive late might also become really popular! For example, think about percentages of books sold. Some old books certainly have sold the most copies (e.g. the Bible), but some newcomers have also sold huge percentages of the market, at least in recent years (e.g. Harry Potter).

   In this question, we'll create graphs in such a way that newcomers have a better shot of becoming popular.

Write a function "preferential_new". The function should take two inputs: a number of timesteps, and a "fitness" f. This function should generate a graph in the following way: Begin with a complete graph of three nodes. At each "timestep", add a new node to the graph. We will give these nodes a "virtual degree", to give them a better chance of competing with old nodes. With probability $\frac{1}{2}$, give the new node a virtual degree of f. Otherwise, the node should have a virtual degree of 0. Now, you'll select an old node and attach your new node to it. You will select the old node with some probability. For each node, the probability of selecting it should be proportional to the old node's actual degree plus it's virtual degree.

3. ($\curvearrowright$)Use "preferential_edge" to create a plot of time versus degree. Run "preferential_edge" for 10000 time steps. Plot the time versus degree for the degree of one of the three original nodes. On the same plot, plot the time versus degree for the node created at time step 1000.

   Please include the graph with your written homework.

4. ($\curvearrowright$)Use "preferential_new" to create a plot of time versus degree. Run "preferential_new" for 10000 time steps. Do this with fitness values of 5, 10, and 50. Plot the time versus degree for the degree of one of the three original nodes. On the same plot, plot the time versus degree for a node created after time step 1000. Make sure this second node starts with a fitness value not equal to 0.

   Please include the graphs (3 in total, one for each f) with your written homework.

Part C — Concepts at large

Exercise 6: How to avoid death by success? (7 pt) You are working in a social media start-up whose main product is a web content recommendation platform that outperform current method. The service was launch a year ago and you are now, after an initial adoption phase, in a steady increase of the number of users.

The main challenge you are facing is to keep up with the demand from an evergrowing amount of users. The current recommendation technique that you use is based on a set of shortest paths from each node to a set of landmark (which are representative nodes). The main cost per node comes from computing coefficients during each hop of these path, so that your cost of running the recommendation is a function of the number of hops from this node to a set of others.

A start-up claims to make a recommendation as accurate as yours, with a brand new technique which does not use path but only direct neighbors. The computation cost of this method per node is then a function of the total number of neighbors. You run a test and conclude that *today* with the current graph of friendship, the two methods have roughly the same computation cost, and similar accuracy.

1. ($\curvearrowright$) As you expect the graph to expand, and hence its topology to evolve in the coming months, how can you predict the evolution of the computation cost for each method? Should you buy the recommendation engine developed by this other start-up?

2. ($\curvearrowright$) This other start-up mentions that they are developing an extension that allows to reuse computed results whenever a new node is belonging to a triangle between two already computed node. Do you think that this method can result in important savings?