Homework 3

EE232E - Graphs and Network Flows

Submission: Please submit a zip file containing your codes and report to "ee232e.spring2016@gmail.com". The zip file should be named as "HW1_UID1_UID2_..._UIDn.zip" where UIDx are student ID numbers of team members. If you had any questions you can send an email to the same address.

In this assignment, we will study a real network. The data is available on this *Dropbox Link* [1]. We are going to study the properties of this real network. The data is given as a directed edge list format, where each line has three items: node1, node2 and the weight of the edge from node1 to node2.

1. Is this network connected? If not, find out the giant connected component. And in the following, we will deal with this giant connected component.

2. Measure the degree distribution of in-degree and out-degree of the nodes.

3. We would like to measure the community structure of the network. First, we need to convert it into an undirected network. Then we use `fastgreedy.community` and `label.propagation.community` to measure the community structure. Are the results of these two methods similar or not?

[1] `https://www.dropbox.com/s/oybns7fvztfy76q/sorted_directed_net.txt?dl=0`

Note that this network has edge weights, so it is not trivial to convert it from directed to undirected, especially if there are two directed edges between node i and node j. We at least have two options.

In option 1, we keep the number of edges unchanged, and just remove the directions. The resulting undirected network is not simple. `label.propagation.community` command can be used to compute a weighted, undirected, non-simple network's community structure.

In option 2, we merge the two directed edges between i and j so that the resulting network is simple. Suppose the weights are w_{ij} and w_{ji} respectively, and we set the weight for the merged undirected edge to be $\sqrt{w_{ij}w_{ji}}$. We can use both `fastgreedy.community` and `label.propagation.community` to measure the community structure.

4. Find the largest community computed from `fastgreedy.community`. Isolate the community from other parts of the network to form a new network, and then find the community structure of this new network. This is the sub-community structure of the largest community.

5. Find all the sub-community structures of the communities with sizes larger than 100.

6. Both `fastgreedy.community` and `label.propagation.community` assume that each node belongs to only one community. In practice, a node can belong to two or more communities at the same time. There is no command in igraph that can detect overlapped communities. Here we are going to use personalized PageRank to study the overlapped communities structures.

   Suppose that we start the random walk from a node i with the damping parameter 0.85 in the original directed network. We will obtain the visiting probabilities of all nodes in the network. These visiting probabilities reflect the relation between the other nodes and node i. On the other hand, we already have some knowledge of the community structure obtained from the above question. Therefore, we can compute

$$\vec{M}_i = \sum_j v_j \vec{m}_j \ ,$$

where v_j is the visiting probability of node j and $\vec{m}_j$ is its community membership computed from 3. Here, the community membership is expressed as a vector so that it can be adapted to multiple memberships. Suppose you have found n communities in 3, and $\vec{m}_j$ is a n dimensional vector with only one element being 1. The resulting $\vec{M}_i$ is a measurement of the multiple memberships.

To include all nodes in the network in the sum might make the computation amount too large. So you can take the largest 30 v_j to include in the sum.

A threshold is needed to remove the memberships that have very small values in $\vec{M}_i$. Choose a proper threshold.

Note that visit probabilities of the random walk started from a node can be obtained by a personalized pagerank with a teleportation probability distribution that is 1 to the node under examination and 0 to every other node.

Try this method to see whether you can find examples of nodes that belong to multiple communities. List at least 3 examples if you find such nodes.