MLSS '13: Network Modeling and Information Propagation

Manuel Gomez Rodriguez

1 Introduction to SNAP: Network Modeling

Building the MLSS package

First, you need to ssh to the MLSS machine (Windows users, please download Putty):

```
$ ssh mlssX@172.16.172.16
(password: mlss379)
```

Then, download the MLSS practical code package, unpack it and compile it:

```
$ wget ftp://172.16.172.16/pub/networks/code-networks.tgz
$ tar zvfx code-networks.tgz
$ cd code-networks/
$ make
```

Several warnings will appear when you compile the code package. As long as you do not get an Error message, you are fine. You can also download the code locally to your laptop, it will help to have a local copy to look up some library functions and files. Now you can go to the subfolder `mlss-13/` where you will start creating networks and coding!

Creating graphs and networks

We will start by creating manually two small examples of a graph and a network (*i.e.*, a graph with some data stored in the nodes and/or edges) using SNAP:

```
$ ./generate_network
```

The above program should print out the nodes and connectivity, and generate four files `graph.png`, `graph.dot`, `network.png` and `network.dot`.

Question 1.1. Do you recognize the graph and the network structures from the print out? Draw the graph and the networks by hand. Check whether you are right by downloading to your computer graph.png and network.png via ssh. If your computer is a linux/macos, open a console, go to the folder you would like to download the images and type:

```
$ scp mlssX@172.16.172.16:~/code-networks/mlss-13/graph.png .
$ scp mlssX@172.16.172.16:~/code-networks/mlss-13/network.png .
```

If you have windows, download winscp at http://winscp.net/. ■

Question 1.2. Open the source code, generate_network.cpp, and find out how the code works. Simple editors which are available in the MLSS machine are vim, joe or pico to open the file. To better understand the code, you can check out a quick intro to SNAP at http://snap.stanford.edu/snap/quick.html. ■

Coding 1.1. Change generate_network.cpp to create a different graph and network. In particular, the code you need to modify is delimited by TODO comments. Once you have changed the file, compile the code by running make at the current folder (mlss-13/). After re-running generate_network, check whether the outputted files graph.png and network.png look like you thought.

If you create graphs and networks with a large number of nodes or edges, you will notice that graph.png and network.png, created using Graphviz[1], soon become messy. Gephi will help us to visualize such large graphs later. ■

Creating large random graphs and networks

Now, we will generate large random networks that mimic the structure of real-world social networks which we will use in the next section to simulate propagation processes or cascades. We initially consider two models of directed real-world social networks: the **Forest Fire** (scale free) model [2] and the **Kronecker Graph** model [9]. We will generate networks that have a weight associated to each edge, which we will use for our propagation models. To make it easier for you, we have coded a program (generate_random_network) which generates both Forest Fire and Kronecker networks.

Question 1.3 Generate several Forest Fire networks with different network parameters and compare their properties (in- and out-degree distribution, clustering coefficient, number of triangles, etc...) using compute_properties. In particular, you should try different burning probabilities (option -g), number of nodes and edge weights range (options -la and -ua; generated uniformly at random). To help you get started, check the following command line, which generates a For-

[1]http://www.graphviz.org

est Fire network with 1,000 nodes with a burning forward probability of 0.2 and a burning backward probability of 0.17:

```
$ ./generate_random_network -t:1 -g:"0.2;0.17" -n:1000 -o:"ff-
    network" -st:1 -sg:1
```

The program will output two files: `ff-network.txt` and `ff-network.gexf`, which include a text and Gephi version of the network. Once you have generated several networks, can you tell what are the burning probabilities controlling for? ■

You can learn how the Forest Fire model actually works by reading Sections 4.2.1. and 4.2.2 of Leskovec et al. ([10], `http://www.cs.cmu.edu/~jure/pubs/powergrowth-tkdd.pdf`).

Question 1.4 Generate several Kronecker networks with different parameter matrices, such as Erdős-Rényi random [4] (parameter matrix $[0.5\ 0.5; 0.5\ 0.5]$), hierarchical [3] ($[0.9\ 0.1; 0.1\ 0.9]$) and core-periphery [11] ($[0.9\ 0.5; 0.5\ 0.3]$) networks. You should choose 2^n nodes and any number of edges you wish by setting the options `-n` and `e`. Compare their properties using `compute_properties`. To help you started, we provide you the command line to generate a Erdős-Rényi random network with 512 nodes and ∼1024 edges[2] drawn from the Kronecker Graph model:

```
$ ./generate_random_network -t:0 -g:"0.5␣0.5;␣0.5␣0.5" -n:512 -e
    :1024 -o:"random-network" -st:1 -sg:1
```

As previously, the command will output two files: `random-network.txt` and `random-network.gexf`. Once you have generate several of these networks, can you tell which network properties are more influenced by the choice of parameter matrix? ■

You can learn how the Kronecker graph model actually works by reading Section 3.1 of Leskovec et al. ([9], `http://arxiv.org/pdf/0812.4905v2.pdf`).

Coding 1.2 (optional): implement another model of directed random networks using SNAP. You will only need to edit `mlss.cpp` at the `TODO` comment (there is only one within the file). It may be helpful to use some of the random graph generators provided by SNAP, check out `code-networks/snap-core/ggen.h`. You will need to call the functions writing something like `TSnap::method(Graph, ...)`. Do not forget to compile the code again by typing `make` after you finish editing.

Take into account that some of the graph generators produce an undirected graph (`PUNGraph`), and you will have to convert it to a directed graph (`PNGraph`).

[2]The Kronecker Graph model will create 1,024 edges but some of them may be self loops and are deleted.

In those cases, decide the direction of each original undirected edge uniformly at random using `TFlt::Rnd.GetUniDev()`. ■

2 Introduction to Gephi: Network visualization

Setting up Gephi

Download Gephi to your personal computer from `http://www.gephi.org/users/download/`[3], install it and open it.

Layout, properties and manipulation

Use Gephi to open any of the `.gexf` network files which are located in the folder `code-networks/mlss-13/data/`. They are networks based on real data. Additionally, you can also open the random networks you generated previously or generate new ones by running again `./generate_random_network` (use `-sg:1` to generate a Gephi file. You can also convert the networks text files (in case you did not use `-sg:1`) using `convert_to_gephi`.

Question 2.1. Visualize the networks using different layout algorithms. For this, you should use the Layout box on the left bottom corner. Which of them do you think works *better*? Can you draw any conclusions based on a particular layout? Can you guess what do the real networks represent? If you would like to know more about Gephi layouts, have a look to the tutorial `http://gephi.org/tutorials/gephi-tutorial-layouts.pdf`.■

Question 2.2. Manipulate the networks. For this, use the Filters box on the right top corner. Have a look at the topology menu and try several filters (e.g., degree range, giant component, ego networks). You will need to drag and drop it the filter to the query box on the right bottom corner. Find out how to stack filters (create subfilters).■

Question 2.3. Compute properties of the network. For this, go to the Window menu and select Statistics to open up the Statistics menu. Compute several statistics, including modularity. Then, partition the network and rank nodes using the Partition and Rank boxes on the top left corner; use different measurements for the partition and rank. Do you find clear partitions or clusters in the networks? Which measures work *better*? Do the networks have a hierarchical or core-periphery structure?■

[3] There are Linux, macOS and Windows versions

You can learn more about Gephi by reading the wiki at `http://wiki.gephi.org/`. You can also find an example of a research project which uses Gephi extensively at `http://snap.stanford.edu/infopath/`, where all information networks images (`http://snap.stanford.edu/infopath/graphs.html`) and videos of dynamic information networks (`http://snap.stanford.edu/infopath/videos.html`) have been generated with Gephi.

3 Information Propagation and Influence Maximization

Until here, you have learned a bit about SNAP and Gephi, and get to know two mathematical models of social and information networks. Now, we will use this knowledge to study information propagation over networks, and study the influence (spread) maximization problem. In particular:

1. You will simulate the propagation of contagions over networks using a widely known probabilistic models of influence and information propagation: the discrete time independent cascade model [6]. You can think of a *contagion* as an information unit which appears at some node of a network and then spreads like an epidemic from node to node over the edges of the underlying network. In case of information diffusion, the contagion represents a piece of information [8] and infection events correspond to times when nodes mention or copy the information from one of their neighbors in the network.

2. You will solve the influence maximization problem: you will look for the most influential source node set of a given size in a network. A contagion that starts spreading in such an influential set of nodes is expected to reach the greatest number of nodes in the network.

3. If time remains, given a fixed source set we will look for the optimal set of edges to add to a network in order to reach the greatest number of nodes in the network.

Discrete-time independent cascade model

The remainder of the practical use the discrete-time independent cascade model, and it will help you to understand it well. The model works as follows: A spreading process (or cascade) starts when a source node set A becomes infected at epoch $t = 0$. Then, source nodes have a single chance to try to infect their children (*i.e.*, neighbors that they can reach directly through an outgoing edge) at epoch $t = 1$ with some probability. When a child becomes active, it has a single chance of activating each currently inactive children at epoch $t = 2$, and so on. The activation attempts succeeds with probability p_{vw}, where v denotes parent and w denotes children. Here, note that the time is modeled only implicitly through the epochs – we will now extend the independent cascade model to continuous time domain.

Question 3.1 (optional). Look at the implementation of the independent cascade model in the function `GenCascadeIC(...)` of `mlss.cpp`.■

Influence Estimation

Given a spreading process that started in the set of source nodes A, we define $N(A)$ as the number of nodes infected at the end of the spreading process and then define the influence function $\sigma(A)$ as the average total number of nodes , *i.e.*, $\sigma(A) = \mathbb{E}N(A)$. Now, we will simulate spreading processes and estimate influence.

Question 3.2. Generate a Kronecker core-periphery network with 128 nodes, $\sim$ 256 edges and edge weights (edge probabilities) uniformly distributed between 0 and 0.5. Then, generate 1,000 spreading processes (or cascades) over the network using a random source node set of size 2 (fixed for all cascades). To help you get started, we tell you how to generate the cascades:

```
$ ./simulate_ic -i:"your-core-periphery-network-namefile.txt" -t
    :1 -ns:5 -c:1000 -cg:25 -sg:1 -o:"core-periphery-cascades-128
    "
```

where `your-core-periphery-network-namefile.txt` should be the filename of your core-periphery network. The command will print out the (random) source set, average influence and standard error based on the simulated 1,000 cascades. It will also output several files, including two plots and a Gephi file.

Have a look at the plots, and find out what they show. The code automatically tried to fit the quantities of one of the plots to a functional relationship. Do you know which type of relationship?

Question 3.3. Open the Gephi file of the cascades you just generated in the previous question using Gephi. Then, press the bottom "Enable Timeline", that will help you to visualize the cascades spreading. For this, look for the small shell on the left bottom and play around with different values for the custom bounds and the play settings. Push the play bottom also on the top left, and you should be able to observe cascades spreading over the network for the appropriate bounds and play settings. It may help to choose the right layout.■

Question 3.4. Explore the effect of choosing other source sets. In particular, try different fixed (deterministic) source sets of difference sizes by modifying options `-t:` and `-ns:`. Node id's will be distributed from 0 to 127. Do you find large differences in average influence or cascade size distribution for different source sets?■

Question 3.5. Generate networks with different characteristics, *i.e.*, different network types, edge densities, and edge weights and simulate spreading processes over them. Do you notice large differences in average influence or cascade size distributions? Which are the characteristics that have a greater impact?■

Question 3.6. Here, we will study some property that may help you to come up with algorithms to select optimal source sets which maximize the average influence. Select at random a node u and two source sets S_1 and S_2 with 1 and 10 sources respectively, not including u. Compute the average influence for S_1, S_2, $S_1 \cup u$ and $S_2 \cup u$. Repeat this several times for different u, S_1 and S_2 and save the average influence each time. Can you guess a general property? Do you know whether the property you are observing has a name? Why do you think this is happening? Hint: A lecture you will have on Saturday will be related.■

Influence Maximization: nodes

Now that you know how to simulate spreading processes and understand them, try to find the optimal set of sources that maximize the average influence in a given network. We have implemented a baseline that chooses the K nodes with highest out-degree in `maximize_sources_ic.cpp`.

Coding 3.1. You need to think of other methods to choose influential nodes and implement them in `maximize_sources_ic.cpp`. Your code should go in the space delimited by `TODO` comments. You may like to rank nodes by (i) some network property you can compute directly from the network, (ii) some measure computed on simulated spreading processes, or (iii) hybrid approach combining (i) and (ii). Can you think of any related theoretical computer science NP-hard problem? This may help you to find some efficient approximate solution to the problem with provable theoretical guarantees. Your implementation should work for any number of source nodes K.■

Question 3.7. Test your method(s) in one (or more) of the random networks you have generated in previous sections, generate new ones or try your method(s) in the networks located at `code-networks/mlss-13/data/`. Note that networks in `code-networks/mlss-13/data/` are unweighted and you will have to set an edge probability with option `-p:` both in `maximize_sources_ic` and `simulate_ic`. Once you believe you have found a good method, join the competition.■

Competition. Apply your methods for finding optimal source sets of size $K =$

$1 \ldots 10$ to the networks in `code-networks/mlss-13/data/`. Use option `-p:0.5` both in `maximize_sources_ic` and `simulate_ic`. For every network, create 10 files, one per K, in which you write the node id's (or node id for $K = 1$) of your solution in a single line, separated by a semicolon (;). For example, for $K = 4$ the file should contain only one line:

10;2;3;4

For the filenames, use the following format. For example, for the network `mlss-network.txt`, $K = 3$ and imagine your surnames are Gomez and Rodriguez, then the filename should be called `mlss-network-3-gomez-rodriguez`. Everytime you have a better solution for a network and a fix K, move the file to the folder `/home/networks/results/` at the MLSS machine.■

Whenever you believe you have mastered influence maximization or you would like to try something perhaps more challenging, move on to the next section.

Influence Maximization: edges

Here, we will try to find the optimal set of edges that added to the network maximize the average influence on a network given any random source set. For the added edges, we assume they are always active, *i.e.*, its edge probability is 1.0. We have implemented a baseline that chooses K random edges starting from nodes with the smallest out-degree in `maximize_edges_ic.cpp`.

Coding 3.2. You need to think of other methods to choose optimal edges and implement them in `maximize_edges_ic.cpp`. Your code should go in the space delimited by `TODO` comments. Do you think it is an easier or a harder problem than finding the optimal source set?■

References

[1] N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner Press, 2nd edition, 1975.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[3] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[4] P. Erdős and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science*, 5:17–67, 1960.

[5] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the Temporal Dynamics of Diffusion Networks. In *ICML '11: Proceedings of the 28th International Conference on Machine Learning*, 2011.

[6] D. Kempe, J. M. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[7] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. In *EC' 06: Proceedings of the eigth International Conference on Electronic Commerce*, 2006.

[8] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[9] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

[10] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[11] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08*, 2008.

[12] E. M. Rogers. *Diffusion of Innovations*. Free Press, New York, fourth edition, 1995.