

Social Networks and Online Markets

Homework 1

Due: 22/5/2022, 23:59

Instructions

You must hand in the homeworks electronically and before the due date and time.

The first homework has to be done by each **person individually**.

Handing in: You must hand in the homeworks by the due date and time by an email to `aris@diag.uniroma1.it` that will contain as attachment (**not links to some file-uploading server!**) a .zip or .pdf file with your answers.

After you submit, you will receive an acknowledgement email that your homework has been received and at what date and time. If you have not received an acknowledgement email within 2 days after you submit then contact Aris.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

If you need any technical help, you can email Andrea Mastropietro: `mastropietro@diag.uniroma1.it`.

For information about collaboration, and about being late check the web page.

Problem 1. Consider a graph created according to the Erdős-Rényi $G_{n,p}$ random-graph model.

1. What is the probability that a graph created according to $G_{n,p}$ contains exactly only one triangle (three nodes connected with each other and no other edges exist)?
2. What is the probability that all the n nodes are connected in a line (with no other edges present)?
3. Assume that we create a graph according to $G_{n,p}$, what is its expected number of edges?

Problem 2. In this homework you need to implement some of the models that we have seen, and measure experimentally some of the graph properties. Of course, each model has its own parameters.

1. The Erdős-Rényi G_{np} random graph model. Parameters:
 - n : number of nodes
 - p : probability of an edge to exist
2. The Barabási-Albert preferential attachment model. Parameters:
 - n : number of nodes
 - ℓ : number of neighbors that a newly arrived node comes with.

Assume that the initial graph is a single node. If it makes your life easier, if multiple edges fall on the same node you can ignore the multiple edges.

The goal is to understand these models, for different parameter combinations. Therefore, for each of these models, you should experiment for different values of the parameters. However, the parameter n should always be high (at least $10K$ but it can be up to the order of millions depending on the power of your computer).

For each of the parameter combinations, compute and report in an organized way:

- Degree distribution (you should display it with a plot)
- Diameter
- Clustering coefficient

For each set of parameters create different graphs and check if the behavior and the values you obtain are the same for each of these graphs.

You are allowed to use a library (such as NetworkX in case you use Python) to handle the graph or compute the graph functions. However, you should implement yourselves the graphs and not use library or other code for that. If you have any questions about what is allowed, feel free to ask Aris.

Now calculate the same three parameters for some of the graphs in the SNAP library: <http://snap.stanford.edu/data/index.html>. Compare the output of your models with those of the real networks and comment on them.

Notice that none of these models has values for the above three values that are realistic. By taking inspiration from them models, try to create a new model that has more realistic values for of these parameters. To create the model you may do whatever you want (e.g. you can combine them, or use some of the other concepts we said in the class).

You should hand in a zip file containing the code of your program results and a report, in pdf format, which will contain the results of your findings. In the report, for each model, you should display a table with the different combinations and the values that you obtain, and for each parameter combination a plot depicting the degree distribution.

As an advice for when you test your programs, first try with smaller values of n to make sure that your program works (e.g., 500 or 1000), then you should try the higher ones.

Problem 3. You are a company that wants to monitor what users post in some site. The users in the site are connected through a network. To find out what users' post you can *follow* some users; when you follow a user, you know what the user posts as well as what her friends post. Assume that you have a budget of k users that you can follow. Provide an approximation algorithm for selecting the set of (at most k) users to follow, such that you maximize the number of users that you know what they post. Assume that you have complete knowledge of the network.

Problem 4. Recall that, given an undirected graph $G = (V, E)$, the densest subgraph $D \subseteq V$ is the set that maximizes

$$\frac{|E \cap (D \times D)|}{|D|}$$

and the sparsest cut $S \subseteq V$ is the set that minimizes

$$\frac{|E \cap (S \times (V \setminus S))|}{\min\{|S|, |V \setminus S|\}}.$$

Give an example of a graph such that $D = S$ and of a graph such that $D \notin \{S, V \setminus S\}$.

Problem 5. Graph Learning on Communication Networks

It is well known that graphs are everywhere. Complex systems are defined by the relationships among their elements and so they can be represented as networks. Such complex systems cover all scales, from protein-protein interaction networks in cells to social interaction networks among people. For this exercise, you will explore and work with a communication network. In this network nodes are people that are connected with an edge if they had an interaction via email. You can find the dataset at: <https://snap.stanford.edu/data/email-Eu-core.html>

The task you have to perform is **community detection** via **node classification**. You have to classify a node/person as belonging to a **department** (multi-class classification). For this exercise you need to define Graph Neural Network. You can use the notebook from the lab as a reference (you can get ideas from it but cannot use the very same code!). We saw GCNs in class, but you are free to try different architectures such as GATs, GINs if you like :-).

1. **Data Preparation.** You will start by preparing your data:

- **Download data** from the website;
- **Data processing:** In the webpage provided you will find two files: an edge list with email communications and a file with labels (communities/classes) for each person. Create a PyTorch Geometric dataset as show in the lab.

2. **Model training.** Compare the performances of your model using different features for the nodes: you can use no features, centrality measures, Node2Vec embeddings and so on. Compare **at least** two different configurations.

Remember to split your data into train, validation and test sets. The performances on the test are the ones that matter! Compare the classification performances in terms of **accuracy**, **recall**, **precision** and **F1-score**. Which features were able to better characterize the graph and deliver the highest performances? Mind that the answer may not be obvious. You have to hand in the code along with a report (about 3 pages, **max 5**) in which you describe all the steps made (plots are welcome). In particular, show how you handled the data, describe the model, the features and provide tables with the performance metrics. Plot also a **confusion matrix**. Feel free to add any comment/observation you think to be relevant.

3. **Bonus – Comparing different models.** As a bonus, you can implement more models (either GNN-based models or standard machine learning models) and compare their performances, properly commenting your results. You can use 1 extra page in the report for this part.