Social Networks and Online Markets

Homework 1

Due: 9/6/2024, 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, for Problem 6, you can email Gianluca De Carlo: `decarlo@diag.uniroma1.it`. For other questions, email Aris.

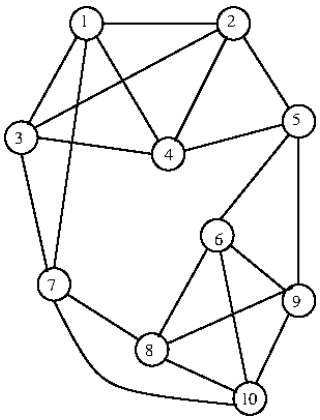
For information about collaboration, and about being late check the web page.

Problem 1. Consider the following modification of the Barabassi–Albert preferential attachement model that we did in class: When a new node arrives at time t again it comes with ℓ edges. However, this time each edge selects a node v with probability proportional to the degree d_v plus a constant c, that is, the probability equals

$$\frac{d_v + c}{(t-1)(2\ell + c)},$$

where $c \geq -\ell$, as we describe at the end of Chapter 4 in the notes (so for $c = 0$ this is the Barabassi-Albert model). Show that the degree distribution that we obtain as $t \to \infty$ is approximately a power law with exponent $3 + c/\ell$.

Problem 2. We are given the following graph, $G = (V,E)$.



1. Find the densest subgraph using the greedy algorithm we saw in class.

2. Find a minimum cut.

3. Demonstrate (by calculating λ_2, $\phi(G)$, etc.) that Cheeger's inequalities hold for this graph.

4. Find the cut that satisfies Part 3 (and show that it does).

You may use a computer to compute eigenvalues and eigenvectors, but you must specify in your solution how (what program and what code/commands you used).

Problem 3. As Aris Gionis mentioned in class, an interesting phenomenon in social networks is that a random person's expected degree is smaller than the degree of her peers: "Your friends are more popular than you are!" Given an undirected graph $G = (V,E)$, select a random node and let X be the random variable that equals to its degree and Y to be the random variable that equals the average degree of the node's neighbors.

1. Prove that $\mathbf{E}[X] \leq \mathbf{E}[Y]$.

2. When do we have that $\mathbf{E}[X] = \mathbf{E}[Y]$?

Hint. Prove that for any $a,b \neq 0$ we have that $\frac{a}{b} + \frac{b}{a} \geq 2$.

Problem 4. We are monitoring a graph, which arrives as a stream of edges $\mathcal{E} = e_1, e_2, \ldots$. We assume that exactly one edge arrives at a time, with edge e_i arriving at time i, and the stream is starting at time 1. Each edge e_i is a pair of vertices (u_i,v_i), and we use V to denote the set of all vertices that we have seen so far.

We assume that we are working in the *sliding window* model. According to that model, at each time t only the w most recent edges are considered *active*. Thus, the set of active edges $E(t,w)$ at time t and for window length w is

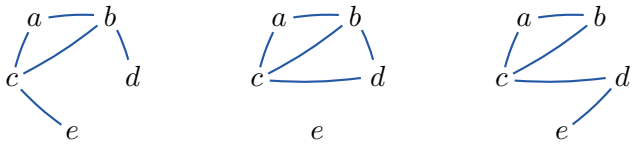
$$E(t,w) = \begin{cases} e_{t-w+1}, \ldots, e_t, & \text{if } t > w, \\ e_1, \ldots, e_t, & \text{if } t \leq w. \end{cases}$$

We then write $G(t,w) = (V,E(t,w))$ to denote the graph that consists of the active edges at time t, given a window length w.

As an example, given the stream of edges

$$e_1 = (c,e), e_2 = (b,d), e_3 = (a,c), e_4 = (c,b), e_5 = (a,b), e_6 = (c,d), e_7 = (d,e)$$

the graphs $G(5,5)$, $G(6,5)$ and $G(7,5)$ are shown below:



$G(5,5)$ $G(6,5)$ $G(7,5)$

Notice that for all $t \geq w$ the graph $G(t+1,w)$ results from $G(t,w)$ by adding one edge and deleting one edge.

We want to monitor the connectivity of the graph $G(t,w)$. In other words, we want to design an algorithm that quickly decides, at any time t, if the graph $G(t,w)$ is connected. In the previous example, the graphs $G(5,5)$ and $G(7,5)$ are connected, while the graph $G(6,5)$ is not connected.

1. Propose a streaming algorithm for deciding the connectivity of $G(t,w)$.

   Hint: An efficient streaming algorithm takes advantage of the fact that the graph $G(t+1,w)$ changes very little compared to $G(t,w)$. Therefore, our algorithm should be able to efficiently update the connectivity of $G(t+1,w)$ when a new edge e_{t+1} arrives at time $t+1$, given that the connectivity of $G(t,w)$ has already been computed.

2. How much space does your algorithm use? There is a simple approach, which uses space $O(n^2)$, and a smarter one that uses space $O(n)$, where $n = |V|$.

3. What is the update time of your algorithm?

   Hint for 2 and 3: The space of your algorithm is the maximum amount of space used at any given moment. The update time is the time needed to compute the output at time $t+1$, given the state at time t and the new edge e_{t+1} that arrives at time $t+1$.

   You should provide your answer using the $O(\cdot)$ notation, written as a function of the window length w and the number of vertices n.

Problem 5. In most opinion-formation models it is assumed that the difference of opinions of two individuals who interact in the social graph decreases during the opinion-formation process. In other words, people's interaction leads to improving agreement. However, in real life, the opposite phenomenon is observed: individuals whose initial opinions are sufficiently far apart, tend to disagree more when they interact with each other. In other words, the difference of opinions of two individuals who disagree "sufficiently enough" tends to increase even more, during the opinion-formation process. This is known as the back-fire phenomenon.

Propose an opinion-formation model that incorporates the idea of back-fire. State your assumptions and motivate your choices. Argue why the model would lead to back fire.
Extra credit: Perform simulations to illustrate that your model behaves as you expect it to do.

Problem 6. Applying Graph Neural Networks for Node Classification and Link Prediction Using the PubMed Dataset
Dataset Overview:

In this exercise, we utilize the PubMed dataset, a citation network with nodes representing scientific publications from the PubMed database, primarily in the biomedical field. Edges indicate citations between these publications. Each node features a TF-IDF weighted word vector from the publication's abstract and a class label that denotes the publication's subject category. You can otain the dataset by using the *Planetoid* package: `https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.Planetoid.html`
Tasks:

1. Node Classification: Construct and train a Graph Neural Network (GNN) to categorize publications within the PubMed dataset into their corresponding subject categories. Test different GNN architectures like GCN, GraphSAGE, and GAT.

2. Link Prediction: Build a GNN-based model to predict the presence of citation links between publications.

Exploratory Data Analysis (EDA):

Conduct an EDA to familiarize yourself with the PubMed dataset's characteristics. Key aspects to explore include:

1. Network Structure: Assess the number of publications (nodes), citation links (edges), average citations per publication, citation distribution, and clustering coefficient.
2. Node Features: Examine the spread of TF-IDF values and explore potential correlations between word frequencies and publication categories.
3. Class Distribution: Analyze the quantity of publications within each subject category and check for any class imbalances.

Evaluation Metrics:

1. Node Classification: Measure the effectiveness of your models using metrics like accuracy, precision, recall, and F1-score.
2. Link Prediction: Evaluate your model's proficiency at distinguishing between existing and non-existing links using metrics such as AUC-ROC, F1-score, and ranking metrics (e.g., MRR, Hits@K).

Challenge: Aim to surpass these benchmark scores, achieved by Gianluca:

1. Node Classification Accuracy: 80.2%
2. Link Prediction AUC-ROC: 91.0%.

Deliverables:

1. Code Implementation: Submit a Jupyter Notebook or Python script that includes your code, implementation specifics, and outcomes.
2. Project Report: Provide a succinct report detailing your methodology, discoveries, obstacles faced, and the results achieved in both tasks.

Help. If you need support for this exercise, you can contact Gianluca: `decarlo@diag.uniroma1.it`.