Homework for the course

Algorithmic methods for mining large graphs

Bertinoro International Spring School 2016

Aristides Gionis, Aalto University

Due: Sun, Apr 10, 2016

Return your answers by email to `aristides.gionis@aalto.fi`

Typed solutions are encouraged, especially using LaTeX.

It is allowed to discuss the problems with other students, however, you should write the answers by yourself. Mention the name of the people that you discussed the problems.

Partial credit will be given for partial solutions, but not for long off-topic discussion that leads nowhere. Overall, think before you write, and try to give concise and crisp answers.

If something is unclear, you can write an email to the course instructor for clarifications. Please `cc` the whole class mailing list, as your question maybe relevant to the other students, as well.

If you decide to do a project, instead of the homework, contact the course instructor immediately, make a project proposal, and agree on a plan and a deadline.

Problem 1

We mentioned in class that citation networks (networks representing citations between research articles) are, in theory, *directed and acyclic graphs* (dags).

Explain why we expect citation networks to be directed and acyclic.

However, in practice, we expect that citation networks have a small number of cycles. Why?

Assume that we want to test whether a given citation network has cycles. Propose an algorithm to detect if a directed graph has a cycle. What is the running time of your algorithm?

Problem 2

Assume that a graph is stored in the "edge incidence" model, that is, all edges incident to one vertex are stored sequentially.

Our sampling algorithm for estimating the triangle coefficient relies on sampling one connected triple of the graph with *uniform probability*. Namely, all connected triples should have *equal* probability to be sampled.

Describe a three-pass algorithm to sample a connected triple uniformly at random.

Note: This is already outlined in the slides, you are asked to describe the method in more detail.

Problem 3

Describe an algorithm to compute *exactly* the *diameter* of a given graph. What is the running time of your algorithm?

Describe an algorithm that uses the Flajolet-Martin sketching scheme to compute *approximately* the *diameter* of a given graph. What is the running time of your algorithm? How can you control the trade-off between accuracy and speed?

Problem 4

Consider X to be the space of *colored graphs*. Each object $G \in X$ is a graph $G = (V, E, c)$ where V is a set of edges, $E \subseteq V \times V$ is a set of edges, and $c : V \to \{1, \ldots, \ell\}$ is a "coloring" function that maps each vertex of the graph G to one of ℓ distinct colors.

Question 4.1. Propose a distance function for pairs of graphs $G_1 = (V, E_1, c_1)$ and $G_2 = (V, E_2, c_2)$. Note that the two graphs are defined over the same set of vertices V.

You are free to define your distance function in any way you want, and you can be as creative as you want. However, you should provide a justification for your definition, and you should argue that your proposed distance function is "intuitive".

Provide an example of how to use such a distance function in a real-world application scenario.

You should also try to define your distance function so that it satisfies all the *metric* properties (non-negativity, identity, symmetry, and triangle inequality).

Is your distance function a metric?

How do you compute the function your defined, and what is the running time?

Question 4.2. Propose a distance function for pairs of graphs $G_1 = (V_1, E_1, c_1)$ and $G_2 = (V_2, E_2, c_2)$, where $V_1 \cap V_2 = \emptyset$. Your function should be flexible to the fact that the two vertex sets V_1 and V_2 are disjoint, for example, when computed on the following two graphs it should return 0.

G_1 G_2

In this example G_1 is defined over $V_1 = \{1, 2, 3, 4\}$ and G_2 over $V_2 = \{5, 6, 7, 8\}$.

Consider again all the issues asked in the previous question: (i) what is the intuition of the distance function your proposed? (ii) provide a real-world example of using such a distance function, (iii) is your distance function a metric? (iv) how do you compute the distance function you proposed? (v) what is the running time of your algorithm?

Problem 5

Show that the graph cut function is submodular.

Problem 6

Question 6.1. Show that a symmetric matrix has real eigenvalues and orthogonal eigenvectors.

Question 6.2. Let $G = (V, E)$ be an undirected d-regular graph, let A be the adjacency matrix of G, and let $L = I - \frac{1}{d}A$ be the normalized Laplacian of G. Prove that for any vector $\mathbf{x} \in \mathbb{R}^{|V|}$ it is

$$\mathbf{x}' L \mathbf{x} = \frac{1}{d} \sum_{(u,v)\in E} (x_u - x_v)^2.$$

Problem 7

Random walks on undirected graphs: Consider a fully-connected undirected graph $G = (V, E)$. Consider a random walk performed on the graph G.

Question 7.1. Prove that the probability of visiting a vertex v in the random walk is proportional to the degree of v in G.

Random walks on directed graphs: Consider an directed graph $G = (V, E)$, which is strongly connected, that is, for all vertices u and v there is a directed path from u to v. Consider a random walk performed on the graph G. The objective of this part of the question is to understand that in property claimed by Question 7.1. does not necessarily hold for directed graphs.

Question 7.2. Provide an example of a strongly-connected directed graph in which there is a vertex with very small degree and very high probability of being visited in the random walk.

Question 7.3. Provide an example of a strongly-connected directed graph in which there is a vertex whose probability of being visited in random walk is *exponentially* small.

Problem 8

We are monitoring a graph, which arrives as a stream of edges $\mathscr{E} = e_1, e_2, \ldots$. 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

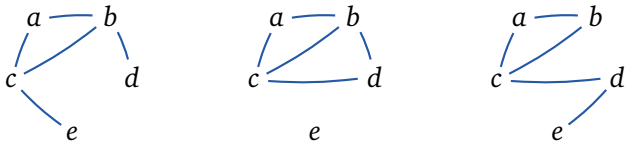
$$E(T,W) = \begin{cases} e_{T-W+1}, \ldots, e_T, & \text{if } T > W, \\ e_1, \ldots, e_T, & \text{if } T \le W. \end{cases}$$

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.

Question 8.1. Propose a streaming algorithm for deciding the connectivity of $G(T,W)$.

Hint for 8.1: 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.

Question 8.2. How much space does your algorithm use?

Question 8.3. What is the update time of your algorithm?

Hint for 8.2 and 8.3: 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 $\mathcal{O}$ notation, written as a function of the window length W and the number of vertices $N = |V|$.