Efficient Approximate PageRank

TAs: Longqi Cai, Tzu-Ming Kuo

Out 10/20/2016 14:50
Due 11/3/2016 23:59

Guidelines for Answers: Please answer to the point. Please state any additional assumptions you make while answering the questions. You need to submit a tar file containing source files and a pdf version of report separately to autolab. Please make sure you write the report legibly for grading.

Rules for Student Collaboration: The purpose of student collaboration in solving assignments is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is allowed to seek help from other students in understanding the material needed to solve a homework problem, provided no written notes are taken or shared during group discussions. The actual solutions must be written and implemented by each student alone, and the student should be ready to reproduce their solution upon request. You may ask clarifying questions on Piazza. However, under no circumstances should you reveal any part of the answer publicly on Piazza or any other public website. Any incidents of plagiarism or collaboration without full disclosure will be handled severely.

Rules for External Help: Some of the homework assignments used in this class may have been used in prior versions of this class, or in classes at other institutions. Avoiding the use of heavily tested assignments detracts from the main purpose of these assignments, which is to reinforce the material and stimulate thinking. Because some of these assignments may have been used before, solutions to them may be available online or from other people. It is explicitly forbidden to use any such sources or to consult people who have solved these problems before. You must solve the homework assignments completely on your own. We will mostly rely on your wisdom and honor to follow this rule. However, if a violation is detected, it will be dealt with harshly.

- Did you receive any help whatsoever from anyone in solving this assignment? Yes/No
- If you answered yes, give full details:_____
(e.g. "Jane explained to me what is asked in Question 3.4")
- Did you give any help whatsoever to anyone in solving this assignment? Yes/No
- If you answered yes, give full details:_____
(e.g. "I pointed Joe to section 2.3 to help him with Question 2")

1 Background

A "snowball sample" of a graph starts with some set of seed nodes of interest, and then repeatedly adds some neighbors of the seed nodes and their incident edges. The idea is to come up with some version of the "local neighborhood" of a node so that one can do analysis of, say, the Facebook friend graph of a small subcommunity. Doing this is unfortunately tricky for a large graph. This assignment uses some of the ideas in a 2006 FOCS paper "Local graph partitioning using PageRank vectors" by Andersen, Chung, and Lang to do a sort of snowball sampling of a large graph—one which you have on disk.

Some notation first.

- G is a graph, V the vertices, E the edges, $n = |V|$, and $m = |E|$.
- We will use indices i for vertices when convenient, so v_i has index i.
- $d(v)$ is the degree of $v \in V$, and D is a matrix with $D_{i,i} = d(v_i)$.
- χ_v is a unit vector with all weight on vertex v_i.
- A is an adjacency matrix for G. $W = \frac{1}{2}(I + D^{-1}A)$ is a "lazy random walk" matrix, where there is probability $1/2$ of staying at vertex v, and probability $1/2$ of moving to some other vertex u connected to v.
- We consider a "lazy" version of personalized PageRank, which is the unique solution to

$$pr(\alpha, s) = \alpha s + (1 - \alpha) pr(\alpha, s) W \tag{1}$$

where α is a "teleportation constant" and s is a "seed" distribution. Note that s and $pr(\alpha, s)$ are row vectors.

- It's easy to show that

$$pr(\alpha, s) = \alpha s + (1 - \alpha) pr(\alpha, sW) \tag{2}$$

(Note the subtle difference from Eq 1 - this statement is true, but not obvious.)

2 Approximating PageRank with "pushes"

The personalized PageRank vector $pr(\alpha, s)$ can be incrementally approximated as the following.

We maintain a pair of vectors p (the current approximation) and r (the "residual"). Initially $r = \chi_v$ and p is an all-zeros vector. This guarantees the following equality holds.

$$p + pr(\alpha, r) = pr(\alpha, \chi_v) \tag{3}$$

Now we repeatedly apply Eq 2 to move probability mass from r to p, but maintain the equality in Eq 3.

We define a push(u, p, r) operation as

$$\begin{aligned} p' &= p + \alpha r_u \\ r' &= r - r_u + (1 - \alpha) r_u W. \end{aligned}$$

where u is a node with non-zero weight in r and r_u is a vector which is zero everywhere except with weight $r(u)$ on node u. A push operation moves α of u's weight from r to p, and then distribute the remaining $(1-\alpha)$ weight within r as if a single step of the random walk associated with W were performed. This operation maintains the equality Eq 3. Notice that you only need $d(u)$ to decide if you push, and the neighbors of u to perform push operation.

Let $apr(\alpha, \epsilon, v_0)$ be an "approximate PageRank" vector which is the result of performing "pushes" repeatedly, in any order, until there is no vertex u such that $r(u)/d(u) \geq \epsilon$ (and then using p as the approximation). Then you can show that

- Computing $apr(\alpha, v_0)$ takes $O(\frac{1}{\epsilon\alpha})$ time

- $\sum_{v:p(v)>0} d(v) \leq \frac{1}{\epsilon\alpha}$

It can also be shown that if there is a small, low-conductance set of vertices that contains v_0, then for an appropriately chosen α and ϵ, the non-zero elements of p will contain that set.

3 Approximating PageRank on a very large graph

This suggests a scheme for approximating PageRank on a very large graph — one too large for even a complete vertex-weight vector to fit in memory. Compute $apr(\alpha, \epsilon, v_0)$ by repeatedly scanning through the adjacency-list of the graph. Whenever you scan past a node u with neighbors $v_1, \ldots, v_k$ in the stream, push u if $r(u)/d(u) > \epsilon$, and otherwise ignore u.

In more detail, let the graph be stored in a file where each line contains

$$u, d(u), v_1, \ldots, v_k$$

where the v_i's are the neighbors of u. The algorithm is then

- Let $p = 0$ and $r = \chi_{v_0}$.
- Repeat the following until no pushes are made in a complete scan:
  - For each line in the graph file
    - If $r(u)/d(u) > \epsilon$ then let $p, r = push(u, p, r)$

Finally, take the nodes that have non-zero weight in p, and include all the edges that are incident on these nodes.

4 Building a low-conductance subgraph

Some more notation:

- The "volume" of a set S is the number of edges incident on S, i.e.

$$volume(S) = \sum_{u \in S} d(u)$$

- The "boundary" of a set S are the edges from a node $u \in S$ to a node $v \notin S$.

$$boundary(S) \equiv \{(u, v) \in E : u \in S, v \notin S\}$$

- The "conductance of S" for a small set S is the fraction of edges in S that are not in the boundary.

$$\Phi(S) = \frac{|boundary(S)|}{volume(S)}$$

  More generally

$$\Phi(S) = \frac{|boundary(S)|}{\min(volume(S), |E| - volume(S))},$$

  but if S is small then the min in the denominator is the same as $volume(S)$.

Intuitively, if a node u is in a low-conductance set S that contains a seed node v_0, then it's plausible that u would have a high score in $pr(\alpha, \chi_{v_0})$. If that's true one way to find such a set would be the following.

- Let $S = \{v_0\}$ and let $S^* = S$
- For all nodes $u \neq v_0$, in decreasing order of the personalized PageRank score $p(u)$:
  - Add u to S.
  - If $\Phi(S) < \Phi(S*)$, then let $S^* = S$.
- Return S^*.

Andersen, Chung and Lang call this operation "sweep", and show that it will find a small, low-conductance set S if one exists. Note that $boundary(S)$, and hence $\Phi(S)$, can be computed incrementally: $boundary(S + \{u\})$ is the edges in $boundary(S)$, after removing the set of edges that enter u, and adding the edges from u to any node $v \notin S + \{u\}$.

5 Data

You will be given a tarball containing stater code, data and script for visualization. Data are located in the *data* directory:

- *polblogs.adj* is a file containing the adjacency list of a graph.
- *polblogs-nodes.txt* are the actual blog sites. The k-th line contains the site name of the node in the k-th line of *polblogs.adj*. This file is NOT used for visualization. Those who are curious may want to take a look.

The format of *polblogs.adj* comes as following:

- Each line represents a node and the outlinks of the node.
- Within each line, there will be multiple node indexes separated by $\backslash t$.
- The first index is the index of the node, while others are the index of destinations of edges starting from the node.

Data for test is not given to you, and **note that the indices are not necessarily numbers**. You should consider indices as strings. An example of test dataset is located at here[1]. It will be very large, so do NOT try to load the entire graph into memory at once.

6 Assignment

In this assignment you are going to implement the snowball algorithm and visualize the two sub-communities induced by two different seed.

A visualization software (Gephi)[2] is available for download. You may use other visualization tool as you like but you are expected to produce a similar visualization of the graph as the one in Figure 1.

7 Autolab Implementation Details

We will use the following command to evaluate the correctness of your code so please adhere to the order of command line arguments:

[1] /afs/cs.cmu.edu/project/bigML/wikiGraph/outlink.adj
[2] https://gephi.org/

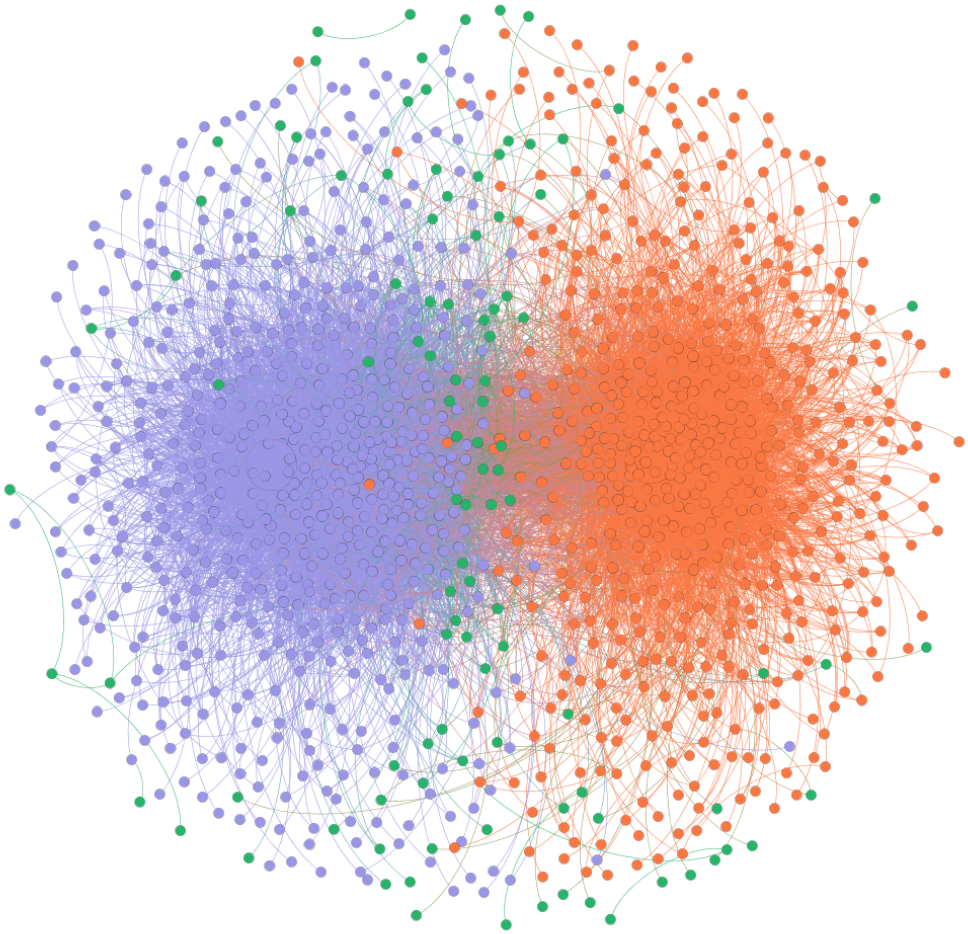


Figure 1: A sample visualization of communities

```
java -cp .:* ApproxPageRank input-path seed alpha epsilon
```

Here the input-path and the seed are strings. The input-path is the path to the adjacency list file whose format is identical to that of *polblogs.adj*. The alpha and epsilon are doubles. The final output of ApproxPageRank class should be the list of nodes present in the lowest conductance subgraph that your code finds. The format of the output per line should be the *string id of the node followed by its Pagerank value*. Please use **tab** as the delimiter and

output to **standard output**. You do not have to worry about the order as we will sort your output before evaluating it.

8 Visualization Guide

After finishing your code and put it in the starter code directory, run

```
make gephi
```

It will use your code to discover two sub-communities induced by two predefined seed and generate **polblogs.gdf** under **vis** folder. Open this file with *Gephi*. You may want to go through the quick start tutorial [3] before visualization.

A few more notes

- The official tutorial is based on an older version, but there's no big differences. *Partition* and *Ranking* tabs are merged into *Appearance* tab in the new version.
- You may want to adjust node color and size under *Appearance* tab.
- Node color can be adjusted by *class* attribute of nodes, which can either be *left*, *right*, or *neutral*.
- Node size can be adjusted by *pagerank* attribute. (Please set *Min Size* to be 20, and *Max Size* to be 100)
- Layout your graph with **Fruchterman Reingold**.

9 Deliverables

Submit your implementations via AutoLab. You should implement the algorithm by yourself instead of using any existing machine learning toolkit. You should upload your code (including all your function files) as a tar file. In addition, you should submit a **report**, which should solve the following problems. (You can consult the lectures of the readings in solving these problems.)

[3] https://gephi.org/users/quick-start/

1. (3 points) Let T be the total number of push operation performed, and let d_i be the degree of the vertex u used in ith push. We can show that

$$\sum_{i=1}^{T} d_i \leq \frac{1}{\epsilon\alpha}.$$

Use this property to explain why the algorithm is guaranteed to finish after finite number of push operations.

Solution: Note that $d_i \geq 1, \forall i$ since a node must have at least one neighbor to be pushed, and the right side of inequality is a constant. T is at most floor$[]\frac{1}{\epsilon\alpha}$.

2. (4 points) Show that

$$pr(\alpha, s) = sR_\alpha,$$

where

$$R_\alpha = \alpha \sum_{t=0}^{\infty} (1-\alpha)^t W^t.$$

This implies $pr(\alpha, s)$ is linear to s. (Hint: show that sR_α satisfies Eq 1).

Solution:

$$\begin{aligned}
\alpha s + (1-\alpha) pr(\alpha, s) W &= \alpha s I + \alpha s \sum_{t=1}^{\infty} (1-\alpha)^t W^t \\
&= \alpha s \sum_{t=0}^{\infty} (1-\alpha)^t W^t \\
&= sR_\alpha \\
&= pr(\alpha, s)
\end{aligned}$$

3. (4 points) Note that

$$\begin{aligned}
R_\alpha &= \alpha \sum_{t=0}^{\infty} (1-\alpha)^t W^t \\
&= \alpha I + (1-\alpha) W R_\alpha.
\end{aligned}$$

Use the property above to show that

$$pr(\alpha, s) = \alpha s + (1-\alpha) pr(\alpha, sW).$$

Solution:

$$\begin{aligned} pr(\alpha, s) &= sR_\alpha \\ &= \alpha s + (1-\alpha)sWR_\alpha \\ &= \alpha s + (1-\alpha)pr(\alpha, sW) \end{aligned}$$

4. (4 points) Show that the equality below holds,

$$p' + pr(\alpha, r') = p + pr(\alpha, r),$$

where

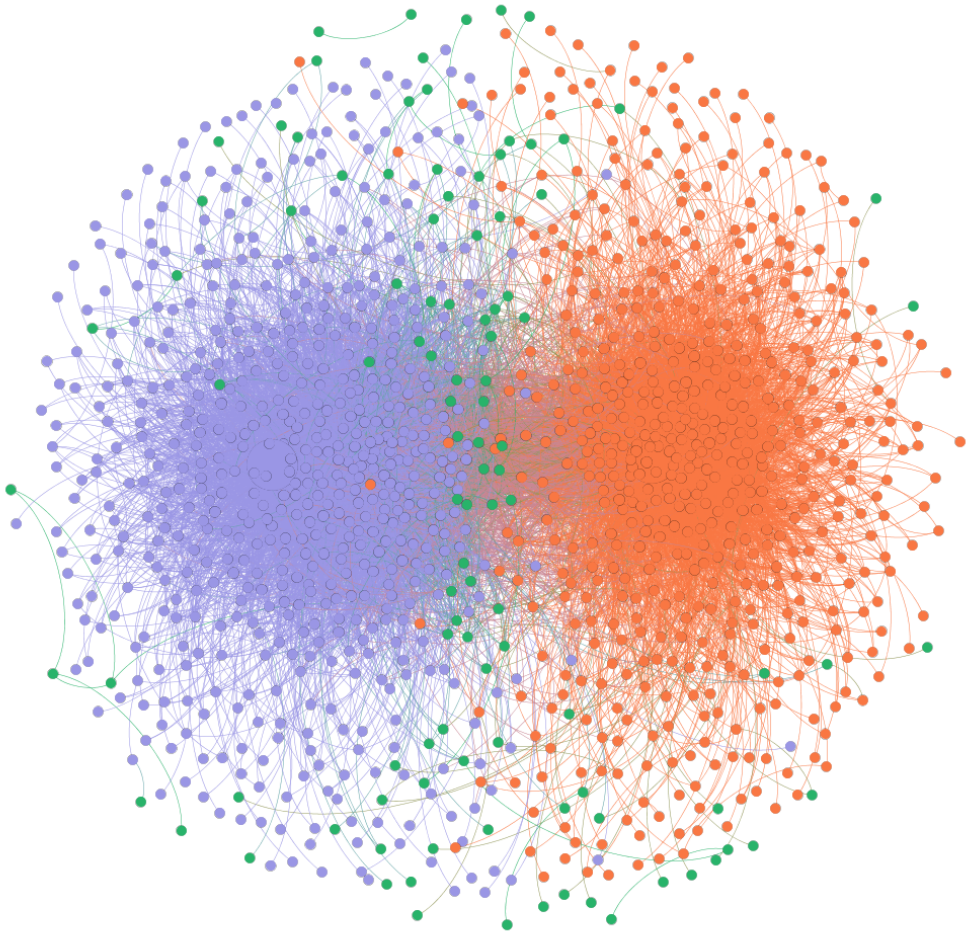
$$\begin{aligned} p' &= p + \alpha r_u \\ r' &= r - r_u + (1-\alpha)r_u W. \end{aligned}$$

This implies that after each push operation, the invariant Eq 3 still holds. Hint: From what we derive above, we know $pr(\alpha, s)$ is linear to s (i.e., $pr(\alpha, r) = pr(\alpha, r - r_u) + pr(\alpha, r_u)$), and $pr(\alpha, s) = \alpha s + (1-\alpha)pr(\alpha, sW)$.

Solution:

$$\begin{aligned} p + pr(\alpha, r) &= p + pr(\alpha, r - r_u) + pr(\alpha, r_u) \\ &= p + pr(\alpha, r - r_u) + \alpha r_u + (1-\alpha)pr(\alpha, r_u W) \\ &= p + pr(\alpha, r - r_u) + \alpha r_u + pr(\alpha, (1-\alpha)r_u W) \\ &= p + \alpha r_u + pr(\alpha, r - r_u + (1-\alpha)r_u W) \\ &= p' + pr(\alpha, r') \end{aligned}$$

5. (15 points) Generate a visualization of the graph with Gephi. Please follow the instruction in Section 8. Solution:



6. Answer the questions in the collaboration policy on page 1.

Marking breakdown

- Code Efficiency (30 points)
- Code Correctness (40 points)
- Report Questions (30 points in total)