# BigML homework 6: Efficient Approximate PageRank

Due: Mon., Apr. 14, 2014 13:29 EST via Autolab

Late submission with 50% credit: Wed., Apr. 16, 2014 13:29 EST via Autolab

## Policy on Collaboration among Students

These policies are the same as were used in Dr. Rosenfeld's previous version of 10601 from 2013. The purpose of student collaboration is to facilitate learning, not to circumvent it. Studying the material in groups is strongly encouraged. It is also allowed to seek help from other students in understanding the material needed to solve a particular homework problem, provided no written notes are shared, or are taken at that time, and provided learning is facilitated, not circumvented. The actual solution must be done by each student alone, and the student should be ready to reproduce their solution upon request. The presence or absence of any form of help or collaboration, whether given or received, must be explicitly stated and disclosed in full by all involved, on the first page of their assignment. Specifically, each assignment solution must start by answering the following questions in the report:

- 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".

Collaboration without full disclosure will be handled severely, in compliance with CMU's Policy on Cheating and Plagiarism. As a related point, 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 will detract 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 (or may have been) 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. I will mostly rely on your wisdom and honor to follow this rule, but if a violation is detected it will be dealt with harshly. Collaboration with other students who are currently taking the class is allowed, but only under the conditions stated below.

## 1 Background

Subsampling 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 subsampling 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 $m = |E|$.
- I'll 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 $D_{i,i} = d(v_i)$.
- $\chi_v$ is a unit (row) vector with all weight on vertex $v$.
- $A$ 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 $s$ is a "seed" distribution (row vector), $\alpha$ is a "teleportation constant".

- It is easy to see that $pr(\alpha, s)$ is a linear function of $s$

  $$pr(\alpha, s) = \alpha \sum_{t=0}^{\infty} (1-\alpha)^t s W^t = s[\alpha \sum_{t=0}^{\infty} (1-\alpha)^t W^t] = s\alpha[I-(1-\alpha)W]^{-1} \tag{2}$$

- It's easy to show that

  $$pr(\alpha, s) = \alpha s + (1-\alpha) pr(\alpha, sW) \tag{3}$$

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

## 2 Approximating PageRank with "pushes"

The personalized PageRank (row) 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 $r = s$ and $p$ is an all-zeros vector. This guarantee that the following equality is satisfied

$$p + pr(\alpha, r) = pr(\alpha, s) \tag{4}$$

Now we repeatedly apply Eq 3 to move probability mass from $r$ to $p$, but maintain the equality in Eq 4.

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

$$p' = p + \alpha r_u$$

$$r' = r - r_u + (1-\alpha) r_u W$$

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 move

$\alpha$ of $u$'s weight from $r$ to $p$, and then distributing 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 in Eq 4. Notice that to do a "push" on $u$, we need to know $d(u)$ and the neighbors of $u$, but we don't need to know anything else about the graph.

Let $apr(\alpha, \epsilon, v_0)$ be an "approximate PageRank" 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 time $O(\frac{1}{\epsilon\alpha})$
- $\sum_{v:p(v)>0} d(v) \leq \frac{1}{\epsilon\alpha}$

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 $\alpha$ and $\epsilon$, 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 ignode $u$.

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. Since both $p$ and $r$ are sparse, they can be kept in memory.

## 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 on the boundary.

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

  More generally

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

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 is 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

An adjacent matrix of wikipedia concepts is available at `/afs/cs.cmu.edu/project/bigML/wikiGraphUDG/`. The file `wikiGraph.adj` contains an adjacent matrix of a subset of the wikipedia graph. Note that this is an undirected graph. Each line is a tab-separated list of wikipedia pages, where the first page has links to each one of the following pages.

## 6 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) along with a **report**, which should solve the following problems:

1. include a visualization of the seed "Machine_learning" with $\alpha = 0.3$ and $\epsilon = 10^{-5}$ (8 points)
2. include a visualization of two other seeds of your choice, with $\alpha$ and $\epsilon$ of your choice. (8 points)
3. how sensitive is the result graph to parameters $\alpha$ and $\epsilon$? What will happen if you change the values of these two hyperparameters? (4 points)
4. Answer the questions in the collaboration policy on page 1.

### Visualizing a subgraph

In this assignment we are going to implement the approximate PageRank algorithm and visualize the result of a few seeds.

A visualization software (Gephi) is available for download[1]. You may use other visualization tool as you like.

Hint 1: to load data into gephi I find GDF format is the easiest[2]

Hint 2: Gephi's UI is a little bit confusing. I find 'ForceAtlas 2' with scaling=500 and gravity=500 gives reasonably good result. You can export figures from the preview tab.

[1]https://gephi.org/users/download

[2]https://gephi.org/users/supported-graph-formats/gdf-format

### Autolab Implementation Details

Overall your program may have the following 4 steps:

- run approximated PageRank with a seed page $s$, and parameters $\alpha$, $\epsilon$.
- create a subgraph that involves nodes in $pr(\alpha, s)$
- do the "sweep" operation and find a small, low-conductance subgraph
- convert the low-conductance subgraph into the GDF format required by Gephi. For a node $v$ use $max(1, \log(p(v)/\epsilon))$ as its node size.

You must have the **ApproxPageRank.java** class file. We will use the following command to evaluate the correctness of your code so please adhere to the order of command line arguments:

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

Here the input-path and the seed are strings. 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 string id of the node followed by its pagerank value. Please use tab as the delimiter. You don't have to worry about the order as we will sort your output before evaluating it.

## 7 Deliverables

You should tar the following items into **hw6.tar** and submit to the homework 6 assignment via Autolab:

- ApproxPageRank.java
- and all other auxiliary functions you have written
- report.pdf

Tar the files directly using "tar -cvf hw6.tar *.java report.pdf". Do **NOT** put the above files in a folder and then tar the folder. You do not need to upload the saved temporary files. Please make sure your code is working fine on linux.andrew.cmu.edu machines before you submit.

## 8 Submission

You must submit your homework through Autolab via the "Homework6" link. In this homework, we provide an additional tool called "Homework6-validation":

- Homework6-validation: You will be notified by Autolab if you can successfully finish your job on the Autolab virtual machines. Note that this is not the place you should debug or develop your **algorithm**. All development should be done on linux.andrew.cmu.edu machines. This is basically a Autolab debug mode. There will be **NO** feedback on your **performance** in this mode. You have unlimited amount of submissions here. To avoid Autolab queues on the submission day, the validation link will be closed 24 hours prior to the official deadline. If you have received a score of 1000, this means that you code has passed the validation.

- Homework6: This is where you should submit your validated final submission. You have a total of **10 possible submissions**. Your performance will be evaluated, and feedback will be provided immediately.

## 9 Grading

You will be graded based on the runtime (40 points) for your code, and the correctness of your implementation (40 points). The correctness of your implementation will be reflected by the precision, and recall of the pages in your generated subgraph, as well as the corresponding values' Spearman's correlation. The report will be graded manually (20 points).