Efficient Approximate PageRank

TAs: Iosef Kaver, Ankit Agarwal, Kavya Srinet, BinBin Xiong

Out November 12, 2015
Due November 24, 2015 via Autolab

Guidelines for Answers: Please answer to the point. Please state any additional assumptions you make while answering the questions. You need to submit a single tar file on autolab, which should include your report. 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.

Code of Conduct Declaration

- 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 $m = |E|$.
- 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 $D_{i,i} = d(v_i)$.
- χ_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), α 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

α 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 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 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/wikiGraph`. The file `outlink.adj` contains an adjacent matrix of wikipedia graph. Each line is a tab-separated list of wikipedia pages, where the first page has links to each one of the following pages. For debug purpose we provide another file `test.adj`, which is an subset of wikipedia graph.

6 Assignment

In this assignment we are going to implement the snowball 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. Figure 1 gives an example result by using "Machine_learning" as seed node. You are expected to produce similar graph for this seed and other seeds. Note that the figure below represents a subset of the graph; you are expected to submit a visualization of the entire graph.

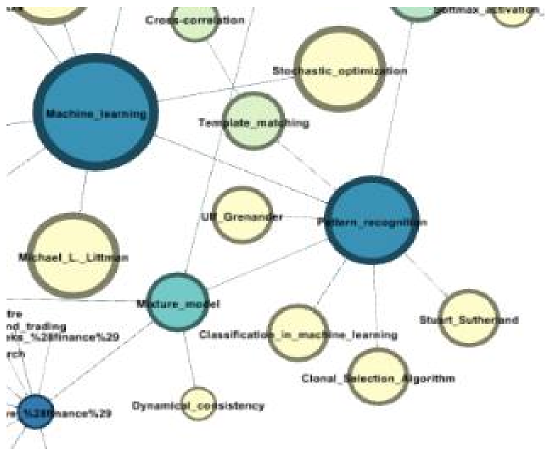

Figure 1: Snowball result for "Machine_learning" with $\alpha = 0.3$ $\epsilon = 10^{-5}$.

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

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

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

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.

Overall your program may have the following 4 steps:

- run approximated PageRank with a seed page s, and parameters α, ϵ.
- 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.

7 Autolab Implementation Details

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.

8 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) Pacman is in a circular maze with $n+1$ rooms, each with a pac-dot in it. Starting from room 0, whose pac-dot is already eaten by Pacman,

Pacman has equal probability to move to either room on the left or on the right, and eats the pac-dot if there is one in the room. Whats the probability that pac-dot k ($1 \leq k \leq n$) is the last to be eaten?

[5 points]

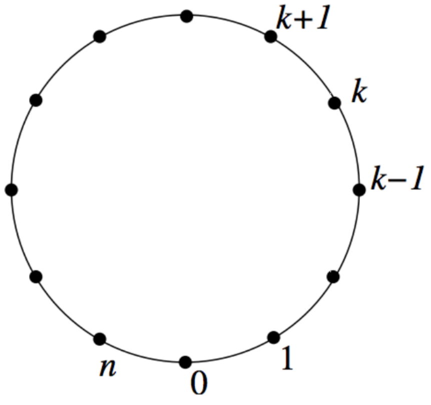

Figure 2: Random walk on a circle

Part(a) We can represent the problem as random walk on a circle (Fig. 1). To eat k last, Pacman must have eaten every other pac-dots, including $k-1$ and $k+1$ (but not k). There are two cases:

Case 1: Eating $k-1$ before $k+1$: The sequence would look like $0 \rightarrow$ (anything but k or $k+1$) $\rightarrow k-1 \rightarrow$ (anything but k) $\rightarrow k+1 \rightarrow$ anything $\rightarrow k$.

Case 2: Eating $k+1$ before $k-1$: The sequence would look like $0 \rightarrow$ (anything but k or $k-1$) $\rightarrow k+1 \rightarrow$ (anything but k) $\rightarrow k-1 \rightarrow$ anything $\rightarrow k$.

Lets consider case 1, in which Pacman eats $k-1$ first. We can further break it down to two subsequence:

Seq 1: $0 \rightarrow$ (anything but k or $k+1$) $\rightarrow k-1$
Seq 2: $k-1 \rightarrow$ (anything but k) $\rightarrow k+1$ (after this Pacman will eat k as last dot with probability 1 as step tends to infinity.)

Whats the probability that Pacman, starting from 0, eats $k-1$ before $k+1$? (Hint: you can use the fact that symmetric random walk starting from 0 visits a before visiting b with probability b for all $a+b$, $a>0$, $b>0$)

Part(b) After Pacman eats $k-1$, whats the probability that it eats $k+1$ without first eating k? (After this all dots are gone except k; we are almost there...)

Part(c) Whats the probability that pac-dot k is the last to be eaten? Briefly justify your answer using results from previous parts. Make sure you consider both cases or your probability wont sum to 1. Is the answer what you would expect?

(2) How can we evaluate a static-graph sampling algorithms. Please answer in no more than 5 sentences. (Hint: Do the readings on course wiki!)

[*5 points*]

(3) Kesden wants to optimize the PageRank value of his website V. Assume all other nodes and edges stay the same, and no spam detection is performed. How will the following methods affect the PageRank value pr(V) (increase/decrease/no effect/it depends)? Why?

[*5 points*]

- Place more outlinks from V
- Obtain more inlinks to V
- Split V to two nodes Va and Vb without any link between Va and Vb. (Copy all links of V to Va and Vb, pr(V) = max(pr(V a), pr(V b)))
- Create 10 nodes $V_1 \ldots V_{10}$ Add a bidirectional link between each (V, V_i) (This is called Sybil attack)

(4) Visualization with Gephi.

[5 points]

- Include a visualization of the seed "Machine learning" with $\alpha = 0.3$ and $\epsilon = 10^{-5}$. You can apply any beautification you want. Make sure to add node label so it's easy to see which wiki pages you've chosen.

- Gephi goes beyond graph visualization. It also provides useful plug-ins like the statistics toolbox in the right column under the 'Overview' tab, such as "Average Degree", "Graph Density" etc. Take a look and answer the following :

  - Which tool would you use to cluster nodes into communities / clusters based on their connectivity?
  - Which tool would you use to use to find authoritative nodes which are the influencers in the network?

(5) Answer the questions in the collaboration policy on page 1.

9 Marking breakdown

- Code correctness and makefile functionality [**80 points**].

- Report Questions [**20 points**]