Snowball Sampling a Large Graph

William Cohen

January 6, 2012

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 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.

2 Approximating PageRank with "pushes"

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.) We'll use this to approximate PageRank incrementally as follows. 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.

We repeatedly update p and r by picking a node u with non-zero weight in r and moving α of u's weight from $r(u)$ to $p(u)$, and then distributing the remaining $(1-\alpha)$ weight within $r(u)$ as if a single step of the random walk associated with W were performed. This is called a "push" operation, and it will maintain the invariant

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

Precisely, define $push(u, p, r)$ to return a pair p', r' which is computed as follows.

- Let p' be a copy of p and r' be a copy of r.
- Update p' and r' as follows:
  - $p'(u) = p(u) + \alpha r(u)$
  - $r'(u) = \frac{1}{2}(1-\alpha) r(u)$
  - For each v such that $(u, v) \in E$:
    - $r'(v) = r(v) + \frac{1}{2} \frac{(1-\alpha) r(u)}{d(u)}$

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.

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

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

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 $v \notin S + \{u\}$.