SCOTTYRANK.JL: AN IMPLEMENTATION OF PAGERANK & HITS

SIYUAN CHEN AND MICHAEL ZHOU

E-mail addresses: `siyuanc2@andrew.cmu.edu, mhzhou@andrew.cmu.edu`.
Date: November 2021.

1. Mathematical Background

1.1. **Linear Algebra.**

1.1.1. *Definitions.* Positive matrices are defined as matrices with positive entries.

Markov matrices are defined as square matrices with nonnegatives entries and column sum 1 across all of its columns. Note that for a $n \times n$ matrix M, the latter condition is equivalent to $M^T\vec{1} = \vec{1}$, where $\vec{1} \in \mathbb{R}^n$ has all ones as components.

Positive Markov matrices are defined as, well, positive Markov matrices.

1.1.2. *Facts.* (Perron-Frobenius theorem) Let A be a positive square matrix. Let λ_1 be A's maximum eigenvalue in terms of absolute values. Then λ_1 is positive and has algebraic (and subsequently geometric) multiplicity 1.

Let M be a Markov matrix. Let λ_1 be M's maximum eigenvalue in terms of absolute values. Then $\lambda_1 = 1$.

Let M' be a positive Markov matrix. Let λ_1 be M''s maximum eigenvalue in terms of absolute values. Then $\lambda_1 = 1$ and has algebraic (and subsequently geometric) multiplicity 1.

1.1.3. *Usage.* Let M be a $n \times n$ Markov matrix. Then M specifies a dicrete memoryless transition process between n states, namely the process where

$$(\forall (t,i,j) \in \mathbb{N} \times [n] \times [n]) \left[\Pr(\text{state } i \text{ at time } t+1 \mid \text{state } j \text{ at time } t) = M_{ij}\right].$$

Let $\vec{v} \in \mathbb{R}^n$ such that $\vec{v}$ has nonnegative components and $\vec{v}^T\vec{1} = 1$ (a stochastic vector). Then $\vec{v}$ specifies an (initial) discrete probability distribution over the n states, namely the distribution where

$$(\forall i \in [n])[\Pr(\text{state } i \text{ at time } 0) = \vec{v}_i].$$

Then the probability distribution over the n states after t steps of the transition process specified by M is precisely $M^t\vec{v}$, or equivalently

$$(\forall (t,i) \in \mathbb{N} \times [n]) \left[\Pr(\text{state } i \text{ at time } t) = \left(M^t\vec{v}\right)_i\right].$$

1.2. **Graph Theory.**

1.2.1. *Definitions.* A simple directed graph is defined as an unweighted directed graph without self-referential edges or multiple edges between the same origin destination pair.

For a simple directed graph with n vertices, the adjacency matrix $\mathcal{A}$ is defined to be the $n \times n$ matrix where

$$(\forall (i,j) \in [n] \times [n]) \left(A_{ij} = \begin{cases} 1 & \text{there is an edge to } i \text{ from } j \\ 0 & \text{otherwise} \end{cases} \right).$$

1.2.2. *Facts.* For a simple directed graph with n vertices and its adjacency matrix $\mathcal{A}$,

$$(\forall j \in [n]) \left[\text{number of outgoing neighbors from vertex } j = \text{out}(j) = (\mathcal{A}_{*j})^T\vec{1}\right]$$

$$(\forall i \in [n]) \left[\text{number of incoming neighbors to vertex } i = \text{in}(i) = (\mathcal{A}_{i*})^T\vec{1}\right].$$

2. Algorithms

2.1. **The network model.**

Both algorithms, PageRank and HITS, model the network of interest as a simple directed graph with websites as vertices and links as edges. This implies that there will be no self-referential links, no duplicate links between the same origin and destination pair, and no priority difference between links.

2.2. **PageRank.**

2.2.1. *The random walk.* PageRank models the behavior of a typical web surfer as a damped random walk.

(1) The surfer starts out by visiting a random site out of all sites with equal probability.
(2) At every step, the surfer has a probability λ of continuing surfing and a complementary $1 - \lambda$ probability of losing interest, for a predetermined λ.
   (a) If the surfer continues . . .
      (i) . . . and there are links exiting the current site, the surfer clicks on a random link (and visits the site it points to) out of those links with equal probability.
      (ii) . . . and there aren't any links exiting the current site, the surfer simply visits a random site out of all sites with equal probability.
   (b) If the surfer loses interest, they simply visits a random site out of all sites with equal probability.

To best model a typical surfer's probability of continuing surfing, λ, also known as the damping factor, is empirically determined to be around 0.85.

2.2.2. *Matrix representation.* Let n be the number of websites in the network of interest. Let $\mathcal{A}$ be the adjacency matrix for the network of interest. Let $\langle \vec{v}_t \rangle_{t\in\mathbb{N}}$ be the probability distributions describing the website the surfer is visiting at time t. Let M be the transition matrix for the random walk process.

Then $\vec{v}_0 = \vec{1}/n$, M is the $n \times n$ matrix where

$$(\forall (i,j) \in [n] \times [n]) \left[M_{ij} = \begin{cases} \frac{\lambda}{\operatorname{out}(j)} + \frac{1-\lambda}{n} & \mathcal{A}_{ij} = 1 \\ \frac{1-\lambda}{n} & \mathcal{A}_{ij} = 0 \wedge \operatorname{out}(j) > 0 \\ \frac{\lambda}{n} + \frac{1-\lambda}{n} & \operatorname{out}(j) = 0 \end{cases} \right],$$

and

$$(\forall t \in \mathbb{N}) \left(\vec{v}_t = M^t \vec{v}_0 \right).$$

Note that in this case M is a positive Markov matrix, assuming reasonable λ.

2.2.3. *Definition.* The PageRank score for a given website in the network of interest is defined as the probabilty of a typical surfer visiting that website after an indefinitely long damped random walk. In matrix form,

$$(\forall i \in [n]) \left[\operatorname{PageRank}(i) = \lim_{t\to\infty} (\vec{v}_t)_i = \lim_{t\to\infty} \left(M^t \vec{v}_0 \right)_i \right].$$

Note that the limits exist: convergence is guaranteed as M has a unique maximal eigenvalue of 1 and thus an steady attracting state.

2.3. **HITS.**

2.3.1. *Authorities and hubs.* filler

2.3.2. *Matrix representation.* filler

```julia
# export read_graph

"""
    read_graph(filepath::String="data/medium-el.txt");
↪   filetype::String="el", zero_index::Bool=false) -> Graph

Reads a graph from an edge list/adjacency list file

# Arguments
- `filepath::String="data/medium-el.txt"`: the path to the source file
↪   (default: Wikipedia PageRank graph)

# Keywords
- `filetype::String="el"`: "el" for edge list, "al" for adjacency list
- `zero_index::Bool=false`: whether the input file is zero-based

# Returns
- `Graph`: the graph from the source file
"""
function read_graph(filepath::String="data/medium-el.txt";
    filetype::String="el", zero_index::Bool=false)
  if filetype == "el"
    read_edge_list(filepath, zero_index)
  elseif filetype == "al"
    read_adjacency_list(filepath, zero_index)
  else
    error("invalid filetype")
  end
end
```