9.1 PageRank

"A modern classic." – Professor Gittens

We're going to talk a bit about the PageRank algorithm. This algorithm serves as a good example of various ways in which to consider computations on graphs. It also has many varied applications, and it gives us a shallow dive into a more "modern" subfield of graph theory via *Spectral Graph Theory*. The major application of PageRank is to compute some metric of *centrality* (i.e., importance) for all vertices in a directed graph.

9.1.1 Random Walk Model

A **random walk** on a graph is a walk that starts at some v and randomly selects some $u \in N(v)$ to hop to from v. Then, some $w \in N(u)$ is selected and subsequently hopped to. This process iterates for some number of steps. Random walks are a surprisingly powerful tool, and they form the basis for a number of approximation and sampling-based algorithms; doing multiple random walks on a small portion of a large graph is a good way to measure some properties that might be extrapolated to the full graph.

In the first PageRank model, we're considering our graph to be a series of web pages connected through directed hyperlinks. We have a hypothetical **surfer** who randomly clicks on links while browsing, and they hypothetically browse the web for an infinite time.

The PageRank of a webpage is defined as the probability that this surfer will be viewing that given page at any point in time. We can therefore calculate an estimate for the PageRank of a page simply by performing a 'sufficiently long' random walk and tracking how often the page is visited versus the total number of page visits.

Issues arise for any page that doesn't have any outgoing links (out degree is zero, called a **sink**) or incoming links (called a **source**), or if the graph D isn't *strongly connected* ($\exists u, v - \text{paths} \forall u, v \in V(D)$). To address these issues, we include some additional considerations. Whenever a random surfer reaches a sink, they randomly jump to any other page in the graph. It's common to additionally consider that this surfer will randomly jump with some probability after every link they traverse; this can improve convergence and stability when calculating PageRanks, particularly when the graph isn't strongly connected. The probability that a surfer clicks a link versus randomly jumping is termed as the **damping factor**. For now, we'll just consider the basic model without the damping factor component.

```
for all v ∈ V(G) do
    P(v) ← 1/n                                  ▷ Initialize PR equally among vertices
    if |N⁺(v)| = 0 then
        sink ← sink + P(v)                      ▷ Total PR of all sinks
for some number of iterations do
    sink_n ← 0                                  ▷ Sink contribution on next iteration
    for all v ∈ V(G) do
        P(v) ← sink/n                           ▷ First get an equal portion from the sinks
        for all u ∈ N⁻(v) do
            P(v) ← P(v) + P(u)/|N⁺(u)|          ▷ All u in N⁻(v) shares P(u) with N⁺(u)
        if |N⁺(v)| = 0 then
            sink_n ← sink_n + P(v)              ▷ Sink contribution for next iteration
    swap(sink, sink_n)
```

9.1.2 Graph Algorithm Model

Graph algorithmic computations can be typified as performing some kind of per-vertex or per-edge iterative update computation on some per-vertex or per-edge *state*. From the perspective of each vertex during a PageRank computation, we have some initial PageRank value that gets updated through multiple iterations. Going back to the random surfer model, a surfer arrived on that vertex/page either from an in-edge or it was the destination of a random jump. As the surfer travels, we can consider that it takes with it a portion of the prior vertex's PageRank. So for a given vertex, the PageRank is the sum of PageRanks incoming through in-edges plus the PageRanks from potential random jumps and from zero out-degree vertices.

The algorithm above gives how we would calculate PageRank using this model.

9.1.3 Linear Algebraic Model

We can also use the adjacency matrix of the graph to formulate this problem from an algebraic perspective. As the transitions of our random surfer through the graph essentially form a Markov chain, we can create a stochastic matrix M of transition probabilities from a given vertex to its neighbors. We define M as:

$$M = (D^{-1}A)^T$$

where D is a diagonal matrix of out degrees and A is the adjacency matrix. We can use a modified adjacency matrix where each vertex with zero out-degree is changed to have an out-degree of $n-1$ and links to all other vertices in the graph, in order to mirror our prior

models. We can then compute updated PageRanks in vector $\mathbf{p}_{i+1}$ as the matrix-vector product of the current PageRank values $\mathbf{p}_i$ and the transition probability matrix.

$$\mathbf{p}_{i+1} = M\mathbf{p}_i$$

with steady state solution

$$\mathbf{p}_\infty = M\mathbf{p}_\infty$$

This should look familiar if you've taken a linear algebra class[1]. Recall that an eigenvector $\mathbf{v}$ for some matrix A is defined as

$$A\mathbf{v} = \lambda\mathbf{v}$$

where λ is a corresponding eigenvalue. So, our PageRank steady state solution is actually just an eigenvector of the transition probability matrix with eigenvalue of 1. There's an entire subfield of graph theory devoted to studying the eigenvectors and eigenvalues on matrices derived from graph adjacency matrices. This is called *Spectral Graph Theory*. In this class, this lecture is the deepest we'll go into that field, however.

[1]If you have not taken a linear algebra class, you should, immediately. Linear algebra forms the basis for most subfields of CS. Also recall that "everything is a graph" and all graphs are matrices, so literally everything is also linear algebra.