Mining Massive Datasets:

Week 2: PageRank

Ian Quah (itq)

June 11, 2017

Topic 1

Link Analysis and PageRank

Web as a graph:

1. nodes: webpage
2. edge: hyperlink

Link Analysis Approach:

1. Page Rank - Google
2. Hubs and Authorities
3. Topic-Specific Page Rank
4. Web Spam Detection Algorithms

Topic 2

PageRank: Flow Formulation

1. Let r$_j$ be the rank of website j (normalized to 1)
2. Flow Algorithm: r$_j$ = $\sum_{i \to j} \frac{r_i}{|D|}$, where $|D|$ is num of nodes pointing to j
3. Vote from important page is worth more
4. However, when there are a lot of pages, this computation becomes expensive, so we need to reformulate it.

Topic 3

PageRank: Matrix

1. r$_j$ = $\sum_{i \to j} \frac{r_i}{|D|}$ $\to$ r = M $\cdot$ r
2. If we constrain the sum of the outnodes to 1, then the matrix is called Column Stochastic
3. Largest Eigenvalue of M is 1, since M is column stochastic

   A x = λx
4. We can now efficiently solve for r using Power Iteration

Topic 4

PageRank: Power Iteration

Topic 4 continued on next page. . .

Power Iteration: iterative scheme

1. Given web graph with N nodes.
   (a) N web pages
   (b) **Initialize**: $r^{(0)} = [1/N, ..., 1/N]^T$
   (c) **Iterate**: $r^{(t+1)} = M \cdot r^t$
   (d) **Halting condition**: $| r^{(t+1)} - r^t | < \epsilon$, here using L_1 norm but can use any other norm
2. Intuition: keep multiplying it until it converges (reaches stationary state)

Formulation as a random walk

1. At time t, random walker is on page i
2. At t + 1, walker randomly picks a page out (out-link)
3. Ends up on some page j from i (M_{ji})
4. Repeats indefinitely

Since the out-links are normalized, this is a probability distribution

Topic 5

Google formulation of Page Rank

Page rank: 3 Questions

1. Does the result converge? Not necessarily
2. Does it converge to what we want? Not always
3. Are the results reasonable? Not always

Google's Solutions

1. Spider Traps: random surfers teleporting out with probability β
2. Dead Ends: random surfer teleports out with probability 1, to another node (uniformly)

Why do random surfers solve our problem?

1. It makes our transition matrix, M, stochastic

   Because what ends up happening is that because there is some chance that it can teleport to any page (uniformly), it becomes a dense graph, and because it can do it with equal probability, this does not actually change the probabilities beyond just making sure that our matrix in the limit does not become 0 or 1 (caught in spider trap)
2. Makes M aperiodic
   (a) **Def** A chain is aperiodic if there exists $k > 1$ s.t interval between two visits to some state S is always a multiple of k
3. Makes M irreducible

(a) **Def** Non-0 probability of transitioning out of any node

The Google Matrix

New form of our PageRank equation

$$r_j = \Sigma_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta)\frac{1}{n}$$

Google Matrix A, modifying M

$$A = \beta M + (1 - \beta)\frac{1}{n} e \cdot e$$

where e is vector of 1's

We know that A is stochastic, aperiodic and irreducible some

$$r^{t+1} = A \cdot r^{(t)}$$

in practice, β is typically 0.8 or 0.9, so our surfer makes about 5 steps and then jumps

Topic 6

Computing PageRank for web-scale

Matrix A vs M

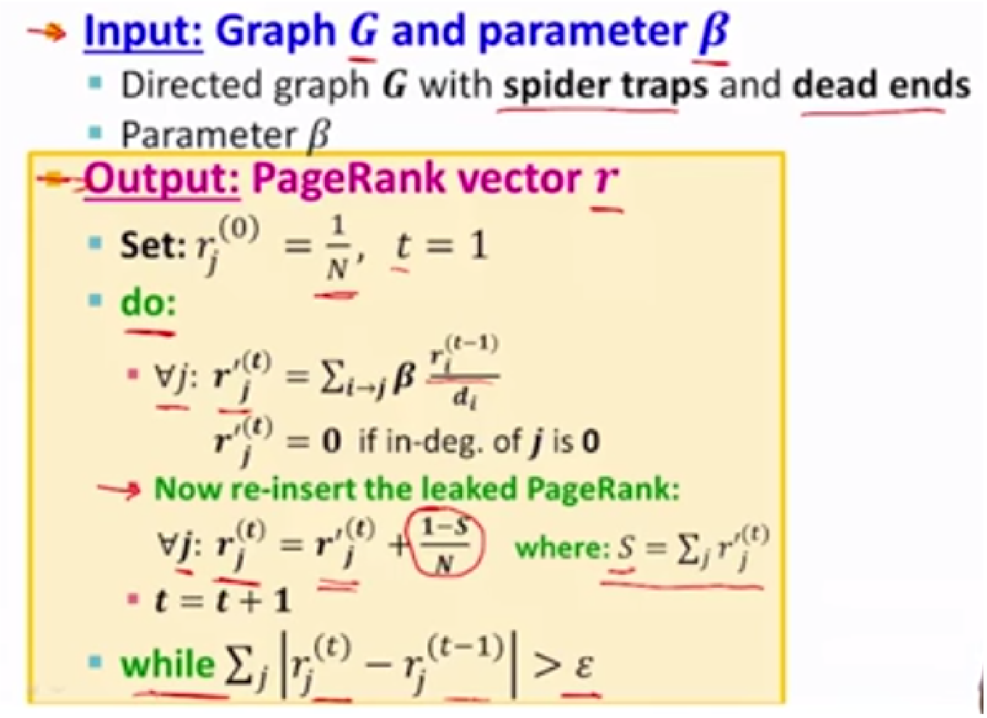
We would prefer to work with matrix M rather than matrix A just because M is full of 0's which is cheap to store, rather than matrix A which is dense

Investigating the matrix

$$\begin{aligned}
\text{Recall: r} = A \cdot r \text{ where } A_{ij} &= \beta M_{ij} + \frac{1 - \beta}{N} \\
r_i &= \Sigma_{j=1}^{N} A_{ij} \cdot r_j \\
r_i &= \Sigma_{j=1}^{N} [\beta M_{ij} + \frac{1 - \beta}{N}] \cdot r_j && \text{(Substitute in A)} \\
r_i &= \Sigma_{j=1}^{N} \beta M_{ij} r_j + \Sigma_{j=1}^{N} \frac{1 - \beta}{N} \cdot r_j && \text{(Distribute the sums)} \\
&= \Sigma_{j=1}^{N} \beta M_{ij} r_j + \frac{1 - \beta}{N} && \text{(bc sum of r}_j = 1) \\
r &= \beta M \cdot r + [\frac{1 - \beta}{N}]_N
\end{aligned}$$

$[\frac{1-\beta}{N}]_N$ is a vector of length N, where all elems have value $\frac{1-\beta}{N}$

Full Page Rank Algorithm



Input: Graph G and parameter β

- Directed graph G with **spider traps** and **dead ends**
- Parameter β

Output: PageRank vector r

- **Set:** $r_j^{(0)} = \frac{1}{N}$, $t = 1$
- **do:**
  - $\forall j$: $r'^{(t)}_j = \sum_{i \to j} \beta \frac{r_i^{(t-1)}}{d_i}$
    $r'^{(t)}_j = 0$ if in-deg. of j is 0
  - Now re-insert the leaked PageRank:
    $\forall j$: $r_j^{(t)} = r'^{(t)}_j + \boxed{\frac{1-S}{N}}$ where: $S = \sum_j r'^{(t)}_j$
  - $t = t + 1$
- **while** $\sum_j \left| r_j^{(t)} - r_j^{(t-1)} \right| > \varepsilon$