More on PageRank

CS 111: Introduction to Computational Science

Spring 2019 Lecture #13

Ziad Matni, Ph.D.

Administrative

- Homework #6
  - Will be issued today. Due **WEDNESDAY (5/29) @ 6:00 pm**

- Your MIDTERM #2 exam is on Thursday
  - » In case you forgot…

Midterm #2 Preparation

- What's on it?
  - Everything from **lectures 9, 10, 11, and 12**
  - Review homework 4 and 5
  - Review the practice questions I put up on our web site
  - Today's lecture is NOT on the midterm exam

- SAME rules apply as from Midterm #1
  - Closed book − no electronics present
  - Can bring one 8.5" x 11" sheet of notes (both sides ok)

Remember: What do You Need to Know?

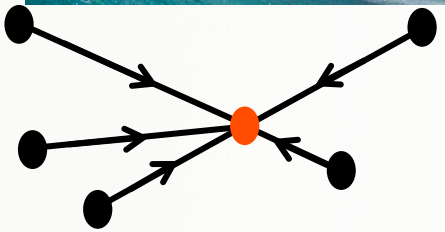
- The functions we've used in class
  - Not their "insides", but how do we use them
- How to write (short) Python code to solve the problems we've encountered in class
  - With the CORRECT syntax
- How to do linear algebra on problems we've encountered in class

Midterm Info

- We start at 11:00 AM SHARP!
- We have 75 minutes
- You should arrive 10 minutes EARLY

Google and the Random Surfer

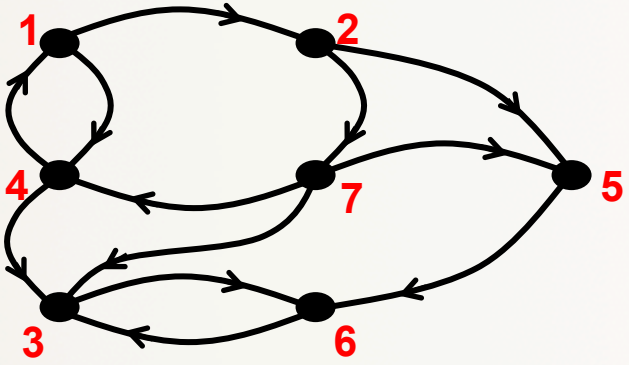
How does Google figure out which web pages are most important?



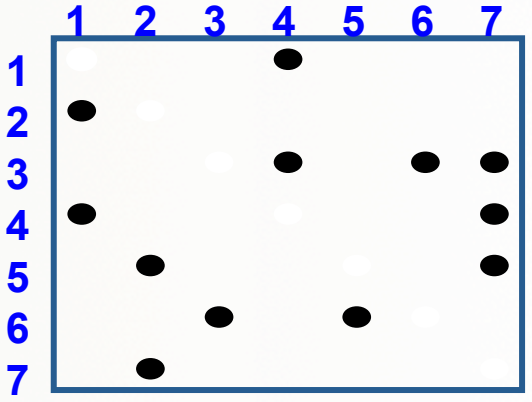
- An important page is one that lots of important pages point to.
- Start at any web page and follow links at random. Forever.
  - » Without wiping-out…
- You'll see "important" pages more often than unimportant ones.

Analyzing the Web with Graphs and Matrices

Graph



Matrix



- Graph nodes are web pages
- Arrows between nodes are links between web pages
- Matrix entries are links from "column" pages to "row" pages
- Page Rank comes from algebra on the matrix
- The Google matrix has over 130,000,000,000,000 rows & columns (as of 2016)

The Random Surfer Model

Let:

- **W** = set of web pages; **n** = # of web pages in W;
  E = the nxn adjacency matrix for W
- For a large W, n is typically in the billions (10^9) and in the trillions (10^{12})
- E will be huge and sparse with $e_{ij} = 1$ if i → j, or 0 otherwise
- So, $r_i = \Sigma_j e_{ij}$ (in-degree) and $c_j = \Sigma_i e_{ij}$ (out-degree of the j^{th} page)

- **p** = the probability that the "random walk" follows a link (typ. p = 0.85)
- We'll call **m** = 1 − p, the prob. that an arbitrary page is chosen

The Random Surfer Model

- **p** = the probability that the "random walk" follows a link (typ. p = 0.85)
- We'll call **m** = **1** − **p**, the prob. that an arbitrary page is chosen

- If we define a matrix **M** where: $m_{ij} = p.e_{ij}/c_j + \delta$ where: $\delta = (1-p)/n$
- Then we get a matrix with *most* of its entries as δ

- **M** is called the "transition probability matrix of the Markov Chain"
  - Sometimes referred to as the "Markov matrix" or "PageRank matrix"
  - It's a column stochastic matrix
- Previously, we had defined **A** as the "Link Matrix" (just: $a_{ij} = e_{ij}/c$)

The Random Surfer Model

- **M** is where: $a_{ij} = p.e_{ij}/c_j + \delta$ (where: $\delta = (1-p)/n$)
- Example:
  - n = 3 billion pages (3×10^9)
  - p = 0.85 (so, m = 1 − p = 0.15)
  - $\delta = (0.15 \times 10^{-9})/3 = 5 \times 10^{-11}$

Summary of Types of Matrices We've Used in PageRank

- The Adjacency Matrix, E
  - Spells out all links in a network
- The Link Matrix, A
  - Shows the links weighted by node out-degree
- The Markov Matrix, M
  - Further weighs the nodes in probabilistic terms

Using the PageRank Matrix in Python

```
def make_M_from_E(E):
    """Make the PageRank matrix from the adjacency matrix of a graph.
        Not for sparse matrices.
    """
    n = E.shape[0]
    outdegree = np.sum(E,0)
    for j in range(n):
        if outdegree[j] == 0:
            E[:,j] = np.ones(n)
            E[j,j] = 0
    A = E / np.sum(E,0)
    S = np.ones((n,n)) / n
    m = 0.15
    M = (1 - m) * A + m * S
    return M
```

Iterative Power Method

- Per the Perron-Frobenius theorem:

$$\mathbf{x} = \mathbf{A}\mathbf{x}$$

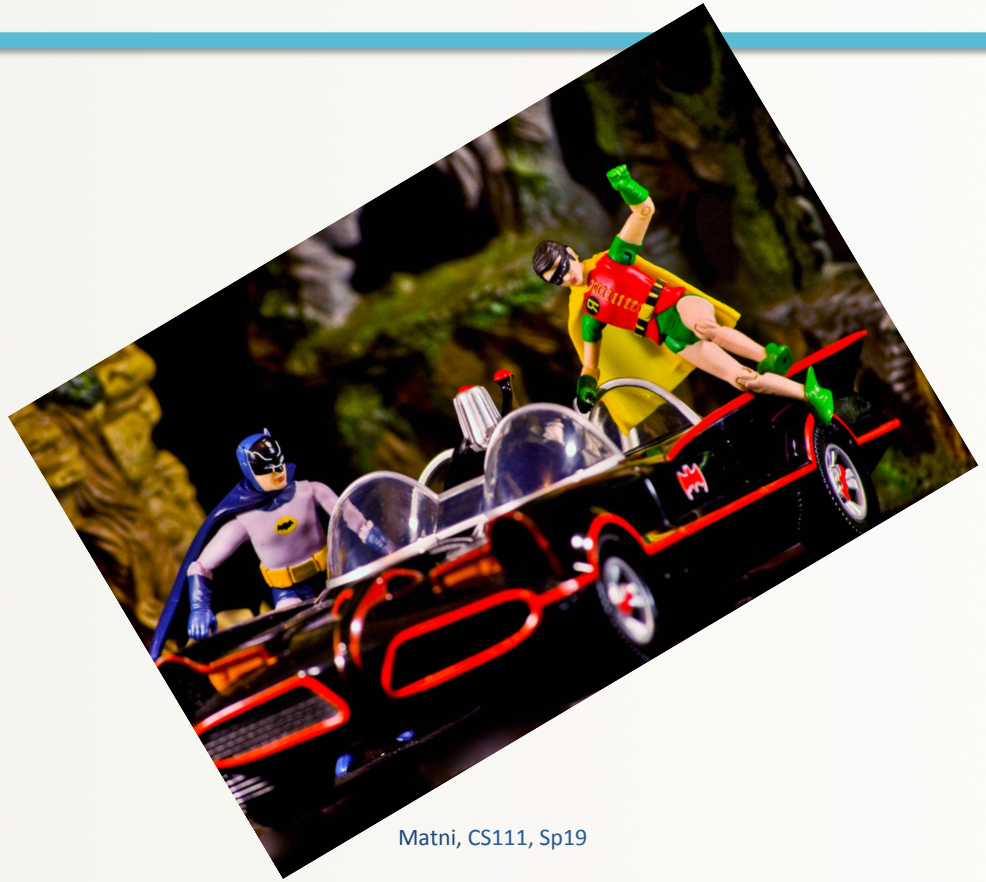
- Where **x** is the PageRank solution as long as:

$$\Sigma_i x_i = 1$$

(that is, normalized)

```
x = some_initial_value
for i in range(N):
    x = M @ x
    x = x / npla.norm(x)
# will approximate iteratively the value for x
```

• • •



Your To-Dos

- **Study for Midterm #2**
- **Homework 6 due next week WEDNESDAY**

</LECTURE>