Numerical Linear Algebra – PageRank continued

CS370 Lecture 28 – March 20, 2017

Page Rank recap

Given a graph of a network, compute a corresponding transition (Markov) matrix…

$$M = \alpha\left(P + \frac{1}{R}ed^T\right) + (1-\alpha)\frac{1}{R}ee^T$$

Then, evolve a probability vector p^i via $p^{n+1} = Mp^n$ towards a steady state, to approximate a “random surfer”.

The site with the highest probability of being visited is considered most important/influential.

Questions to Ponder…

- Do we actually know if it will settle (*converge*) to a fixed final result?
- If yes, then how long will it take? Roughly how many *iterations* are needed before we can stop?
- Can we implement this *efficiently* (e.g. for very large networks?)

Making Page Rank **Efficient**

A naïve implementation of Page Rank involves repeatedly multiplying massive matrices with > 1billion×1billion entries.

How can we implement this in a way that is actually computational feasible?

Later: How do we know it will converge at all? And if so, in how many iterations?

First step: Precomputation

The ranking vector p^{∞} can be pre-computed once and stored, *independent* of any specific query.

To search for “lobster hats”, Google finds **only** the subset of pages matching the keyword(s), and ranks those by their values in the (**precomputed**) $\boldsymbol{p}^{\infty}$.

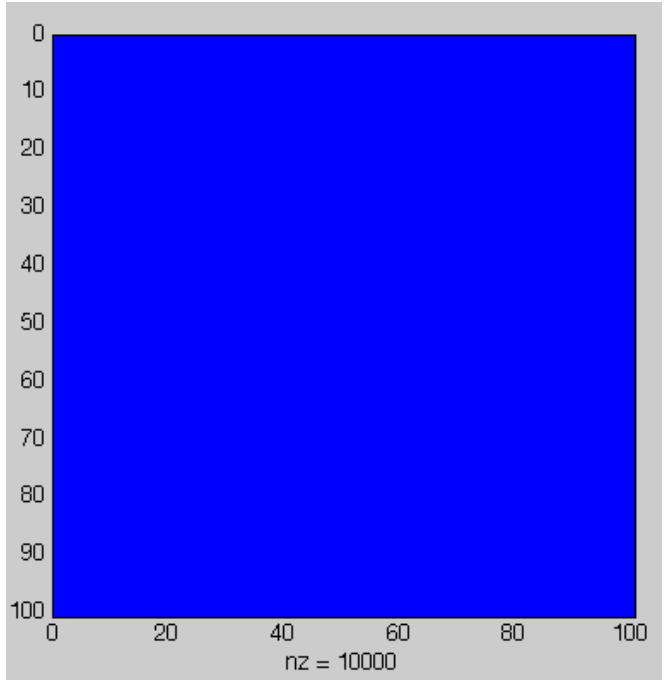
Matrix Sparsity

In numerical linear algebra, we often deal with two kinds of matrices.

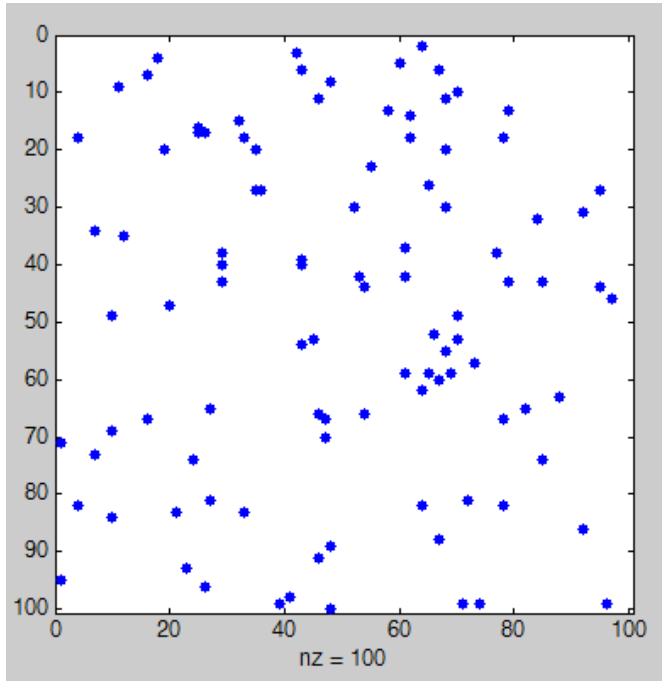
Dense: Most or all entries are **non-zero**. Store in an $N \times N$ array, manipulate "normally".

Sparse: Most entries are **zero**. Use a "sparse" data structure to save space (and time).

Prefer algorithms that avoid "destroying" sparsity (i.e., filling in zero entries).



Non-zeros in a dense matrix



Non-zeros in a sparse matrix

Sparse Matrix-Vector multiplication

Multiplying a sparse matrix with a vector can be done efficiently!

Only non-zero matrix entries are accessed/used.

$$\begin{pmatrix} 2 & & & 1 \\ & & 1 & \\ & 3 & & \\ & & & -2 \\ 1 & & & \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} = \begin{pmatrix} 2 \cdot 1 + 1 \cdot 4 \\ -2 \cdot 5 \\ 1 \cdot 3 \\ 3 \cdot 2 \\ 0 \\ 1 \cdot 1 + 1 \cdot 5 \end{pmatrix} = \begin{pmatrix} 6 \\ -10 \\ 3 \\ 6 \\ 0 \\ 6 \end{pmatrix}$$

Second Step: Exploiting Sparsity

To implement Page Rank efficiently, it is crucial to **exploit sparsity.**

Sadly, our M matrix was **fully dense**. No zero entries at all!

A dense matrix-vector multiply with $1{,}000{,}000{,}000^2$ entries is slooooooow.

The trick: Use linear algebra manipulations to perform the main iteration

$$\boldsymbol{p}^{n+1} = M\boldsymbol{p}^n$$

without ever creating/storing M!

Let's derive this…

Algorithm

Given this efficient/sparse iteration, loop until the max change in probability vector per step is small ($< \ tol$) – easy!

Page Rank Algorithm

$\mathbf{p}^0 = \mathbf{e}/R$

For $k = 1, \ldots,$ until converged

$\quad \mathbf{p}^k = M\mathbf{p}^{k-1}$ (7.32)

$\quad$ If $\max_i |[\mathbf{p}^k]_i - [\mathbf{p}^{k-1}]_i| < tol$ then quit

EndFor

Google Search: Other Factors

Page Rank can be “tweaked” to incorporate other (commercial?) factors.

Replace standard teleportation $\frac{1-\alpha}{R}ee^T$ with $(1-\alpha)ve^T$, where a special probability vector v places extra weight on whatever sites you like.

In reality, many factors besides link-based ranking can come into play.

Search Engine Optimization (SEO) is big business, motivating people to try to guess/determine all factors at work.

e.g., http://backlinko.com/google-ranking-factors

Convergence of Page Rank

Remaining questions:

- How can we be sure that Page Rank will ever “settle down” to a fixed probability vector?
- If it does, how many iterations will it take?

We will need some additional facts about Markov matrices, involving **eigenvalues** and **eigenvectors**.

Review: Eigenvalues and Eigenvectors

Recall from linear algebra:

An *eigenvalue* λ and corresponding *eigenvector* $\boldsymbol{x}$ of a matrix Q are a non-zero scalar and vector, respectively, which satisfy

$$Q\boldsymbol{x} = \lambda\boldsymbol{x}.$$

Review: Eigenvalues and Eigenvectors

Recall: If a matrix is invertible/non-singular, the equation **Ax** = **0** has only the trivial solution **x** = **0.**

Equivalently, this can be written

$$Q\boldsymbol{x} = \lambda I\boldsymbol{x}$$

where I is the identity matrix.

Rearranging gives

$$(\lambda I - Q)\boldsymbol{x} = 0$$

which implies that the matrix $\lambda I - Q$ must be *singular* for λ and Q to be eigenvalues/eigenvectors.

Review: Eigenvalues and Eigenvectors

A *singular* matrix A satisfies $\det A = 0$.

Thus to find the eigenvalues λ of Q, we can solve the ***characteristic polynomial*** given by

$$\det(\lambda I - Q) = 0\,.$$

Example: Find the eigenvalues/eigenvectors of

$$\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}.$$

Review: Eigenvalues and Eigenvectors

Example: Find the eigenvalues/eigenvectors of

$$\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}.$$

Answer:

$$\lambda_1 = 4, \boldsymbol{u_1} = [1,1]^T,$$
$$\lambda_2 = -3, \boldsymbol{u_2} = [2,-5]^T.$$

e.g., $\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$

Review: Eigenvalues and Eigenvectors

Note: The eigenvalues and eigenvectors are not necessarily always *real.*

e.g., the two eigenvalues of $\begin{bmatrix} 2 & -1 \\ 1 & 2 \end{bmatrix}$ are $2 \pm i$.

Why eigen-stuff again?

$$M = \begin{bmatrix} \frac{1}{40} & \frac{1}{6} & \frac{37}{120} & \frac{1}{40} & \frac{1}{40} & \frac{1}{40} \\ \frac{9}{20} & \frac{1}{6} & \frac{37}{120} & \frac{1}{40} & \frac{1}{40} & \frac{1}{40} \\ \frac{9}{20} & \frac{1}{6} & \frac{1}{40} & \frac{1}{40} & \frac{1}{40} & \frac{1}{40} \\ \frac{1}{40} & \frac{1}{6} & \frac{1}{40} & \frac{1}{40} & \frac{9}{20} & \frac{7}{8} \\ \frac{1}{40} & \frac{1}{6} & \frac{37}{120} & \frac{9}{20} & \frac{1}{40} & \frac{1}{40} \\ \frac{1}{40} & \frac{1}{6} & \frac{1}{40} & \frac{9}{20} & \frac{9}{20} & \frac{1}{40} \end{bmatrix}$$

The Page Rank process is actually converging towards a specific ***eigenvector*** of the Markov matrix, M.

Example from last time, 10 iterations gave:

$$[0.05205, \quad 0.07428, \quad 0.05782, \quad 0.34797, \quad 0.19975, \quad 0.26810]^T$$

The eigenvector of M corresponding to an eigenvalue of 1 is (approximately):

$$[0.05170, \quad 0.07367, \quad 0.05741, \quad 0.34870, \quad 0.19990, \quad 0.26859]^T$$

Convergence of Page Rank

To show that Page Rank converges we first need a few more properties & definitions involving Markov matrices…

1. Every Markov matrix Q has 1 as an eigenvalue. (Th'm 7.5)
2. Every eigenvalue of a Markov matrix Q satisfies $|\lambda| \leq 1$. So 1 is its *largest* eigenvalue. (Th'm 7.6)
3. A Markov matrix Q is a *positive* Markov matrix if $Q_{ij} > 0 \ \forall i, j$ (Def'n 7.7).
4. If Q is a positive Markov matrix, then there is **only one** linearly independent eigenvector of Q with $|\lambda| = 1$. (Th'm 7.8)

1. Every Markov matrix Q has 1 as an eigenvalue.

Eigenvalues of Q and Q^T are equal, since $\det(Q) = \det(Q^T)$.
Now, notice that $Q^T \boldsymbol{e} = \boldsymbol{e}$; why?

Since the columns of Q sum to 1, so do rows of Q^T.
For example:

$$Q^T \boldsymbol{e} = \begin{bmatrix} \frac{1}{4} & \frac{1}{8} & 0 \\ \frac{1}{2} & \frac{7}{8} & 0 \\ \frac{1}{4} & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{8} & \frac{7}{8} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

1. Every Markov matrix Q has 1 as an eigenvalue.

Since $Q^T \boldsymbol{e} = (1)\boldsymbol{e}$, 1 is therefore an eigenvalue of Q^T, with eigenvector $\boldsymbol{e}$.

We already said that the eigenvalues of Q and Q^T are equal, since $\det(Q) = \det(Q^T)$. (However, eigenvectors can differ.)

So 1 is *also* an eigenvalue of Q.

2. Every eigenvalue of a Markov matrix Q satisfies $|\lambda| \leq 1$. So 1 is its *largest* eigenvalue. (Th'm 7.6)

We will show that $|\lambda| \leq 1$ for Q^T (and therefore also for Q).

Let's work it through…

Super Fun Course Evaluation Time! ☺

http://evaluate.uwaterloo.ca/

Any and all (preferably constructive) suggestions are welcome.

Feedback from course evaluations is used for:

1. Improving future offerings of CS370.
2. Improving future versions of Christopher (i.e., identify areas/ways to enhance my teaching skills).
3. Evaluating my teaching performance (e.g., for granting of tenure, promotions, raises, etc.)

So please be careful and thorough. Thank you!

Items 3. & 4.

3. Definition: A Markov matrix Q is a positive Markov matrix if $Q_{ij} > 0 \; \forall i, j$ (Def'n 7.7).

(This is just a definition, no proof req'd.)

4. If Q is a positive Markov matrix, then there is **only one** linearly independent eigenvector of Q with $|\lambda| = 1$. (Th'm 7.8)

(We won't prove this. See notes for a reference.)

Implication: If Q is positive Markov, then $Q\boldsymbol{x} = \boldsymbol{x}$ for some $\boldsymbol{x}$.

If also $Q\boldsymbol{y} = \boldsymbol{y}$, then $\boldsymbol{y} = c\boldsymbol{x}$ for some scalar c. i.e. $\boldsymbol{y}$ is a multiple of $\boldsymbol{x}$.

Eigenvector with $\lambda = 1$ is *unique!*

Page Rank Convergence

We can now prove that Page Rank *will* converge.

Let’s do this...

Convergence Rate

The number of iterations required for Page Rank to converge to the final vector p^{∞} depends on the size of the *2nd largest* eigenvalue, $|\lambda_2|$.

Can you see why?

$$\boldsymbol{p^k} = \left(M^k\right)\boldsymbol{p^0} = c_1\boldsymbol{x_1} + \sum_{l=2}^{R} c_l(\lambda_l)^k\,\boldsymbol{x_l}$$

The 2nd largest eigenvalue dictates the *slowest* rate at which the "unwanted" components of $\boldsymbol{p^0}$ are shrinking.

Convergence Rate

It turns out that for our google matrix, $|\lambda_2| \approx \alpha$. (We won't prove.)

Recall: α dictated the balance between following real links, and teleporting randomly.

e.g., if $\alpha = 0.85$, then $|\lambda_2|^{114} \approx |0.85|^{114} \approx 10^{-8}$. What does this say?

After 114 iterations, any vector components of $\boldsymbol{p^0}$ *not* corresponding to the eigenvalue $|\lambda_1|$ will be scaled down by about $\sim 10^{-8}$ (or smaller!)

The resulting vector $\boldsymbol{p^{114}}$ is likely to be a good approximation of the dominant eigenvector, $\boldsymbol{x_1}$.

Effect of α

A small value of $|\lambda_2| \approx \alpha$ implies faster convergence.

So speed it up by choosing as small α as possible?

No! $\alpha = 0$ implies **only** random teleportation!

This ignores the web's link structure completely, so the ranking is meaningless (all equal).

Page Rank

We have answered our remaining questions about Page Rank:

- It **can** be implemented efficiently by exploiting sparsity intelligently.
- It **does** converge, given the properties of the transition matrix M.
- Its **convergence rate** depends on the choice of the random teleportation parameter α.

Page Rank as *Power iteration*

This algorithm to compute the dominant eigenvector is called the power iteration (or power method).

It has theoretical implications and extensions, as well as practical applications beyond ranking of web pages.

e.g., Recommender systems (for services like Facebook, Twitter, Amazon, etc.):

WTF: The Who to Follow Service at Twitter

Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, Reza Zadeh

Twitter, Inc.
@pankaj @ashishgoel @lintool @aneeshs @dongwang218 @reza_zadeh

Page Rank

This concludes our theoretical and practical tour of Page Rank.

Page Rank…

… is a direct application of the classic power iteration; the key insight was to interpret web links as importance indicators, and use this knowledge to express page ranking as finding an eigenvector.

…offers a nice example of how numerical linear algebra techniques can make possible the solution of large-scale problems in real-world applications.