Lecture 20: PageRank and Gaussian Elimination

Leili Rafiee Sevyeri

Based on lecture notes by me and many previous CS370 instructors

Winter 2024

Cheriton School of Computer Science
University of Waterloo

Second Step: Exploiting Sparsity

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

Sadly, our google matrix M was **fully dense**. No zeros at all!

A dense matrix-vector multiply with $1,000,000,000^2$ entries is slooooooooooooooooO00O...O00OOooooOO000OOOoooooow.

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

$$p^{n+1} = Mp^n$$

without ever creating/storing M!

Exploiting Sparsity in M

dense column in "dead end"

We have $M = \alpha \left(P + \frac{1}{R} e d^T \right) + \frac{(1-\alpha)}{R} e e^T$

sparse, not all pages are linked together

fully dense

Consider computing

$$M p^n = \underset{(1)}{\alpha P p^n} + \underset{(2)}{\frac{\alpha}{R} e d^T p^n} + \underset{(3)}{\frac{(1-\alpha)}{R} e e^T p^n}$$

Output p^{n+1} is a vector, and a sum of 3 vectors:

(1) is a sparse matrix-vector multiply. It can be done efficiently.

(3) involves $ee^Tp^n = e(\underbrace{e^Tp^n}_{\text{Scalar}})$ which requires the "dot-product" e^Tp^n.

This is just 1, since p^n is a probability vector.

$\therefore$ we can simply add $\left(\frac{1-\alpha}{R}\right)e$ for (3).

(2) is similar: Compute $\frac{\alpha}{R}(\underbrace{d^Tp^n}_{\text{Scalar}})e$

$$d^Tp^n = [\qquad]\begin{bmatrix} \\ \\ \end{bmatrix} = [\;]$$

So, $p^{n+1} = Mp^n = (1) + (2) + (3)$, with no dense matrix-vector multiplication. We never form M explicitly.

Algorithm

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

Page Rank Algorithm

$$\begin{aligned}
&\mathbf{p}^0 = \mathbf{e}/R \\
&\text{For } k = 1, \ldots, \text{ until converged} \\
&\qquad \mathbf{p}^k = M\mathbf{p}^{k-1} \\
&\qquad \text{If } \max_i |[\mathbf{p}^k]_i - [\mathbf{p}^{k-1}]_i| < tol \text{ then quit} \\
&\text{EndFor}
\end{aligned} \tag{7.7.1}$$

Google Search: Other Factors

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

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

In modern search engines, many factors besides pure link-based ranking can come into play.

(Hence, Search Engine Optimization (SEO) is a lucrative business.)

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 $\mathbf{x}$ of a matrix Q are a scalar and non-zero vector, respectively, which satisfy

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

Review: Eigenvalues and Eigenvectors

Equivalently, this can be written

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

where I is the identity matrix.

Rearranging gives

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

$x \neq 0$

which implies that the matrix $\lambda I - Q$ must be *singular* for λ and $\mathbf{x}$ to be an eigenvalue/eigenvector pair, since we want $\mathbf{x} \neq \mathbf{0}$.

Quick Review: Eigenvalues and Eigenvectors

$\lambda I - Q$

$p(\lambda) = \det(\lambda I - Q) = 0$

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

Solution: To find eigenvalues, we solve $\det(\lambda I - Q) = 0$

$$\det(\lambda I - Q) = \det \begin{bmatrix} \lambda - 2 & -2 \\ -5 & \lambda + 1 \end{bmatrix} = \lambda^2 - \lambda - 12 = 0$$

factors as $(\lambda - 4)(\lambda + 3) = 0$, so roots are:

$\lambda_1 = 4$, $\lambda_2 = -3$.

To find corresponding eigenvectors, plug λ back in:

$Qx = \lambda x$. So,

$$\begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4x_1 \\ 4x_2 \end{bmatrix}$$

1st row says $2x_1 + 2x_2 = 4x_1 \implies x_1 = x_2$

(Second row tells the same thing!)

Therefore, any vector $\vec{u}_1 = c_1 \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ for arbitrary non-zero scalar c_1 is an eigenvector for $\lambda_1 = 4$.

Likewise, for $\lambda_2 = -3$, we get $\vec{u}_2 = c_2 \begin{bmatrix} 2 \\ -5 \end{bmatrix}$ which is the other eigenvalue.

Review: Eigenvalues and Eigenvectors

Note: In the general case, the eigenvalues are not necessarily always real.

e.g., the two eigenvalues of $\begin{bmatrix} 2 & -1 \\ 1 & 5 \end{bmatrix}$ are $2 \pm i$.

Why eigen-stuff again?

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

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

With the earlier example 10 iterations gave:

$$M^{10} p^{0} = [0.05205, 0.07428, 0.05782, 0.34797, 0.19975, 0.26810]^T$$

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

$$\text{eigenvector} \rightarrow [0.05170, 0.07367, 0.05741, 0.34870, 0.19990, 0.26859]^T$$

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 $Q^T e = e$; why? ✓

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

For example:

$$\begin{bmatrix} \frac{1}{4}+\frac{1}{2}+\frac{1}{4} \\ \frac{1}{8}+\frac{7}{8}+0 \\ 0+0+1 \end{bmatrix}$$

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

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

$Q^T e = e \longrightarrow \left.\begin{array}{l} Q^T e = (1).e \\ Q^T e = \lambda e \end{array}\right\} \begin{array}{l} \lambda = 1 \\ \vec{x} = \vec{e} \end{array}$

Since $Q^T e = (1)e$, then $\lambda = 1$ is therefore an eigenvalue of Q^T, with eigenvector e (by def'n).

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.

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...

Markov Matrices Q Satisfy $|\lambda_i| \le 1$

First show $|\lambda_i| \le 1$ for Q^T instead...

Let λ and $\vec{x}$ be an eigenvalue/vector pair for Q^T.

$\therefore$ $Q^T \vec{x} = \lambda \vec{x}$

Let k be the index of the largest magnitude entry of $\vec{x}$. So, $|x_j| \le |x_k|$ for all j.

We have:

$$(Q^T x)_k = \sum_{j=1}^{n} \underbrace{Q_{jk}}_{Q^T_{kj}} x_j = \lambda x_k$$

$$|\lambda x_k| = |\lambda||x_k| = \left|\sum_{j=1}^{n} Q_{jk}\, x_j\right|$$

$$\leq \sum_{j=1}^{n} Q_{jk} |x_j|$$

used Δ inequality and Q's entries being non-negative.

$$\leq \sum_{j=1}^{n} Q_{jk} |x_k|$$

Since $|x_j| \leq |x_k|$

$$\leq |x_k| \underbrace{\left(\sum_{j=1}^{n} Q_{jk}\right)}_{1}$$

since column sums of Q are 1.

So, $|\lambda||x_k| \leq |x_k|$, and $|\lambda| \leq 1$ for Q^T, and also for Q, since they have the same eigenvalues.

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 if curious.)

Implication: If Q is positive Markov, then $Q\mathbf{x} = (1)\mathbf{x}$ for some $\mathbf{x}$. $x \neq 0$

If also $Q\mathbf{y} = \mathbf{y}$, then $\mathbf{y} = c\mathbf{x}$ for some scalar c. i.e. $\mathbf{y}$ is a multiple of $\mathbf{x}$. $y \neq 0$

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

Page Rank Convergence

With all these facts, we can now prove that Page Rank *will* converge.

Let's do it!

Page Rank Convergence

If M is a positive Markov matrix, PageRank converges to a unique vector p^∞ for initial probability vector p^0.

Let $\vec{x}_\ell$ be the corresponding eigenvector for λ_ℓ, for all ℓ. Assume we can write p^0 as a linear combination of eigenvectors $\vec{x}_\ell$. Then:

$$p^0 = \sum_\ell c_\ell \vec{x}_\ell$$ for scalar c_ℓ.

Assume eigenvalues are in sorted order. So,

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \dots$$

Page Rank Convergence

Then $\vec{x}_1$ corresponds to λ_1.

PageRank computes

$$(M^k)p^0 = M^k \sum_{\ell=1}^{R} c_\ell \vec{x}_\ell$$

$$= \sum_{\ell=1}^{R} (M^k) c_\ell \vec{x}_\ell$$

$$= \sum_{\ell=1}^{R} \lambda_\ell^k c_\ell \vec{x}_\ell$$

Since $\vec{x}_\ell$ is an eigenvector of M.

$$M\vec{x}_\ell = \lambda \vec{x}_\ell$$

$$= \underbrace{c_1 \vec{x}_1}_{\lambda_1 = 1} + \sum_{\ell=2}^{R} \lambda_\ell^k c_\ell \vec{x}_\ell$$

Theorem 7.8 said that $|\lambda_\ell| < 1$ for $\ell > 1$, since $\lambda_1 = 1$ was unique. Hence

$$\lim_{k\to\infty} \lambda_\ell^k = 0 \quad \text{for } \ell > 1.$$

$$\therefore\ p^\infty = \lim_{k\to\infty} (M^k) p^0 = c_1 \vec{x}_1$$

Other components are scaled towards 0.

If we start with a different probability vector

$$q^0 = \sum b_\ell \vec{x}_\ell$$

we find $\quad q^\infty = b_1 \vec{x}_1$

Since q^∞ and p^∞ are probability vectors, both sum to 1. Then

$$\sum_{i=1}^{R} b_1 \, x_1(i) = \sum_{i=1}^{R} c_1 \, x_1(i) = 1$$

$$b_1 \cancel{\left(\sum_{i=1}^{R} x_1(i) \right)} = c_1 \cancel{\left(\sum_{i=1}^{R} x_1(i) \right)}$$

$$\Longrightarrow b_1 = c_1$$

$\therefore\ p^\infty = q^\infty$, so PageRank converges to a unique vector. ✓

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?

$$\mathbf{p}^k = (M^k)\mathbf{p}^\mathbf{0} = c_1\mathbf{x_1} + \sum_{\ell=2}^{R} c_\ell(\lambda_\ell)^k\mathbf{x_1}$$

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

Convergence Rate

It turns out that for our google matrix, $|\lambda_2| \approx \alpha$ (We won't prove it.)
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 $\mathbf{p^0}$ not corresponding to the eigenvalue $|\lambda_1|$ will be scaled down by about $\sim 10^{-8}$ (or smaller!)

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

Effect of α

$$M = \underbrace{\alpha p'}_{=0} + \frac{1-\alpha}{n} e e^T$$

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).

Essentially, α trades off accuracy for efficiency.

End of Lecture 20

Numerical Linear Algebra
Gaussian Elimination

Numerical Linear Algebra

The study of algorithms for performing linear algebra operations ***numerically*** (i.e., approximately, on a computer, with floating point).

- Matrix/vector arithmetic
- Solving linear systems of equations
- Taking norms
- Factoring & inverting matrices
- Finding eigenvalues/eigenvectors (e.g. PageRank!)
- Etc.

Our *numerical* methods often differ from familiar exact methods, due to efficiency concerns, floating point error, stability, etc.

Applications

Application areas include:

- Fitting polynomials and splines.
- Implicit time integration.
- Optimization problems.
- Machine learning, statistics.
- Engineering.
- Computational finance.
- Computational biology.
- Image processing.
- Data mining & search.
- Computer vision.
- Etc.

Nearly everywhere numerical computation is used, numerical linear algebra plays some role.

Solving Linear Systems of Equations

Many many many practical problems rely on solving systems of linear equations of the form

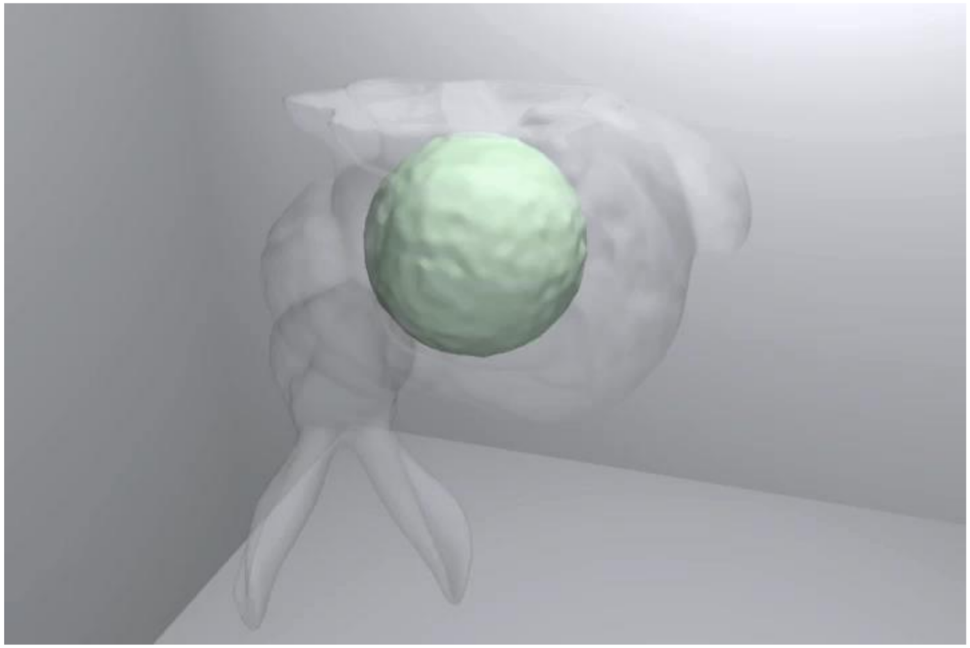
$$Ax = b$$

where A is a matrix, b is a right-hand-side (column) vector, and x is a (column) vector of unknowns.

Example: Animating Fluids

Computing one frame of animation requires solving a linear system with > **one million unknowns**.

- i.e. matrix A has dimensions $> 1,000,000 \times 1,000,000$.



Must be done once per frame; animations are usually played back at 30 frames / second.

- e.g. for 10 seconds of video, must solve 300 linear systems with size $1,000,000^2$ each.

So: We need methods to solve linear systems **efficiently** and **accurately**.

Review: Gaussian Elimination

In your linear algebra class, you would have seen *Gaussian Elimination*.

This involves:

- eliminating variables via row operations, until only one remains.
- back-substituting to recover the value of all the other variables.

This was done by applying combinations of:

1. Multiplying a row by a constant.
2. Swapping rows.
3. Adding a multiple of one row to another row.

Gaussian Elimination: $Ax = b$

(Some) numerical algorithms use Gaussian elimination, too. But it is interpreted differently...

Our view will be the following:

1. **Factor** matrix A into $A = LU$, where L and U are triangular.
2. **Solve** $Lz = b$ for intermediate vector z.
3. **Solve** $Ux = z$ for x.

Gaussian Elimination as Factorization

Solve $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 2 \\ 1 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 4 \\ 2 \end{bmatrix}$ for the vector $\vec{x}$.

Solution