谷歌如何从网络的大海里捞到针

顾国勇
http://maths.nju.edu.cn/~ggu/

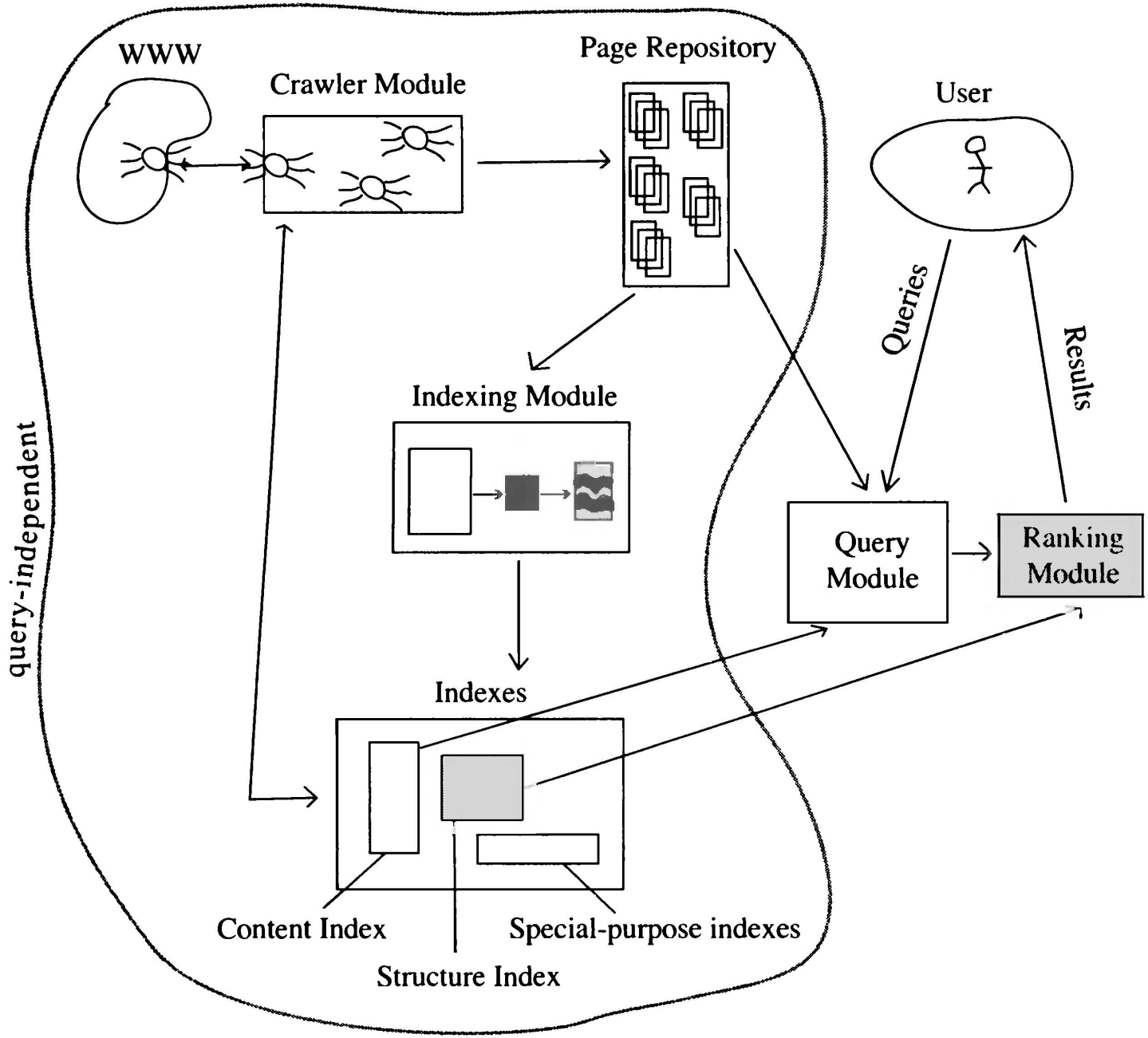
南京大学 数学系

Outline

Outline

谷歌简介

- 1998年，正值网络的增长步伐已经超过当时搜索引擎的能力范围，Google 由 Sergey Brin 和 Lawrence Page 共同创建（当时在斯坦福大学攻读理工博士）
- Google 名字来源于一个数学大数 googol（数字 1 后有 100 个 0，即自然数 10^{100}）单词错误的拼写方式，象征着为人们提供搜索海量优质信息的决心
- Google 官方的公司使命为：To organize the world's information and make it universally accessible and useful
- 换言之，他们首先希望，将搜索引擎引入一个更开放的、更学术化的环境，来改进搜索引擎的设计
- 其次，他们感到其搜索引擎产生的统计数据能够为学术研究提供很多的有趣信息


WWW
Crawler Module
Page Repository
User
Queries
Results
Indexing Module
query-independent
Query Module
Ranking Module
Indexes
Content Index
Special-purpose indexes
Structure Index

网络蜘蛛

- 网络蜘蛛（Web spider）也叫网络爬虫（Web crawler）、蚂蚁（ant）、自动检索工具（automatic indexer）、网络疾走（WEB scutter），是一种“自动化浏览网络"的程序，或者说是一种网络机器人（Internet bot）
- 网络爬虫始于一张被称作种子的地址（URLs）列表. 当网络爬虫访问这些地址时，它们会甄别出页面上所有的超链接，并将它们写入一张 " 待访列表 " ，即 " 爬行疆域 " （crawl frontier）. 此疆域上的地址将被按照一套策略循环访问
- 互联网资源卷帙浩繁，这意味着网络爬虫在一定时间内只能下载有限数量的网页，因此它需要优化它的下载方式.
- 互联网资源瞬息万变，这也意味着网络爬虫下载的网页在使用前就已经被修改甚至是删除了

Outline

如何辨别谁重要

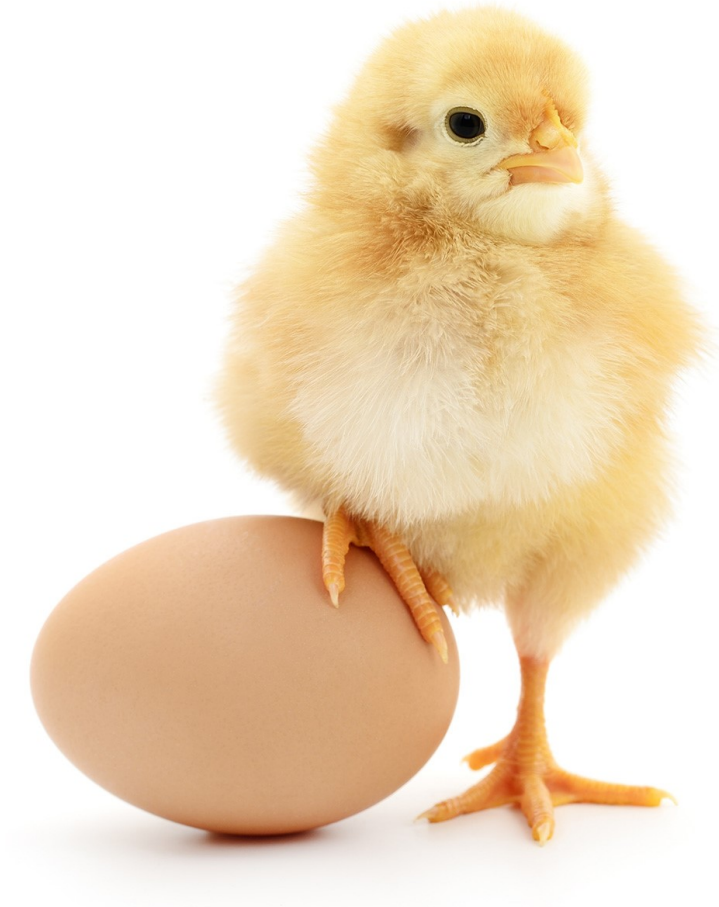
- 如果你曾建立过一个网页，你应该会列入一些你感兴趣的链接，它们很容易使你点击到其它含有重要、可靠信息的网页. 这样就相当于你肯定了你所链接页面的重要性
- Sergey Brin 和 Lawrence Page 的基本想法是：
  一个网页的重要性是由链接到它的其他网页的数量及其重要性来决定
- 谷歌的网页排序算法每隔一段时间在所有网页中进行一次受欢迎程度的评估，以确定哪些网页最重要

重要性量化

- 我们对任意一个网页 P，以 $I(P)$ 来表述其重要性
- 假定网页 P_j 有 l_j 个链接. 如果这些链接中的一个链接到网页 P_i，那么网页 P_j 将会将其重要性的 $1/l_j$ 赋给 P_i
- 网页 P_i 的重要性就是所有指向这个网页的其他网页所贡献的重要性的加和
- 换言之，如果我们记链接到网页 P_i 的网页集合为 B_i，那么

$$I(P_i) = \sum_{P_j \in B_i} \frac{I(P_j)}{l_j}.$$

这或许让你想起“先有鸡还是先有蛋”的问题：
为了确定一个网页的重要性，我们首先得知道所有指向它的其他网页的重要性



超链矩阵

建立一个矩阵，称为超链矩阵（hyperlink matrix），$H = [H_{ij}]$，其中第 i 行第 j 列的元素为

$$H_{ij} = \begin{cases} \dfrac{1}{l_j}, & \text{如果} P_j \in B_i \\ 0, & \text{上述条件不成立} \end{cases}$$

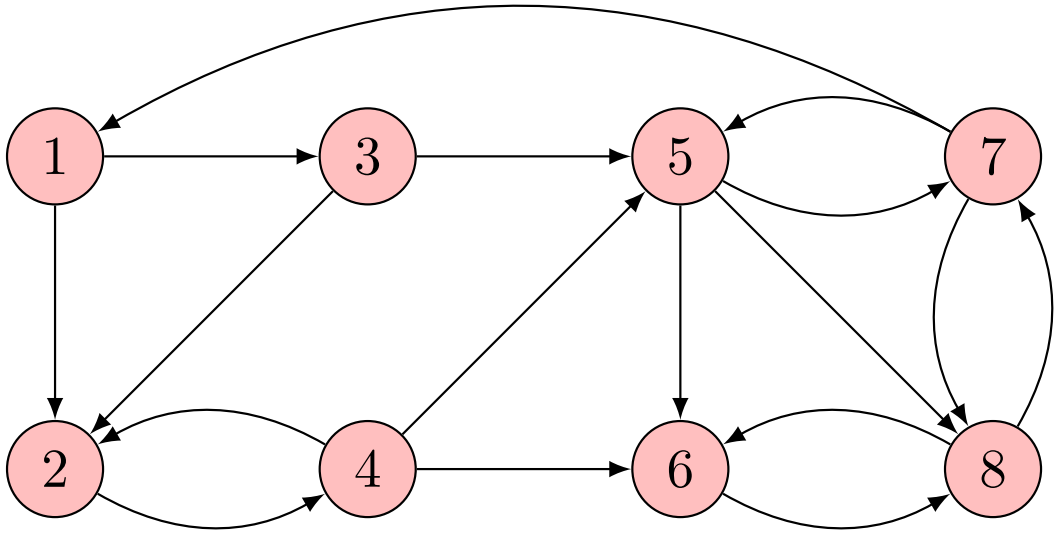
注意到 H 有一些特殊的性质：

- 所有的元都是非负的
- 除非对应这一列的网页没有任何链接，它的每一列的和为 1
- 所有元均非负且列和为 1 的矩阵称为随机矩阵

平稳向量

- 我们还需要定义向量 $I = [I(P_i)]$，其元素为所有网页的网页排序— 重要性的排序值
- 前面定义的网页排序可以表述为 $I = HI$
- 换言之，向量 I 是矩阵 H 对应特征值 1 的特征向量
- 我们也称之为矩阵 H 的平稳向量（stationary vector）

一个例子

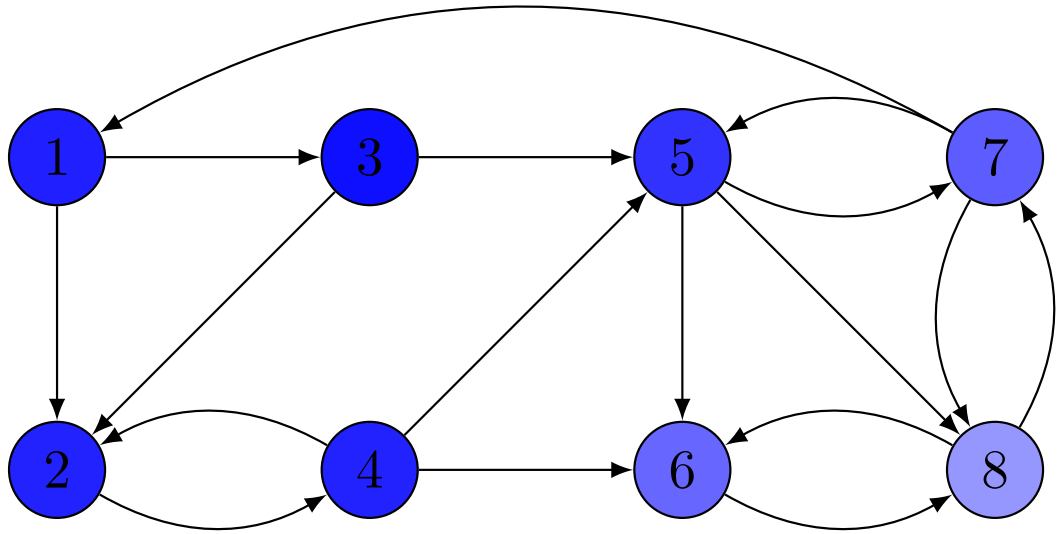


$H^T \cdot 1 = 1$

其相应的超链矩阵 H 为

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/3 & 0 \end{bmatrix}, \quad I = \begin{bmatrix} 0.0600 \\ 0.0675 \\ 0.0300 \\ 0.0675 \\ 0.0975 \\ 0.2025 \\ 0.1800 \\ 0.2950 \end{bmatrix}$$

下图是阴影化的图，其中网页排序值越高的网页阴影越浅



Outline

幂法（power method）原理

- 设 $A \in \mathbb{R}^{n\times n}$ 可对角化
- $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$ 为特征值（计重数），$v_1, v_2, \ldots, v_n$ 为相对应的单位化的特征向量
- 初始向量 b_0 可以分解为 $b_0 = c_1v_1 + c_2v_2 + \cdots + c_nv_n, (c_1 \neq 0)$

$$\begin{aligned} A^k b_0 &= c_1A^kv_1 + c_2A^kv_2 + \cdots + c_nA^kv_n \\ &= c_1\lambda_1^k v_1 + c_2\lambda_2^k v_2 + \cdots + c_n\lambda_n^k v_n \\ &= c_1\lambda_1^k\left(v_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \cdots + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k v_n\right) \end{aligned}$$

$b_k = \frac{A^k b_0}{\|A^k b_0\|}$ 收敛于 v_1，收敛速度取决于 $\frac{|\lambda_2|}{|\lambda_1|}$

Perron-Frobenius theorem

For real square matrix with positive entries

- has a unique largest real eigenvalue
- the corresponding eigenvector has strictly positive components

every non-negative matrix can be obviously obtained as a limit of positive matrices

- the existence of an eigenvector with non-negative components
- obviously the corresponding eigenvalue will be non-negative and greater or equal in absolute value than all other eigenvalues

primitive matrices $\Longleftrightarrow$ irreducible aperiodic non-negative matrices

statements of the Perron-Frobenius theorem for positive matrices remain true for primitive matrices

Gershgorin circle theorem

- $A \in \mathbb{C}^{n \times n}$, $R_i = \sum_{j \neq i} |a_{ij}|$
- $D(a_{ii}, R_i)$ be the closed disc centered at a_{ii} with radius R_i
- every eigenvalue of A lies within at least one of the Gershgorin discs $D(a_{ii}, R_i)$
- the eigenvalues must also lie within the Gershgorin discs corresponding to the columns of A

超链矩阵 H 的特点

- 有很多方法可以找到一个方阵的特征向量
- 我们面对的是一个特殊的挑战，因为矩阵 H 是一个这样的方阵，它的每一列都对应谷歌检索到的一个网页
- 也就是说，H 大约有 $n = 250$ 亿行和列（2.5×10^{10}，2012年）
- 不过其中大多数的元都是 0，即 H 稀疏
- 研究表明每个网页平均约有 10 个链接，换言之，平均而言，每一列中除了 10 个元外全是 0

我们将选择被称为幂法（power method）的方法来找到矩阵 H 的平稳向量 I

幂法（power method）

超链矩阵 H（无全 0 列）有一个特征值为 1，所有特征值的模 ≤ 1

首先选择 I 的备选向量 I^0，进而按下式产生向量序列 I^k

$$I^{k+1} = HI^k, \quad k = 0, 1, 2, \ldots$$

一般原理：序列 I^k 将收敛到平稳向量 I

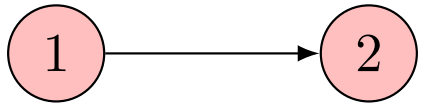
I^0	I^1	I^2	I^3	I^4	$\cdots$	I^{60}	I^{61}
1	0	0	0	0.0278	$\cdots$	0.06	0.06
0	0.5	0.25	0.1667	0.0833	$\cdots$	0.0675	0.0675
0	0.5	0	0	0	$\cdots$	0.03	0.03
0	0	0.5	0.25	0.1667	$\cdots$	0.0675	0.0675
0	0	0.25	0.1667	0.1111	$\cdots$	0.0975	0.0975
0	0	0	0.25	0.1806	$\cdots$	0.2025	0.2025
0	0	0	0.0833	0.0972	$\cdots$	0.18	0.18
0	0	0	0.0833	0.3333	$\cdots$	0.295	0.295

三个重要的问题

自然而然产生的三个问题是：

- 序列 I^k 总是收敛吗？（即运算多次后，I^k 和 I^{k+1} 几乎是一样的）
- 收敛后的平稳向量是否和初始向量 I^0 的选取没有关系？
- 重要性排序值是否包含了我们想要的信息？

对目前的方法而言，上述三个的答案都是否定的！



其矩阵为 $H = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$，幂法的过程：

I^0	I^1	I^2	I^3
1	0	0	0
0	1	0	0

- 两个网页的重要性排序值均为 0，这样我们无法获知两个网页之间的相对重要性信息
- 问题在于网页 P_2 没有任何链接. 因此，在每个迭代步骤中，它从网页 P_1 获取了一些重要性，但却没有赋给其他任何网页
- 没有任何链接的网页称为悬挂点（dangling nodes），显然在我们要研究的实际网络中存在很多这样的点

H 的概率化解释

- 想象我们随机地在网上跳转网页
- 也就是说，当我们访问一个网页时，一秒钟后我们随机地选择当前网页的一个链接到达另一个网页
- 例如，我们正访问含有 l_j 个链接的网页 P_j，其中一个链接引导我们访问了网页 P_i，那么下一步转到网页 P_i 的概率就是 $1/l_j$
- 由于跳转网页是随机的，我们用 T_j 表示停留在网页 P_j 上的时间那么我们从网页 P_j 转到网页 P_i 的时间为 T_j/l_j
- 如果我们转到了网页 P_i，那么我们必然是从一个指向它的网页而来. 这意味着

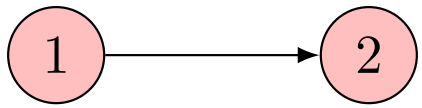
$$T_i = \sum_{P_j \in B_i} \frac{T_j}{l_j}$$

其中求和是对所有链接到 P_i 的网页 P_j 进行的

H 的概率化解释 cont'd

- 注意到这个方程与定义网页排序值的方程相同，因此 $I(P_i) = T_i$. 那么一个网页的网页排序值可以解释为随机跳转时花在这个网页上的时间
- 当然，这种表述中还存在一个问题：如果我们随机地跳转网页，在某种程度上，我们肯定会被困在某个悬挂点上，这个网页没有给出任何链接
- 为了能够继续进行，我们需要随机地选取下一个网页；也就是说，我们假定悬挂点可以链接到其他任何一个网页
- 这个效果相当于将超链矩阵 H 做如下修正：将其中所有元都为 0 的列替换为所有元均为 $1/n$ 的列
- 前者就对应于网页中的悬挂点. 这样修正后悬挂点就不存在了. 我们称修正后的新矩阵为 S

我们之前的例子，现在就变成了



其对应矩阵为 $S = \begin{bmatrix} 0 & 1/2 \\ 1 & 1/2 \end{bmatrix}$ 及特征向量 $I = \begin{bmatrix} 1/3 \\ 2/3 \end{bmatrix}$

换言之，网页 P_2 的重要性是网页 P_1 的两倍，符合我们的直观认知了

- 矩阵 S 有一个很好的性质，即其所有元均非负且每列的和均为1. 即，S 为随机矩阵
- 随机矩阵具有一些很有用的性质. 例如，随机矩阵总是存在平稳向量（对应特征值 1 的特征向量）

幂法实现

注意到 S 是由 H 通过一个简单的修正得到. 定义矩阵 A 如下：
对应于悬挂点的列的每个元均为 $1/n$，其余各元均为 0. 则

$$S = H + A$$

- 一般而言，幂法是寻找矩阵对应于绝对值最大的特征值的特征向量
- 就我们而言，我们要寻找矩阵 S 对应于特征值 1 的特征向量
- 首先要说到的是最好的情形. 在这种情形下，其他特征值的绝对值都小于 1；也就是说，矩阵 S 的其它特征值都满足 $|\lambda| < 1$

我们假定矩阵 S 的特征值为 λ_j 且

$$1 = \lambda_1 > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|,$$

对矩阵 S，假设对应于特征值 λ_j 的特征向量存在一个基向量 v_j. （这一假设在一般情况下并不一定要成立）

将初始向量 I^0 写成 $I^0 = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$，那么

$$\begin{aligned}
I^1 &= SI^0 = c_1 v_1 + c_2 \lambda_2 v_2 + \cdots + c_n \lambda_n v_n, \\
I^2 &= SI^1 = c_1 v_1 + c_2 \lambda_2^2 v_2 + \cdots + c_n \lambda_n^2 v_n, \\
&\cdots\cdots \\
I^k &= SI^{k-1} = c_1 v_1 + c_2 \lambda_2^k v_2 + \cdots + c_n \lambda_n^k v_n.
\end{aligned}$$

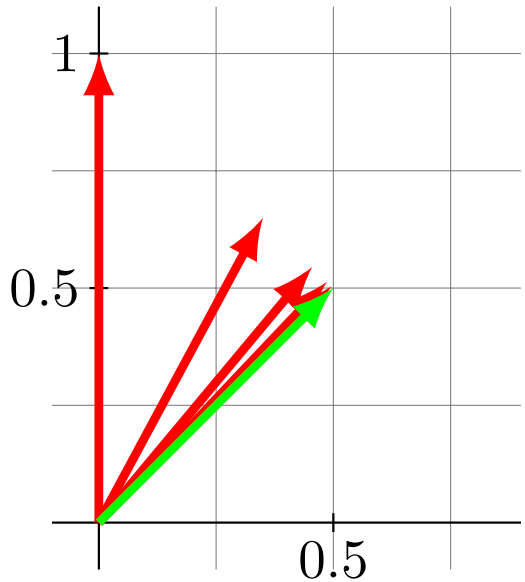
当 $j \geq 2$ 时，因为所有特征值的绝对值小于1，因此 $\lambda_j^k \to 0$. 从而 $I^k \to I = c_1 v_1$，后者是对应于特征值 1 的一个特征向量.

- $I^k \to I$ 的速度由 $|\lambda_2|$ 确定.
- 当 $|\lambda_2|$ 比较接近于 0 时，那么 $\lambda_2^k \to 0$ 会相当快.

例如，考虑下述矩阵$S = \begin{bmatrix} 0.65 & 0.35 \\ 0.35 & 0.65 \end{bmatrix}$

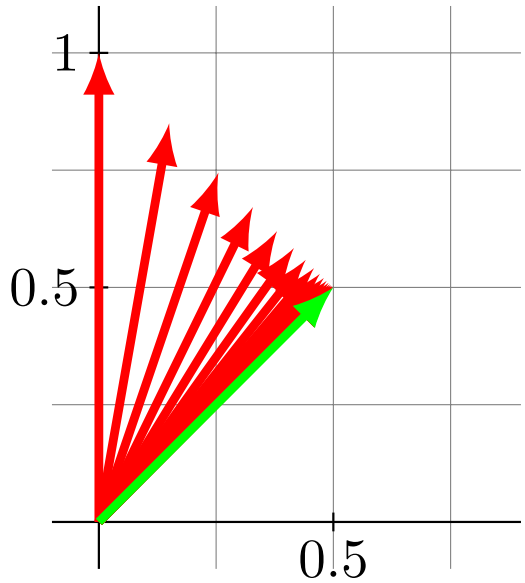
这个矩阵的特征值为 $\lambda_1 = 1$ 及 $\lambda_2 = 0.3$

可以看出用红色标记的向量 I^k 收敛到用绿色标记的平稳向量 I.



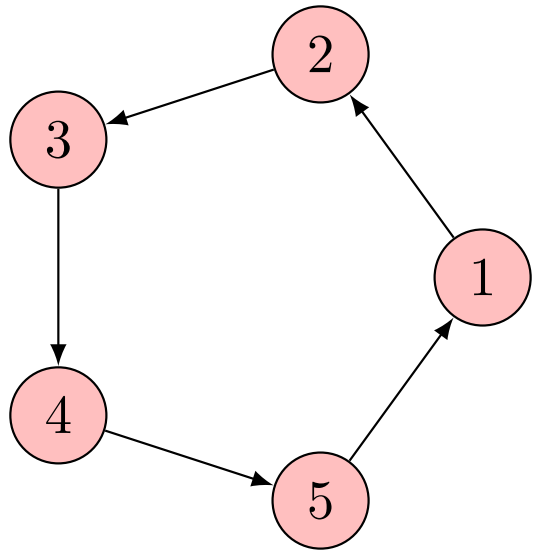
再考虑矩阵$S = \begin{bmatrix} 0.85 & 0.15 \\ 0.15 & 0.85 \end{bmatrix}$

其特征值为 $\lambda_1 = 1$ 及 $\lambda_2 = 0.7$. 可以看出，本例中向量 I^k 收敛到平稳向量 I 的速度要慢很多，因为它的第二个特征值较大



不顺之时

矩阵 S 需要满足 $\lambda_1 = 1$ 且 $|\lambda_2| < 1$. 然而，这一点并不总成立



矩阵 S 为

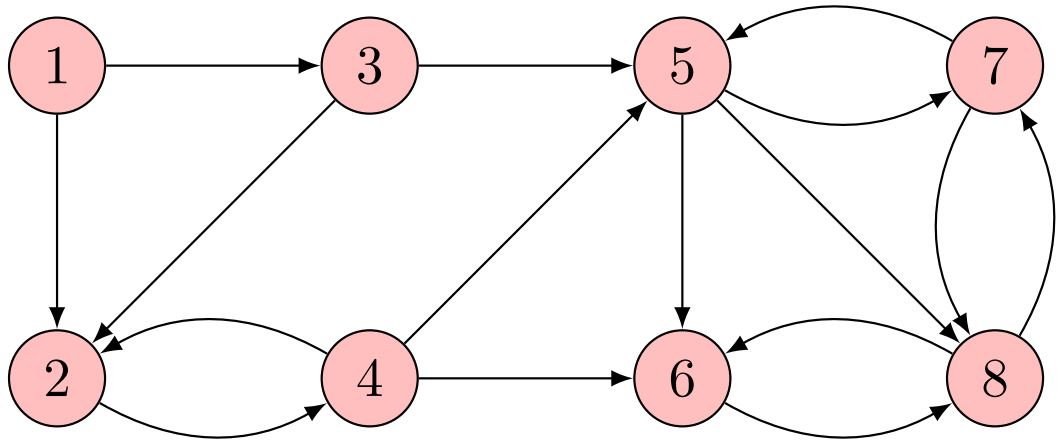
$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

那么我们可以得到

I^0	I^1	I^2	I^3	I^4	I^5
1	0	0	0	0	1
0	1	0	0	0	0
0	0	1	0	0	0
0	0	0	1	0	0
0	0	0	0	1	0

在这种情况下，向量序列 I^k 不再收敛.

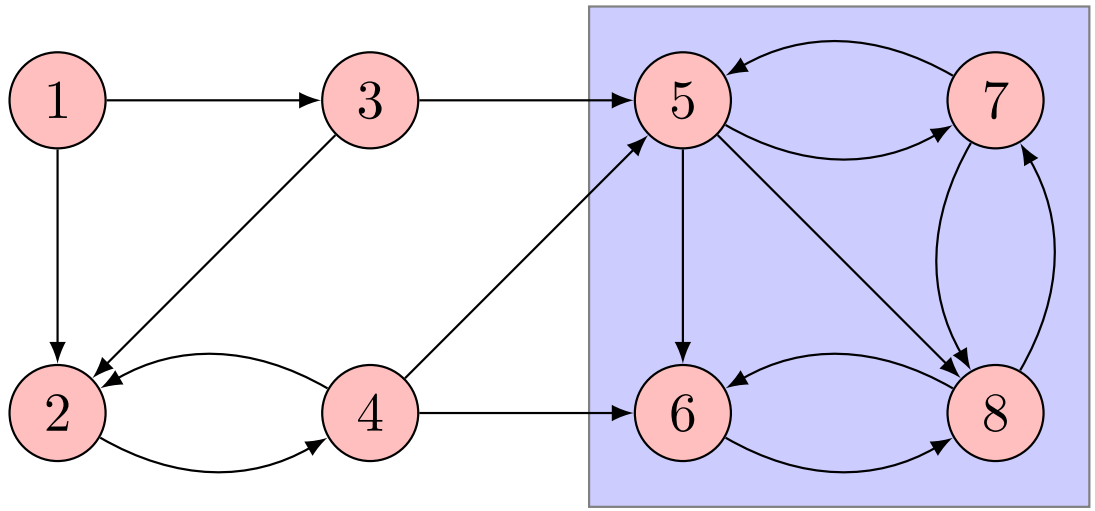
- 注意到矩阵 S 的第二个特征值满足 $|\lambda_2| = 1$，因此前述幂法的前提不再成立
- 为了保证 $|\lambda_2| < 1$，我们需要矩阵 S 为本原（primitive）矩阵
- 这意味着，对某个 m，S^m 的所有元均为正，换言之，若给定两个网页，那么从第一个网页经过 m 个链接后可以到达第二个网页
- 我们将看到如何修正矩阵 S 以获得一个本原随机矩阵，从而满足 $|\lambda_2| < 1$



矩阵 S 及平稳向量 I 为

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 1/3 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/3 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 1/3 & 1 & 1/2 & 0 \end{bmatrix}, \quad I = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.12 \\ 0.24 \\ 0.24 \\ 0.4 \end{bmatrix}$$

- 注意到前四个网页的排序值均为 0. 这使我们感觉不对：每个页面都有其它网页链接到它，显然总有人喜欢这些网页
- 一般来说，我们希望所有网页的重要性排序值均为正
- 问题在于，本例包含了一个小网络，即下图中蓝色方框部分



- 在这个方框中，有链接进入到蓝色方框，但没有链接转到外部
- 正如前述中关于悬挂点的例子一样，这些网页构成了一个“重要性水槽”，其他四个网页的重要性都被”排”到这个“水槽”中

- 这种情形发生在矩阵 S 为可约（reducible）时；也即，S 可以写成如下的块形式

$$S = \begin{bmatrix} * & 0 \\ * & * \end{bmatrix}$$

- 可以证明：如果矩阵 S 不可约，则一定存在一个所有元均为正的平稳向量
- 对一个网络，如果任意给定两个网页，一定存在一条由链接构成的路使得我们可以从第一个网页转到第二个网页，那么称这个网络是强连通的（strongly connected）
- 显然，上面最后的这个例子不是强连通的. 而强连通的网络对应的矩阵 S 是不可约的

矩阵 S 是随机矩阵，即意味着它有一个平稳向量. 我们还需要 S 满足

- 本原，从而 $|\lambda_2| < 1$；
- 不可约，从而平稳向量的所有元均为正.

最后一个修正

- 为得到本原且不可约的矩阵，我们修正随机跳转网页的方式
- 我们的随机跳转模式由矩阵 S 确定
  - 或者是从当前网页上的链接中选择一个
  - 或者是对没有任何链接的网页，随机地选取其他网页中的任意一个
- 为了做出修正，首先引入一个介于 0 到 1 之间的参数 α. 然后假定随机跳转的方式略作变动
- 遵循矩阵 S 的方式跳转的概率为 α
- 而随机地选择下一个页面的概率是 $1-\alpha$
- 注意到 G 为随机矩阵，因为它是随机矩阵的组合.
- 进而，矩阵 G 的所有元均为正，因此 G 为本原且不可约.

记所有元均为 1 的 $n \times n$ 矩阵为 J，则谷歌矩阵（Google matrix）：

$$G = \alpha S + (1-\alpha)\frac{1}{n}J$$

- 若$\alpha = 1$，则 $G = S$. 这意味着我们面对的是原始的网络超链结构.
- 若$\alpha = 0$，则 $G = 1/nJ$. 也即我们面对的是一个任意两个网页之间都有连接的网络，它已经丧失了原始的网络超链结构
- 显然，我们将会把 α 的值取得接近于 1，从而保证网络的超链结构在计算中的权重更大
- 然而，还有另外一个问题. 请记住，幂法的收敛速度是由第二个特征值的幅值 $|\lambda_2|$ 决定的
- 而对谷歌矩阵，已经证明了第二个特征值的幅值为 $|\lambda_2| \leq \alpha$
- 这意味着当 α 接近于 1 时，幂法的收敛速度将会很慢
- 作为这个矛盾的折中方案，$\alpha = 0.85$

$|\lambda_2| \leq \alpha$

存在可逆矩阵 $\begin{bmatrix} 1 & P \end{bmatrix}$（$P \in \mathbb{R}^{n \times n-1}$），其逆矩阵记成

$$\begin{bmatrix} q^T \\ Q^T \end{bmatrix} \quad \text{则} \quad \begin{bmatrix} q^T \\ Q^T \end{bmatrix} \begin{bmatrix} 1 & P \end{bmatrix} = \begin{bmatrix} q^T 1 & q^T P \\ Q^T 1 & Q^T P \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & I_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} q^T \\ Q^T \end{bmatrix} S^T \begin{bmatrix} 1 & P \end{bmatrix} = \begin{bmatrix} q^T \\ Q^T \end{bmatrix} \begin{bmatrix} 1 & S^T P \end{bmatrix} = \begin{bmatrix} q^T 1 & q^T S^T P \\ Q^T 1 & Q^T S^T P \end{bmatrix} = \alpha \begin{bmatrix} 1 & q^T S^T P \\ 0 & Q^T S^T P \end{bmatrix}$$

$$\begin{bmatrix} q^T \\ Q^T \end{bmatrix} (1/n) J^T \begin{bmatrix} 1 & P \end{bmatrix} = \begin{bmatrix} q^T \\ Q^T \end{bmatrix} \begin{bmatrix} 1 & (1/n) J^T P \end{bmatrix} = \begin{bmatrix} q^T 1 & (1/n) q^T J^T P \\ Q^T 1 & (1/n) Q^T J^T P \end{bmatrix}$$

$$= \begin{bmatrix} 1 & (1/n) q^T J^T P \\ 0 & (1/n) Q^T 1 1^T P \end{bmatrix} \quad (1-\alpha) \begin{bmatrix} 1 & (1/n) q^T J^T P \\ 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} q^T \\ Q^T \end{bmatrix} \underbrace{\left[\alpha S^T + (1-\alpha) \tfrac{1}{n} J^T \right]}_{G^T} \begin{bmatrix} 1 & P \end{bmatrix} =$$

计算排序向量 I

这个方法需要应用到一个维数约为 250 亿的方矩阵（2012年）！

- 幂法特别适用于这种情形,回想随机矩阵 S 可以写成下述形式

$$S = H + A.$$

- 从而谷歌矩阵有如下形式（其中 J 是元素全为 1 的矩阵）

$$G = \alpha H + \alpha A + \frac{1-\alpha}{n} J,$$

- 从而

$$GI^k = \alpha H I^k + \alpha A I^k + \frac{1-\alpha}{n} J I^k.$$

- 矩阵 H 的绝大部分元都是 0；平均而言，一列中只有 10 个元是非零数. 求 HI^k 的每个元时，只需要知道 10 个项即可
- 和矩阵 J 一样，矩阵 A 的行元素都是相同的. 求 AI^k 与 JI^k 相当于添加悬挂点或者所有网页的当前重要性排序值

当 α 取值接近于 0.85，布林和佩奇指出，需要 50 到 100 次迭代来获得对向量 I 的一个足够好的近似（$0.85^{50} \approx 0.00029576$）

- 计算到这个最优值需要几天才能完成
- 当然，网络是不断变化的. 首先，网页的内容，尤其是新闻内容，变动频繁
- 其次，网络的隐含超链结构在网页或链接被加入或被删除时也要相应变动
- 有传闻说，谷歌大约 1 个月就要重新计算一次网页排序向量 I
- 由于在此期间可以看到网页排序值会有一个明显的波动，一些人便将其称为谷歌舞会（Google Dance）

Outline

现状

- Google originally applied the PageRank algorithm to a web of 24 million pages in 1998.
- By 2014, the size of the indexed web had grown thousand-fold to the order of tens of billions.
- As the web continues to grow, challenges arise in the computation of PageRank.
- Although these challenges are somewhat mitigated by improving computer technology, they are also being addressed through academic research on algorithm efficiency.
- Recent approaches include a distributed randomized updating of PageRank within a web-aggregation framework.

Outline

HITS algorithm

还有一些其他使用网络的超链结构来进行网页排序的算法

值得一提的例子是 Hyperlink-Induced Topic Search (HITS) 算法，由 Jon Kleinberg 提出，它是 Teoma 搜索引擎的基础

- a good hub represented a page that pointed to many other pages
- a good authority represented a page that was linked by many different hubs

谢谢！