lec18.2 PageRank Algorithm - Markov

PageRank算法：马尔可夫链视角

马尔可夫链模型中的概念

- 节点 (node)：图中的每个网页被视为一个状态
- 弧 (arc)：超链接被视为状态之间的转移
- 在一个固定的状态下，所有转移的概率相等：即在一个网页上，点击任何一个链接的概率是相等的

随机网页冲浪者模型

这个框架模拟了一个随机网页冲浪者（Random Web Surfer）的行为：

- 一个虚拟的冲浪者随机点击链接
- 所有链接被点击的概率相等
- 浏览器上的“后退”按钮不使用，冲浪者也不输入URL
- 冲浪者点击了许多次后，最终停留在某个特定网页上的概率是多少

算法

状态转移概率矩阵 A 表示冲浪者从一个状态（页面） i 转移到另一个状态（页面） j 的概率

- $A_{ij} = \frac{1}{O_i}$，如果 $(i, j) \in E$，即如果页面 i 有一个链接指向页面 j，则转移概率是页面 i 的出链数量 O_i 的倒数
- 否则，$A_{ij} = 0$，即页面 i 没有链接指向页面 j

初始概率分布向量 P_0 表示一个冲浪者在每个状态（页面）上的初始概率分布

$$P_0 = (P_0(1), \ldots, P_0(n))^T$$

是一个 n 维列向量，表示开始时冲浪者在每个页面上的概率

于是我们有

$$\sum_{i=1}^{n} P_0(i) = 1$$
$$\sum_{j=1}^{n} A_{ij} = 1$$

这表示

- 在初始时刻，冲浪者处于某个状态（页面）上的概率和为 1，即冲浪者必然在某个页面上
- 对于每个状态 i，冲浪者总是会点击某个链接转移到其他状态。状态 i 的所有出链的概率和为1

> *实际上，上述状态转移概率矩阵的条件并不总是成立，因为有些网页没有出链。这些网页被称为悬挂页面（dangling pages）*

如果矩阵 A 对于每个 $i = 1, \ldots, n$ 都满足上述条件，即所有出链的概率和为 1，我们称 A 为马尔科夫链的随机矩阵（stochastic matrix）

> *在马尔可夫链模型中，给定初始概率分布 P_0，问经过 m 步或转移后，马尔可夫链（随机冲浪者）在每个状态 j 的概率是多少？*

在进行一次转移后，随机冲浪者处于状态 j 的概率可以计算为：

$$P_1(j) = \sum_{i=1}^{n} A_{ij}(1) P_0(i)$$

其中，$A_{ij}(1)$ 是在一次转移中，从 i 转移到 j 的概率；$A_{ij}(1) = A_{ij}$
求和是因为随机冲浪者一开始的位置可以是任何一个节点，而不是固定在某一个特定的节点。因此，我们需要考虑所有可能的初始位置及其相应的转移概率

上式的矩阵形式为：

$$P_1 = A^T P_0$$

经过 k 次转移后：

$$P_k = A^T P_{k-1}$$

根据马尔可夫链的遍历定理（Ergodic Theorem）：
如果 A 是不可约 (irreducible) 且 非周期 (aperiodic) 的，则由随机转移矩阵 A 定义的有限马尔可夫链（finite Markov chain）具有 唯一 的 平稳概率分布（stationary probability distribution）

这意味着：
经过一系列的状态转移后，概率分布 P_k 将收敛到一个稳态概率向量 Π，无论初始概率向量 P_0 是什么

$$\lim_{k \to \infty} P_k = \Pi$$

当我们达到稳态时，有

$$P_k = P_{k-1} = \Pi$$

因此，

$$\Pi = A^T \Pi$$

这意味着 Π 是矩阵 A^T 对应特征值为 1 的主特征向量
又

> *一个随机矩阵（stochastic matrix）总有一个特征值为1。所有其他特征值的绝对值都小于或等于1*

如果我们将 Π 视为 `PageRank` 向量 P，我们便得到了之前的方程

$$P = A^T P$$

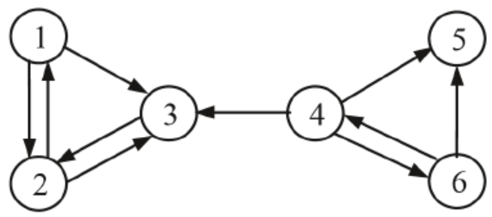
我们希望矩阵 A 满足以下三个条件：

1. Stochastic: 随机性。每一行的和为 1
2. Irreducible: 不可约性。意味着图是连通的，即从任何一个节点可以到达任何另一个节点。
3. Aperiodic: 非周期性。意味着图没有固定的周期性返回

这些条件在真实的网络图中并不总是满足
然而，我们可以通过修改矩阵 A 使其满足所有三个条件

1. 使 A 成为随机矩阵

- 对于每一个悬空页（dangling page）i，添加一组完整的出链接
- 从 i 到每一个页面的转移概率是 $\frac{1}{n}$

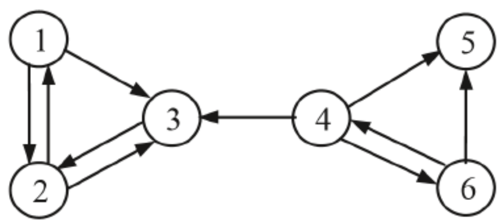


$$\bar{A} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Hyperlink graph

Transition probability matrix

$\Downarrow$



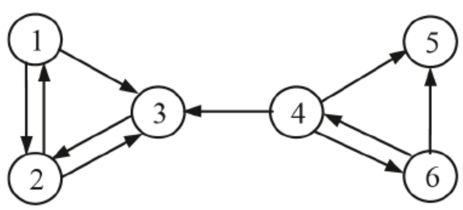
$$\bar{A} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Hyperlink graph

Transition probability matrix

2. 使 A 拥有不可约与非周期性

从每个页面添加一个链接到每个页面，并且给每个链接赋予一个由参数 d 控制的小转移概率



$$(1-d)\frac{\bar{E}}{n} + d\bar{A}^T = \begin{pmatrix} 1/60 & 7/15 & 1/60 & 1/60 & 1/6 & 1/60 \\ 7/15 & 1/60 & 11/12 & 1/60 & 1/6 & 1/60 \\ 7/15 & 7/15 & 1/60 & 19/60 & 1/6 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 1/60 \end{pmatrix}$$

where d is a number between 0 and 1, and $\bar{E}$ is a $n \times n$ square matrix of all 1's.

$(1-d)\frac{E}{n} + dA$ 是随机的（但已转置）、不可约的和非周期性的；其中 E 是 $n \times n$ 全 1 方阵

备注

1. 阻尼因子 (d) 的典型值
   - 在实际应用中，阻尼因子的典型值是 0.85

2. PageRank 算法的应用
    - PageRank 算法可以应用于任何图（不仅限于网页图）以对顶点进行排名
3. PageRank 算法的性质
    - PageRank 算法是基于图中的随机游走 (random walks) 的众多算法之一
4. 算法收敛性
    - 由于我们只对页面的排名感兴趣，实际收敛可能不是必要的。算法可以在达到一些最大迭代次数后终止，即使在达到容差 ϵ 之前
5. 实际应用中的迭代次数
    - 据报道，在包含 3.22 亿个链接的数据库上，该算法在大约 52 次迭代后收敛到可接受的容差

例子

假设有一个由5个网页组成的有向图，如下所示：

- A 指向 B 和 C
- B 指向 C 和 D
- C 指向 A
- D 指向 C 和 E
- E 指向 A 和 C

图的表示如下：

mathematica 复制代码

```mathematica
A → B, C
B → C, D
C → A
D → C, E
E → A, C
```

初始PageRank值

假设每个网页的初始PageRank值相同，设为1。即：

css 复制代码

```css
P(A) = 1
P(B) = 1
P(C) = 1
P(D) = 1
P(E) = 1
```

计算PageRank值

根据PageRank公式：

```scss
P(a) = (1 - d) / N + d * (Σ P(x) / O(x))
```

其中 d 是阻尼因子（通常设为0.85），N 是总的网页数量，$P(x)$ 是页面 x 的PageRank值，$O(x)$ 是页面 x 的出链数量。

第一次迭代

1. **计算P(A)**：
   - 来自C和E的贡献
   - $P(C)/O(C) = 1/1 = 1$
   - $P(E)/O(E) = 1/2 = 0.5$
   - $P(A) = (1 - 0.85)/5 + 0.85 * (1 + 0.5) = 0.03 + 0.85 * 1.5 = 0.03 + 1.275 = 1.305$

2. **计算P(B)**：
   - 来自A的贡献
   - $P(A)/O(A) = 1/2 = 0.5$
   - $P(B) = (1 - 0.85)/5 + 0.85 * 0.5 = 0.03 + 0.425 = 0.455$

3. **计算P(C)**：
   - 来自A、B和D的贡献
   - $P(A)/O(A) = 1/2 = 0.5$
   - $P(B)/O(B) = 1/2 = 0.5$
   - $P(D)/O(D) = 1/2 = 0.5$
   - $P(C) = (1 - 0.85)/5 + 0.85 * (0.5 + 0.5 + 0.5) = 0.03 + 0.85 * 1.5 = 0.03 + 1.275 = 1.305$

4. **计算P(D)**：
   - 来自B的贡献
   - $P(B)/O(B) = 1/2 = 0.5$
   - $P(D) = (1 - 0.85)/5 + 0.85 * 0.5 = 0.03 + 0.425 = 0.455$

5. **计算P(E)**:

- 来自D的贡献
- $P(D)/O(D) = 1/2 = 0.5$
- $P(E) = (1 - 0.85)/5 + 0.85 * 0.5 = 0.03 + 0.425 = 0.455$

第二次迭代

用第一次迭代后的PageRank值继续计算，直到值收敛:

1. **计算P(A)**:

- $P(A) = (1 - 0.85)/5 + 0.85 * (P(C) + P(E))$
- $P(C) = 1.305, P(E) = 0.455$
- $P(A) = 0.03 + 0.85 * (1.305 + 0.455) = 0.03 + 0.85 * 1.76 = 0.03 + 1.496 = 1.526$

2. **计算P(B)**:

- $P(B) = (1 - 0.85)/5 + 0.85 * (P(A))/2$
- $P(A) = 1.526$
- $P(B) = 0.03 + 0.85 * 0.763 = 0.03 + 0.64855 = 0.67855$

3. **计算P(C)**:

- $P(C) = (1 - 0.85)/5 + 0.85 * (P(A)/2 + P(B)/2 + P(D)/2)$
- $P(A) = 1.526, P(B) = 0.67855, P(D) = 0.455$
- $P(C) = 0.03 + 0.85 * (1.526/2 + 0.67855/2 + 0.455/2) = 0.03 + 0.85 * (0.763 + 0.339275 + 0.2275) = 0.03 + 0.85 * 1.329775 = 0.03 + 1.13030875 = 1.16030875$

4. **计算P(D)**：
   - $P(D) = (1 - 0.85)/5 + 0.85 * (P(B)/2)$
   - $P(B) = 0.67855$
   - $P(D) = 0.03 + 0.85 * 0.339275 = 0.03 + 0.28838375 = 0.31838375$

5. **计算P(E)**：
   - $P(E) = (1 - 0.85)/5 + 0.85 * (P(D)/2)$
   - $P(D) = 0.31838375$
   - $P(E) = 0.03 + 0.85 * 0.159191875 = 0.03 + 0.13531309375 = 0.16531309375$

如此进行若干次迭代，直到PageRank值收敛，即各个页面的PageRank值变化很小。