Lecture 13

PageRank

Google assigns to each web page a number called the PageRank, calculated using eigenvectors; pages with higher rank come higher in search results.
We will describe a simplified version.

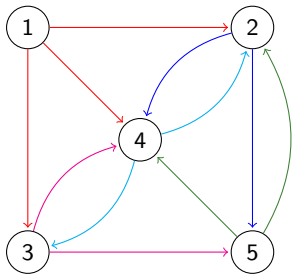
- Imagine pages $S_1, \ldots, S_n$, with some links between them.
- Say S_j links to N_j different pages, and assume $N_j > 0$.
- We want rankings $r_i \geq 0$, normalised so that $\sum_i r_i = 1$; so r is a probability vector in $\mathbb{R}^n$.
- A link from S_j to S_i is a vote by S_j that S_i is important.
- Links from important pages should count for more; links from pages with many links should count for less.
- We use this rule: a link from S_j to S_i contributes r_j/N_j to r_i.
- Thus, the following consistency condition should be satisfied:

$$r_i = \sum_{\text{pages } S_j \text{ that link to } S_i} \quad .$$

PageRank as an eigenvector

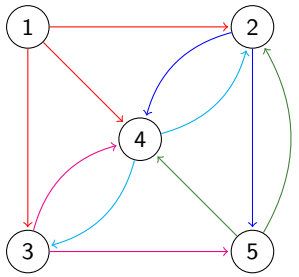
Pages $S_1, \ldots, S_n$; rankings $r_i \geq 0$ with $\sum_i r_i = 1$; S_j links to N_j pages;
Consistency condition $r_i = \sum_{\text{pages } S_j \text{ that link to } S_i} r_j/N_j$.

Define matrix P by $P_{ij} = \begin{cases} 1/N_j & \text{if there is a link from } S_j \text{ to } S_i \\ 0 & \text{otherwise.} \end{cases}$
Consistency condition is $r_i = \sum_j P_{ij} r_j$, so $r = Pr$, so r is an eigenvector for P with eigenvalue . Column j has N_j entries of $1/N_j$ so P is stochastic.



$$P = \begin{bmatrix} 0 & 0 & 0 & & \\ 1/3 & 0 & 0 & & \\ 1/3 & 0 & 0 & & \\ 1/3 & 1/2 & 1/2 & & \\ 0 & 1/2 & 1/2 & & \end{bmatrix}.$$

PageRank as a Markov chain



$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 1/2 & 1/2 \\ 1/3 & 0 & 0 & 1/2 & 0 \\ 1/3 & 1/2 & 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 & 0 & 0 \end{bmatrix}.$$

Imagine a surfer who clicks a randomly chosen link on the current page once per minute. This gives a Markov chain X with transition matrix P.
The PageRank vector r must satisfy $r_i \geq 0$ and $\sum_i r_i = 1$ and $Pr = r$, so it is a stationary distribution for X.
Take $q = \begin{bmatrix} 1/n & \cdots & 1/n \end{bmatrix}^T$ (distribution for a uniformly random page).
Typically there is a unique stationary distribution r, and $P^k q$ converges quickly to r as $k \to \infty$. When n is millions or billions, this is the best way to find r.
Conceptually: r_i is the long run average proportion of time that a random surfer spends on page i.

Calculating PageRank in Maple

```
with(LinearAlgebra):
n := 5;
P := << 0 | 0 | 0 | 0 | 0 >,
 <1/3 | 0 | 0 | 1/2 | 1/2 >,
 <1/3 | 0 | 0 | 1/2 | 0 >,
 <1/3 | 1/2 | 1/2 | 0 | 1/2 >,
 < 0 | 1/2 | 1/2 | 0 | 0 >>;
NS := NullSpace(P - IdentityMatrix(n));
r := NS[1];
r := r / add(r[i],i=1..n);
r := evalf(r);
```

Result: $r = \begin{bmatrix} 0.0 \\ 0.2777777778 \\ 0.1666666667 \\ 0.3333333333 \\ 0.2222222222 \end{bmatrix}$; so page 1 has rank 0.0
page 2 has rank 0.2777777778
page 3 has rank 0.1666666667
page 4 has rank
page 5 has rank

Calculating PageRank as a limit

```
with(LinearAlgebra):
n := 5;
P := << 0 | 0 | 0 | 0 | 0 >,
 <1/3 | 0 | 0 | 1/2 | 1/2 >,
 <1/3 | 0 | 0 | 1/2 | 0 >,
 <1/3 | 1/2 | 1/2 | 0 | 1/2 >,
 < 0 | 1/2 | 1/2 | 0 | 0 >>;
q := Vector(n,[1/n $ n]);
r := evalf(P^10 .  q);
```

Result: $r = \begin{bmatrix} 0.0 \\ 0.2783203125 \\ 0.1667317708 \\ 0.3332682292 \\ 0.2216796875 \end{bmatrix}$, close to the exact value of $\begin{bmatrix} 0.0 \\ 0.2777777778 \\ 0.1666666667 \\ 0.3333333333 \\ 0.2222222222 \end{bmatrix}$

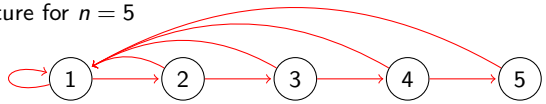
---

q is a vector of length n, whose entries are $1/n$, repeated n times.
We have seen that $r = \lim_{k\to\infty} P^k q$, so $r = P^{10}q$ should be approximately right.

PageRank example

- Suppose that every page (including page 1) links to page 1.
- Also page 1 links to page 2, page 2 links to page 3 and so on, until page $n-1$ links to page n.
- There are no other links.

Picture for $n = 5$



Page 2 only gets credit from page 1, but page 1's credit is shared equally between the link to page 2 and the link back to itself, so $r_2 =$.
Similarly, page 3 only gets credit from page 2, but page 2's credit is shared equally between the link to page 3 and the link back to page 1, so $r_3 =$ $=$. Similarly, $r_k = r_{k-1}/2 =$ for $2 \le k \le n$.
The total rank is

$$\sum_{k=1}^{n} r_1/2^{k-1} = r_1 \sum_{j=0}^{n-1} (\tfrac{1}{2})^j = r_1 \frac{1-(\frac{1}{2})^n}{1-\frac{1}{2}} = 2r_1(1-2^{-n}).$$

This must be equal to , so $r_1 =$.

Damping

Google found it useful to modify the PageRank algorithm with a *damping factor* d, where $0 < d < 1$. Consider a surfer who clicks a random link on the current page with probability d, but with probability $1-d$ chooses a uniformly random page (whether or not there is a link to it).
This gives a new transition matrix:

$$Q_{ij} = \begin{cases} \frac{d}{N_j} + \frac{1-d}{n} & \text{if there is a link from } S_j \text{ to } S_i \\ & \text{otherwise.} \end{cases}$$

Equivalently: let R be the stochastic matrix with $R_{ij} = 1/n$ for all i and j; then $Q = dP +$. Now the PageRank vector r should satisfy $(Q - I_n)r = 0$. We can approximate r by finding $Q^k q$ for large q.

```
d := 0.85;
R := Matrix(n,n,[1/n $ n^2]);
Q := d * P + (1-d) * R;
NS := NullSpace(Q - IdentityMatrix(n));
r := NS[1];
r := r / add(r[i],i=1..n);
```

or

```
r := Q^10 .  q;
```

Damped PageRank example

Consider a world in which every page (including Facebook) links to Facebook, and no page links anywhere else. Facebook is page 1, with rank a; the other $n-1$ pages all have rank b. The total rank $a+(n-1)b$ must be one, so $b=(1-a)/(n-1)$.

$$P=\begin{bmatrix} 1 & \\ \hline 0 & \end{bmatrix} \qquad R=\begin{bmatrix} 1/n & \\ \hline 1/n & \end{bmatrix}$$

The equation $Qr=r$ or $(dP+(1-d)R)r=r$ becomes

$$\begin{bmatrix} d+(1-d)/n & \\ \hline (1-d)/n & \end{bmatrix}\begin{bmatrix} a \\ \hline b \end{bmatrix}=\begin{bmatrix} a \\ \hline b \end{bmatrix}.$$

Looking at the top entry we get $(d+(1-d)/n)(a+(n-1)b)=a$, but $a+(n-1)b=1$ so $a=d+(1-d)/n$ and

$$b=\frac{1-a}{n-1}=\frac{1-d-(1-d)/n}{n-1}=\frac{(1-d)(1-1/n)}{n-1}=\frac{1-d}{n}.$$