26. Computing the Rank of a Webpage

Google PageRank
More Practice with 2D Array OPs
More Practice with numpy

Functions and 2D Arrays

Assume
from random import uniform as randu
from numpy import *

Let’s write a function `randuM(m,n)` that returns an m-by-n array of random numbers, each chosen from the uniform distribution on [0,1].

A Function that Returns an n-by-n Array of Random Numbers

```python
def randuM(m, n):
    A = zeros((m, n))
    for i in range(m):
        for j in range(n):
            A[i, j] = randu(0, 1)
    return A
```

Probability Arrays

A nxn probability array has the property that its entries are nonnegative and that the sum of the entries in each column is 1

```
.2 .6 .2
.7 .3 .3
.1 .1 .5
```

A Function that Returns a Random Probability Array

```python
def probM(n):
    A = randuM(n, n)
    for j in range(n):
        s = 0;
        for i in range(n):
            s += A[i, j]
        if s != 0:
            for i in range(n):
    return A
```

Probability Arrays

To generate a random probability array, generate a random matrix with nonnegative entries and then divide the numbers in each column by the sum of the numbers in that column

```
  5  6  1
  2  0  3
  4  3  1
```

```
  5/11  6/9  1/5
  2/11  0/9  3/5
  4/11  3/9  1/5
```
A network consists of nodes connected by directed edges. Each node has a transition probability associated with it, indicating the likelihood of a person moving from one node to another.

Think of a node as an island or a Web page. Transition probabilities represent the probability of moving from one node to another. For example, with a probability of 0.1, a person on island 1 will hop to island 2.

A Random Process

Suppose there are 1000 people on each node. At the sound of a whistle, they hop to another node in accordance with the "outbound" probabilities.

At Node 0

At Node 1

Suppose there are 1000 people on each node. At the sound of a whistle, they hop to another node in accordance with the "outbound" probabilities.
At Node 2

The Population Distribution

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 0</td>
<td>1000</td>
<td>1000</td>
</tr>
<tr>
<td>Node 1</td>
<td>1000</td>
<td>1300</td>
</tr>
<tr>
<td>Node 2</td>
<td>1000</td>
<td>700</td>
</tr>
</tbody>
</table>

Repeat

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 0</td>
<td>1000</td>
<td>1120</td>
</tr>
<tr>
<td>Node 1</td>
<td>1300</td>
<td>1300</td>
</tr>
<tr>
<td>Node 2</td>
<td>700</td>
<td>580</td>
</tr>
</tbody>
</table>

After 100 Iterations

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 0</td>
<td>1142.85</td>
<td>1142.85</td>
</tr>
<tr>
<td>Node 1</td>
<td>1357.14</td>
<td>1357.14</td>
</tr>
<tr>
<td>Node 2</td>
<td>500.00</td>
<td>500.00</td>
</tr>
</tbody>
</table>

Appears to reach a Steady State

In terms of popularity: Island 1 > Island 0 > Island 2

[1142.85, 1357.14, 500.0] is the "stationary array"
Computing the Stationary Array Involves a Probability Array

Transition Probability Array

$$P_{\text{old}} = \begin{bmatrix} .2 & .6 & .2 \\ .7 & .3 & .3 \\ .1 & .1 & .5 \end{bmatrix}$$

$$P[i,j]$$ is the probability of hopping from node $$j$$ to node $$i$$

Formula for Updating the Distribution Array

$$P = \begin{bmatrix} .2 & .6 & .2 \\ .7 & .3 & .3 \\ .1 & .1 & .5 \end{bmatrix}$$

$$w[0] = .2v[0] + .6v[1] + .2v[2]$$
$$w[1] = .7v[0] + .3v[1] + .3v[2]$$

V is the old distribution array, w is the updated distribution array

A Function that Computes the Update

```python
def Update(P,v):
    n = len(x)
    w = zeros((n,1))
    for i in range(n):
        for j in range(n):
            w[i] += P[i,j]*v[j]
    return w
```

V is the old distribution vector, w is the updated distribution vector
Background

Index all the pages on the Web from 0 to N-1. (N is around 50 billion.)

The PageRank algorithm orders these pages from “most important” to “least important”.

It does this by analyzing links, not content.

Key Ideas

The Transition Probability Array

A Very Special Random Walk

The Connectivity Array

A Random Walk on the Web

Repeat:

You are on a webpage.
There are m outlinks.
Choose one at random.
Click on the link.

The Connectivity Array

$G[i,j]$ is 1 if there is a link on page $j$ to page $i$

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

The Probability Array

\[
\begin{align*}
\text{a} &= \frac{1}{3} \\
\text{b} &= \frac{1}{2} \\
\text{c} &= \frac{1}{4}
\end{align*}
\]

$$
\begin{bmatrix}
0 & a & 0 & 0 & b & 0 & c & 0 \\
a & 0 & 0 & 0 & 0 & 0 & c & 1 \\
a & 0 & a & 0 & 0 & b & 0 & 0 \\
a & 0 & a & b & 0 & a & 0 & 0 \\
0 & 0 & 0 & b & 0 & 0 & c & 0 \\
0 & a & a & 0 & 0 & a & 0 & 0 \\
a & 0 & 0 & 0 & 0 & 0 & c & 0 \\
0 & 0 & a & 0 & 0 & a & 0 & 0
\end{bmatrix}
$$
### PageRank From the Stationary Array

<table>
<thead>
<tr>
<th>Stationary Array</th>
<th>PageRank</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5723</td>
<td>3</td>
</tr>
<tr>
<td>0.8206</td>
<td>1</td>
</tr>
<tr>
<td>0.7876</td>
<td>2</td>
</tr>
<tr>
<td>0.2609</td>
<td>5</td>
</tr>
<tr>
<td>0.2064</td>
<td>7</td>
</tr>
<tr>
<td>0.8911</td>
<td>0</td>
</tr>
<tr>
<td>0.2429</td>
<td>6</td>
</tr>
<tr>
<td>0.4100</td>
<td>4</td>
</tr>
</tbody>
</table>

Webpage 5 has a PageRank of 0.