Athens Exercise1

Gard Hillestad

November 2015

1 Task 1

```
# Page Rank Exercise

M = [[0,0,0,0,1,1.0/3],
     [1,0,0,0,0,1.0/3],
     [0,1,0,0,0,0],
     [0,0,0.5,0,0,1.0/3],
     [0,0,0,1,0,0],
     [0,0,0.5,0,0,0]]

for i in M:
    print(i)

rk = [1.0/6,1.0/6,1.0/6,1.0/6,1.0/6,1.0/6]

epsilon = 0.01;
norm = 1;

rkp1 = [0,0,0,0,0,0]
iterations = 0;

while norm>epsilon:
    iterations += 1
    for i in range(0,6):
        for j in range(0,6):
            rkp1[i] += M[i][j]*rk[j]
    norm = 0;
    for i in range(0,len(rk)):
        norm += abs(rk[i] - rkp1[i])
    rk = rkp1
    rkp1 = [0,0,0,0,0,0]
    print(norm)

print(rk)
print("Iterations", iterations)
```

Results of Page Rank Two example run-throughs had these results:

```
epsilon = 0.01;
rk =
[0.17946264701773101,
0.21645443561702987,
0.21490031054717265,
0.14134887577478025,
0.14177645366559974,
0.10605727737768632]
Iterations 22

epsilon = 0.001;
[0.17841442676364427,
0.2141072868645124,
0.21431789107614901,
0.1430063848479438,
0.1428844564753232,
0.10726955397242699]
Iterations 39
```

Both solutions are correct since they maintain their hierarchy, but only if you disregard the digits after the 1/100. In the first solution node 2 ¿ node 3, which is clearly incorrect as node 2 only refer to, but if we only examine the digits including 1/100, then we see that 2 = 3, which is obviously true if 2 only referrs to 3. The nodes which are referred to by nodes that are both important (have many references to it) and don't make many votes should dominate, hence a vote from 6 is less important than a vote from 5 even though they recieve the same amount of votes. In a large graph where you expect pages to have more similar / nuanced values, it will be important to tune down the epsilon.

Uniqueness As there is one page rank that will match the stationary distribution, which is unique, we can say that there is a unique page rank vector.

2 Task 2

If we want to keep the rule of inheritance, we should let the leaf nodes which have to be excluded from the algorithm inherit importance in the same fashion. Assume that before the reduction on $\mathbf{r}$ and $\mathbf{M}$ is reverted, the values of the missing rows in $\mathbf{r}$ is set to zero and the matrix $\mathbf{M}$ is given by the initial graph. My proposition is simply running over the algorithm once more to propagate the values in $\mathbf{r}$ to find the final $\mathbf{r}$

1. remove dead ends

$$r = \hat{r} \tag{1}$$

$$M = \hat{M} \tag{2}$$

example

$$M = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \rightarrow \hat{M} \begin{bmatrix} a & c \\ g & i \end{bmatrix} \tag{3}$$

$$r = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \rightarrow \hat{r} \begin{bmatrix} a \\ c \end{bmatrix} \tag{4}$$

2. compute page rank

$$\hat{r_{sol}} = \begin{bmatrix} p(a) \\ p(c) \end{bmatrix} \tag{5}$$

3. propegate values

$$r = \begin{bmatrix} p(a) \\ 0 \\ p(c) \end{bmatrix} \tag{6}$$

$$r_{sol} = Mr \tag{7}$$

knowing how matrix multiplication works, it is easy to see that this holds.