Harrison Lam

The Google PageRank Algorithm

In this project, the PageRank algorithm, a link analysis algorithm, is explored and discussed. PageRank measures the relative importance of websites in a network of websites, with directional links between them.

Notation:

The network can be represented by a matrix (A_{ij}), with $A_{ij} = k$ if there are k edges $d_j \to d_i$

Out-degree of a node i is the number of outgoing edges from node i

A node with out-degree 0 is a dangling node

PageRank can also be used to represent a voting system, where each page can distribute a total vote of 1 to other pages. The votes are then weighted according to the importance of the voters, which suggests the following recursion:

$w_i = \sum_{j=1}^{N} S_{ij} w_j$ where $S_{ij} = \frac{A_{ij}}{\sum_{q=1}^{N} A_{qj}}$ and $w_i > 0$

S can be considered as the transition matrix of a Markov chain that surfs the network by picking an outgoing edge at random. The recursion suggests that the score vector is the invariant measure. Starting at $w = (1, \dots, 1)$ and applying the transition matrix S many times to it might allow us to find the solution. However, there are cases where this fails to converge:

$$(A_{ij}) = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = (S_{ij})$$

Then $S\begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}0\\2\\1\end{pmatrix}, S^2\begin{pmatrix}1\\1\\1\end{pmatrix} = S\begin{pmatrix}0\\1\\2\end{pmatrix}, \dots$

So the recursion alternates between $\begin{pmatrix}0\\2\\1\end{pmatrix}$ and $\begin{pmatrix}0\\1\\2\end{pmatrix}$ and does not converge.

Random surfer model for user behaviour:

Choose a random page according to some pre-defined distribution π

At every step, if there are no outgoing links, choose a random page according to π

If there are outgoing links, with probability d, choose an outgoing link at random and with probability $1 - d$, choose a random page according to π

Consider the case with $A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}, d = 0.85, \pi$ uniform

The code below simulates 100 sample paths for 1000 steps each, shown in figure 1:

```
piprob = [1/4 1/4 1/4 1/4];
d = 0.85;
A = [0 1 0 0;1 0 0 0;1 0 0 1;0 0 0 0];
pi_cprob = cumsum([0, piprob]);
[~, ~, start] = histcounts(rand(1,100),pi_cprob);
%Each column of paths stores a sample path
paths = zeros(1000, 100);
paths(1,:) = start;
for i = 1:1000
    for j = 1:100
        current = paths(i,j);
        if sum(A(:,current)) == 0 %No outgoing links
            [~, ~, paths(i+1,j)] = histcounts(rand, pi_cprob);
        else
            if rand < d
                %Choose an outgoing link at random
                jumpprob = A(:,current)'/sum(A(:,current));
                cprob = cumsum([0, jumpprob]);
                [~, ~, paths(i+1,j)] = histcounts(rand, cprob);
            else
                [~, ~, paths(i+1,j)] = histcounts(rand, pi_cprob);
            end
        end
    end
end
```

Figure 1: generating 100 sample paths

The average time spent on each node until time t for each sample path is then computed and it is plotted for the first sample path, t = 1, …,1000 (shown in figure 2). For sample path j, the average time spent on the k^{th} node from the beginning until time t is denoted $\mu_{jt}^{(k)}$

```
mu = zeros(100,1000,4);
for j = 1:100
    pathj = paths(:,j);
    for k = 1:4
        for t = 1:1000
            mu(j,t,k) = sum(pathj(1:t)==k)/t;
        end
    end
end
```

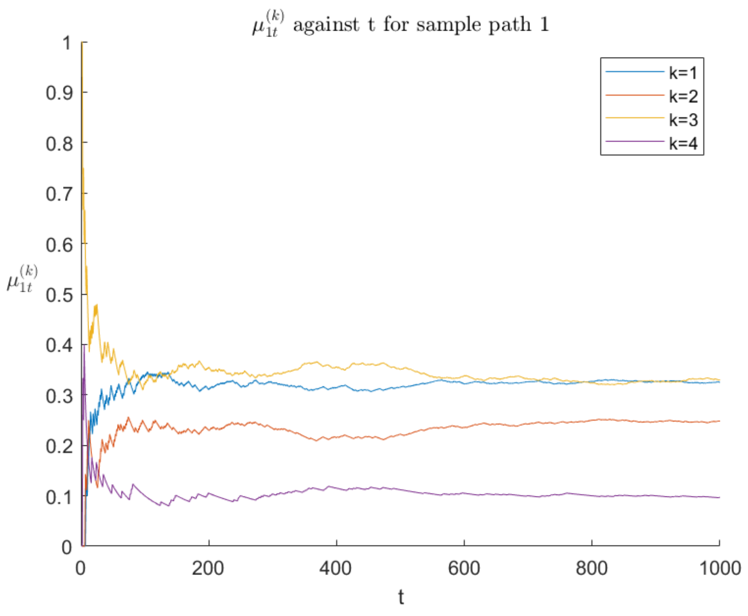


Figure 2: average time spent on each node until time t for sample path 1

Then, the variance is computed over the sample paths and plotted against t (figure 3):

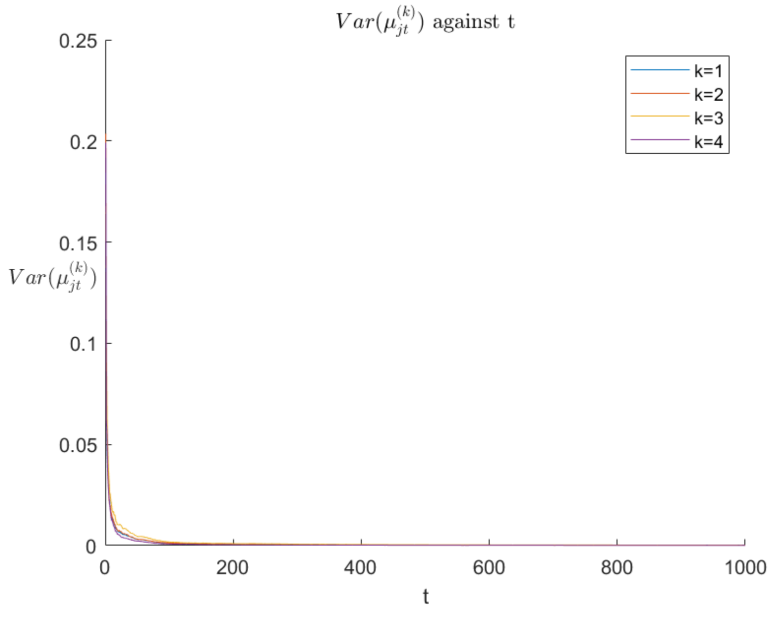


Figure 3: variance against time

To adjust to the random surfer model given above, we adjust the initial recurrence equation:

At node j:

If $\sum_q A_{qj} = 0$: $\mathbb{P}(j \text{ moves to } i) = \pi_i$

If $\sum_q A_{qj} \neq 0$: $\mathbb{P}(j \text{ moves to } i) = dS_{ij} + (1-d)\pi_i$ where $\mathrm{S_i}$ defined previously

$$\text{So } \omega_i = \sum_j \hat{S}_{ij}\omega_j \text{ where } \hat{S}_{ij} = \begin{cases} \pi_i & \text{if } \sum_q A_{qj} = 0 \\ \dfrac{dA_{ij}}{\sum_q A_{qj}} + (1-d)\pi_i & \text{if } \sum_q A_{qj} \neq 0 \end{cases}$$

$$\Rightarrow \hat{S}_{ij} = dS_{ij} + \left(1 - d1\left\{\sum_q A_{qj} \neq 0\right\}\right)\pi_i \text{ transition probabilities for new Markov chain}$$

$d \neq 0,1$ and $\pi_i > 0$, so the chain can go from any state to any state in 1 move
$\Rightarrow$ irreducible and aperiodic

- Markov chain finite and irreducible
  $\Rightarrow$ positive recurrent and has unique invariant distribution $\lambda_i = \dfrac{1}{\mathbb{E}_i T_i} > 0\ \forall i$
  $\Rightarrow \omega_i = \sum_j \hat{S}_{ij}\omega_j$ has unique solution p where p distribution and $p_i = \dfrac{1}{\mathbb{E}_i T_i}$
  By Ergodic theorem, $p_i = \dfrac{1}{\mathbb{E}_i T_i} =$ average time spent in i
- Markov chain irreducible, positive recurrent and aperiodic
  $\Rightarrow$ Convergence to equilibrium p from any initial distribution

This is then implemented in pagerank.m, shown below in figure 4:

```
function [scores] = pagerank(A,iterations,piprob)
    dim = size(A);
    nodes = dim(1);
    if nargin == 2
        piprob = ones(1,nodes)*(1/nodes);
    end
    d = 0.85;
    S = zeros(nodes);
    %Computing S matrix
    for col = 1:nodes
        if sum(A(:,col)) == 0
            S(:,col) = piprob';
        else
            S(:,col) = d*A(:,col)/sum(A(:,col))+(1-d)*piprob';
        end
    end
    %Run the recursion
    scores = ones(1,nodes)'*(1/nodes);
    for i = 1:iterations
        scores = S*scores;
    end
    scores = nodes*scores;
end
```

Figure 4: PageRank function

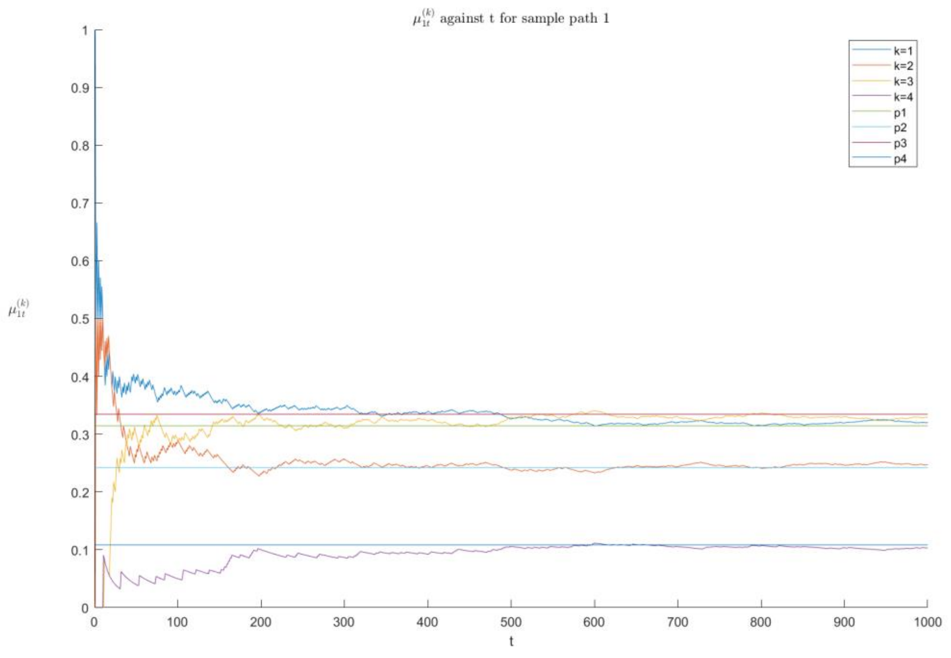
This gives the PageRank score, after 10000 iterations (using more iterations gives the same answer):

```
>> main

scores =

    1.2581
    0.9691
    1.3383
    0.4344
```

Plotting the value of $p = \frac{\omega}{N}$ obtained on top of figure 2 shows that the result is consistent with question 2. μ_{1t}^{k} should converge to the average time in node k ($= p_k$ from last part) as $t \to \infty$ by the law of large numbers. From the plot below, μ_{1t}^{k} seems to approach p_k for each k as t increases.



To test pagerank.m, we first write a function that generates a random adjacency matrix of size N, with the out-degree of each node being an independent Poisson random variable with mean k and all graphs occur with equal probability conditional on the out-degrees.

A graph is uniquely characterised by the out-arrows, so given the out-degrees, all graphs being equiprobable is equivalent to a uniform distribution over every combination of out-arrows from node i that sum up to the out-degree of i. Instead of finding all the possible (N-1)-combinations that sum up to the out-degree, which is very computationally inefficient, we use the fact that

$X_{N-1}(k) = X_{N-2}(0) + X_{N-2}(1) + \cdots + X_{N-2}(k)$,
$X_N(k) =$ # of combinations of N numbers that sum up to k

by considering the possible values of the first element.

$X_{N-2}(i)$ correspond to the combinations in $X_{N-1}(k)$ with first element being $k - i$

So, letting $p_i = \frac{X_{N-2}(k-i)}{X_{N-1}(k)}$ be the probability mass function of the first digit (and then repeating it for all the digits except for the last one) is equivalent to random generation where all graphs are equiprobable. This is done using the code below:

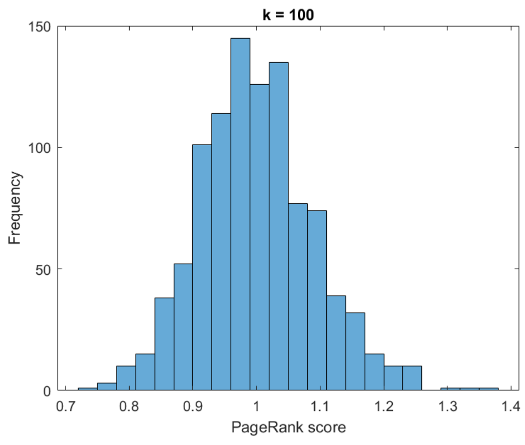
```
function [A] = randomadjacency(N,k)
    A = zeros(N,N);
    %Generating Poisson
    outdeg = poissrnd(k,1,N);
    maxdeg = max(outdeg);
    numcombinations = zeros(maxdeg+1,N-1);
    numcombinations(:,1) = 1;
    for i = 2:N-1
        for j = 1:maxdeg+1
            numcombinations(j,i) = sum(numcombinations(1:j,i-1));
        end
    end
    for i = 1:N
        currenttotal = outdeg(i);
        out = zeros(1,N-1);
        for j = 1:N-2
            prob = numcombinations(1:currenttotal+1,N-1-
j)'/numcombinations(currenttotal+1,N-j);
            prob = flip(prob);
            cprob = cumsum([0 prob]);
            [~, ~, out(j)] = histcounts(rand, cprob);
            out(j) = out(j)-1;
            currenttotal = currenttotal-out(j);
        end
        out(N-1) = outdeg(i)-sum(out(1:N-2));
        A(:,i) = [out(1:i-1)';0;out(i:N-1)'];
    end
end
```

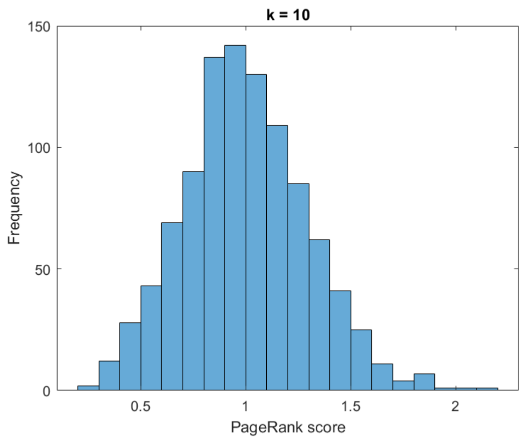
The eigenvalues and eigenvectors are found for the modified transition matrix using the code below.

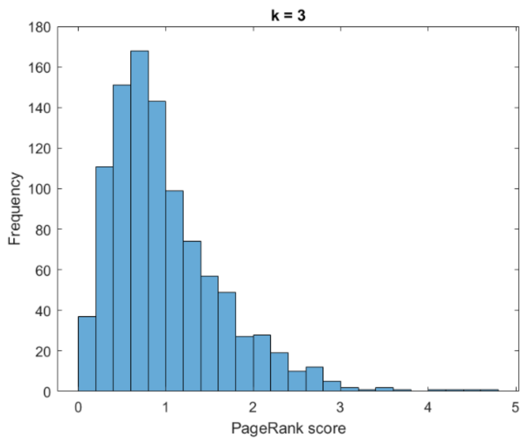
```
A =  randomadjacency(1000,0.5);
pr = pagerank(A,10000);
pi = ones(1,1000)*1/1000;
newA = zeros(1000);
for i = 1:1000
    if sum(A(:,i)) > 0
        newA(:,i) = A(:,i)*0.85/sum(A(:,i)) + 0.15*pi';
    else
        newA(:,i) = pi';
    end
end
[V,D] = eig(newA);
ev = V(:,1)*1000/sum(V(:,1));
```

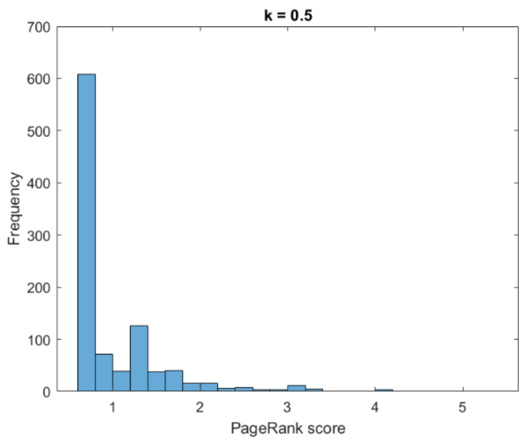
It has an eigenvalue 1, with the eigenvector corresponding to the stationary distribution of the system and should give the PageRank score when multiplied by N=1000 (from question 3). Using the pagerank.m function in question 4 with 10000 iterations shows that the PageRank scores are nearly identical, with the difference being $\approx 10^{-14}$. This suggests that pagerank.m works as expected.

As k decreases, the distribution of scores seem to be more positively skewed, with the higher PageRank scores being more extreme, as seen below using histogram plots:









This is because when the mean number of out-arrows decreases, there will be more nodes without any nodes pointing to it. Those webpages are 'irrelevant' and have lower than average PageRank scores. For very small values of k, there is only a small group of webpages with links pointed towards them, so these pages are a lot more 'important' than the other webpages in comparison and have much higher scores.

However, this model is unrealistic, as the assumption that all graphs are equiprobable is not true in real-life networks. In general, webpages are added into the system slowly and not all at once and a newly created website is a lot more likely to point towards a well-known webpage than an unpopular webpage. This results in the votes bunching up, with a very small group of webpages getting a very large proportion of the votes.

Now, we explore a citation dataset from the Arxiv high energy physics theory section. citations.dat represent the directional links. A denotes the adjacency matrix and z denotes the number of articles per journal. The objective is to compare and analyse the relationship between two scoring approach.

Approach 1:

Eigenfactor (EF) score, which corresponds to PageRank score above and article influence (AI) score, which is calculated by EF score divided by z_i

Approach 2:

Total citations (TC) score, which represents the in-degree of a node and impact factor (IF) score, which is calculated by TC score divided by z_i

Using the given dataset, A and z are extracted using the following code:

```
z = zeros(1,272);
for i = 1:272
    z(i) = sum(ids(:,2)==i);
end
A = zeros(272);
for i = 1:size(links,1)
    id1 = ids(ids(:,1)==links(i,1),2);
    id2 = ids(ids(:,1)==links(i,2),2);
    A(id2,id1) = A(id2,id1)+1;
end
```

The top 19 journals are identical for the two scoring systems, with the difference in rank being less than 1. However, the difference is a lot more significant for the lower ranked journals, with a mean difference of 12 and a max difference of 94. This suggests that the EF and TC scores contain different information. The EF score of a journal takes into account the scores of the other journals that cited their articles, whereas the TC score weights them equally and only counts the number of citations. Using pagerank.m, the EF scores are computed and the TC scores are calculated using the matrix A, which then gives a correlation coefficient of 0.9987. Although they are very highly correlated, the reason could be that they are linked by a common factor, e.g. journal size. For example, a journal with a high TC score is likely going to be a large and well-known journal, which is then also likely to have a large EF score (or vice versa), but that does not necessarily mean that the two scoring systems capture the same information. To avoid this, we compute the AI/IF scores by dividing the EF/TC scores by the number of articles. This gives a correlation coefficient of 0.9955 between the AI and IF scores, which captures how 'similar' these scores are when we remove the journal size factor. Although this is still very high, it shows that adding a common factor to the comparison (journal size in this case) can make the variables seem more correlated, and there could be other factors also contributing to this effect. The mean difference in rank is now 8.75, with a max difference of 84. Therefore, the EF and TC scores offer different information and cannot be used interchangeably given that there are many journals whose ranks differ drastically under the two scoring systems.

Reference:

https://www.maths.cam.ac.uk/undergrad/catam/II/9pt5.pdf