PSBC Assignment 1: Page Rank

March 16, 2015

Louis Runcieman

Department of Mathematics

University of Manchester

Academic year 2014-2015

Abstract

PageRank is an algorithm used by Google to determine the importance of a webpage. Along with other algorithms, webpages can be searched for based on their relevance and importance to the search query; the importance is based around the number of hyperlinks pointing to a webpage [1, 2]. An innate property of the algorithm considers the importance of a webpage - hyperlinks from webpages with higher PageRank values contribute a greater weight and importance. This paper explores the basic ideas surrounding the theory with all programming done through GNU Octave (please note, this may induce compatibility issues with MATLAB).

Question 1

(a) A portion of the Web can be represented as a directed graph with nodes as webpages and edges as hyperlinks between webpages. This gives rise to a representation of the Web by its connectivity matrix (known as the hyperlink matrix) $H = (h_{ij})$, defined by,

$$h_{ij} = \begin{cases} 1, & \text{if a hyperlink exists to page i from page j,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

The function `form_s` constructs the stochastic matrix S associated with a given hyperlink matrix H, outputting in sparse form should the input be sparse. The stochastic matrix S is a guaranteed solution to the PageRank eigenvalue problem, with the PageRank vector defining the importance of a specific webpage based on the number of hyperlinks pointing to it. An innate property of `form_s` involving the *in-degree* and *out-degree* of an individual page ensures webpages are not given an overwhelming influence by simply hyperlinking to a lot of webpages. The in-degree and out-degree are the row and column sums of H respectively.

Listing 1: The .m file for `form_s`.

```
function S= form_s(H)
%form_s(H) constructs a stochastic matrix S associated with a hyperlink
%matrix H. Should H be given in sparse form, S will return a spare matrix.

binary=[0 1];
assert(ismember(H,binary),'Entries of the hyperlink matrix must be 0 or 1.');
narginchk(1,1); %Number of input arguments must be 1.
assert(issquare(H),'H must be a square matrix.'); %Ensures square hyperlink
                                                  %matrix input.
N=size(H); n=N(1,1);
e=ones(n,1); %Outputs column vector of ones, length n.
c=sum(H); %Sums columns of H.
for j=1:n;
      if  c(j)==0;
          a(j)=1;
          d(j)=0;
      else
          a(j)=0; %In the case of a dangling node at page j.
          d(j)=1/c(j);
      end %Outputs vector a as row, NOT column.
end
D=diag(d);
S=H*D+(e*a)/n;
if issparse(H)==1
```

```
    S=sparse(S);
end %Returns S in sparse form if H is in sparse form.
end
```

Listing 2: Test of `form_s` using the directed graph from Figure 1 of [3].

```
>> H=[0 0 0 0 0 0 1; 1 0 0 0 0 0 0; 1 1 0 0 0 0 0; 0 0 1 0 0 0 0;
      1 0 0 0 0 0 0; 1 0 0 0 1 0 0; 0 0 0 0 0 0 0];
>> form_s(H)
ans =

   0.00000   0.00000   0.00000   0.14286   0.00000   0.14286   1.00000
   0.25000   0.00000   0.00000   0.14286   0.00000   0.14286   0.00000
   0.25000   1.00000   0.00000   0.14286   0.00000   0.14286   0.00000
   0.00000   0.00000   1.00000   0.14286   0.00000   0.14286   0.00000
   0.25000   0.00000   0.00000   0.14286   0.00000   0.14286   0.00000
   0.25000   0.00000   0.00000   0.14286   1.00000   0.14286   0.00000
   0.00000   0.00000   0.00000   0.14286   0.00000   0.14286   0.00000
```

(b) Although the stochastic matrix S given from the hyperlink matrix H by `form_s` does indeed solve the PageRank eigenvalue problem, the solution is not unique. The Google matrix is another adjustment which gives rise to a unique solution to the PageRank eigenvalue problem by ensuring irreducibility and is given by the stochastic matrix,

$$G(\alpha) = \alpha S + (1-\alpha)ve^T, \quad 0 < \alpha < 1, \tag{2}$$

where $e, v \in \mathbb{R}^n$ such that $e = [1, \ldots, 1]^T$ and v, known as the *personalisation vector*, has nonnegative entries such that $e^T v = 1$. The Google matrix then defines a Markov chain interpreted by:

- a surfer having probability α to choose a hyperlink at random;
- and probability $1 - \alpha$ to choose a random webpage from any webpage on the Web.

Typically $\alpha = 0.85$ and $v = e/n$. The following figure is the plot of eigenvalues of $G(\alpha)$ and S for the hyperlink matrix associated with the directed graph from Figure 1 of [3]; α has values in the range 0.3 to 0.9, stepsize 0.1 and $v = e/7$.

Listing 3: Octave code used to produce Figure 1.

```
H=[0 0 0 0 0 0 1; 1 0 0 0 0 0 0; 1 1 0 0 0 0 0; 0 0 1 0 0 0 0;
   1 0 0 0 0 0 0; 1 0 0 0 1 0 0; 0 0 0 0 0 0 0];
S=form_s(H); e=ones(7,1); v=1/7*e;

hold all
for k=0.3:0.1:0.9
for i=1:7 %Ensures legend properly labels distinct sets of eigenvalues.
```

```
Figure_1(i)=plot(eigs(k*S+(1-k)*v*e'),'.','markersize',13);
end
plot(eigs(S),'*','color','k','markersize',10);
end
hold off
axis([-0.5,1.2,-0.5,0.5,]);
set(gca,'linewidth',2,'FontSize',11);
xlabel('Real axis');
ylabel('Imaginary axis');
legend('Eigenvalues of G(0.3)','Eigenvalues of G(0.4)',...
'Eigenvalues of G(0.5)','Eigenvalues of G(0.6)','Eigenvalues of G(0.7)',...
'Eigenvalues of G(0.8)','Eigenvalues of G(0.9)','Eigenvalues of S');
box on;
set(findall(gcf,'type','text'),'FontSize',11,'FontName','CMU Serif')
```

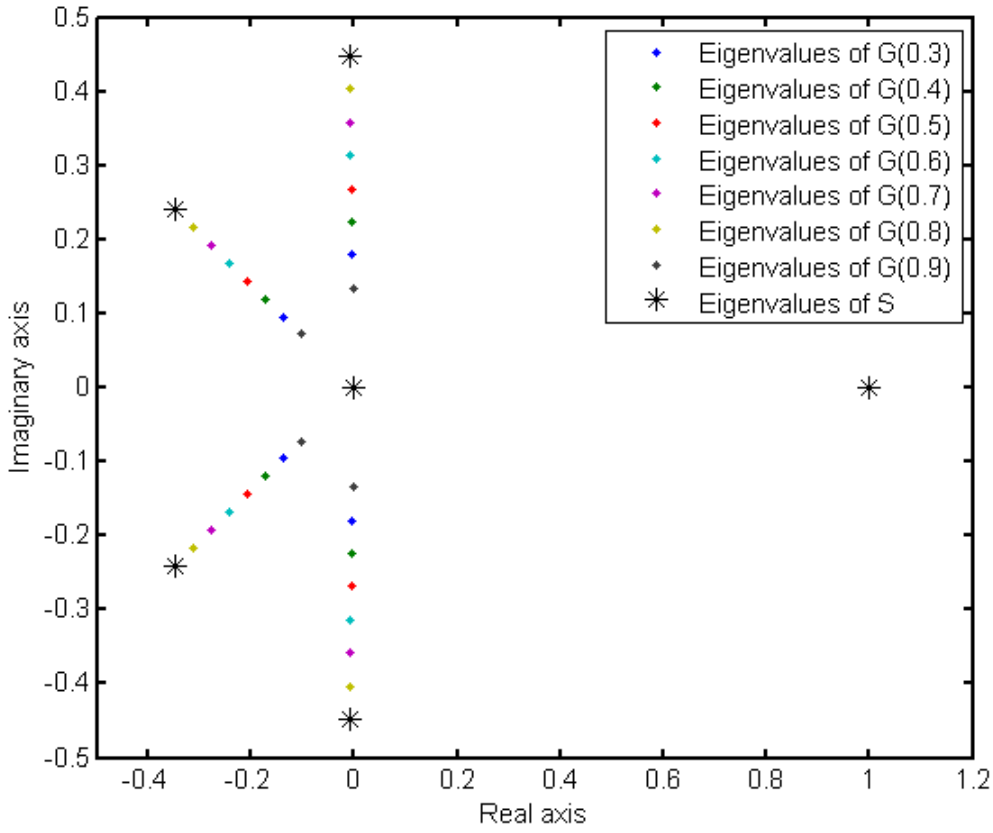

Figure 1: Eigenvalues of $G(\alpha)$ and S.

(c) The observed eigenvalues are consistent with the theory from [4]. All matrices S, $G(\alpha)$ have an eigenvalue equal to one, which is proven for column stochastic matrices in *Theorem 1* [4], and all other eigenvalues are less than or equal to one in magnitude, as observed in Figure 1 (in particular, these

eigenvalues have modulus less than 0.5). Furthermore, for each increase in value of α the modulus of non-unit eigenvalues is constantly scaled by a value of α. This coincides with *Theorem 2* [4] which states that if S has eigenvalues $\{1, l_2, \ldots, l_n\}$ then $G(\alpha)$ has eigenvalues $\{1, \alpha l_2, \ldots, \alpha l_n\}$. The cycle repeats, as represented in Figure 1, with values of α between 0.8 and 0.9. This is consistent with the usual choice of $\alpha = 0.85$, implying this values gives the quickest convergence of the power method.

Question 2

(a) Assume $e^T p^{(0)} = 1$, (ie. $p^{(0)}$ is column stochastic) and consider for $k \in \mathbb{N}$ the iteration,

$$p^{(k)} = G(\alpha)p^{(k-1)}. \tag{3}$$

This approach is known as the *power method*. We can redefine iteration (3) by,

$$p^{(k)} = G(\alpha)p^{(k-1)} = G(\alpha)^2 p^{(k-2)} = \cdots = G(\alpha)^k p^{(0)}. \tag{4}$$

since $G(\alpha)$ is column stochastic, $e^T G(\alpha) = e^T$. Combining this with (4) leads to,

$$e^T p^{(k)} = e^T G(\alpha)^k p^{(0)} = e^T p^{(0)} = 1. \tag{5}$$

with the last line evaluated by the assumption $e^T p^{(0)} = 1$. Let the eigenvalues $l_1, l_2, \ldots l_n$ of $G(\alpha)$ be such that,

$$1 = |l_1| > |l_2| \geqslant \cdots \geqslant |l_n|, \tag{6}$$

with linearly independent corresponding eigenvectors $p, v_2, \ldots v_n$. Since $\mathbb{R}^n$ is dimension n and there are n linearly independent eigenvalues, $p^{(0)}$ can be represented as a linear combination of them; namely, $p^{(0)} = \alpha_1 p + \sum_{i=2}^{n} \alpha_i v_i$, for constants $\alpha_i \in \mathbb{R}$. Assume $\alpha_1 \neq 0$. Then,

$$\begin{aligned} p^{(k)} &= G(\alpha)^k p^{(0)} \\ &= G(\alpha)^k (\alpha_1 p + \sum_{i=2}^{n} \alpha_i v_i) \\ &= \alpha_1 G(\alpha)^k p + \sum_{i=2}^{n} \alpha_i G(\alpha)^k v_i \\ &= \alpha_1 p + \sum_{i=2}^{n} \alpha_i l_i^k v_i, \end{aligned} \tag{7}$$

with the last line evaluated using the relationship between eigenvalues and corresponding eigenvectors, $G(\alpha)v_i = l_i v_i$ implying $G(\alpha)^k v_i = l_i^k v_i$. Specifically for $i = 1$, $G(\alpha)^k p = 1^k p = p$. Since $|l_i| < 1$ for all $i \geqslant 2$, then $l_i^k \to 0$ as $k \to \infty$, directly implying,

$$\lim_{k \to \infty} p^{(k)} = \alpha_1 p. \tag{8}$$

We can deduce $\alpha_1 = 1$ since $e^T p^{(k)} = 1 = e^T p$, concluding that the limiting probability that an infinitely dedicated random surfer visits a specific webpage is given by its PageRank.

Replacing $G(\alpha)$ by definition (2) in iteration (3) finally gives rise to,

$$\begin{aligned} p^{(k)} &= (\alpha S + (1-\alpha)ve^T)p^{(k-1)} \\ &= \alpha S p^{(k-1)} + (1-\alpha)ve^T p^{(k-1)} \\ &= \alpha S p^{(k-1)} + (1-\alpha)v, \end{aligned} \tag{9}$$

with the final line holding from (5) for $k-1$ case. Should n be large, H sparse, (9) is advantageous over (3) since the iteration will converge faster with less memory needed due to the sparse form of H and hence S.

(b) The function `randweb` produces a pseudo-random hyperlink matrix H representing n webpages with a total of k hyperlinks. 55A characteristic of the function prevents any non-zero entries on the diagonal; this translates to no pages having self-pointing links. Although the PageRank vector is not affected by multiples of the identity matrix added to H (only eigenvalues will change), it has been implemented to ensure less memory is needed to store all entries, and more importantly that k represents the true number of hyperlinks significant to the system. Should a sparse output be required the inbuilt function `sparse` can be implemented as follows, `sparse(randweb(n,k))`.

Listing 4: The .m file for `randweb`.

```
function H=randweb(n,k)
%randweb(n,k) create a psuedo-random hyperlink matrix of size n and with k
%hyperlinks.

narginchk(2,2) %Number of input arguments must be 2.
assert(k<=n*(n-1),...
'Hyperlink matrices assumed to have zero diagonal, k cannot exceed n*(n-1).');

I=eye(n); %Outputs identity matrix size n.
ind=find(~I); %Outputs indices of all 0 entries (ensuring diagonal is zero).
n_0=n*(n-1); %The fixed number of non zero elements in I.
hype=randperm(n_0,k); %Chooses randomly which k entries will be 1.
H=zeros(n); %Outputs zero matrix size n.
H(ind(hype))=1; %Assigns the k randomly chosen entries to value 1.
end
```

(c) The function `pm_pagerank` outputs the PageRank vector by the power method defined in (3).

Listing 5: The .m file for `pm_pagerank`.

```
function [p,cnt] = pm_pagerank(H,v,alpha)
%PM_PAGERANK PageRank by power method. p = pm_pagerank(H,v,alpha) computes the
%PageRank vector p associated with the connectivity matrix H, scaling parameter
```

```
%ALPHA and personalization vector V. To also output the number of iterations
%command [p,cnt] = pm_pagerank(H,v,alpha).

n = length(H); %Number of pages.
S = form_s(H); %Associated stochastic matrix with H.
u = (1-alpha)*v;
p = ones(n,1)/n;
p_prev = zeros(n,1);
cnt = 0;
while max(abs(p-p_prev)) > 1e-5
p_prev=p;
p=alpha*S*p_prev+u; %see equation (2) on handout
cnt=cnt+1;
end
```

(d) The function `randweb`, with $n = 100$, $k = 1000$ outputs a hyperlink matrix H representing 100 webpages with an average of 10 outlinks on each webpage. Applying `pm_pagerank` outputs the corresponding PageRank vector, p.

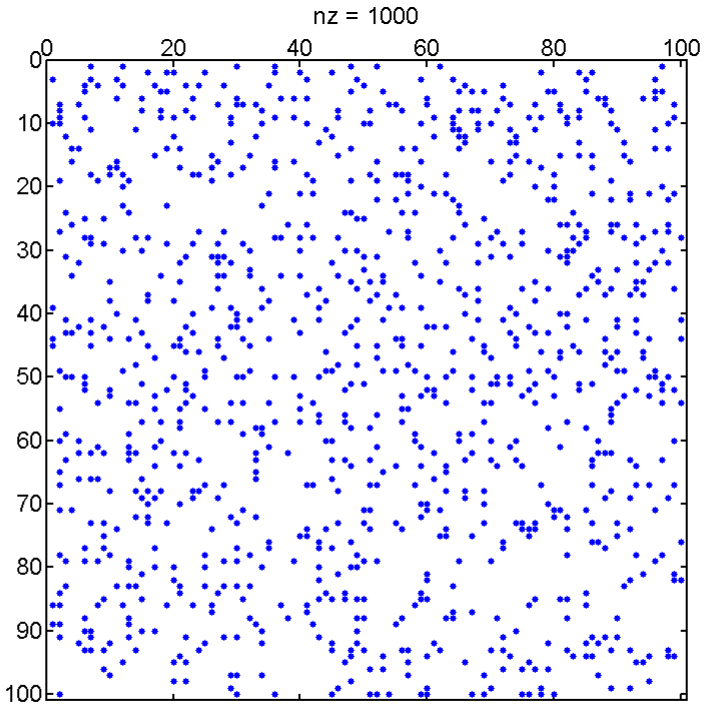

Figure 2: Representation of the connectivity graph of H using `spy`.

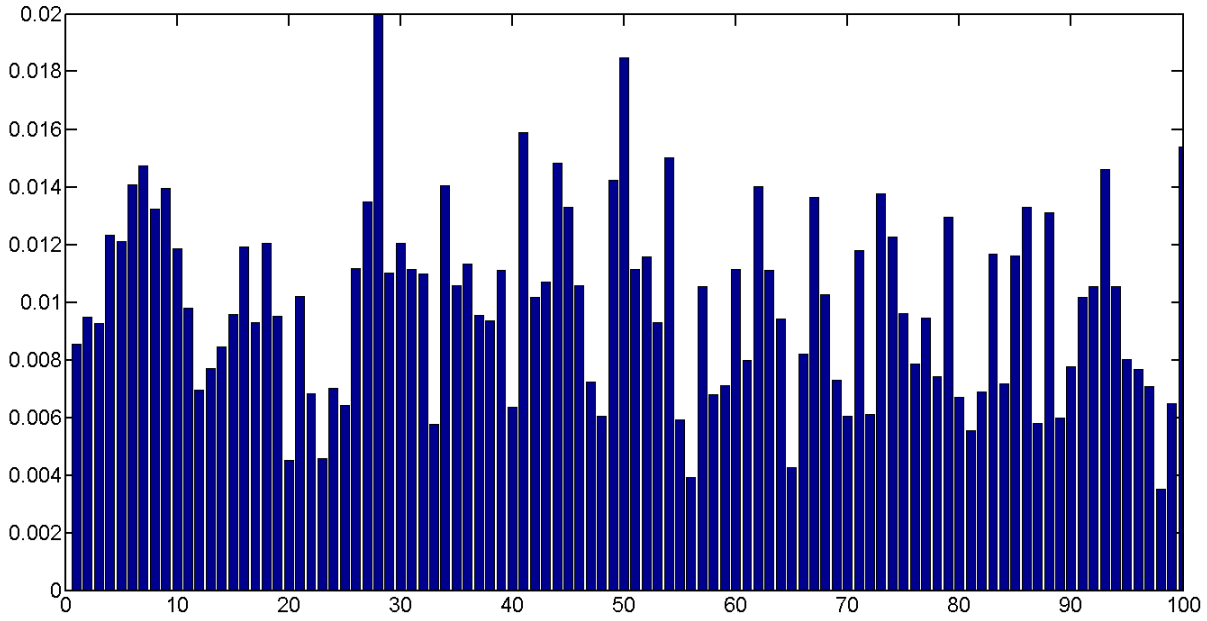

Figure 3: PageRank vector p represented by a bar graph. Number of iterations required: 7.

Listing 6: Octave code used to produce Figures 2 and 3.

```
H=randweb(100,1000); v=ones(100,1)/100; alpha=0.85;
[p,cnt]=pm_pagerank(H,v,alpha);
figure(1); spy(H)
set(findall(gcf,'type','text'),'FontSize',18,'FontName','CMU Serif')
set(gca,'XAxisLocation','top','linewidth',2,'FontSize',18);
figure(2); bar(p)
set(gca,'linewidth',2,'FontSize',18);
axis([0,100,0,0.02]);
cnt
```

Question 3

(a) Consider a weighted directed graph, with sum of weights equalling one representing a portion of the Web. The edge from node i to node j having weight k describes a surfer having probability k to follow the hyperlink from webpage i to webpage j. The process is stochastic in that the evolution of the system is randomly determined; the process can be statistically analysed but not predicted precisely. We then consider the process of a random walk on a weighted graph.

(b) The PageRank vector represents a probability distribution describing the likelihood of a surfer arriving at a particular page through random hyperlink choices. This process is directly compatible

with the above description, with the random walkers represented by web surfers randomly choosing hyperlinks and entries of the iterated PageRank vector representing the weight along edges. It is not necessary to follow and hence record the path taken by any web surfer since the system choose paths at random. The probability of transitioning to another webpage depends only on where the walker currently is; this is since Markov chains obey the Markovian property. It is however vital the number of surfers at each vertex, or webpages, is recorded since this gives rise to the PageRank vector through multiple walks (or iterations). As the system converges through multiple iterations the more influential and important webpages become apparent.

(c) The function `surfer` simulates web surfers undergoing the random walk defined above; the number of surfers and steps are input arguments and initially there are an equal number of surfers at all vertices.

Listing 7: The .m file for `surfer`.

```
function W=surfer(H,s,w)%Hyperlink matrix H, s surfers at each webpage, w walks.

N=size(H,1);
x=s*ones(N,1); %Number of surfers at each webpage (equal number).
H=diag(1./sum(H,2))*H;

for j=1:w
y=zeros(N,1); %Where surfers terminate walk.
  for ivertx=1:N
  r=rand(x(ivertx),1);
  ws=cumsum(H(ivertx,:)); %Cumulative sum of rows.
  u=histc(r,[0,ws]);
   y=y+u(1:end-1);
  end
W=y;
end
```

(d) Using a psuedo-random hyperlink matrix produced by `randweb` with 10 webpages and an average of 10 outgoing hyperlinks per webpage, the function `surfer` does not converge. This is expected since the number of outgoing hyperlinks on every webpage is near to the mean of 10, meaning no single page is more important than another.

References

[1] Brin, S. and Page, L. (1998). *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* In: *Seventh International World-Wide Web Conference (WWW 1998), 14/18 April 1998, Brisbane, Australia.*

[2] Page, L. (1998). *Method for node ranking in a linked database.* US Patent Number 6285999.

[3] Lionheart, W.R.B. (2015). *PSBC Assignment 1: Page Rank.* Accessed: 24 February 2015, <https://oldwww.ma.man.ac.uk//~bl/teaching/PSBC/private/ass1_2015.pdf>.

[4] Lionheart, W.R.B. and Tisseur, F. (2013). *An Introduction to PageRank.* Accessed: 24 February 2015, <https://oldwww.ma.man.ac.uk//~bl/teaching/PSBC/private/pagerank.pdf>.