605_HW10.Rmd

Kumudini Bhave

April 15, 2017

PAGE RANK

Problem Set 1 :

From a 6 page universe, we compute the page rank for each of the web pages. The nodes represent the websites, and the links outgoing from each of these nodes are as follows

Webpage Node	Outgoing Link To
Node 1	Node 2
Node 1	Node 3
Node 2	NA
Node 3	Node 1
Node 3	Node 2
Node 3	Node 5
Node 4	Node 5
Node 4	Node 6
Node 5	Node 4
Node 5	Node 6
Node 6	Node 4

Form the A matrix. Then, introduce decay and form the B matrix as we did in the course notes

Solution :

Based on the above table of website node outgoing links, we derive the matrix that represent the nodes with their links

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

We observe that Node 2 deos not have any outgoing links. hence it is the dangling node. Hence there is an equal probability that a use on Node 2 may jump to an other node. This probability is equally likely i.e. 1/6 Hence to make it row stochastic, we alter the row 2 by replacing it with the equal probabilities (These sum to 1)

$$A = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

The decayed matrix B can be defined as

$$B = 0.85 \times A + (0.15 / n)$$

where n is the number of nodes

```
knitr::opts_chunk$set(message = FALSE, echo = TRUE)

library(DT)
```

```
## Warning: package 'DT' was built under R version 3.3.3
```

```
matrixA <- matrix(c(0, 1/2, 1/2, 0, 0, 0, 1/6, 1/6, 1/6, 1/6, 1/6, 1/6, 1/3, 1/3, 0, 0, 1/3, 0, 0, 0, 0
    0, 0), nrow = 6, byrow = T)

# The decay being defined as 0/.85

decay <- 0.85

matrixB <- decay * matrixA + (0.15/6)

matrixB
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
## [1,] 0.0250000 0.4500000 0.4500000 0.0250000 0.0250000 0.0250000
## [2,] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
## [3,] 0.3083333 0.3083333 0.0250000 0.0250000 0.3083333 0.0250000
## [4,] 0.0250000 0.0250000 0.0250000 0.0250000 0.4500000 0.4500000
## [5,] 0.0250000 0.0250000 0.0250000 0.4500000 0.0250000 0.4500000
## [6,] 0.0250000 0.0250000 0.0250000 0.8750000 0.0250000 0.0250000
```

---

2 Start with a uniform rank vector r and perform power iterations on B till convergence. That is, compute the solution r = Bn * r. Attempt this for a sufficiently large n so that r actually converges.

Solution :

An 1 by n vector r which represents the PageRank of all the n webpage nodes.

$$r_i = \begin{bmatrix} 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \end{bmatrix}$$

We iteratively perform A * r such that , the page rank converges for all nodes, at which point

$$r = B \times r$$

```
knitr::opts_chunk$set(message = FALSE, echo = TRUE)

# function converge() @param1 probability matrix @param2 initial page rank vector

converge <- function(mat, rvec) {
    converged = FALSE
    iter <- 0

    mat <- matrixB
    rvec <- rinit

    # loop through till the pagerank vector stabilizes
    while (converged == FALSE) {
        iter <- iter + 1
        newrvec <- crossprod(mat, rvec)

        if (identical(newrvec, rvec)) {
            converged <- TRUE
        } else {
            rvec <- newrvec
            converged <- FALSE
        }

    }

    # return the pagerank vector and the no of iterations it took to converge
    return(list(newrvec, iter))
}


# Start with considering equal page rank for each of the nodes
rinit <- matrix(c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6), nrow = 6, byrow = T)

# Testing function converge()
r <- converge(matrixB, rinit)

# Converged Vector
r[[1]]
```

```
##            [,1]
## [1,] 0.05170475
## [2,] 0.07367926
## [3,] 0.05741241
## [4,] 0.34870369
## [5,] 0.19990381
## [6,] 0.26859608
```

```
# No of iterations to converge
r[[2]]
```

```
## [1] 67
```

---

3. Compute the eigen-decomposition of B and verify that you indeed get an eigenvalue of 1 as the largest eigenvalue and that its corresponding eigenvector is the same vector that you obtained in the previous power iteration method. Further, this eigenvector has all positive entries and it sums to 1.

Solution :

```
knitr::opts_chunk$set(message = FALSE, echo = TRUE)

# Finding Eigen Values for decayed matrix B
eigen((matrixB))$values
```

```
## [1]  1.00000000+0i  0.57619235+0i -0.42500000+0i -0.42500000-0i
## [5] -0.34991524+0i -0.08461044+0i
```

```
# Verifying if max value is 1
which.max(eigen((matrixB))$values)
```

```
## Warning in which.max(eigen((matrixB))$values): imaginary parts discarded in
## coercion
```

```
## [1] 1
```

```
# Finding Eigen Vectors for decayed matrix B
eigen(t(matrixB))$vectors
```

```
##              [,1]          [,2]                         [,3]
## [1,] 0.1044385+0i  0.2931457+0i  2.945054e-15+5.546010e-22i
## [2,] 0.1488249+0i  0.5093703+0i -1.227106e-15-0.000000e+00i
## [3,] 0.1159674+0i  0.3414619+0i -2.257875e-15-6.033336e-22i
## [4,] 0.7043472+0i -0.5890805+0i -7.071068e-01+0.000000e+00i
## [5,] 0.4037861+0i -0.1413606+0i  7.071068e-01+0.000000e+00i
## [6,] 0.5425377+0i -0.4135367+0i  0.000000e+00-2.145851e-08i
##                              [,4]           [,5]            [,6]
## [1,]  2.945054e-15-5.546010e-22i -0.06471710+0i -0.212296003+0i
## [2,] -1.227106e-15+0.000000e+00i  0.01388698+0i  0.854071294+0i
## [3,] -2.257875e-15+6.033336e-22i  0.07298180+0i -0.363638739+0i
## [4,] -7.071068e-01+0.000000e+00i -0.66058664+0i  0.018399984+0i
## [5,]  7.071068e-01-0.000000e+00i  0.73761812+0i -0.304719509+0i
## [6,]  0.000000e+00+2.145851e-08i -0.09918316+0i  0.008182973+0i
```

```
# Taking the corresponding vector for largest eigen value Considering only the real values
evec <- Re(eigen(t(matrixB))$vectors[, 1])

# Diving by the sum of the vector
evec <- matrix(evec/sum(evec))

# Comparing it to the output from the function converge()
identical(round(evec, 8), round(r[[1]], 8))
```

```
## [1] TRUE
```

We find that the converge() returns the same pagerank matrix as the eigenvector corresponding to the highest eigenvalue of the decayed matrix B

---

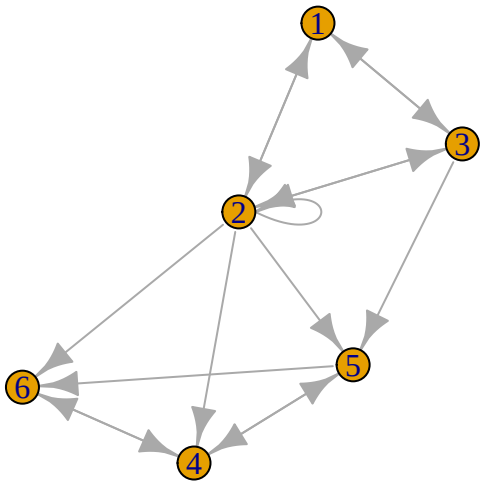
4. Use the graph package in R and its page.rank method to compute the Page Rank of the graph as given in A. Note that you don't need to apply decay. The package starts with a connected graph and applies decay internally. Verify that you do get the same PageRank vector as the two approaches above.

Solution :

```
knitr::opts_chunk$set(message = FALSE, echo = TRUE)
# loading the igraph package to plot the directed graph of nodes
library(igraph)

g <- graph.adjacency(matrixA, "directed", weighted = TRUE)


plot(g)
```




```
knitr::opts_chunk$set(message = FALSE, echo = TRUE)

# Verifying if the page rank matrix obtained through this matches the one through converge()

pgrankmat <- matrix(page.rank(g)$vector)

identical(round(pgrankmat, 8), round(evec, 8))
```

```
## [1] TRUE
```

We observe that igraph package internally handles decay using a damping factor of 0.85 and automatically

assigns a uniform random probability to dangling nodes, and derives the pagerank matrix which is identical to the one derived from function converge() and the eigenvector of th decayed matrix

---