DATA 605 : Final Exam

Ramnivas Singh

12/14/2021

Problem 1 - Playing with PageRank

You'll verify for yourself that PageRank works by performing calculations on a small universe of web pages. Let's use the 6 page universe that we had in the previous discussion For this directed graph, perform the following calculations in R.

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

Answer : Based on the Toy example with 6 URLs of 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

```
input <- c("Webpage Node","Outlinks",
"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")
input <- matrix(input,ncol=2,byrow=TRUE)
input <- as.table(input, row.names=FALSE)
input
```

```
##   A            B
## A Webpage Node Outlinks
## B Node 1       Node 2
## C Node 1       Node 3
## D Node 2       NA
## E Node 3       Node 1
## F Node 3       Node 2
## G Node 3       Node 5
## H Node 4       Node 5
## I Node 4       Node 6
## J Node 5       Node 4
## K Node 5       Node 6
## L Node 6       Node 4
```

Transition Matrix : We derive the matrix that represent the nodes with their links based on the above table of website node (Webpage Node) outgoing links (Outlinks)

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

Initial rank matrix r is given below for this representation

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

Above representation shows - Node 2 does not have any outgoing links. So probability (1/6) for the nodes uses Node 2 may passes to these nodes equally.

Decay matrix B : The decay matrix B can be defined as below

$$d = 0.85B = d * A + (1 - d)/n$$

n is the number of nodes

```r
matrix_A  <- 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,
```

```r
#As one column is having a dead end, we need to convert the matrix in stochastic

n=dim(matrix_A)[1]
stochastic = apply(matrix_A,1,  function(x) if(sum(x)!=1) return(x+ (1/n)) else return(x))
stochastic
```

```
##      [,1]      [,2]      [,3] [,4] [,5] [,6]
## [1,]  0.0 0.1666667 0.3333333  0.0  0.0    0
## [2,]  0.5 0.1666667 0.3333333  0.0  0.0    0
## [3,]  0.5 0.1666667 0.0000000  0.0  0.0    0
## [4,]  0.0 0.1666667 0.0000000  0.0  0.5    1
## [5,]  0.0 0.1666667 0.3333333  0.5  0.0    0
## [6,]  0.0 0.1666667 0.0000000  0.5  0.5    0
```

```r
damping_factor = .85 #Damping factor
identity_matrix = matrix(c(rep(1,36)),ncol =6) #Identity matrix
number_nodes=dim(matrix_A)[1] #Number of Nodes
matrix_B= damping_factor*matrix_A+((1-damping_factor)*(identity_matrix/number_nodes)) #Decay from trans
matrix_B
```

```
##           [,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
```

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

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

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

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

$$r \quad = \quad B \quad \times \quad r$$

Lets develop a function ‘converge()’ which loop through till the pagerank vector stabilizes and return the pagerank vector

```
#@param1 probability matrix
#@param2 initial page rank vector
convergence <- function(probability_matrix, page_rank_vector)
{
     converged = FALSE
     iter <- 0
     page_rank_vector <- rinit
     while(converged == FALSE)
     {
          iter <- iter + 1
          newrvec <- crossprod(matrix_B, page_rank_vector)

          if (identical(newrvec, page_rank_vector))
          {
               converged <- TRUE
          }
          else{
               page_rank_vector <- newrvec
               converged <- FALSE
          }
     }
     return( list(newrvec, iter))
}
```

Initial rank matrix r is given below for this representation. We consider 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)
```

Test function convergence()

```
vector <- convergence(matrix_B, rinit)
vector
```

```
## [[1]]
##            [,1]
## [1,] 0.05170475
## [2,] 0.07367926
## [3,] 0.05741241
## [4,] 0.34870369
## [5,] 0.19990381
## [6,] 0.26859608
##
## [[2]]
## [1] 72
```

Converged Vector

```
converge<-vector[[1]]
converge
```

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

Total iterations to converge

```
iterations<-vector[[2]]
iterations
```

```
## [1] 72
```

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.

Answer : Compute the eigen-decomposition of B

```
eigen((matrix_B))$values
```

```
## [1]  1.00000000+0i  0.57619235+0i -0.42500000+0i -0.42500000-0i -0.34991524+0i
## [6] -0.08461044+0i
```

Lets verify if largest value is 1

```
which.max(eigen((matrix_B))$values)
```

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

Find eigenvector for decayed matrix B

```
eigen(t(matrix_B))$vectors
```

```
##              [,1]          [,2]                         [,3]
## [1,] 0.1044385+0i  0.2931457+0i  9.162390e-16+5.425344e-23i
## [2,] 0.1488249+0i  0.5093703+0i -8.139802e-16-0.000000e+00i
## [3,] 0.1159674+0i  0.3414619+0i -7.362635e-17-5.073466e-23i
## [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+00-1.460109e-08i
##                            [,4]           [,5]            [,6]
## [1,]  9.162390e-16-5.425344e-23i -0.06471710+0i -0.212296003+0i
## [2,] -8.139802e-16+0.000000e+00i  0.01388698+0i  0.854071294+0i
## [3,] -7.362635e-17+5.073466e-23i  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+1.460109e-08i -0.09918316+0i  0.008182973+0i
```

Eigenvector for real values

```
evec <-Re(eigen(t(matrix_B))$vectors[,1])
evec <-matrix(evec/sum(evec))
evec <-round(evec,8)
evec
```

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

Comparing outputs

```
converge<-round(vector[[1]],8)
identical(evec, converge)
```

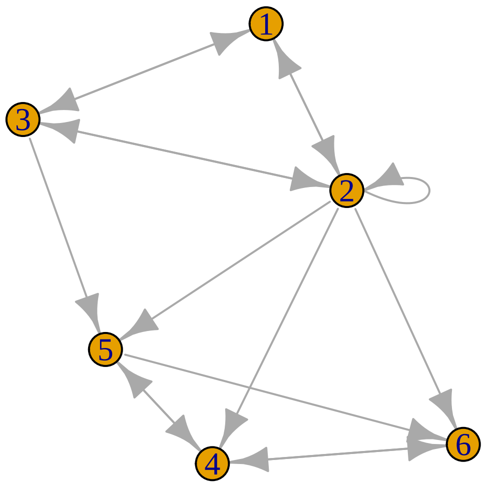
```
## [1] TRUE
```

After rounding off the values to 8th place, we found that the eigenvector to highest eigenvalue of the matrix B returns same pagerank matrix as the function converge() above in this exam. ***

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.

Answer :

```
adjacency_graph<- graph.adjacency(matrix_A, 'directed', weighted=TRUE)
plot(adjacency_graph)
```



```
#Page Rank of resultant vector of the graph
page_rank(adjacency_graph)$vector
```

```
## [1] 0.05170475 0.07367926 0.05741241 0.34870369 0.19990381 0.26859608
```

Verifying Page Rank vector as the two approaches is same

```
page_rank_matrix <- round(matrix(page.rank(adjacency_graph)$vector),8)
identical(page_rank_matrix, converge)
```

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

This verification proves that the page rank vector from both the approaches is identical