XIAO_Lin_Solutions_lab4

Lin Xiao

September 21, 2015

Teammates: Lei Qian & Arpita Mandan

1. Read in the file titled `pageRank.RData`. Use the `ls()` command and ensure that the link matrix `myData` is present in the environment.

```
load("/Users/LinXIAO/Downloads/pageRank.RData")
ls()
```

```
## [1] "myData"
```

2. Count the number of links for each page (at each row) and place these into a diagonal matrix. Call this matrix `M`.

```
M<-diag(apply(myData,1,sum)) #sum link matrix by row
```

3. Invert the matrix `M` and store the inverse into the variable `MInv`.

```
MInv<-solve(M)
```

4. Based on the link matrix and the matrix `M` compute the Broken Rank. What is the problem here? Hint: Before performing the calculation remember to transpose your link matrix since the you need the links to go from j $\rightarrow$ i

```
BRmatrix<-t(myData) %*% MInv
BRmatrix
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    0    1    0    0    0
## [2,]    0    0    1    0    0
## [3,]    1    0    0    0    0
## [4,]    0    0    0    0    1
## [5,]    0    0    0    1    0
```

```
#get corresponding eigenvectors with eigenvalue 1
BR<-eigen(BRmatrix)$vectors[,(which(signif(eigen(BRmatrix)$values,4)==1))]
#normalize the Broken Rank vectors by sum of columns
normalizedBR<-scale(BR,center=F,scale=as.numeric(apply(as.matrix(BR),2,sum)))
normalizedBR
```

```
##        [,1]          [,2]
## [1,] 0.0+0i 0.3333333+0i
## [2,] 0.0+0i 0.3333333+0i
## [3,] 0.0+0i 0.3333333+0i
## [4,] 0.5+0i 0.0000000+0i
## [5,] 0.5+0i 0.0000000+0i
## attr(,"scaled:scale")
## [1]  1.414214 -1.732051
```

The BrokenRank vector p exists but is nonunique. There are two eigenvectors of BRmatrix with eigenvalue 1 and they are totally opposite. We can confirm this by transforming the BRmatrix into a graph, it's not strongly connected.

5. Create a dampening parameter `d` and set this to 0.85. Next, count the number of webpages present in the link matrix and store this in the variable n. Next, initialize a matrix `E` with dimensions equal to the link matrix that contains only 1s. Now combine these elements to compute the Page Rank vector for the link matrix. How are this different from the Broken Rank. Hint: Before performing the calculation remember to transpose your link matrix since the you need the links to go from j $\rightarrow$ i.

```
d<-0.85
n<-ncol(myData) #count the number of webpages
E<-matrix(rep(1,n*n),ncol=n)
PRmatrix<-(1-d)/n*E + d*t(myData) %*% MInv
PRmatrix
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] 0.03 0.88 0.03 0.03 0.03
## [2,] 0.03 0.03 0.88 0.03 0.03
## [3,] 0.88 0.03 0.03 0.03 0.03
## [4,] 0.03 0.03 0.03 0.03 0.88
## [5,] 0.03 0.03 0.03 0.88 0.03
```

```
#get corresponding eigenvector(s) with eigenvalue 1
PR<-eigen(PRmatrix)$vectors[,(which(signif(eigen(PRmatrix)$values,4)==1))]
#normalize the Page Rank vectors by sum of columns
normalizedPR<-scale(PR,center=F,scale=as.numeric(apply(as.matrix(PR),2,sum)))
normalizedPR
```

```
##        [,1]
## [1,] 0.2+0i
## [2,] 0.2+0i
## [3,] 0.2+0i
## [4,] 0.2+0i
## [5,] 0.2+0i
## attr(,"scaled:scale")
## [1] -2.236068
```

There is a small modification with the use of d, in this way we get only one eigenvector with eigenvalue 1.

6. Read in the data from the following source `http://www.ats.ucla.edu/stat/data/mat25.txt` into the variable `myNewData`. Write a function `myPageRank` that takes as its inputs a link matrix and a dampening parameter and outputs the Page Rank vector. Now test this on the the link matrix `myNewData` and the dampening parameter `d`. Hint: Before performing the calculation remember to transpose your link matrix since the you need the links to go from j $\rightarrow$ i

```
fileUrl<-"http://www.ats.ucla.edu/stat/data/mat25.txt"
download.file(fileUrl,destfile='./mat25.txt',method="curl")
myNewData<-as.matrix(read.table('./mat25.txt'))
#download the file and transform the data frame into matrix
#the inputs are d and the link matrix#
#output is the normalized Page Rank vector#
myPageRank<-function(linkmatrix,d){ #Basically doing the same thing as above
```

```
  n<-ncol(linkmatrix)
  E<-matrix(rep(1,n*n),ncol=n)
  M<-diag(apply(linkmatrix,1,sum))
  MInv<-solve(M)
  PRmatrix<-(1-d)/n*E + d*t(linkmatrix) %*% MInv
  PR<-eigen(PRmatrix)$vectors[,(which(signif(eigen(PRmatrix)$values,4)==1))]
  normalizedPR<-scale(PR,center=F,scale=as.numeric(apply(as.matrix(PR),2,sum)))
  print(normalizedPR)
}
myPageRank(myNewData,d)  #test it with myNewData and d
```

```
##                [,1]
##  [1,] 0.05096317+0i
##  [2,] 0.03545182+0i
##  [3,] 0.04341318+0i
##  [4,] 0.02855733+0i
##  [5,] 0.06138548+0i
##  [6,] 0.04926776+0i
##  [7,] 0.02557665+0i
##  [8,] 0.06735780+0i
##  [9,] 0.03449904+0i
## [10,] 0.04142908+0i
## [11,] 0.02761109+0i
## [12,] 0.07174358+0i
## [13,] 0.03078005+0i
## [14,] 0.04893792+0i
## [15,] 0.02291655+0i
## [16,] 0.02499144+0i
## [17,] 0.00600000+0i
## [18,] 0.02527793+0i
## [19,] 0.04136165+0i
## [20,] 0.02794755+0i
## [21,] 0.05808002+0i
## [22,] 0.05580470+0i
## [23,] 0.02626592+0i
## [24,] 0.05023857+0i
## [25,] 0.04414171+0i
## attr(,"scaled:scale")
## [1] 4.659195
```