151bhw5 coding

Lucy Shao

June 7, 2017

```
#mu eigenvalue, x eigenvector
power1<-function(n,A,x,TOL,N){
  k<-1
  xp<-norm(x,type="I")
  x<-x/xp
  while(k<=N){
    y<-A%*%x

    yp<-norm(y,type="I")
     mu<-yp
    if(yp==0){
      return (list(0,x,k))
    }
    err<-norm(x-(y/yp),type="I")
    x<-y/yp
    if(err<TOL){
      return (list(mu,x,k))

    }
    k<-k+1
  }

}


 power2<-function(n,A,x,TOL,N){
  k<-1
  xp<-norm(x,type="2")
  x<-x/xp
  while(k<=N){
    y<-A%*%x

    yp<-norm(y,type="2")
     mu<-yp
    if(yp==0){
      return (list(0,x))
    }
    err<-norm(x-(y/yp),type="2")
    x<-y/yp
    if(err<TOL){
      return (list(mu,x,k))

    }
    k<-k+1
  }

 }
```

```r
#check wether invertible
f <- function(m) class(try(solve(m),silent=T))=="matrix"

invpower<-function(n,A,x,TOL,N){
  q<-as.numeric((t(x)%*%A%*%x)/(t(x)%*%x))
  k<-1
  xp<-norm(x,type="I")
  x<-(x/xp)
  while(k<=N){
    if(f(A-q*diag(n))==FALSE){
      return(q)
    }
    y<-solve(A-q*diag(n))%*%x
    yp<-norm(y,type="I")
    mu<-yp
    err<-norm((x-(y/yp)), type="I")
    x<-(y/yp)
    if(err<TOL){
      mu<-(1/mu)+q
      return(list(mu,x,k))
    }
    k=k+1
  }
  return("unsuccessful")
}
```

```r
p=0.85
#first
one<-c(0.01,0.01,0.86,rep(0.01,12))
#2
two<-c(0.435,rep(0.01,9),0.435,rep(0.01,4))
three<-c(0.01,0.435,0.01,0.01,0.435,rep(0.01,10))
four<-c(17/60,17/60,rep(0.01,10),17/60,0.01,0.01)
five<-c(rep(0.01,5),0.435,0.01,0.435,rep(0.01,7))
six<-c(rep(0.01,6),17/60,17/60,0.01,17/60,rep(0.01,5))

#7th
seven<-c(rep(0.01,9),0.86,rep(0.01,5))
eight<-c(rep(0.01,8),0.435,0.435,rep(0.01,5))
nine<-c(rep(0.01,4),0.435,rep(0.01,5),0.435, rep(0.01,4))
ten<-c(rep(0.01,8),17/60,0.01,17/60,0.01,0.01,17/60,0.01)
eleven<-c(rep(0.01,11),0.435,0.01,0.435,0.01)
#12th
twelve<-c(rep(0.01,12),0.86,rep(0.01,2))
thirteen<-c(rep(0.01,10),0.435,0.01,0.01,0.01,0.435)
#14th
fourteen<-c(rep(0.01,12),0.86,rep(0.01,2))
#15th
fifteen<-c(rep(0.01,13),0.86,0.01)
C<-rbind(one,two,three,four,five,six,seven,eight,nine,ten,eleven,twelve,thirteen,fourteen,fifteen)

#from the smallest to the largest eigenvalue
order(Re(eigen(C)$values))
```

```
##  [1]  2  3 10 11  5  6  8  9 15 13 14 12  7  4  1
```

```
#Use three methods ps. the first returned value is the dominant eigenvalue
power1(15,C,matrix(c(rep(1,15))),0.00001,100)
```

```
## [[1]]
## [1] 0.9975118
##
## [[2]]
##               [,1]
## one      0.9867283
## two      0.9943495
## three    0.9844182
## four     0.9647446
## five     0.9690685
## six      0.9473609
## seven    0.9712973
## eight    0.9800354
## nine     0.9868228
## ten      0.9663151
## eleven   1.0000000
## twelve   1.0000000
## thirteen 1.0000000
## fourteen 1.0000000
## fifteen  1.0000000
##
## [[3]]
## [1] 18
```

```
power2(15,C,matrix(c(rep(1,15))),0.00001,100)
```

```
## [[1]]
## [1] 0.997506
##
## [[2]]
##               [,1]
## one      0.2590408
## two      0.2610394
## three    0.2584324
## four     0.2532679
## five     0.2544005
## six      0.2487011
## seven    0.2549844
## eight    0.2572789
## nine     0.2590606
## ten      0.2536761
## eleven   0.2625182
## twelve   0.2625182
## thirteen 0.2625182
## fourteen 0.2625182
## fifteen  0.2625182
##
## [[3]]
## [1] 16
```

```r
invpower(15,C,matrix(c(rep(1,15))),0.00001,100)
```

```
## [[1]]
## [1] 0.9975107
##
## [[2]]
##            [,1]
##  [1,] 0.9867129
##  [2,] 0.9943389
##  [3,] 0.9844070
##  [4,] 0.9647330
##  [5,] 0.9690632
##  [6,] 0.9473571
##  [7,] 0.9712949
##  [8,] 0.9800315
##  [9,] 0.9868190
## [10,] 0.9663133
## [11,] 1.0000000
## [12,] 1.0000000
## [13,] 1.0000000
## [14,] 1.0000000
## [15,] 1.0000000
##
## [[3]]
## [1] 3
```

```r
#Exact,the largest is 0.9975
eigen(C)$values
```

```
##  [1]  9.975107e-01+0.000000e+00i -4.250000e-01+7.361216e-01i
##  [3] -4.250000e-01-7.361216e-01i  6.067639e-01+0.000000e+00i
##  [5] -2.677320e-01+4.637329e-01i -2.677320e-01-4.637329e-01i
##  [7]  5.313449e-01+0.000000e+00i -3.512316e-03+4.840569e-01i
##  [9] -3.512316e-03-4.840569e-01i -2.968709e-01+1.847287e-01i
## [11] -2.968709e-01-1.847287e-01i  6.109695e-04+0.000000e+00i
## [13]  3.470695e-07+6.011453e-07i  3.470695e-07-6.011453e-07i
## [15] -6.941389e-07+0.000000e+00i
```

```r
#random initial
set.seed(123)
power1(15,C,matrix(rnorm(15)),0.00001,100)
```

```
## [[1]]
## [1] 0.9975101
##
## [[2]]
##               [,1]
## one      0.9867085
## two      0.9943361
## three    0.9844051
## four     0.9647316
## five     0.9690592
```

```
## six      0.9473537
## seven    0.9712923
## eight    0.9800296
## nine     0.9868165
## ten      0.9663097
## eleven   0.9999923
## twelve   1.0000000
## thirteen 0.9999978
## fourteen 1.0000000
## fifteen  0.9999923
##
## [[3]]
## [1] 73
```

```
power2(15,C,matrix(rnorm(15)),0.00001,100)
```

```
## [[1]]
## [1] 0.9975098
##
## [[2]]
##                [,1]
## one      -0.2590341
## two      -0.2610355
## three    -0.2584272
## four     -0.2532620
## five     -0.2544005
## six      -0.2487018
## seven    -0.2549855
## eight    -0.2572786
## nine     -0.2590612
## ten      -0.2536784
## eleven   -0.2625241
## twelve   -0.2625192
## thirteen -0.2625215
## fourteen -0.2625192
## fifteen  -0.2625241
##
## [[3]]
## [1] 73
```

```
invpower(15,C,matrix(rnorm(15)),0.00001,100)
```

```
## [1] "unsuccessful"
```

A PDF report

I have explored that the approximate eigenvalues are all very close to the exact one. The rank is shown by computation that the smallest eigenvalue to the largest eigenvalue: 2 3 10 11 5 6 8 9 15 13 14 12 7 4 1 which is significant if we need more information about the webpages. The tolerance was set to be 10^-5. If the initial value is set to be a vector of 1's, the iterations it take for power1 is 18, for power2 is 16, for invpower is 3. This result has shown that for this approximation, the inverse power method takes less iterations and reaches better efficiency. If the initial value is set to be random, the inversepow method's iteration exceeds 100.