Temas selectos de econometría y finanzas (modulo de matrices aleatorias)

J. Antonio García Ramirez, Tarea 4

23 de Octobre, 2018

Ejercicio 2

Considere la descomposición espectral $H = OXO^t$ *, donde* H *es una matriz simétrica de dimensión* $n \times n$*,* O *es la matriz ortogonal que contiene los eigenvectores de* H*, y* X *es una matriz diagonal que contiene sus eigenvalores. Los diferenciales de* H *se pueden expresar como:*

$$dH = \prod_{i<j} |\lambda_i - \lambda_j (dX)(O^t dO)$$

Numéricamente, las perturbaciones en X *y* O *se calculan a través de las perturbaciones en* H*. Como analistas numéricos siempre pensamos en* H *como la entrada, y en* $\{X, O\}$ *como la salidas, por lo que es natural hacerse preguntas en esa dirección. Asumiendo que la descomposición espectral es única después de fijar la base de las columnas de* O*, la perturbación a primer orden en* $\{X, O\}$ *debido a la perturbación en* H *está dada por:*

$$\frac{(dX)(O^t dO)}{dH} = \frac{1}{\prod_{i<j} |\lambda_i - \lambda_j|} = \frac{1}{\Delta(X)}$$

Donde $\Delta(X)$ *es el valor absoluto del determinante de Vandermonde.*

Basandose en la expresión anterior, escriba un código para obtener numéricamente el jacobiano, y compare el resultado con el valor exacto al calcular el determinante de Vandermonde para una matriz fija de dimensión 10×10 *de tal manera que el error relativo sea menor a* 10^{-3}*. Se recomienda seguir los siguientes pasos:*

i. *Construya una matriz simétrica:* H

ii. *Obtenga los valores y vectores propios de la matriz construida:* $\{X, O\}$

iii. *Determine la dimensión de la matriz jacobiana:* $n(n+1)/2 \times n(n+1)/2$

iv. *De manera iterativa realizar los siguientes pasos:*

   1. *Generar una perturabación* E *en el elemento* (i, j) *de la matriz* $H : H' = H + \epsilon E$
   2. *Obtener los valores y vectores propios de* $H' : \{X', 0'\}$
   3. *Calcular los valores propios perturbados:* $dX = (X - X')/\epsilon$
   4. *Calcular los vectores propios perturbados:* $O^t dO = O^t(O' - O)/\epsilon$
   5. *Llenar las primeras* n *columnas de la matriz jacobiana con* dX*, y las restantes con* $O^t dO$

v. *Calcular el valor absoluto de la matriz jacobiana resultante y comparar el resultado con el valor del determinante de Vandermonde de* H*, calculado con alguna función preestablecida en el lenguaje de su preferencia.*

vi. *Asegurarse que el error relativo sea menor a* 10^{-3}

Después de implementar la simulación propuesta, fijar una semilla y al intentar fijar un ϵ adecuado de perturbación para después de fijar la matriz obtener un error relativo menor a 10^{-3} la única dificultad fue que la función eigen del kernel base del lenguaje R es demasiado precisa por lo que al comparar el error relativo este siempre era menor a la precisión de la máquina.

Por lo que se recurrió a una implementación menos precisa de la descomposición de valores y vectores propios, la función eigjacobi del paquete *pracma*, que realiza reflexiones y rotaciones para obtener la descomposición.

Incluimos a continuación el código de la simulación las únicas salidas que desplegamos son la matriz fija H del tipo GOE, con entradas redondeadas a dos decimales y el error relativo al cálculo del determinante de Vandermonde con una perturbación de $\epsilon = 0.001$.

```
rm(list = ls())
set.seed(0) # fijamos una semilla
n <- 10     # fijamos las dimensiones de la matriz
    #Construimos una matriz simetrica GOE
M <- matrix( rnorm(n*n), ncol = n, nrow = n )
H <- (M +t(M))/2
round(H, 2)
```

```
##        [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10]
##  [1,]  1.26  0.22  0.55  0.52  1.09 -0.64 -0.28 -0.14 -0.40  1.85
##  [2,]  0.22 -0.80 -0.39 -0.42  0.13 -0.39  0.12 -0.32  0.84 -1.06
##  [3,]  0.55 -0.39  0.13  0.19 -0.25  1.47  0.07 -0.67 -0.26  0.03
##  [4,]  0.52 -0.42  0.19 -0.65 -0.05  0.18  0.44 -0.27  0.51 -0.58
##  [5,]  1.09  0.13 -0.25 -0.05 -1.17 -0.56 -1.19  0.91  0.20  0.18
##  [6,] -0.64 -0.39  1.47  0.18 -0.56  0.25  0.43  0.46 -1.32 -0.57
##  [7,] -0.28  0.12  0.07  0.44 -1.19  0.43 -1.43  0.25  0.62 -0.11
##  [8,] -0.14 -0.32 -0.67 -0.27  0.91  0.46  0.25 -0.12 -0.59  0.01
##  [9,] -0.40  0.84 -0.26  0.51  0.20 -1.32  0.62 -0.59  1.26  0.44
## [10,]  1.85 -1.06  0.03 -0.58  0.18 -0.57 -0.11  0.01  0.44  1.00
```

```
library(pracma) # utilizamos un algoritmo poco preciso para calcular la
                # eigen descomposicion
eigen <- eigjacobi(H)
X <- eigen$D
O <- eigen$V
Jacobiano <- matrix( rep (0, (n*(n+1)/2)**2),
                         ncol =  n*(n+1)/2, nrow = n*(n+1)/2 )
epsilon <- 1e-3 # fijamos la perturbacion
columna <- 1    # un contador para iterar facilmente
for (i in 1:n)
{
    for(j in i:n)
    {
        E <- diag(rep(0, n))     # generamos la matriz de perturbacion por cada entrada
        E[i, j] <- E[j, i] <- 1  # de la matriz simetrica
        H.prima <- H + epsilon*E # perturbamos
        eigen.aux <- eigjacobi(H.prima) # nueva eigen-descomposicion
        X.prima <- eigen.aux$D
        O.prima <- eigen.aux$V
        d.X <- (X - X.prima)/epsilon    # val. prop. perturbados
        d.O <- (t(O) %*% (O.prima - O) )/epsilon # vect. prop. perturbados
        Jacobiano[1:n, columna] <- d.X    # guardamos las perturbaciones de cada entrada
        Jacobiano[ (n+1):( n*(n+1)/2 ), columna] <- d.O[upper.tri(d.O)]
        columna <- columna + 1
    }
}
#########################
library(matrixcalc)
empirico <- abs( det(Jacobiano) ) # calculamos el valor absoluto
```

```
                                  # de la matriz-Jacobiano que simulamos
teorico <- 1/abs( det( vandermonde.matrix(X, n) ) ) # calculamos el determinante
                                                    # de vandermonde de los val.prop
(error <- abs( (teorico-empirico) / (teorico ))) # Calculo del error relativo
```

```
## [1] 3.902359e-05
```