César Isaí García Cornejo
cesar.cornejo@cimat.mx
Computo Científico

Tarea 3

Semestre 3

CIMAT

Ejercicio 1
Sea Q una matriz unitaria aleatoria de 20$\times$ 20 (eg. con A una matriz de tamaño 20 $\times$20 aleatoria calculen su descomposición QR). Sean $\lambda_1 > \lambda_2 > \cdots > \lambda_{20} = 1 > 0$ y

$$B = Q^* diag(\lambda_1, \lambda_2, \cdots, \lambda_{20})Q, \quad B_\varepsilon = Q^* diag(\lambda_1 + \varepsilon_1, \lambda_2 + \varepsilon_2, \cdots, \lambda_{20} + \varepsilon_{20})Q$$

con $\varepsilon_i \sim N(0, \sigma)$, con σ =0.02λ_{20} = 0.01.

1. Comparar la descomposición de Cholesky de B y de B_ε usando el algoritmo de la tarea 1. Considerar los casos cuando B tiene un buen número de condición y un mal número de condición.
2. Con el caso mal condiciona, comparar el resultado de su algoritmo con el del algoritmo Cholesky de scipy.
3. Medir el tiempo de ejecución de su algoritmo de Cholesky con el de scipy.

Solución:

1. Siguiendo las indicaciones, generamos una matriz aleatoria A donde cada entrada en una realización de una distribución normal estandar. Entonces, por la descomposición QR obtenemos una matriz unitaria. Veamos que para tener una matriz bien condicionada o mal condicionada se requiere que su número de condición sea relativamente grande. El número de condición depende de los valores singulares extremos, es decir, el valor singular mayor y menor. En matrices cuadradas los valores singulares se obtienen con los valores propios. Así, la matriz B y B_ε se construyen para poder manipular el número de condición de las matrices.

   Se propone tomar $\lambda_1 > \lambda_2 > \cdots > \lambda_{20} = 1$ de la forma

   $$\lambda_i = \frac{\alpha^{20}}{\alpha^i}, \qquad i = 1, \cdots 20.$$

   Esta caracterización nos permite obtener valores similares de λ_i para α pequeños cercanos a uno, en contraste con α mayores, que genera valores muy dispares para λ_i.

   De esta forma, para el caso $\alpha = 1.1$ tenemos que el valor mayor de lamda $\lambda_1 = 6.11$ mientras que por construcción $\lambda_{20} = 1$. Es decir el numero de condición es próximo a 6.11 que es una matriz bien condicionada. Para $\alpha = 5$ tenemos que $\lambda_1 \approx 1{,}9 \times 10^{13}$ lo que forma un número de condición malo.

   Pensemos por ahora en el caso bien condicionado. Tras la construcción de las matrices B y B_ε se calcula la matriz R de la descomposición Cholesky implementada en la primer tarea.

   $$B = R^* R \qquad B_\varepsilon = R_\varepsilon^* R_\varepsilon$$

   Esto nos genera dos matrices, que para fines comparativos consideramos la diferencia entre ellas.

   $$\Delta R = R - R_\varepsilon$$

Dicha diferencia se imprimio en pantalla en el código adjunto. Notamos errores relativamente pequeños, y que dicho error ronda el orden de milesimas (-3).

Se hace lo mismo para la matriz B mal condicionada ($\alpha = 5$), nuevamente se imprime en pantalla la diferencia entre las matrices R y notamos que tiene un error menor pero más variable. Estos errores rondan del orden de -9 a -6.

Lo anterior no es concluyente, va contra la intuición. Por ello, consideramos en obtener la norma de matrices para saber con certeza cuáles pares de matrices *distan* más. Usando la norma L^2 implementada con scipy, que consta del valor singular más grande, tenemos que la norma en el caso mal condicionado es mayor al caso bien condicionado, pues

$$\|\Delta R_{BC}\|_2 = 0{,}0461$$
$$\|\Delta R_{MC}\|_2 = 0{,}0532$$

2. Para el próposito indicado, es necesario mandar a llamar la paquetería con el comando

```
from scipy.linalg import cholesky as scipycholesky

R_sci = scipycholesky(B)
R_sci_eps = scipycholesky(B_eps)
```

donde el renombramiento en la función se aplicó para desambiguar de la función local del mismo nombre, y B es la matriz mal condicionada construida con $\alpha = 5$. Al imprimir en pantalla y análizar notamos que dicha matriz R es sustancialmente diferente al obtenido por nosotros. Luego, scipy manejo mejor el malcondicionamiento, pues tambien lo notamos al obtener su norma L^2

$$\|\Delta R_{ScipyMC}\|_2 = 0{,}052$$

que es menor en distancia a la obtenida por nuestro algoritmo.

3. El tiempo de ejecución se obtuvo de manera usual con la función time de python. Al medir el tiempor de nuestro algoritmo, este tiene una duración de **0.00976** nanosegundo contra el tiempo que tarda la función de scipy que demoró **0.00950**

   Estos valores del tiempo difieren a cuando comentamos las impresiones de pantalla de las matrices. En este caso tenemos que el tiempo que tarda nuestro algoritmo es de **0.00199** nanosegundo vs scipy que tarda **0.00086** nanosegundo.

   En ambos situaciones el algoritmo de scipy tarda menos, lo que es razonable considerando que sabemos que dicha librearia esta construida en C.

Ejercicio 2
Resolver el problema de mínimos cuadrados,

$$y = X\beta + \varepsilon, \quad \varepsilon_i \sim N(0, \sigma)$$

usando su implementación de la descomposición QR; β es de tamaño $n \times 1$ y X de tamaño $n \times d$.

Sean $d = 5$, $n = 20$, $\beta = (5, 4, 3, 2, 1)$ y $\sigma = 0{,}13$.

1. Hacer X con entradas aleatorias $U(0, 1)$ y simular y. Encontrar $\hat{\beta}_p$ haciendo $X + \Delta X$, donde las entrads de ΔX son $N(0, \sigma = 0{,}01)$ Comparar a su vez con $\hat{\beta}_c = ((X + \Delta X)^*(X + \Delta X))^{-1}(X + \Delta X)^* y$ usando el algoritmo genérico para inveritr matrices scipy.linalg.inv
2. Lo mismo que el anterior pero con X mal condicionada (ie. con casi colinealidad).

Solución:

1. Procedemos con crear el vector y simplemente usando la relación dada. Con el algoritmo de regresión lineal por medio de QR podemos hacer regresión y calcular el estimador de mínimos cuadrados.

   Vemos que el estimador de mínimos cuadrados es, para una realización de ε

   $$\hat{\beta} = (4{,}95718095; 4{,}03395321; 3{,}02021941; 2{,}01559771; 1{,}04167694)$$

   Con el error ΔX, tenemos una nueva matriz de diseño $X + \Delta X$ cuyo estimador de mínimos cuadrados es

   $$\hat{\beta}_p = (4{,}92740612; 4{,}0334383; 3{,}00621361; 2{,}03061811; 1{,}08462213)$$

   que se obtuvo por regresión con QR.

   Finalmente, para la misma matriz de diseño $X + \Delta X$, obtenemos el estimador de mínimos cuadrados análitico que es

   $$\hat{\beta}_c = (4{,}92740612; 4{,}0334383; 3{,}00621361; 2{,}03061811; 1{,}08462213)$$

   La primera observación a hacer es que $\hat{\beta}_p = \hat{\beta}_c$ y esto tiene sentido pues resolver el sistema de ecuaciones lineales por QR

   $$(X + \Delta X)^*(X + \Delta X)\hat{\beta}_p = (X + \Delta X)^* y$$

   es equivalente al expresado en el enunciado para $\hat{\beta}_c$ siempre que la matriz X sea de rango completo para evitar problemas con la inversa.

   La siguiente observación es que vemos que el valor de los estimadores $\hat{\beta}$ y $\hat{\beta}_p$ difieren muy poco.

2. Para el caso con casí colinealidad o mal condicionado, construímos la matriz de diseño de forma que una de sus columnas es combinación lineal de las otras.

   Construimos la matriz de diseño casi colineal tomando una matriz aleatoria de $n \times d$ y luego sustituimos la segunda columna por la primera más un error que distribuye normal con media cero y varianza pequeña. En este caso, consideramos un error con varianza $\sigma = 0.01$ para tener casi la misma fila. Luego, hacemos lo mismo que el inciso previo.

   El estimador de mínimos cuadrados de la matriz de diseño es

   $$\hat{\beta} = (2{,}01977367; -8{,}83891167; -2{,}37905967; 1{,}06147236; -1{,}09874232)$$

   que pronto vemos difiere bastante de β. Luego, como ya sabemos que $\hat{\beta}_p = \hat{\beta}_c$ reportamos únicamente uno de ellos. De igual forma, se comprobó numericamente que dicha iguadad se satisface en el código correspondiente. Así

   $$\hat{\beta_c} = (1{,}99456172; -8{,}99101306; -2{,}4984814; 1{,}04200205; -1{,}1090989)$$

   donde volvemos a notar que la solución ha explotado lo que representa el hecho de estar mal condicionada.