Cómputo científico para probabilidad y estadística. Tarea 3. Estabilidad.

Juan Esaul González Rangel

Septiembre 2023

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

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

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

   a) 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.

   Para comparar la descomposición Cholesky se usó el siguiente procedimiento:

   - Se construyeron las matrices B y B_ε y se obtuvo su descomposición Cholesky, llamadas B-chol y B_ε-chol.
   - Se obtuvo la diferencia $B' = B$-chol$-B_\varepsilon$-chol, que es una matriz triangular superior y se evaluaron las siguientes métricas:
     - Valor máximo de $|B'|$.
     - Promedio de los valores de B'.
     - Suma de los valores absolutos de las entradas de B'.

   Con base en estas comparaciones, podemos apreciar cómo es que la descomposición Cholesky de B y de B_ε difieren entre sí. Los resultados de la comparación se resumen en la siguiente tabla

Métrica	Caso bien condicionado	Caso mal condicionado
Entrada máxima	0.011919704	0.125
Media	-9.25E-07	0.0001284255518
Suma de valor absoluto	62.69409725	42.63486008

   Notemos que los errores de ambas matrices están centrados en un número cercano a cero, lo que significa que modificar ligeramente a B de manera centrada tuvo un efecto "centrado" también en el error que se ocasiona en su descomposición de Cholesky, pero el caso mal condionado tiene una media mayor, por lo que podemos decir que la perturbación tendió a "sesgar" más la aproximación.

   También podemos notar que, aunque la entrada máxima de la diferencia entre B-chol y B_ε-chol es mayor para la matriz mal condicionada, la suma de los valores absolutos es mayor en el caso bien condicionado. La interpretación es que, a pesar de que en la matriz mal condicionada hay entradas de B_ε-chol que cambian mucho respecto a las de B-chol, en general el error de aproximación ocasiona que las diferencias entre B y B_ε no se vean reflejadas en las factorizaciones de Cholesky de cada una, por lo que podemos decir que en el caso mal condicionado nuestra factorización de Cholesky es menos precisa (los pequeños cambios en la entrada no se reflejan en la salida).

   b) Con el caso mal condicionado, comparar el resultado de su algoritmo con el del algoritmo de Cholesky de scipy.

Al utilizar los mismo pasos que en la comparación anterior, para la matriz mal condicionada y el algoritmo de Scipy, obtenemos la siguiente tabla. Recordemos que lo que se evalua son las entradas de la matriz B-chol $- B_\varepsilon$-chol, donde B-chol es la descomposición Cholesky de B y B_ε-chol, la correspondiente para B_ε.

Métrica	Algoritmo propio	Algoritmo de Scipy
Entrada máxima	0.125	0.212919468
Media	0.0001284255518	-0.001174055979
Suma de valor absoluto	42.63486008	66.79376308

Nuevamente tenemos que las diferencias están relativamente centradas, aunque el algoritmo de Scipy presenta una media un poco más lejana de cero. En esta ocasión, tanto la entrada máxima como la suma en valor absoluto de la matriz obtenida con Scipy son mayores, lo que puede ser un indicio de que el algoritmo de Scipy implemente alguna función de robustez para evitar que el error se propague en problemas mal condicionados puesto que es esperable que los resultados de las descomposiciones para B y B_ε varíen.

c) Medir el tiempo de ejecución de su algoritmo de Cholesky con el de scipy.

El tiempo de ejecución de ambos algoritmos era muy corto, lo que resultaba en mediciones nulas varias veces, por lo que la solución fue medir el tiempo total que se tarda cada algoritmo en obtener la factorización Cholesky un total de 1,000 veces para cada matriz (B o B_ε en los casos bien y mal condicionados). En la siguiente tabla, donde las cantidades se expresan en segundos, se resume dicha información.

	Algoritmo propio	Algoritmo de Scipy	Proporción
B bien condicionada	1.27788901	0.00851154	154.042549
B_ε bien condicionada	1.36065578	0.00852394	142.8957261
B mal condicionada	1.34203696	0.01675606	72.9913062
B_ε mal condicionada	1.22572351	0.00680399	180.0227416

En la columna proporción se indica qué tanto más grande es el tiempo que llevó el cálculo con el algoritmo propio comparado con el cálculo con el algoritmo de Scipy. Notemos que Scipy es en general entre 70 y 180 veces más rápido para resolver estos problemas, y esto puede deberse al lenguaje en que está implementado Scipy.

Una observación es que el algoritmo de Scipy tiene tiempos de ejecución similares para tres de los cuatro casos, excepto el caso B mal condicionado en el que tarda alrededor del doble que en los otros, mientras nuestro algoritmo propio presenta muy poca diferencia en cualquier caso. Una posible explicación es que hay funciones extra del algoritmo de Scipy para lidiar con casos mal condicionados que causan un mayor tiempo de ejecución.

En conclusión, en este ejercicio observamos que al usar matrices bien condicionadas, los efectos de una modificación pequeña en la entrada producen modificaciones de magnitud correspondiente en la salida, mientrs que cuando trabajamos con matrices mal condicionadas, estos efectos pueden no reflejarse, o ser magnificados, causando errores de aproximación.

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 $\sigma = 0.13$

a) Hacer X con entradas aleatorias $U(0, 1)$ y simular y. Encontrar $\hat{\beta}$ y compararlo con el obtenido $\hat{\beta}_p$ haciendo $X + \Delta X$, donde las entradas 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 invertir matrices `scipy.linalg.inv`.

En la siguiente tabla se resumen los estimadores $\hat{\beta}$, $\hat{\beta}_p$ y $\hat{\beta}_c$ para una matriz X bien condicionada.

Valor real	5	4	3	2	1
$\hat{\beta}$	5.08438964	3.90654472	3.07956781	2.01345467	0.93787969
$\hat{\beta}_p$	5.04854568	3.93883121	3.17391487	1.94807777	0.8944391
$\hat{\beta}_c$	5.04854568	3.93883121	3.17391487	1.94807777	0.8944391

Hay varias cosas que podemos observar en ella. Lo primero es que el error absoluto en todos los casos es del orden de 0.1 unidades, lo que significa que el error relativo es de entre 0.02 (para $\beta_1 = 5$) y 0.1 (para $\beta_5 = 1$). Considerando que la matriz aleatoria que se creó tiene entradas uniformes en $(0, 1)$ y que el ruido que se le añadió es Normal(0,0.13), podemos decir que la perturbación promedio de cada observación es de 13 %, por lo que es aceptable que los estimadores tengan una variación de entre 2 % y 10 %.

También podemos notar que al añadir ruido a X (cuando se considera $X + \Delta X$), los estimadores cambian, pero este cambio no afecta a todas las entradas de la misma manera, pues las dos primeras entradas de $\hat{\beta}_p$ son más cercanas a las correspondientes de β que las de $\hat{\beta}$, aunque en las tres restantes la tendencia se revierte. Aunque tomando en cuenta el error total de todas las entradas, $\hat{\beta}$ es más cercano a β que $\hat{\beta}_p$.

Así mismo, notamos que $\hat{\beta}_c$ coincide a la perfección con $\hat{\beta}_p$. El estimador $\hat{\beta}_c$ es el estimador teórico de mínimos cuadrados, y el hecho de que este coincida con β_p nos indica que el algoritmo que estamos utilizando es efectivo para encontrar el estimador que buscamos. La diferencia es que el cálculo explícito de $\hat{\beta}_c$ requiere la inversión de una matriz, lo cual supone un costo computacional muy alto y por lo tanto es menos recomendable, mientras que $\hat{\beta}_p$ fue obtenido mediante un método más eficiente.

En conclusión, con esta matriz notamos que los estimadores aproximan a los valores reales con un nivel de error aceptable (teniendo en cuenta la perturbación), que los cambios en la estimación cuando se realiza un cambio pequeño en la matriz X también son pequeños, y que el método que estamos usando para encontrar el estimador de mínimos cuadrados es equivalente al método analítico clásico, aunque más eficiente numéricamente.

b) Lo mismo que el anterior pero con X mal condicionada (ie. con casi colinealidad).

En la siguiente tabla se resumen los reusltados de este segundo inciso.

Valor real	5	4	3	2	1
$\hat{\beta}$	-73.13937309	-2332.410907	-2411.89615	158.6812259	4673.707698
$\hat{\beta}_p$	4.4716246	4.61787602	2.73137268	3.50345316	-0.34596404
$\hat{\beta}_c$	4.4716246	4.61787602	2.73137268	3.50345316	-0.34596404

Lo primero que llama la atención es que ahora el estimador $\hat{\beta}$ es muy alejado tanto en magnitudes absolutas como relativas de lo que se esperaría, por lo que en la práctica un estimador como este es inutilizable. Además, aún en este caso se conserva que $\hat{\beta}_p = \hat{\beta}_c$, lo que ssignifica que aún en el caso mal condicionado, el método de solución de mínimos cuadrados mediante QR es equivalente al método clásico de solución de mínimos cuadrados.

Es algo sorprendente que ahora $\hat{\beta}_p$ sea más cercano a β que $\hat{\beta}$, y la explicación es que al agregar ruido aleatorio a cada columna de X, la matriz $X + \Delta X$ tiene columnas que están menos próximas a ser linealmente dependientes, y por lo tanto su número de condición es menor. La estimación $\hat{\beta}_p$ es considerablemente mejor que $\hat{\beta}$ para cualquier entrada, pero aún así no llega a ser tan buena como cualquiera de las estimaciones del caso bien condicionado.

Podemos concluir de todo esto que el método de resolución de mínimos cuadrados funciona incluso en el caso mal condicionado, pero que trabajar con una matriz con un mal número de condición puede alterar los resultados numéricos mucho más que trbajar con datos en los que existe un error de aproximación. De hecho, como lo vimos en el caso mal condicionado, una matriz con datos aproximados puede ofrecernos una mejor solución cuando esto disminuye el número de condición.