Tarea 2. Descomposición QR y mínimos cuadrados

Ricardo Chávez Cáliz

September 13, 2017

Problema 1. Implementar el algoritmo de Gram-Schmidt modificado 8.1 del Trefethen (p. 58) para generar la descomposición QR.

Se implementó dicho algoritmo a una matriz A de entradas aleatorias $\sim U(0,1)$ para la obtención de matrices Q y R correspondientes a la descomposición QR de A. Es decir:

1. $A = QR$
2. R es triangular superior
3. $Q^t \cdot Q = I$

Para verificar que 1) se satisface se calcula QR y se compara con A usando el método *allclose* de numpy. Para 2) se usa el método *VerificaTS*. El punto 3) se verifica calculando $Q^t \cdot Q$ y se compara con la matriz identidad con el método *allclose*. Se realiza esta

Problema 2. Implementar el algoritmo que calcula el estimador de mínimos cuadrados en una regresión usando la descomposición QR.

Se implementó el algoritmo en la función *estimadorMC*, el cual halla b tal que $\|Y - Xb\|_2$ sea mínimo, donde X es matriz de diseño obtenida de un vector de observaciones y Y es el vector aleatorio a estimar. El residuo $r = Y - X\beta$ es mínimo cuando $r \in Null(P)$ donde P es un proyector ortogonal con $rango(P) = rango(X)$ esto implica que $P \cdot r = 0$, por lo tanto $X\beta = Py$. La proyección ortogonal es obtenida con $P = Q \cdot Q^t$, donde Q viene de la descomposición QR de X.

$$\begin{aligned} X \cdot \beta &= P \cdot Y \\ Q\dot{R} \cdot \beta &= Q \cdot Q^t \cdot Y \\ R \cdot \beta &= Q^t \cdot Y \end{aligned}$$

Para hacer esto se llama al método que construye la matriz de diseño con un parámetro p, y a algún método que calcula descomposición QR de esta (el descrito en el punto 1 ó el propio de Numpy). En la última ecuación R es triangular superior entonces es posible resolver β usando el método *BackwardSubst*, de esta manera encontramos β y $X \cdot \beta$ será la mejor aproximación de Y.

Problema 3. Generar $\mathbf{Y}$ compuesto de $y_i = sen(x_i) + \epsilon_i$ donde $\epsilon_i \sim N(0, \sigma)$ con $\sigma = 0.1$, para $x_i = \frac{4\pi i}{n}$ para $i = 1, \ldots, n$.

Hacer un ajuste de mínimos cuadrados a $\mathbf{Y}$, con descomposición QR, ajustando un polinomio de grado $p - 1$.

- Considerar los 12 casos: $p = 3, 4, 5, 100$ y $n = 100, 1000, 10000$.
- Graficar el ajuste en cada caso.
- Medir tiempo de ejecución de su algoritmo, comparar con descomposición QR de scipy y graficar los resultados.

Se consideran los casos pedidos y se obtienen los estimadores de Y como se muestra en Figuras 1,2 y 3. Se observa que la elección del parámetro p es delicada, dado que para p muy pequeños no se obtiene una buena aproximación (estamos aproximando por un polinomio de grado pequeño) pero si p es demasiado grande, entonces la aproximación tampoco es buena.

En Figura 4, se observan las distintas aproximaciones de Y cuando $p = 10$ y n varía. En este caso se puede ver que 10 es un parámetro apropiado para aproximar Y.

Se midieron los tiempos para la estimación de mínimos cuadrados y para el algoritmo desarrollado en el punto 1 y el dado por scipy. Se presentan aquí en la Figura 5, graficando el tiempo de ejecución a medida que el n crece. Dejando p fijo en 2 para reducir el tiempo de ejecución de cada estimación y tomando n hasta 100.

Como el tiempo de ejecución depende del Y que tiene un ruido aleatorio, la gráfica presenta un sesgo. Para evitar esto se ejecuta 10 veces para cada tamaño y se toma la mediana (representa de mejor manera a los datos en este caso) y se grafica tomando este en cuenta. Vease Fig. 6 para esto.

Problema 4. Hacer $p = 0.1n$, o sea, diez veces más observaciones que coeficientes en la regresión, ¿Cuál es la n máxima que puede manejar su computadora?

Fue capaz de ejecutar para $n = 10128$ con un tiempo de 610.720312569. Generando los siguientes Runtimewarnings

- RuntimeWarning: overflow encountered in double-scalars
- RuntimeWarning: overflow encountered in subtract
- RuntimeWarning: invalid value encountered in divide

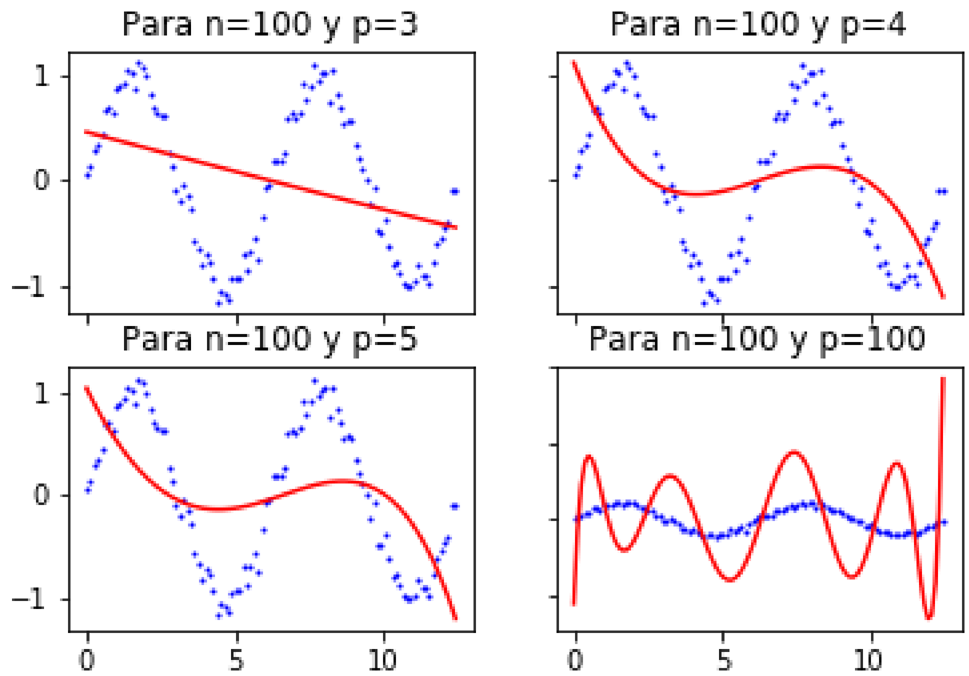

Figure 1: Estimación de mínimos cuadrados con 100 puntos y distinta p

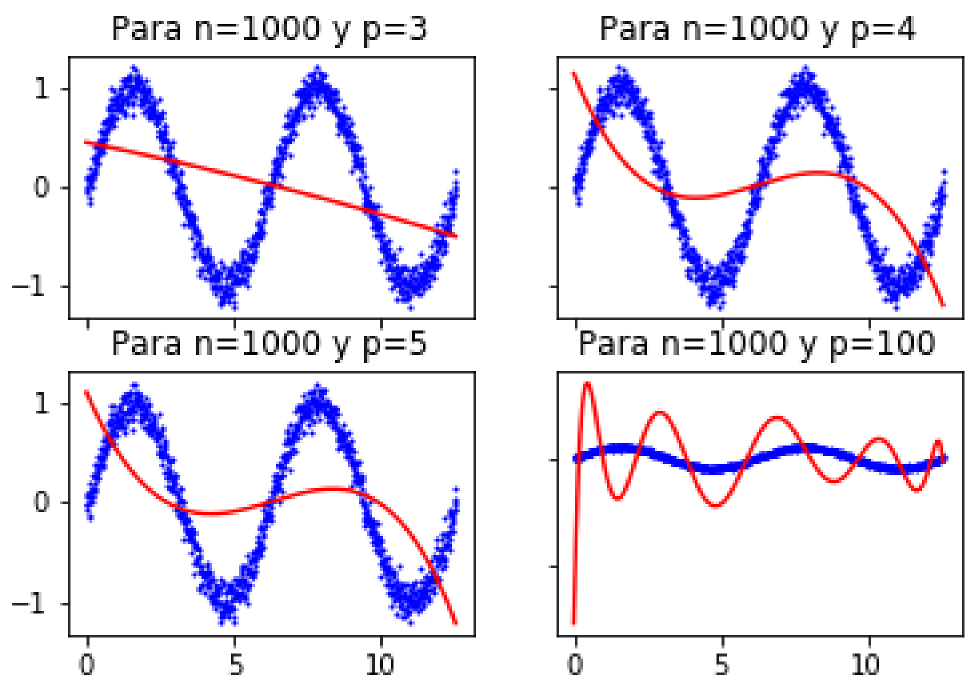

Figure 2: Estimación de mínimos cuadrados con 1000 puntos y distinta p

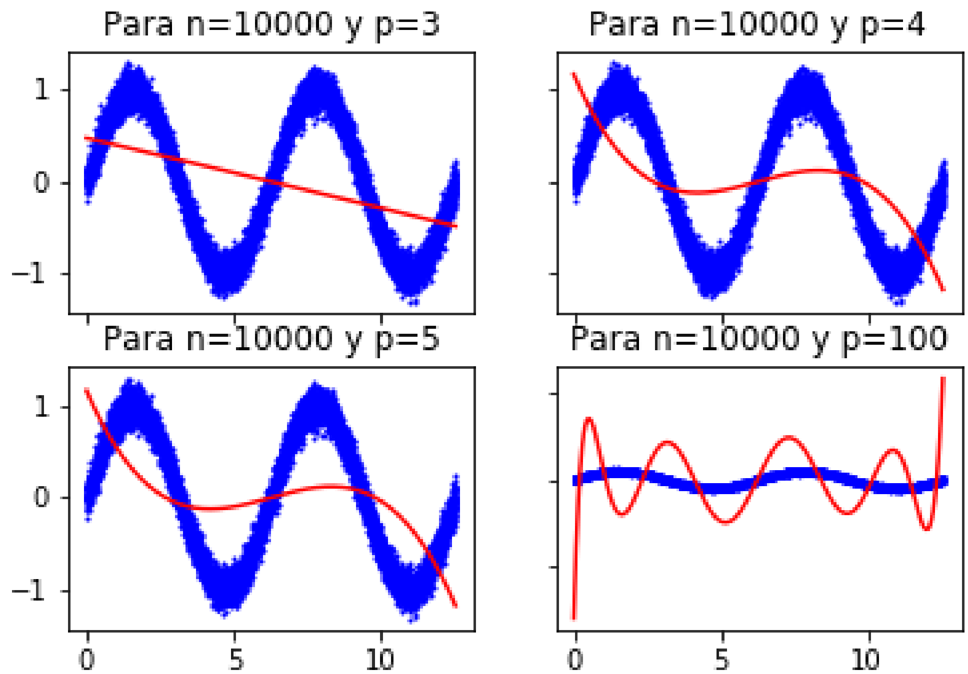

Figure 3: Estimación de mínimos cuadrados con 10000 puntos y distinta p

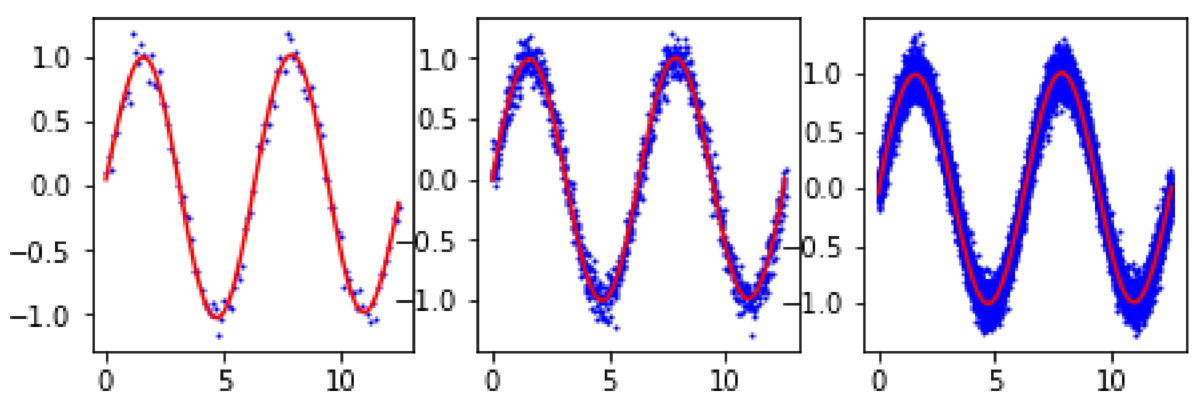

Figure 4: Estimación de mínimos cuadrados con $p = 10$ y distinta n

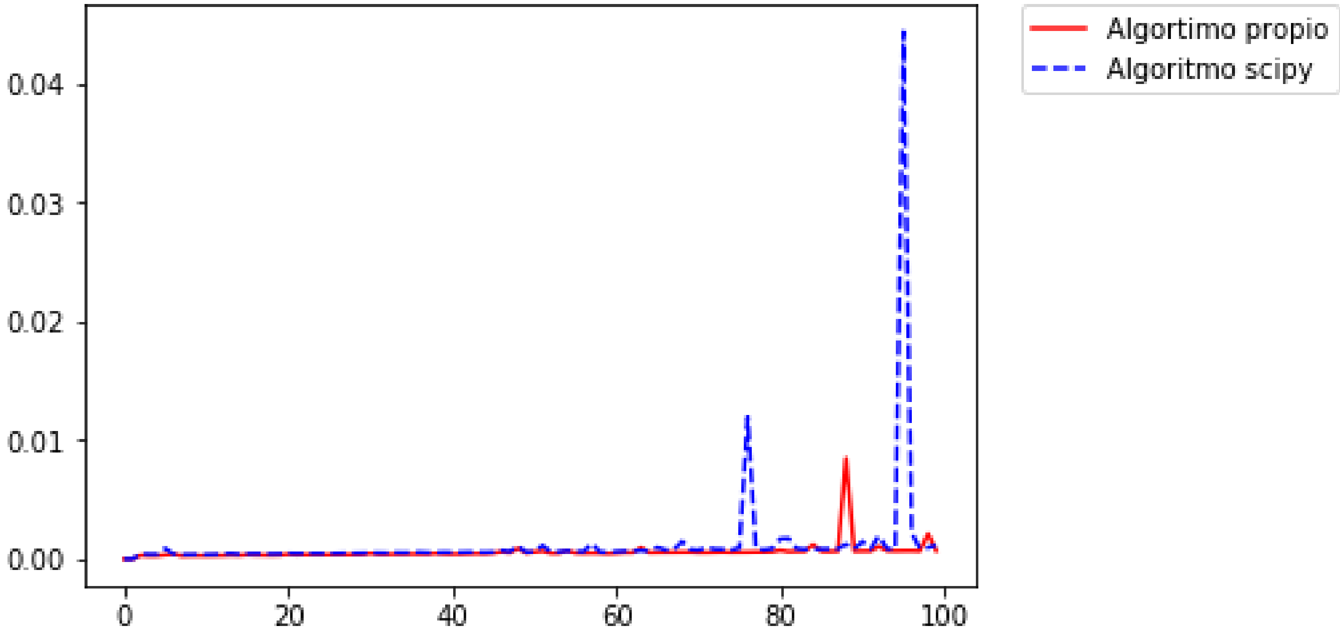

Figure 5: Tiempos de ejecucción para el algoritmo propio y el de scipy

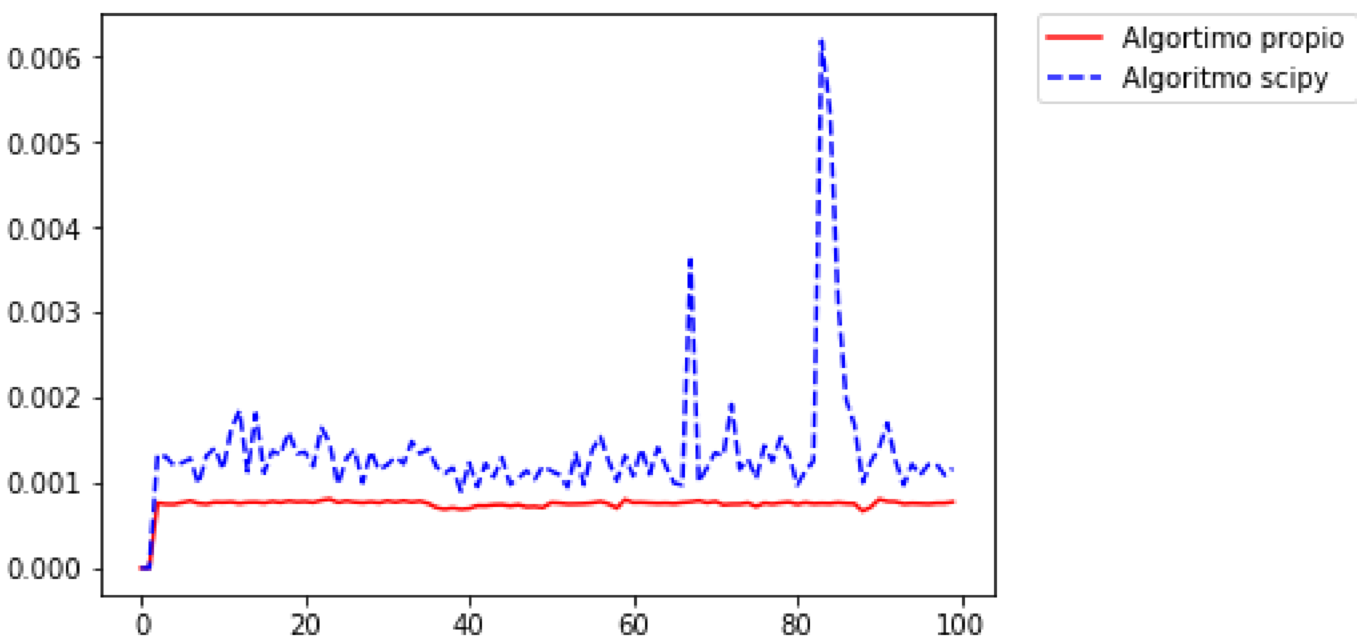

Figure 6: Tiempos medios de ejecucción para el algoritmo propio y el de scipy