Tarea 2
QR y mínimos cuadrados

Joel Chacón Castillo
Cómputo Científico

4 de septiembre de 2019

1. Gram-Schmidt

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

Comentarios

Se implementó el algorimto de Gram-Schidt modificado, la función que realiza este proceso es nombrada QR y utiliza como argumento la matriz a factorizar. Ésta función regresa la dos matrices Q y R en su forma reducida, es decir, $A \in \mathbb{C}^{m\times n}$, $Q \in \mathbb{C}^{m\times n}$ y $R \in \mathbb{C}^{n\times n}$. Se puede ejecutar un ejemplo con la función *Exercise1*.

2. Mínimos Cuadrados

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

Comentarios

Para resolver el problema de mínimos cuadrados partiendo de la factorización $QR = A$ (forma reducida), se procede considera el proyector ortogonal $P = QQ^*$ teniendo:

$$y = Pb = QQ^*b \tag{1}$$

Dado que $y \in range(A)$, el sistema $Ax = y$ tiene una solución exacta. Combinando la factorizació QR y la ecuación (1) se tiene lo siguiente

$$QR = x = QQ^*b \tag{2}$$

Así realizando una multiplicación matricial por la izquierda de (2) se obtiene lo siguiente

$$Rx = Q^*b \tag{3}$$

Finalmente el procedimiento para aplicar el estimador de mínimos cuadrados es el siguiente

1. Calcular la factorización con forma reducida $A = QR$.
2. Calcular el vector $Q^* = b$.
3. Resolver el sistema con forma triangular superior (Backward) $Rx = Q^*b$ para obtener x

La complejidad de implimentar este procedimiento está acotado aproximadamente $O(2mn^2 - \frac{2}{3}n^3)$. En este procedimiento se asume que ya se cuenta con la matriz A, está matriz tiene la forma Vandermonde como se indica a continuación

$$\begin{bmatrix} 1 & x_1 & & x_1^n \\ 1 & x_2 & .. & x_2^n \\ 1 & x_3 & .. & x_3^n \\ & . & . & . \\ 1 & x_m & . & x_m^n \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ .. \\ c_n \end{bmatrix} \approx \begin{bmatrix} y_0 \\ y_1 \\ .. \\ y_n \end{bmatrix} \tag{4}$$

De la ecuación (4) se desea calcular el vector de coeficientes (se podrían ver como pesos), una vez que se obtenga el vector de pesos se pueden estimar valors de la forma $\hat{Y} = c_0 + c_1(x) + c_2(x^2) = \sum_{i=0}^{p} = c_i x^i$. En la implementación la función que realiza este procedimiento es $Least_Squares(X, Y, p), recibe tres argumentos, el primero es el conjunto de datos X_{np}$, *el segundo es la etiqueta de cada dato* $Y_{n\times 1}$ *y el tercer argumento es el grado del polinomio. Se puede ejecutar un ejemplo con la función Exercise2.*

3. Mínimos Cuadrados - Experimento

Generar Y *compuesto de* $y_i = sin(x_i) + \epsilon_i$ *donde* ϵ_i *tiene distrbución* $N(0, 0{,}11)$ *para* $x_i = (4\pi i)/n$ *para* $i = 1, .., n$. *Hacer un ajuste de mínimos cuadrados a* Y, *con descomposición* QR *ajustando un polinomio de grado* $p - 1$

1. *Considerar los 12 casos:* $p = 3, 4, 6, 100$ *y* $n = 100, 1000, 10000$.
2. *Graficar el ajuste en cada caso.*
3. *Medir tiempo de ejecución de su algoritmo, comaprar con descomposición* QR *de scipy y graficar los resultados.*

Comentarios

3.1. Graficación de los modelos ajustados

Con base a lo anterior, en las figuras 1, 2, 3 se muestra el resultado de ajustar 100, 1000, 10000 datos de entrenamiento respectivamente. En la figura 1 se observa que con un número de 100 observaciones el ajuste con $p = 100$ *tiende a ser inestable. Conforme el número de*

observaciones aumenta a 10000 (figura 3) se observa más estabilidad en el ajuste del polinomio $p = 100$*, sin embargo visualmente se observa un sobre-ajuste, en resultado tratar de ajustar un polinomio de grado* $p = 100$ *no es lo mejor para este problema, personalmente probaría tratar de poner una restricción que si se considerase un método de penalización con el lagrangiano resultaría en un término de regularización (para mayor información revisar el libro de Patter Recognition de Bishop). De todos las configuraciones, la mejor fue considerando* $p = 6$ *ya que se ajusta de forma más adecuada a los datos.*

3.2. Tiempos implementación vs Scipy

En la figura 4 se muestra el tiempo requerido en cada una de las 12 configuraciones, principalmente se observa que en los casos de mayor tiempo es en donde se tiene que ajustar un polinomio de grado 100*, esto es intuitivo ya que la dimensión del sistema (número de columnas n) que se requiere factorizar aumenta de forma lineal con respecto a el grado del polinomio. Además, considerando las primeras configuraciones, es decir, con polinomios de menor orden que* $p = 100$ *el tiempo requerido por la libreria de scipy es similar la implementación, sin embargo en el ajuste de polinomios mayores la librería de scipy es más eficiente. Un aspecto importante a observar es sobre revisar el número de condición de la matriz Vandemonde, ya que al aumentar el grado del polinomio (p) el número de consición aumenta significativamente, esto puede generar un problema numérico al realizar la factorización* QR*, en la figura 5 se observa que hay configuraciones en que el número de condición de la matriz es muy elevado.*

4. Experimento - Estabilidad

Hacer $p = 0{,}1n$*, es decir, diez veces más observaciones que coeficientes en la regresión. ¿Cuál es la n máxima que puede manejar su computadora?*

Comentarios

Como se menciona anteriormente, el valor n máximo que se puede utilizar en la computador es de 200*, después de eso existe un problema de overflow, en la figura 5 se observa el número de condición considerando diez veces más observaciones que coeficientes. Una forma de evitar este problema es reformulando el problema, una sugerencia se basa en reescalar, esto es generar* Y *compuesto de* $y_i = sin(x_i) + \epsilon_i$ *donde* ϵ_i *tiene distrbución* $N(0, 0{,}11)$ *para* $x_i = (4\pi i)/10000$ *para* $i = 1, .., n$*. El número de condición de esta reformulación se puede observar en la figura 6, donde se observa una mejor estabilidad en comparación al caso anterior. En la figura 7 se observan los tiempos en comparación al de la librería de scipy que parece ser mucho más eficiente.*

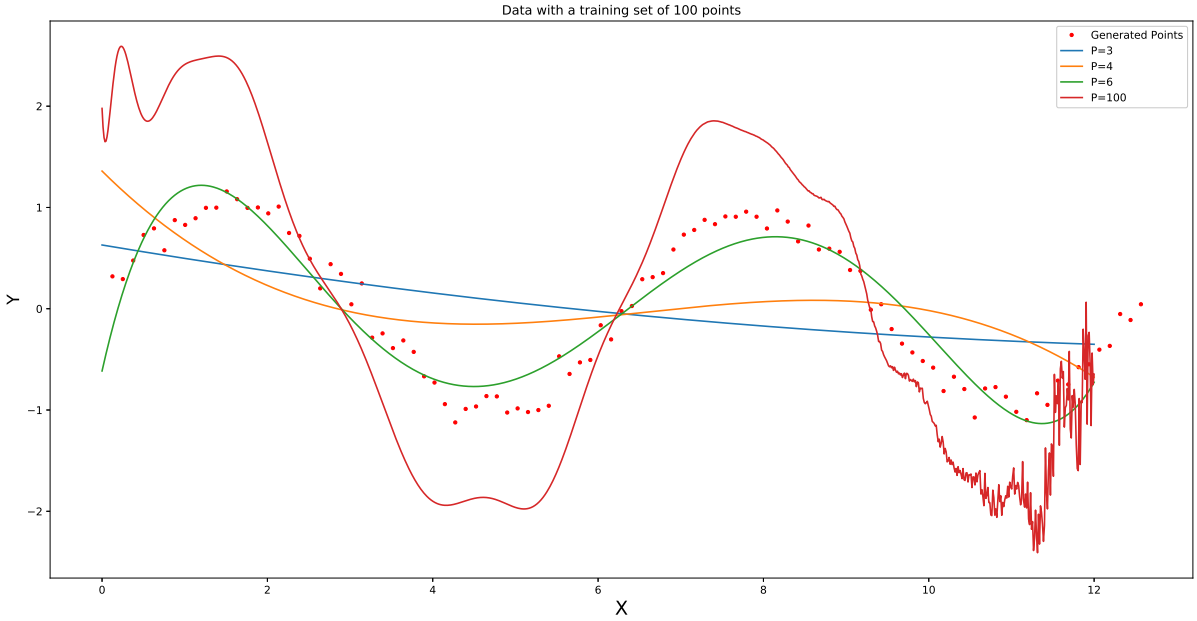

Figura 1: Ajuste con mínimos cuadrados por medio de la descomposición QR considerando 100 datos de entrenamiento y 1000 de prueba.

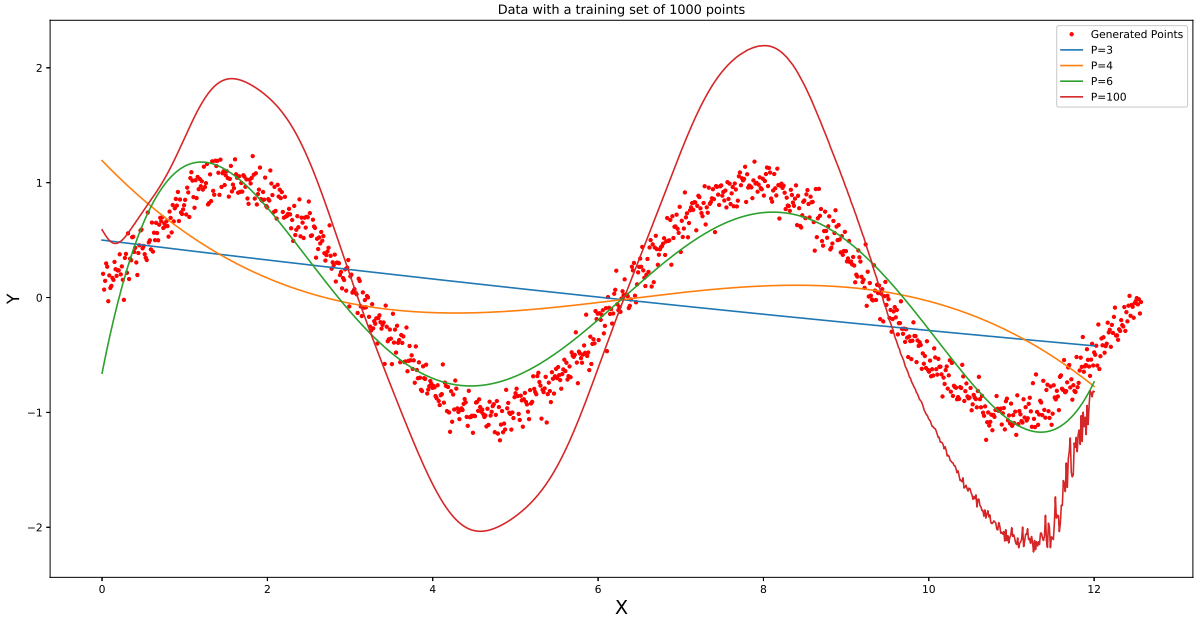

Figura 2: Ajuste con mínimos cuadrados por medio de la descomposición QR considerando 1000 datos de entrenamiento y 1000 de prueba.

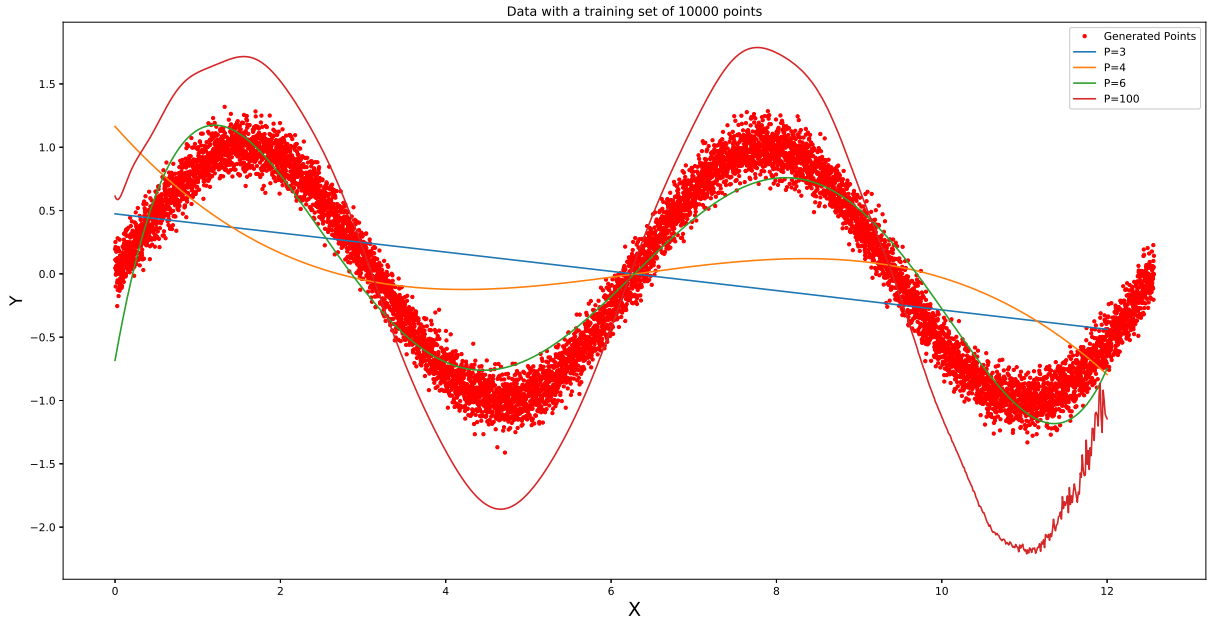

Figura 3: Ajuste con mínimos cuadrados por medio de la descomposición QR considerando 10000 datos de entrenamiento y 1000 de prueba.

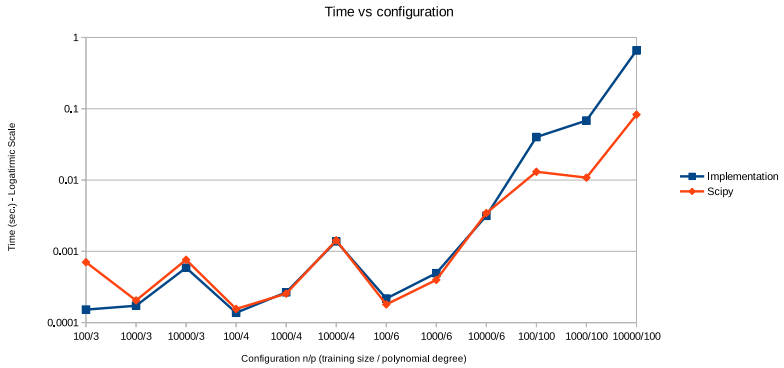

Figura 4: Ajuste con mínimos cuadrados por medio de la descomposición QR considerando 10000 datos de entrenamiento y 1000 de prueba.

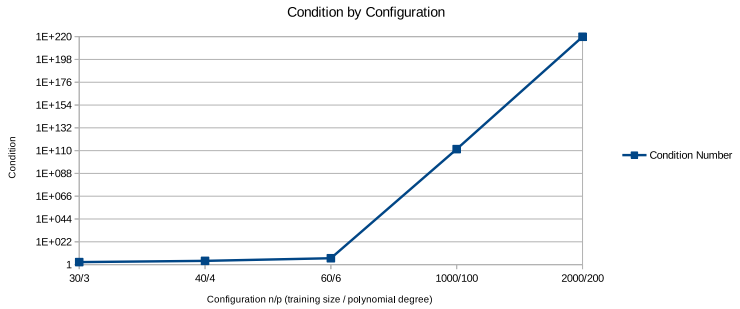

Figura 5: Ajuste con mínimos cuadrados por medio de la descomposición QR considerando 10000 datos de entrenamiento y 1000 de prueba.

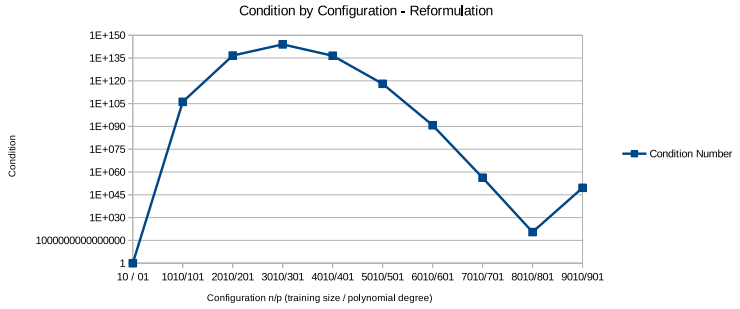

Figura 6: Ajuste con mínimos cuadrados por medio de la descomposición QR considerando 10000 datos de entrenamiento y 1000 de prueba.

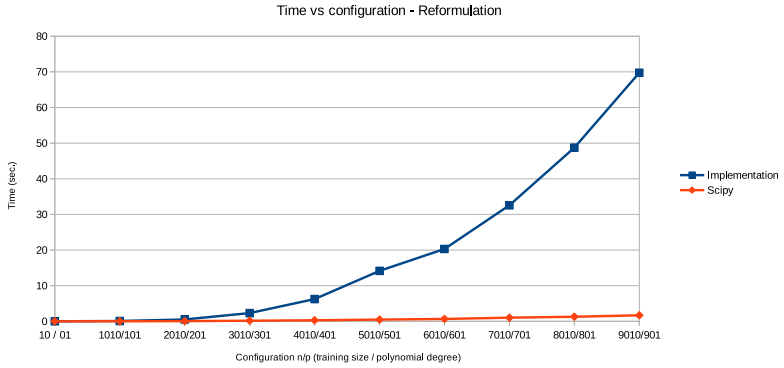

Figura 7: Ajuste con mínimos cuadrados por medio de la descomposición QR considerando 10000 datos de entrenamiento y 1000 de prueba.