Elementos de Cálculo Numérico / Cálculo Numérico

Segundo Cuatrimestre 2020

Noveno ejercicio computacional
07-06-21 al 14-06-21
Recuerde subir el archivo en formato ejercicioX_NOMBREAPELLIDO.py
Recuerde enviar su código al hacer consultas

En este ejercicio aplicaremos el método de cuadrados mínimos para realizar un ajuste de una curva a un conjunto de puntos $\{(x_i, y_i)\}_{i=1}^n$ mediante una combinación lineal de $\phi_1, \ldots, \phi_m$ funciones conocidas. El problema de cuadrados mínimos consiste en encontrar un vector de coeficientes c tal que:

$$\underbrace{\begin{bmatrix} \phi_1(x_1) & \ldots & \phi_m(x_1) \\ \phi_1(x_2) & \ldots & \phi_m(x_2) \\ \vdots & \ddots & \vdots \\ \phi_1(x_n) & \ldots & \phi_m(x_n) \end{bmatrix}}_{A \in \mathbb{R}^{n \times m}} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}}_{c \in \mathbb{R}^m} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{Y \in \mathbb{R}^n}$$

Como suele ocurrir que $m \ll n$, el sistema $Ac = Y$ podría no tener solución. Por lo tanto, se busca $c \in \mathbb{R}^m$ tal que $\|Ac - Y\|_2$ sea mínima. Como hemos visto en clase:

$$c \text{ minimiza } \|Ac - Y\|_2 \iff A^T Ac = A^T Y$$

Para simplificar la notación, llamamos B a la matriz $A^T A$. En este trabajo, implementarems el método de Cuadrados Mínimos para dos tipos de modelos:

- Polinomial: $c_1 + c_2 x + c_3 x^2 + \cdots + c_m x^{m-1}$
- Senoidal: $c_1 sen(x) + c_2 sen(2x) + \cdots + c_m sen(mx)$

Comience importando las librerías que utilizaremos:

```
import numpy as np
import numpy.linalg as npl
import pandas as pd
import matplotlib.pyplot as plt
```

A Construya una función que reciba un vector con valores `x`, un número entero `m` y un argumento `tipo` y construya una matriz `A` tal que:

   1 Si `tipo` es `polinomial` entonces $A_{ij} = x_i^j$ (x_i elevado a j), donde j se mueve entre 0 y $m - 1$.

   2 Si `tipo` es `senoidal` entonces $A_{ij} = sin((j + 1)x_i)$, donde j se mueve entre 0 y $m - 1$.

   Puede usar el siguiente modelo:

```
def matriz_A(x,m,tipo):
    nx = # COMPLETAR #
    A = np.zeros((nx,m))
    if tipo=='polinomial':
        for i in range(nx):
            for j in range(m):
                A[i][j] = ## COMPLETAR ##
    elif tipo=='senoidal':
        for i in range(nx):
            for j in range(m):
                A[i][j] = ## COMPLETAR ##
    return(A)
```

Chequee esta función, usando por ejemplo, `m` $= 3$, `x = np.array([0,2])` para ambos valores de `tipo`.

B Construya una función que reciba los vectores `x` e `y`, un número de coeficientes `m` y un argumento `tipo` como en el punto A, que construya la matriz y resuelva el problema de cuadrados mínimos. Considere usar las funciones `np.dot`, `npl.inv`, `np.transpose`. Puede usar el modelo:

```
def cuadrados_minimos(x,y,m,tipo):
    A = matriz_A(x,m,tipo)
    B = ## COMPLETAR ##
    c = ## COMPLETAR ##
    return(c)
```

Chequee esta función usando, por ejemplo `x=np.array([0,1,2])`, `y = np.array([0,1,4])`, `n=3` y `tipo='polinomial'`. El resultado debería ser muy próximo a $c = (0, 0, 1)$. *¿Por qué?*

C Construya una función que reciba un argumento `tipo` y un vector de coeficientes `c`, y retorne una función que calcule el valor de la aproximación por cuadrados mínimos para el dado tipo. Esta función deberá recibir como argumento un array `s` de valores y devolver un array `resultado` con los valores de ajuste en cada valor de `s`. Puede usar el siguiente modelo, inspirándose en lo aprendido en el ejercicio 7:

```
def genera_ajustador(c,tipo):
    def function(s):
        A = ## COMPLETAR ##
        resultado = ## COMPLETAR ##
        return (resultado)
    return (function)
```

D Para 10 puntos $\{x_i\}_{i=1}^{10}$ equiespaciados en $[-\frac{\pi}{2}, \frac{\pi}{2}]$ se tienen los siguientes datos:

y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}
1.19	0.84	0.825	0.56	0.376	-0.186	-0.663	-0.682	-0.801	-0.996

Tomando `m=2` encontrar la función de tipo `polinomial` y la de tipo `senoidal` que mejor aproximen a los datos en sentido de cuadrados mínimos. Graficar ambos resultados junto con los puntos de la tabla. ¿Cuál de los dos modelos le parece que aproxima mejor?

```
datos_x = #COMPLETAR
datos_y = [1.19,0.84,0.825,0.56,0.376,-0.186,-0.663,-0.682,-0.801,-0.996]
poli_coef = cuadrados_minimos(##COMPLETAR##)
sen_coef = cuadrados_minimos(##COMPLETAR##)

poli_fun = genera_ajustador(##COMPLETAR##)
sen_fun = genera_ajustador(##COMPLETAR##)

X = np.linspace(-np.pi/2, np.pi/2, 100)
plt.scatter(datos_x, datos_y)
plt.plot(##COMPLETAR##, ##COMPLETAR##, label='Ajuste polinomial')
plt.plot(##COMPLETAR##, ##COMPLETAR##, label='Ajuste senoidal')
plt.legend()
plt.savefig('ajuste_tabla.png')
```

E Adjunto al trabajo se encuentra un archivo .csv con los precios diarios de las acciones de Google, desde 2016 hasta 2018, inclusive. Considerando a `X` como el día y a `Y` como el precio de las acciones, para `m` igual a $2, 4, 6, 8$ hallar el modelo de tipo polinomial que mejor aproxima en sentido de cuadrados mínimos a los datos. Graficar los datos junto con los polinomios. ¿Cuál de estos ajustes representa mejor los datos?

```
datos = pd.read_csv('datos_google.csv')
X = np.array(datos['dia'])
Y = np.array(datos['precio de cierre'])

ms = [2, 4, 6, 8]
x_aju = np.linspace(np.min(X), np.max(X), 500)
plt.clf()
plt.plot(X, Y, 'o', ms=1)   # Ploteamos los datos
for m in ms:   # Para cada m hallamos el polinomio y lo graficamos
    coefs = cuadrados_minimos(##COMPLETAR##)
    poli_fun = genera_ajustador(##COMPLETAR##, 'polinomial')
    plt.plot(##COMPLETAR##, ##COMPLETAR##, label='m=' + str(m))
plt.legend()
plt.savefig('ajustes_datos.png')
```