Métodos Cuantitativos para Cs Sociales y Negocios
Prof. Sergio Pernice
Homework N°6

Junghanss, Juan Cruz | Marco, Tomas Guido | Ezequiel Lopez Mondo
Lisandro Austerlitz | Waldemar Diribarne

April 29, 2021

0.1 Homework N°6: Calcular el vector X dual y corroborar que el producto escalar con el vector Y resulte -0.5, es decir, igual al coeficiente beta de la regresión.

Para el desarrollo se considerará el código empleado en la clase práctica y a partir de este emplearemos las modificaciones y adhesiones necesarias para cumplir con el cálculo de la consigna.

[2]:
```
import numpy as np
import scipy.linalg as LG
```

[3]:
```
M = np.array([[1,1,1],[0,1,2]])
M
```

[3]:
```
array([[1, 1, 1],
       [0, 1, 2]])
```

[21]:
```
z = LG.null_space(M)
z
```

[21]:
```
array([[-5.55111512e-17],
       [-8.94427191e-01],
       [ 4.47213595e-01]])
```

[6]:
```
np.dot(M,z)
```

[6]:
```
array([[5.55111512e-17],
       [0.00000000e+00]])
```

[7]:
```
uno = M[[0],:]
x = M[[1],:]
x
```

[7]:
```
array([[0, 1, 2]])
```

[8]:
```
uno_d = np.concatenate((z.T,x), axis=0)
uno_d
```

[8]:
```
array([[ 0.40824829, -0.81649658,  0.40824829],
       [ 0.        ,  1.        ,  2.        ]])
```

[9]:
```
uno_dual_tilde = LG.null_space(uno_d)
uno_dual_tilde
```

[9]:
```
array([[-0.91287093],
       [-0.36514837],
       [ 0.18257419]])
```

[10]:
```
uno_dual = uno_dual_tilde / np.dot(uno, uno_dual_tilde)
uno_dual
```

[10]:
```
array([[ 0.83333333],
       [ 0.33333333],
       [-0.16666667]])
```

[11]:
```
y = np.array([[2],[0.5],[1]])
y
```

[11]:
```
array([[2. ],
       [0.5],
       [1. ]])
```

[12]:
```
np.dot(uno_dual.T,y)
```

[12]:
```
array([[1.66666667]])
```

De la manera anterior se demuestra que, para el vector dual del vector de unos del cálculo de la regresión, su producto escalar con el vector Y da como resultado el coeficiente de ordenada al origen o constante de la regresión lineal.

A continuación, tomando como referencia el procedimiento anterior, simplificaremos en menos celdas el desarrollo, pero ajustando los vectores que se usan en el cálculo para obtener nuestro resultado.

[11]:
```
# Definimos nuevamente nuestra matriz M; el vector x; uno para facilitar la␣
↪lectura del código sin chequear celdas anteriores
M = np.array([[1,1,1],[0,1,2]])
z = LG.null_space(M)
np.dot(M,z)
uno = M[[0],:]
x = M[[1],:]

# Nuestros cálculos serán:

x_d = np.concatenate((z.T,uno),axis=0)

x_dual_tilde = LG.null_space(x_d)

x_dual = x_dual_tilde / np.dot(x, x_dual_tilde)

print("El vector X dual es: \n",x_dual)
```

```
El vector X dual es:
 [[-5.00000000e-01]
 [-3.92523115e-17]
 [ 5.00000000e-01]]
```

[12]:
```
y = np.array([[2],[0.5],[1]])

print("El producto escalar entre X dual y el vector Y es: \n", np.dot(x_dual.
↪T,y))
```

```
El producto escalar entre X dual y el vector Y es:
 [[-0.5]]
```

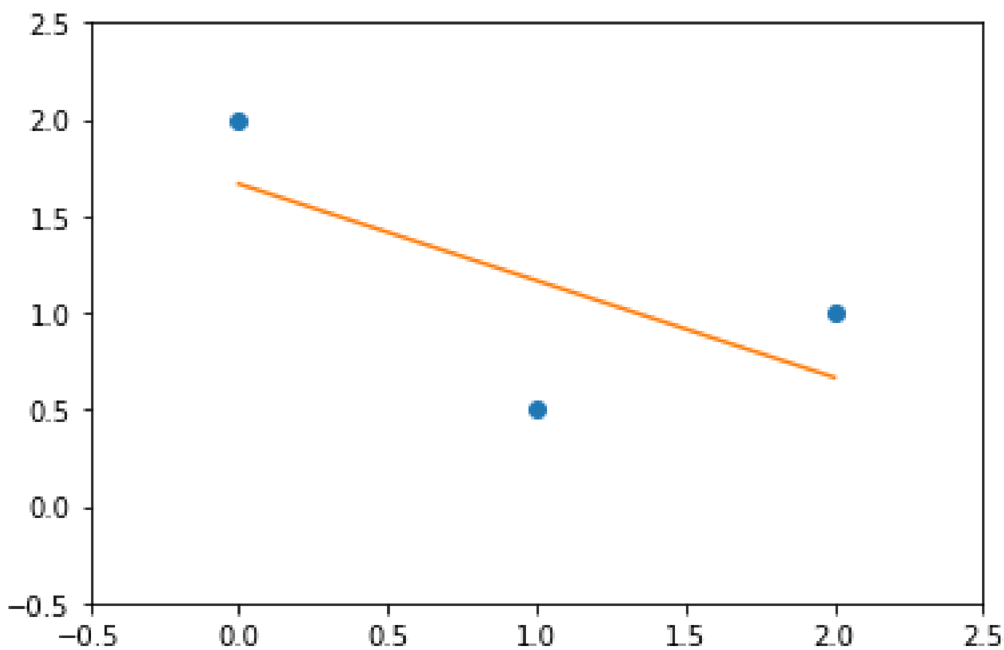
Así, hemos concluido que obtuvimos como resultado el coeficiente β de la regresión, que era:

$$Y_i = \alpha + \beta \cdot X_i + \epsilon_i$$
$$Y_i = 1.666 - 0.5 \cdot X_i$$

Recordemos nuestro problema de regresión lineal simple, que para el ejemplo considera 3 puntos y ajusta minimizando la suma del error cuadratico promedio una función lineal de la forma $f(x) = m \cdot x + b$

[25]:
```python
import matplotlib.pyplot as plt

x = np.array([0, 1, 2])

y = np.array([ 2, 0.5, 1])
plt.plot(x, y, 'o')

m, b = np.polyfit(x, y, 1)

plt.plot(x, m*x + b)
plt.xlim((-0.5,2.5))
plt.ylim((-0.5,2.5))
```

[25]: (-0.5, 2.5)



Se ve a rasgos generales que la función ajustada tiene la forma $Y_i = f(x) = 1.666 - \frac{X_i}{2}$, donde $\beta = 0.5$