Regressão Linear

Prof. Doherty Andrade

www.metodosnumericos.com.br

1 Regressão Linear

Neste Jupyter notebook vamos tratar da técnica de regressão linear. É uma importante técnica numérica para ajustar aos dados uma expressão afim: $y(x) = b_0 + b_1 x$.

A regressão simples ajusta aos dados (x_k, y_k) uma reta da forma $y = b_0 + b_1 x$. O termo constante b_0 é chamado de intercepto e o termo b_1 é chamado de coeficiente linear.

A regressão linear múltipla ajusta aos dados (X_k, y_k), onde $X_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,m}) \in R^m$ uma expressão do tipo $y = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_m x_m$.

Os coeficientes a_i são chamados de parâmetros do modelo ou coeficientes.

A técnica do mínimos quadrados apenas responde à pergunta: dentre todas as retas, qual delas melhor se ajusta aos dados? Aqui não fazermos nenhum estudo sobre a validade do ajuste.

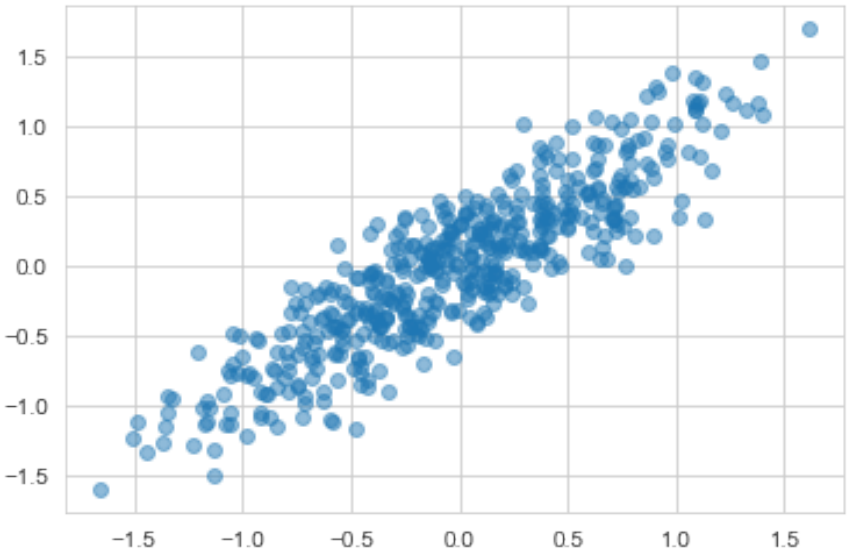
Vamos apresentar um exemplo no caso de regressão linear simples. Vamos gerar dados e obter a reta que melhor se ajusta aos dados.

Antes vamos chamar os pacotes que serão usados.

```
In [1]: import numpy as np
        import seaborn as sns
        sns.set_style("whitegrid")

        import matplotlib.pylab as plt
        %matplotlib inline
        plt.rc('figure', figsize = (12, 5))
```

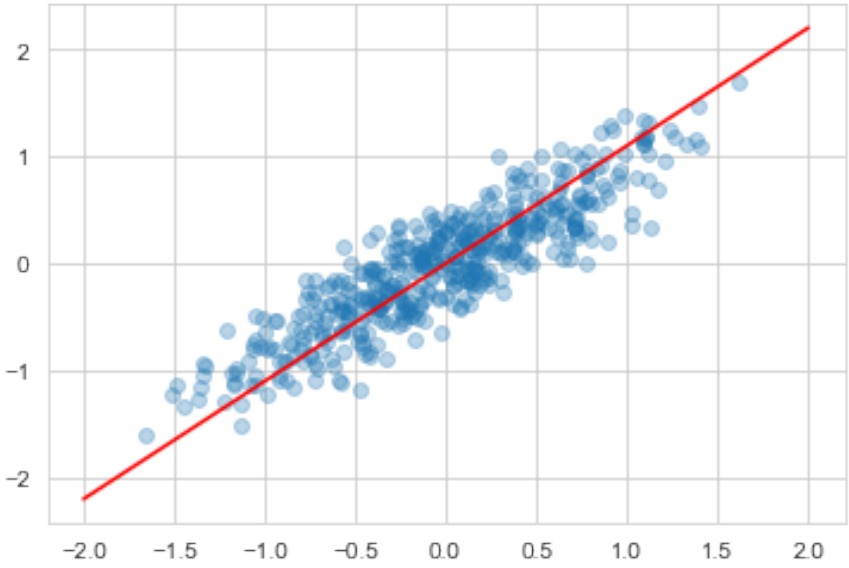
```
In [2]: X1 = np.random.randn(500, 2)  # gera uma amostra de pares
        A = np.array([[0.5, .3], [.3, 0.5]])
        X2 = np.dot(X1, A)
        plt.plot(X2[:, 0], X2[:, 1], "o", alpha = 0.5)
```

```
Out[2]: [<matplotlib.lines.Line2D at 0x22c7f0610f0>]
```



Agora vamos criar o modelo linear e plotar a reta que melhor se ajusta aos pontos.

```
In [3]: model = [0+1.1*x for x in np.arange(-2,3)]
        plt.plot(X2[:, 0], X2[:, 1], "o", alpha = 0.3);
        plt.plot(np.arange(-2,3), model, 'r');
        plt.show()
```



No caso de determinar a reta $y = \alpha_1 + \alpha_2 x$ que melhor se ajusta aos dados $(x_k, y_k), k = 1, 2, \ldots, m$ basta resolver o sistema de equações lineares 2×2 a seguir:

$$\begin{bmatrix} m & \sum x_k \\ \sum x_k & \sum x_k^2 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \sum y_k \\ \sum x_k y_k \end{bmatrix},$$

onde m é a quantidade de pontos.

2 Exemplo 1

Dados os pares (x_i, y_i) abaixo aproxime esses dados por uma reta,

x_i	-1	-0.75	-0.5	-0.25	0	0.25	0.5	0.75	0.9	1	1.5
y_i	3	3.5	4	4.5	5	5.5	6	6.5	6.8	7	8

O sistema linear acima nos fornece:

$$\begin{bmatrix} 11 & 2.40 \\ 2.40 & 6.81 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} = \begin{bmatrix} 59.8 \\ 25.620 \end{bmatrix}.$$

Resolvendo obtemos que $\alpha_0 = 5$ e $\alpha_1 = 2$. Segue que a reta procurada que melhor se ajusta aos dados é

$$\phi(x) = 2x + 5.$$

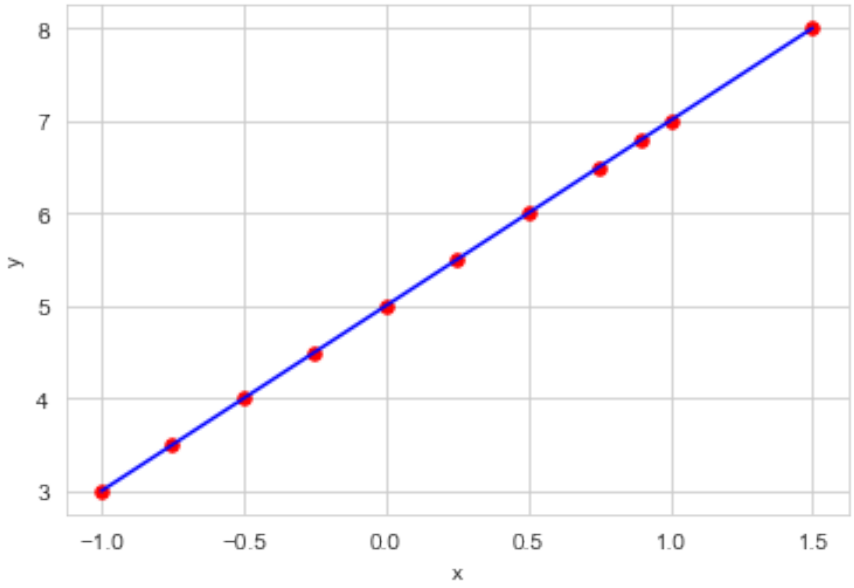
O script a seguir, resolve o sistema linear, apresenta a reta do ajuse linear e plota os pontos e reta.

3 Script Python

```
In [4]: import numpy as np
        import scipy.linalg as la
        import matplotlib.pyplot as plt

        def estimate_coef(x, y):
         # número de pontos
            n = np.size(x)

        #termos do sistema linear
            SS_x = np.sum(x)
            SS_y = np.sum(y)
            SS_xy = np.sum(x*y)
            SS_xx = np.sum(x*x)
        #resolvendo o sistema linear
            A = np.array([[n, SS_x],[SS_x, SS_xx]])
            c =  np.array([SS_y, SS_xy])
            b = la.solve(A,c)
            return(b[0], b[1])

        def plot_regression_line(x, y, b):
            plt.scatter(x, y, color = "r", marker = "o", s = 30)
```

```
        # reta soluçao
            y_pred = b[0] + b[1]*x

        # plotando a reta regressão
            plt.plot(x, y_pred, color = "b")

        #nomeando eixos
            plt.xlabel('x')
            plt.ylabel('y')

        # mostrando o plot
            plt.show()

        def main():
                # observaçõess
                x = np.array([-1   , -0.75, -0.5   , -0.25 , 0    , 0.25 ,  0.5,   0.75,     0.9, 1
                y = np.array([3   ,   3.5,   4    ,    4.5,    5,   5.5,    6,    6.5,     6.8,  7

                # estimando os coeficientes
                b = estimate_coef(x, y)
                print("Os coeficientes estimados são:", 'b_0=', b[0]," e b_1=", b[1])
                # plotando a reta de regressão
                plot_regression_line(x, y, b)

        if __name__ == "__main__":
                main()
```

```
Os coeficientes estimados são: b_0= 4.999999999999999  e b_1= 2.0000000000000004
```



Mesmo procedimento, mais enxuto e sem o gráfico.

```
In [5]: import numpy as np
        import scipy.linalg as la
        import matplotlib.pyplot as plt

        def MMQLinear(x, y):
            n = np.size(x)
            SS_x = np.sum(x)
            SS_y = np.sum(y)
            SS_xy = np.sum(x*y)
            SS_xx = np.sum(x*x)
        #resolvendo o sistema linear
            A = np.array([[n, SS_x],[SS_x, SS_xx]])
            c =  np.array([SS_y, SS_xy])
            b = la.solve(A,c)
            #return(b[0], b[1])
            print('Os coeficientes são: b[0]=', b[0],'e b[1]=', b[1])

In [6]: x = np.array([-1  , -0.75, -0.5  , -0.25 , 0    , 0.25 ,  0.5,   0.75,     0.9, 1 ,1.5])
        y = np.array([3   ,   3.5,   4   ,    4.5,    5,   5.5,    6,    6.5,     6.8,  7,  8])
        MMQLinear(x, y)

Os coeficientes são: b[0]= 4.999999999999999 e b[1]= 2.0000000000000004


In [ ]:
```