Métodos de Estatística Aplicada com Python

Aula 10

Carlos Góes[1]

[1]Pós-Graduação em Ciência de Dados
Instituto de Educação Superior de Brasília

2017

Sumário

Sumário

Introdução à Regressão Linear

Intuição

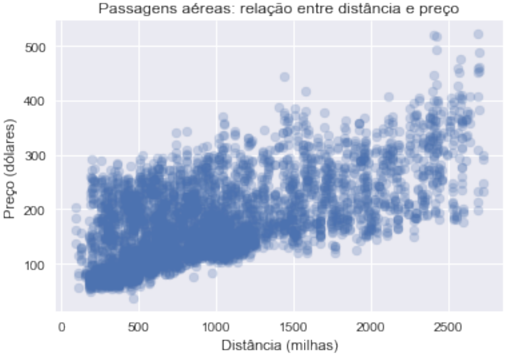
- Análise de regressão linear é um método de analisar associação entre variáveis
- Ela é um aprofundamento da análise de covariância
- O objetivo último da regressão linear é expressar uma variável dependente (y) como uma função de uma variável independente (x).

$$y = f(x)$$

Introdução à Regressão Linear

Intuição

- Como chegar nessa função?
- Você lembra da relação entre distâncias e preços em passagens aéreas?



Introdução à Regressão Linear

Intuição

- Carregue a base de dados (que está no formato do programa estatístico Stata) e veja o cabeçalho:

```
file = 'https://github.com/omercadopopular/cgoes/blob/master/StatsPython/
data/wooldridge/airfare.dta?raw=true'
df = pd.read_stata(file)
```

- Vamos excluir as variáveis que não interessam:

```
df = df.drop(['ldist', 'y98', 'y99', 'y00', 'lfare',
             'ldistsq', 'concen', 'lpassen'], axis=1)
```

- E alterar o nome de uma variável para português:

```
df = df.rename(columns = {'fare':'preco'})
print(df.head())
```

Introdução à Regressão Linear

Intuição

- Sabemos que existe uma associação positiva entre as duas variáveis (quanto maior a distância, maior tende a ser o preço).
- Como?

```
np.corrcoef(df['preco'],df['dist'])
```

- Será que podemos representar os preços como uma função da distância?

$$preco = f(distância)$$

Introdução à Regressão Linear

Intuição

- Sabemos que existe uma associação positiva entre as duas variáveis (quanto maior a distância, maior tende a ser o preço).
- Como?

```
np.corrcoef(df['preco'],df['dist'])
```

- Será que podemos representar os preços como uma função da distância?

$$preço = f(distância)$$

Introdução à Regressão Linear

Definição

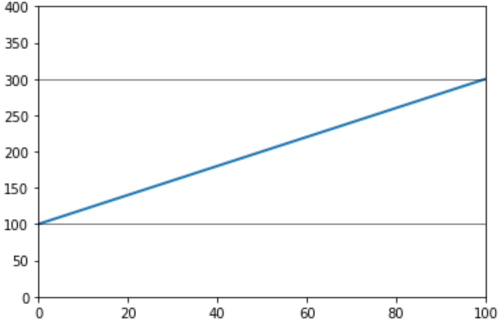
- Lembrando o que é uma função:

$$
\begin{aligned}
y &= f(x) = \alpha + \beta x \\
\alpha &= intercepto \quad (onde \quad cruza \quad a \quad vertical) \\
\beta &= \frac{quanto \quad y \quad muda}{quando \quad x \quad muda}
\end{aligned}
$$

Introdução à Regressão Linear

Definição

- Neste caso:



$$\alpha = 100$$
$$\beta = \frac{300 - 100}{100 - 0} = \frac{200}{100} = 2$$

Introdução à Regressão Linear

Definição

- Uma regressão linear traça uma reta que tenta aproximar o melhor valor de y para cada valor de x:

$$\hat{y} = \alpha + \beta x \tag{1}$$

- Nessa equação, $\hat{y}$ é o valor predito para y, para cada valor de x segundo essa função.

Introdução à Regressão Linear

Definição

- Se quisermos expressar y como uma função de x, precisamos incluir um erro de mensuração, que será definido pela diferença entre o valor real de y e o valor predito ($\hat{y}$).

$$y = \alpha + \beta x + u \quad (2)$$

$$u = y - \hat{y} \quad (3)$$

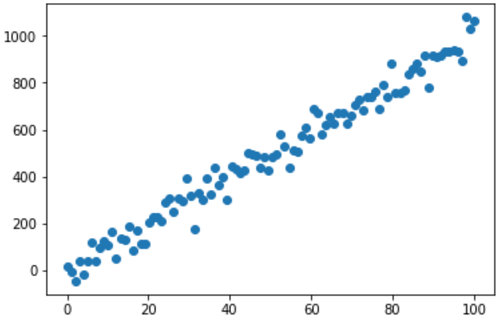
- O exercício de regressão linear consiste em encontrar os melhores valores possíveis para α e β.
- Como fazer isso?

Sumário

Encontrando os coeficientes

Intuição

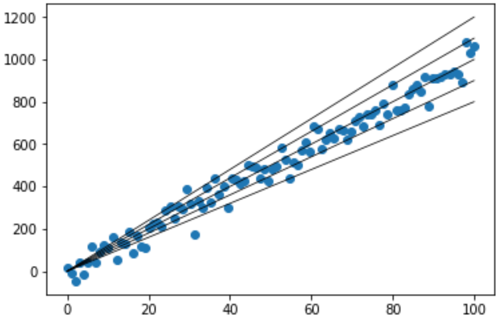
- Considere as variáveis abaixo, qual é a melhor função linear que descreve a relação entre elas?



Encontrando os coeficientes

Intuição

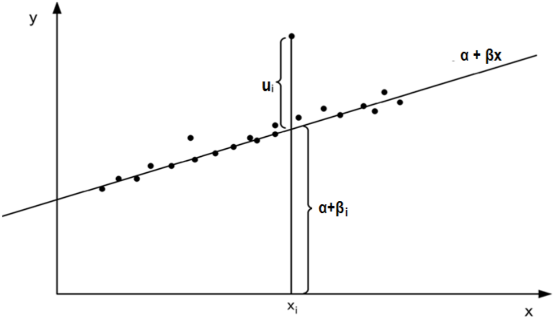
- Mesmo se soubessemos que $\alpha = 0$, como saber qual é o β que descreve a melhor função?



Encontrando os coeficientes

Mínimos quadrados

- O método padrão para alcançar a melhor reta é traçar diversas retas possíveis e ver qual é a soma dos quadrados dos resíduos em cada uma delas.
- O que são os resíduos?



Encontrando os coeficientes

Mínimos quadrados

- Matematicamente, para encontrar os resíduos, traçamos uma função específica que relaciona $x = \{x_1, x_2, \ldots, x_N\}$ e $y = \{y_1, y_2, \ldots, y_N\}$ e depois calculamos a diferença entre os valores observados de y e os valores preditos pela função.

$$
\begin{aligned}
\hat{y} &= \alpha + \beta x \\
u &= y - \hat{y} \\
u &= y - (\alpha + \beta x) \\
logo & \\
y &= \alpha + \beta x + u
\end{aligned}
$$

Encontrando os coeficientes

Mínimos quadrados

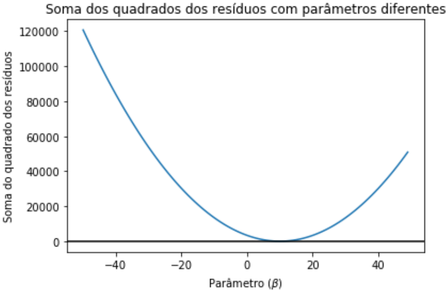
- E quais são os melhores valores para α e β?
- Aqueles que minimizem os quadrados dos resíduos u:

$$\min_{\alpha,\beta} \sum_{i=1}^{N} (u_i)^2 = \min_{\alpha,\beta} \sum_{i=1}^{N} [y_i - (\alpha + \beta x_i)]^2 \tag{4}$$

Encontrando os coeficientes

Mínimos quadrados

- O que isso quer dizer?
- Imagine que nós sabemos que $\alpha = 0$. Para cada valor de β é possível encontrar um valor dos quadrados dos resíduos, que resulta numa função.



- O problema matemático, portanto, é encontrar o ponto em que essa função tem o menor valor.

Encontrando os coeficientes

Mínimos quadrados

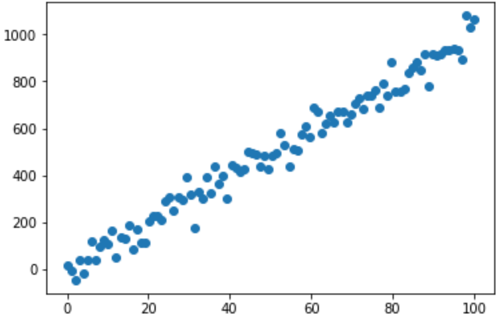
- Vamos, primeiro, simular uma série que é uma função de outra, mas que tem um resíduo aleatório inserido em cada observação:

```
# Sequência de x
x = np.linspace(0,100,100)

# y = função de x mais um erro aleatório.
y = x * 10 + np.random.normal(0,50,len(x))

## Plotar correlação

fig1 = plt.figure()
plt.scatter(x,y)
```

Encontrando os coeficientes

Mínimos quadrados

- Impressão da máquina:



Encontrando os coeficientes

Mínimos quadrados

- Depois, criamos uma função que calcula a soma dos quadrados dos resíduos, para determinado conjunto de x, y e um β (estamos assumindo que $\alpha = 0$):

```
## Cálculo da soma dos quadrados dos resíduos
def sum_sq_resid(beta, x, y, scale=1/10000):
    vec = [(y_i - beta * x_i) ** 2 for (y_i, x_i) in zip(y,x)]
    return scale * sum(vec)
```

- E calculamos o quadrado dos resíduos para diferentes β diferentes:

```
## Estimação da soma dos quadrados dos resíduos com parâmetros diferentes
sample = range(-50,50)

results = []
for i in sample:
    results.append(sum_sq_resid(i, x, y))
```

Encontrando os coeficientes

Mínimos quadrados

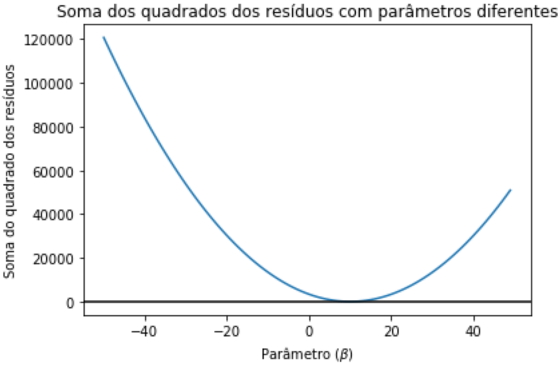
- Plotando:

```
# Plotar função
fig2 = plt.figure()
plt.plot(sample, results)
plt.axhline(np.min(results), color='black')
plt.title('Soma dos quadrados dos resíduos com parâmetros diferentes')
plt.xlabel(r'Parâmetro ($\beta$)')
plt.ylabel('Soma do quadrado dos resíduos')
plt.show()
```

Encontrando os coeficientes

Mínimos quadrados

- Impressão da máquina:



Encontrando os coeficientes

Solução via algoritmo de minimização

- Uma forma de encontrar o mínimo de uma função é traçar um algoritmo de minimização.
- Ele consiste dos seguintes passos
  - Traçar a função que você quer minimizar;
  - Escolher um β inicial;
  - Verificar qual é a derivada da função naquele ponto;
  - Aumentar (reduzir) o β se a derivada por negativa (positiva);
  - Repetir até a derivada chegar a zero (ponto de minimização local).

Encontrando os coeficientes

Solução via algoritmo de minimização

- Como é o algoritmo.

```
def minimize(x, y, alpha=0.001, num_inters=100):
    # Escolher um £\beta£ inicial
    beta = 0
    # Função de mínimos quadrados
    f = lambda beta: sum_sq_resid(beta, x, y)

    path = []
    betas = []
    # Repetir até o número máximo de iterações
    for i in range(0,num_inters):
        # Aumentar (reduzir) o £\beta£ se a derivada por negativa (positiva)
        beta = beta - alpha * derivative(f, x0=beta, dx=1e-10)
        # Armazenar os betas que vão se alterando
        betas.append(beta)
        # Armazenar a soma dos quadrados
        path.append(sum_sq_resid(beta, x, y))

    return(beta, betas, path)
```

Encontrando os coeficientes

Solução via algoritmo de minimização

- Rodando o algoritmo.

```
beta, betas, path = minimize(x,y)
```

- Imprimindo o resultado.

```
print(beta)
```

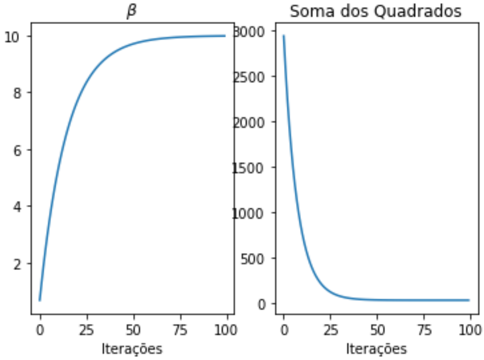
- Imprimindo o caminho do algoritmo.

```
f, ax = plt.subplots(1,2)
ax[0].plot(betas)
ax[0].set_title(r'$\beta$')
ax[0].set_xlabel('Iterações')
ax[1].plot(path)
ax[1].set_title('Soma dos Quadrados')
ax[1].set_xlabel('Iterações')
```

Encontrando os coeficientes

Solução via algoritmo de minimização

- Impressão da máquina:



Encontrando os coeficientes

Solução via algoritmo de minimização

- Alterativa: utilizar o statsmodels.

```
import statsmodels.formula.api as smf
import pandas as pd

df = pd.DataFrame({'x': x, 'y': y})

reg_sem_constante = smf.ols('y ~ x - 1', data=df).fit()

print(reg_sem_constante.summary())
```

Encontrando os coeficientes

Solução via cálculo numérico

- Outra maneira é utilizar cálculo para encontrar a solução.
- Em notação de álgebra linear: $Y = [y_1, \ldots, y_N]$, $X = [x_1, \ldots, x_N]$, $b = [\alpha, \beta]$

$$
\begin{aligned}
\min b \quad & (Y - X'b)^2 \\
2(Y - X'b)(-X') &= 0 \\
-2X'(Y - X'b) &= 0 \\
X'Y - X'Xb &= 0 \\
b &= \frac{X'Y}{X'X}
\end{aligned}
$$

Encontrando os coeficientes

Solução via cálculo numérico

- Isso tudo se converte em:

$$\beta = \frac{cov(X,Y)}{var(X)} \tag{5}$$

$$\alpha = \bar{y} - \beta\bar{x} \tag{6}$$

Encontrando os coeficientes

Solução via cálculo numérico

- Definir função para estimar os parâmetros.

```
def regressao(x, y):
    beta = (np.cov(x,y, ddof=1) / np.var(x, ddof=1))[0,1]
    alpha = np.mean(y) - np.mean(x) * beta
    return alpha, beta
```

- Imprimindo o resultado.

```
alpha_regressao, beta_regressao = regressao(x,y)

print(alpha_regressao, beta_regressao)
```

Encontrando os coeficientes
Solução via algoritmo de minimização

- Alterativa: utilizar o statsmodels.

```
import statsmodels.formula.api as smf
import pandas as pd

df = pd.DataFrame({'x': x, 'y': y})

reg_completa = smf.ols('y ~ x ', data=df).fit()

print(reg_completa.summary())
```

Sumário

Aplicação

Regressão linear univariada

- Voltamos ao nosso problema original.
- Qual será que é a melhor função que aproxima os preços, com base nas informações sobre distâncias de passagens?

Aplicação

Regressão linear univariada

- Rodamos e imprimimos a regressão

```
reg = smf.ols('preco ~ dist', data=df).fit()

print(reg.summary())
```

Aplicação

Regressão linear univariada

- Rodamos e imprimimos a regressão

```
reg = smf.ols('preco ~ dist', data=df).fit()

print(reg.summary())
```

Aplicação

Interpretação

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  preco   R-squared:                       0.389
Model:                            OLS   Adj. R-squared:                  0.389
Method:                 Least Squares   F-statistic:                     2923.
Date:                Wed, 29 Nov 2017   Prob (F-statistic):               0.00
Time:                        00:27:05   Log-Likelihood:                -25225.
No. Observations:                4596   AIC:                         5.045e+04
Df Residuals:                    4594   BIC:                         5.047e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    103.2614      1.643     62.868      0.000     100.041     106.481
dist           0.0763      0.001     54.063      0.000       0.074       0.079
==============================================================================
Omnibus:                      311.033   Durbin-Watson:                   0.529
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              364.797
Skew:                           0.676   Prob(JB):                     6.10e-80
Kurtosis:                       2.726   Cond. No.                     2.21e+03
==============================================================================
```

- Equação:

$$\hat{p} = 103.25 + 0.076 \cdot d \tag{7}$$

Aplicação

Interpretação

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  preco   R-squared:                       0.389
Model:                            OLS   Adj. R-squared:                  0.389
Method:                 Least Squares   F-statistic:                     2923.
Date:                Wed, 29 Nov 2017   Prob (F-statistic):               0.00
Time:                        00:27:05   Log-Likelihood:                -25225.
No. Observations:                4596   AIC:                         5.045e+04
Df Residuals:                    4594   BIC:                         5.047e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    103.2614      1.643     62.868      0.000     100.041     106.481
dist           0.0763      0.001     54.063      0.000       0.074       0.079
==============================================================================
Omnibus:                      311.033   Durbin-Watson:                   0.529
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              364.797
Skew:                           0.676   Prob(JB):                     6.10e-80
Kurtosis:                       2.726   Cond. No.                     2.21e+03
==============================================================================
```

Intervalo de confiança:

$$183.26 \pm 2 \cdot 1.643 = \{100.041, 106.481\} \quad (8)$$

$$0.076 \pm 2 \cdot 0.00141 = \{0.074, 0.079\} \quad (9)$$

Aplicação
Interpretação

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  preco   R-squared:                       0.389
Model:                            OLS   Adj. R-squared:                  0.389
Method:                 Least Squares   F-statistic:                     2923.
Date:                Wed, 29 Nov 2017   Prob (F-statistic):               0.00
Time:                        00:27:05   Log-Likelihood:                -25225.
No. Observations:                4596   AIC:                         5.045e+04
Df Residuals:                    4594   BIC:                         5.047e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    103.2614      1.643     62.868      0.000     100.041     106.481
dist           0.0763      0.001     54.063      0.000       0.074       0.079
==============================================================================
Omnibus:                      311.033   Durbin-Watson:                   0.529
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              364.797
Skew:                           0.676   Prob(JB):                     6.10e-80
Kurtosis:                       2.726   Cond. No.                     2.21e+03
==============================================================================
```

Estatística-t (ver se coeficiente é diferente de zero, medido em e.p.):

$$\frac{x - t_0}{e.p.(x)} = \frac{0.0763 - 0}{0.00141} = 54.06 \tag{10}$$

Aplicação

Interpretação

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  preco   R-squared:                       0.389
Model:                            OLS   Adj. R-squared:                  0.389
Method:                 Least Squares   F-statistic:                     2923.
Date:                Wed, 29 Nov 2017   Prob (F-statistic):               0.00
Time:                        00:27:05   Log-Likelihood:                -25225.
No. Observations:                4596   AIC:                         5.045e+04
Df Residuals:                    4594   BIC:                         5.047e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept    103.2614      1.643     62.868      0.000     100.041     106.481
dist           0.0763      0.001     54.063      0.000       0.074       0.079
==============================================================================
Omnibus:                      311.033   Durbin-Watson:                   0.529
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              364.797
Skew:                           0.676   Prob(JB):                     6.10e-80
Kurtosis:                       2.726   Cond. No.                     2.21e+03
==============================================================================
```

Probabilidade de encontrar uma estatística-t maior ou igual a essa se a hipótese nula de o coeficiente ser igual a zero for verdadeira

Aplicação
Interpretação

- A função que estimamos é essa:

$$\hat{p} = 103.25 + 0.076 \cdot d$$

- Nela, *prêço* são os valores preditos (esperados) pela equação, sendo que:

$$p = 103.25 + 0.076 \cdot d + u \tag{11}$$

- E, portanto:

$$u = p - \hat{p} \tag{12}$$

Aplicação

Interpretação

- Estimar $\hat{p}$.

```
# Imprimir só os coeficientes
print(reg.params)
# Criar valores preditos pela equação
preco_hat = reg.params[0] + reg.params[1] * df['preco']
# Alternativa: usar .predict()
preco_hat = reg.predict()
# Colocar no dataframe
df['preco_hat'] = preco_hat
```

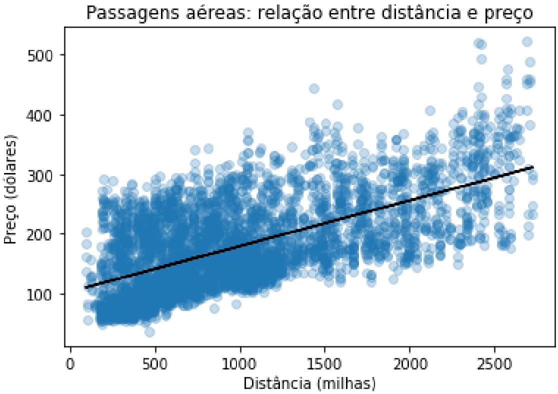
- Plotar resultado

```
plt.scatter('dist', 'preco', data=df, alpha=0.25)
plt.plot('dist', 'preco_hat',
         data=df, color='black')
plt.xlabel('Distância (milhas)')
plt.ylabel('Preço (dólares)')
plt.title('Passagens aéreas: relação entre distância e preço')
plt.show()
```

Aplicação

Interpretação

- Impressão da máquina



Aplicação

Interpretação

- Ou simplesmente usar o seaborn:

```
# Seaborn
import seaborn as sns

sns.regplot('dist', 'preco', data=df, line_kws={'color': 'black'})
plt.show()
```