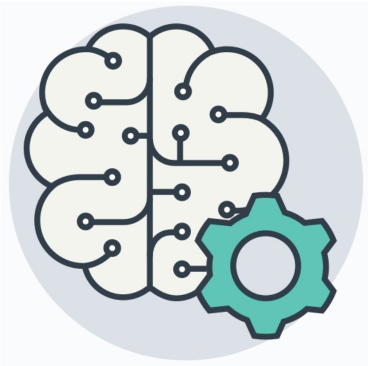
Aprendizado de Máquina

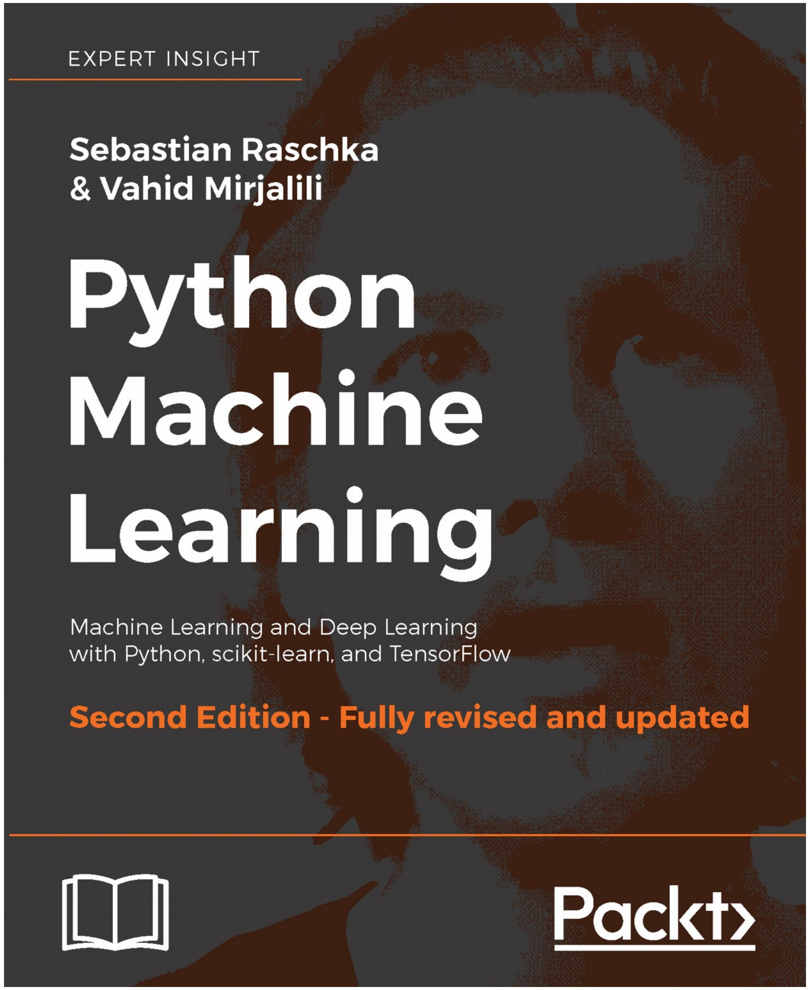
Regressão



Prof. Regis Pires Magalhães

regismagalhaes@ufc.br - http://bit.ly/ufcregis

Main Reference



Python Machine Learning

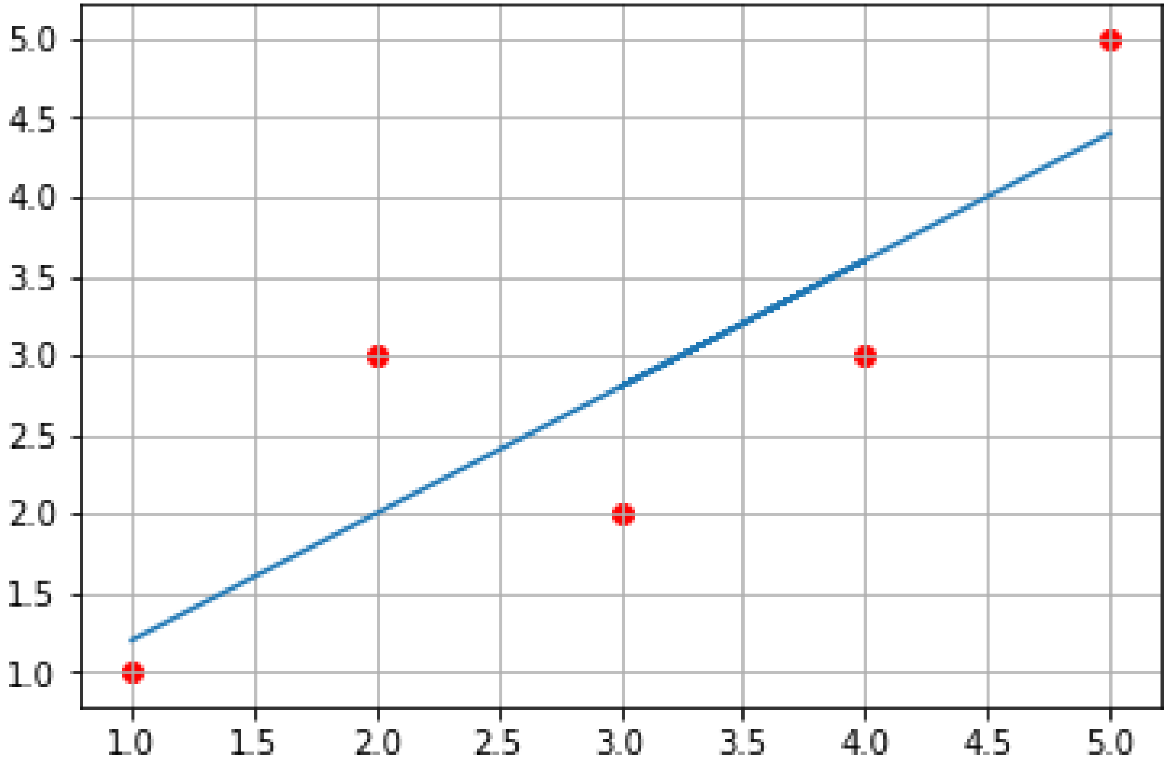
Chapter 10 - Predicting Continuous Target Variables with Regression Analysis

scikit-learn datasets

datasets.load_boston ([return_X_y])	Load and return the boston house-prices dataset (regression).
datasets.load_breast_cancer ([return_X_y])	Load and return the breast cancer wisconsin dataset (classification).
datasets.load_diabetes ([return_X_y])	Load and return the diabetes dataset (regression).
datasets.load_digits ([n_class, return_X_y])	Load and return the digits dataset (classification).
datasets.load_files (container_path[, …])	Load text files with categories as subfolder names.
datasets.load_iris ([return_X_y])	Load and return the iris dataset (classification).
datasets.load_linnerud ([return_X_y])	Load and return the linnerud dataset (multivariate regression).
datasets.load_mlcomp (name_or_id[, set_, …])	DEPRECATED: since the http://mlcomp.org/ website will shut down in March 2017, the load_mlcomp function was deprecated in version 0.19 and will be removed in 0.21.
datasets.load_sample_image (image_name)	Load the numpy array of a single sample image
datasets.load_sample_images ()	Load sample images for image manipulation.
datasets.load_svmlight_file (f[, n_features, …])	Load datasets in the svmlight / libsvm format into sparse CSR matrix
datasets.load_svmlight_files (files[, …])	Load dataset from multiple files in SVMlight format
datasets.load_wine ([return_X_y])	Load and return the wine dataset (classification).

Regressão Linear

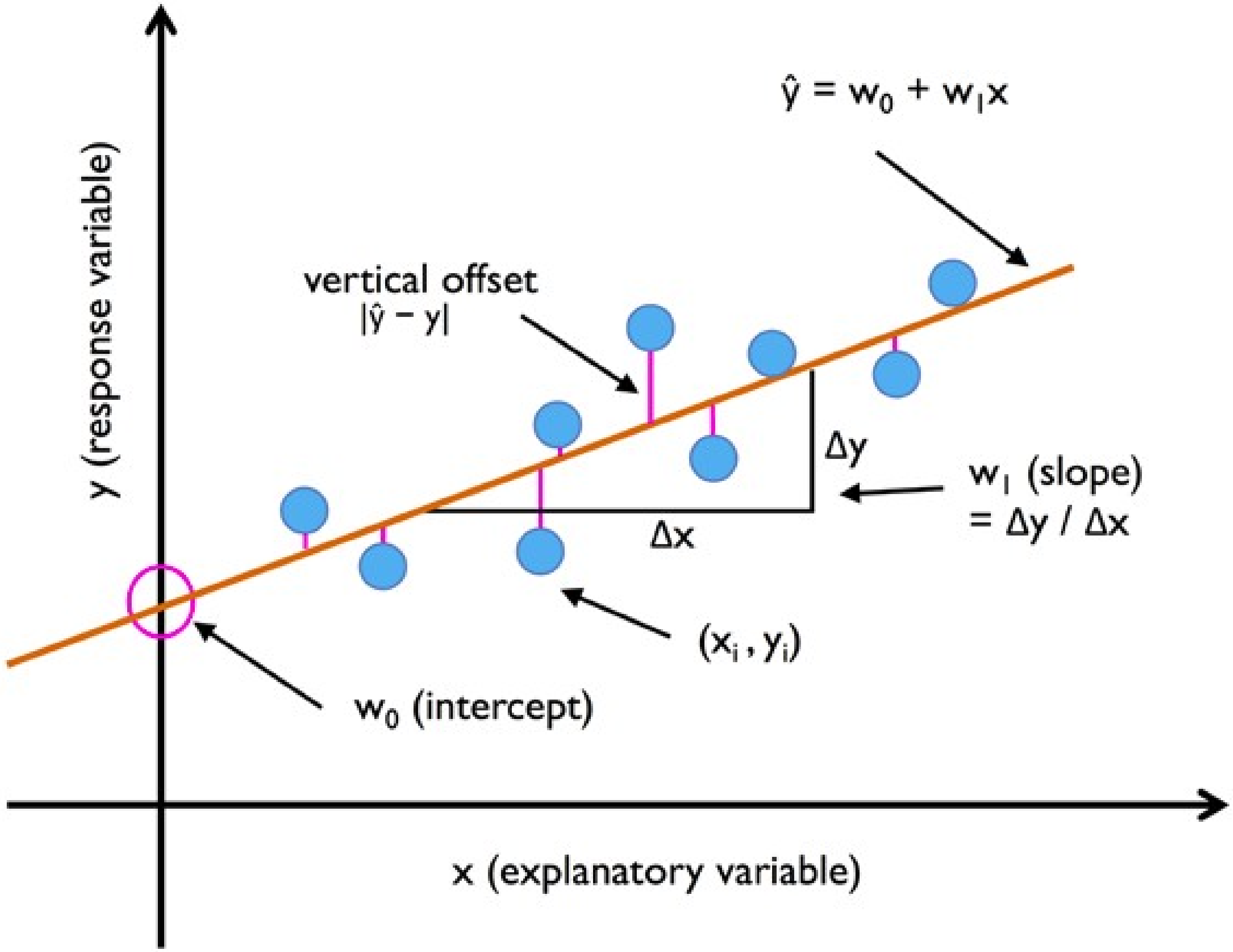
x	y
1	1
2	3
4	3
3	2
5	5



Simple Linear Regression

$$y = w_0 + w_1 x$$

Simple Linear Regression



Simple Linear Regression

- y = B0 + B1 × x

$$B1 = \frac{\sum_{i=1}^{n}(x_i - mean(x)) \times (y_i - mean(y))}{\sum_{i=1}^{n}(x_i - mean(x))^2}$$

$$B1 = \frac{covariance(x, y)}{variance(x)}$$

$$B0 = mean(y) - B1 \times mean(x)$$

$$mean(x) = 3$$
$$mean(y) = 2.8$$

$$B1 = \frac{8}{10}$$
$$B1 = 0.8$$

$$B0 = mean(y) - B1 \times mean(x)$$
$$B0 = 2.8 - 0.8 \times 3$$
$$B0 = 0.4$$

$$y = B0 + B1 \times x$$
$$y = 0.4 + 0.8 \times x$$

RMSE = 0.692820323

Simple Linear Regression

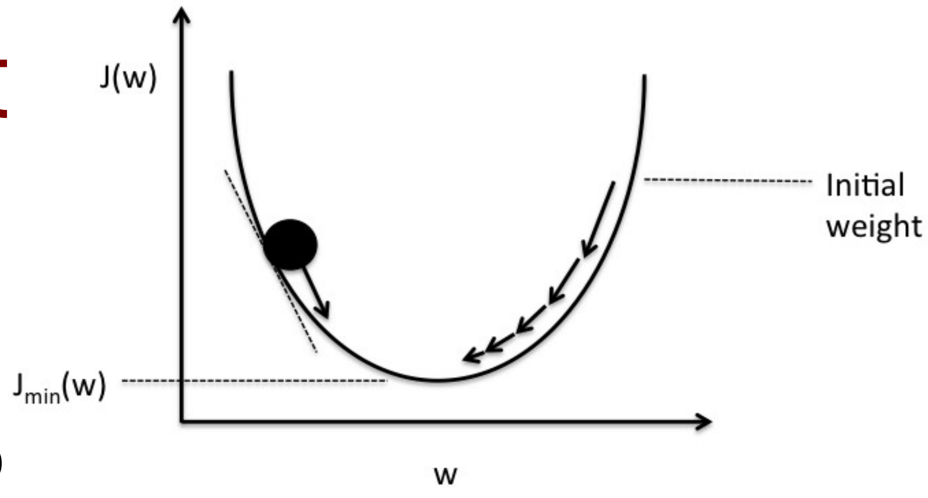
- Atalho

$$B1 = corr(x, y) \times \frac{stdev(y)}{stdev(x)}$$

$$B1 = 0.852802865 \times \frac{1.483239697}{1.58113883}$$

$$B1 = 0.8$$

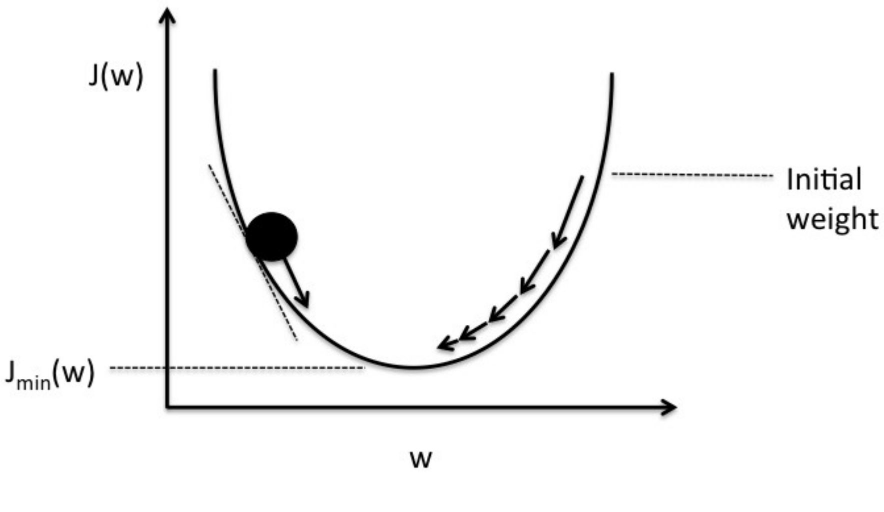
Gradient Descent



Schematic of gradient descent.

- Imagine uma bola rolando ao longo do gráfico de uma função de custo.
- A medida que a bola rola, ela segue a rota mais íngreme, eventualmente chegando ao fundo.
- Em resumo, é isso que ocorre com o gradiente descendente.

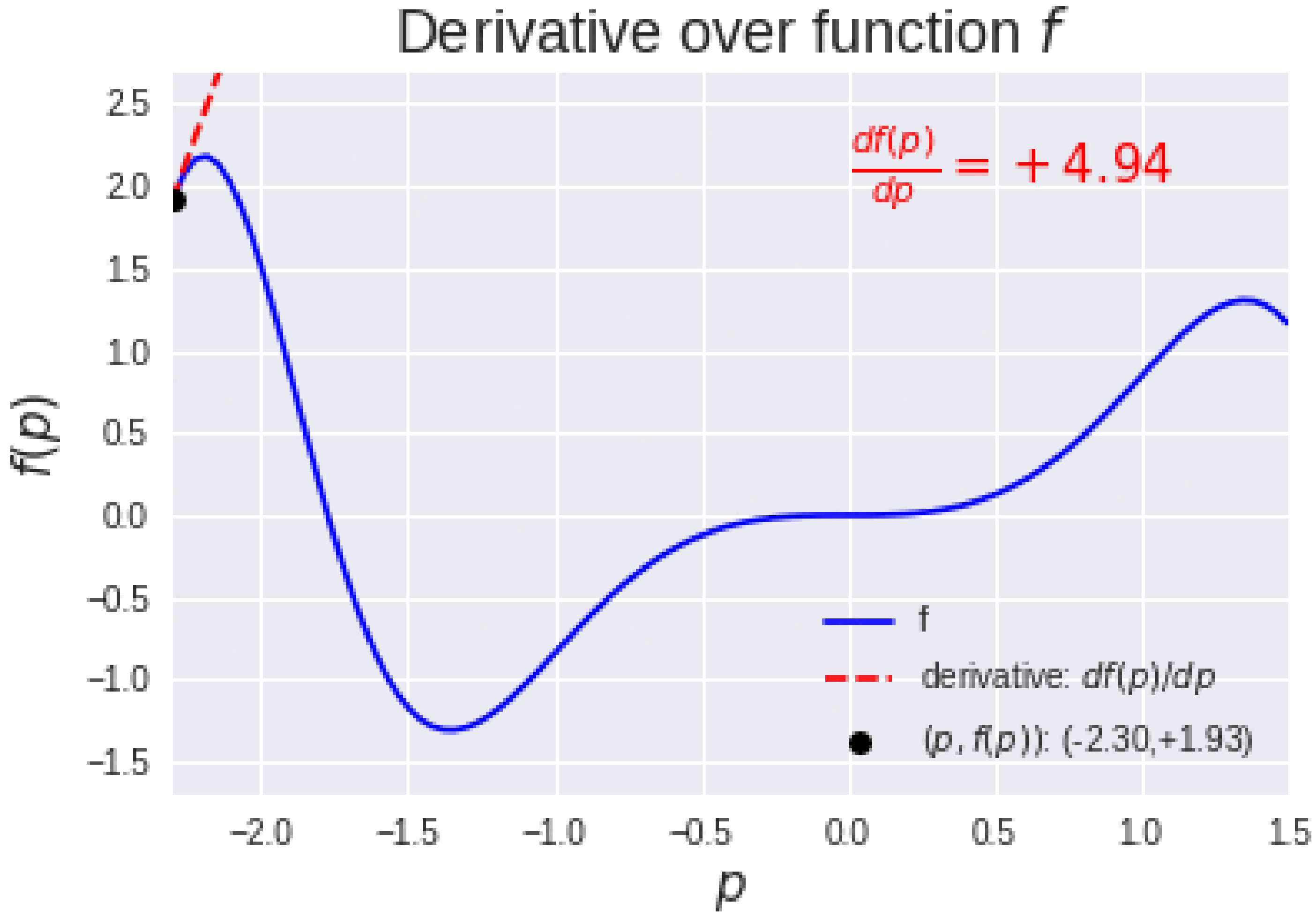
Gradient Descent



Schematic of gradient descent.

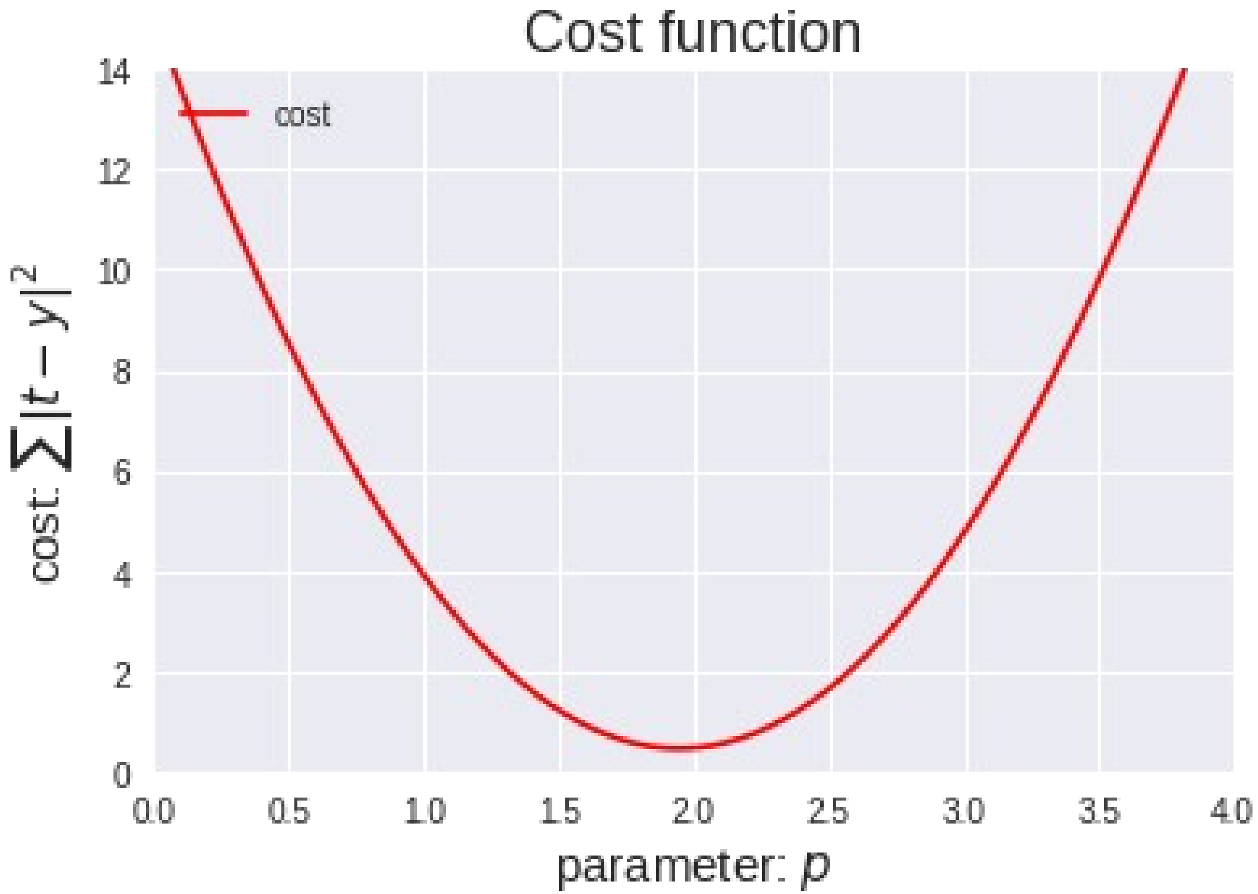
- Escolha um ponto no gráfico, encontre a direção que tem a inclinação mais íngrer naquela direção, e repita o processo.
- Eventualmente, encontraremos um mínimo da função de custo.
- E porque aquele ponto é o mínimo, ele possui os parâmetros necessários para desenhar nossa linha.

Derivada de uma função f



Função de custo

objetivo: minimizar o erro quadrado



Função de Erro

$y = b_1x + b_0$

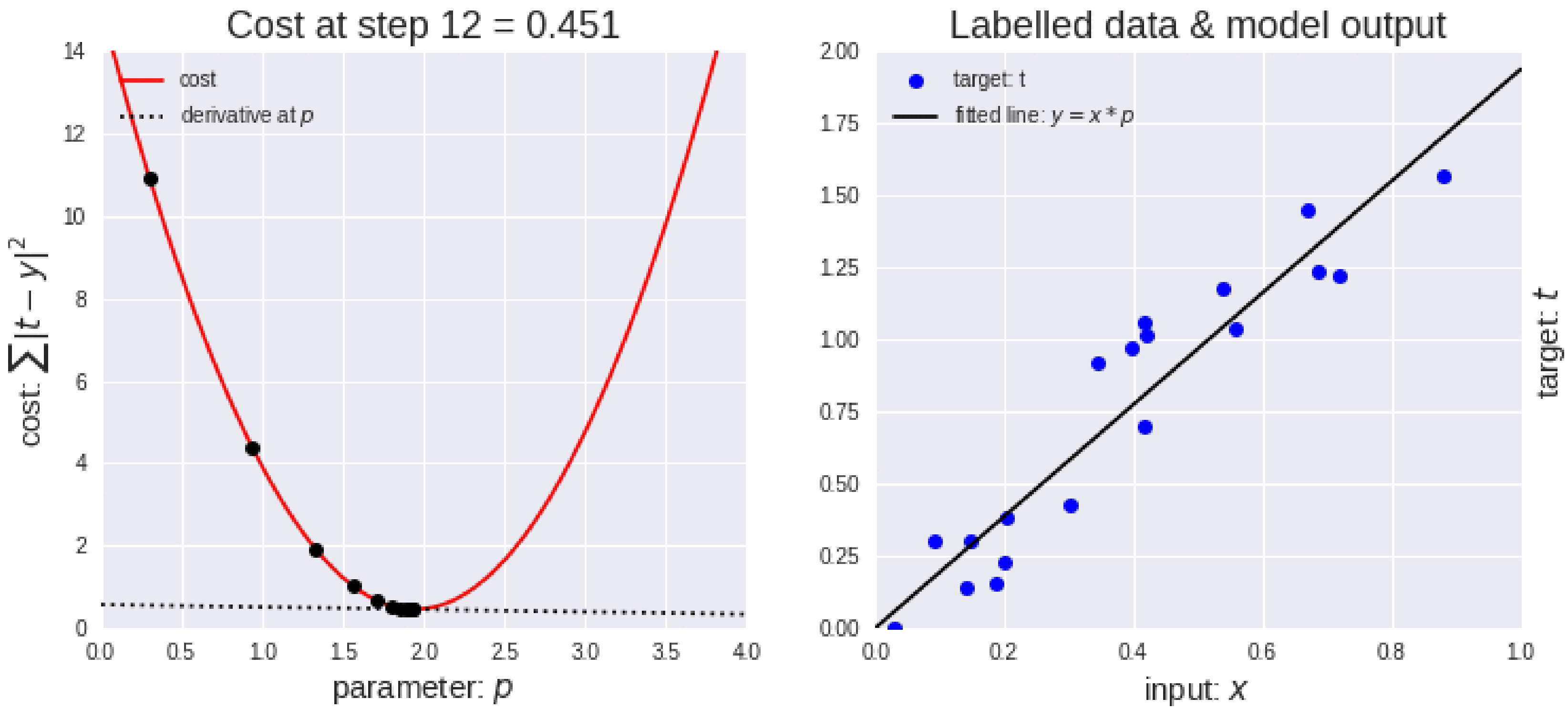
b_1 é a inclinação

b_0 é onde y é interceptado.

$$Error_{\beta_0,\beta_1} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\beta_1 x_i + \beta_0))^2$$

Minimizando a função de custo

$$Error_{\beta_0,\beta_1} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\beta_1 x_i + \beta_0))^2$$



Fonte: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

Minimizando a função de custo

$$Error_{\beta_0,\beta_1} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\beta_1 x_i + \beta_0))^2$$

É necessário calcular a derivada parcial para b0 e b1:

$$\frac{\partial}{\partial \beta_1} = \frac{2}{N}\sum_{i=1}^{N} -x_i(y_i - (\beta_1 x_i + \beta_0))$$

$$\frac{\partial}{\partial \beta_0} = \frac{2}{N}\sum_{i=1}^{N} -(y_i - (\beta_1 x_i + \beta_0))$$

- O erro diminui a cada interação.
- A direção do movimento em cada interação é calculada a partir das 2 derivadas parciais.

Fonte: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

Gradient Descent

```
def compute_error_for_line_given_points(b0, b1, x, y):
    N = len(y)
    totalError = 1/N * np.sum((y - (b1 * x + b0)) ** 2)
    return totalError

def step_gradient(b0_current, b1_current, x, y, learning_rate):
    N = len(y)
    b0_gradient = 2/N * np.sum(-(y - ((b1_current * x) + b0_current)))
    b1_gradient = 2/N * np.sum(-x * (y - ((b1_current * x) + b0_current)))
    new_b0 = b0_current - (learning_rate * b0_gradient)
    new_b1 = b1_current - (learning_rate * b1_gradient)
    return new_b0, new_b1

def gradient_descent_runner(x, y, b0, b1, learning_rate, num_iterations):
    for _ in range(num_iterations):
        b0, b1 = step_gradient(b0, b1, x, y, learning_rate)
    return b0, b1
```

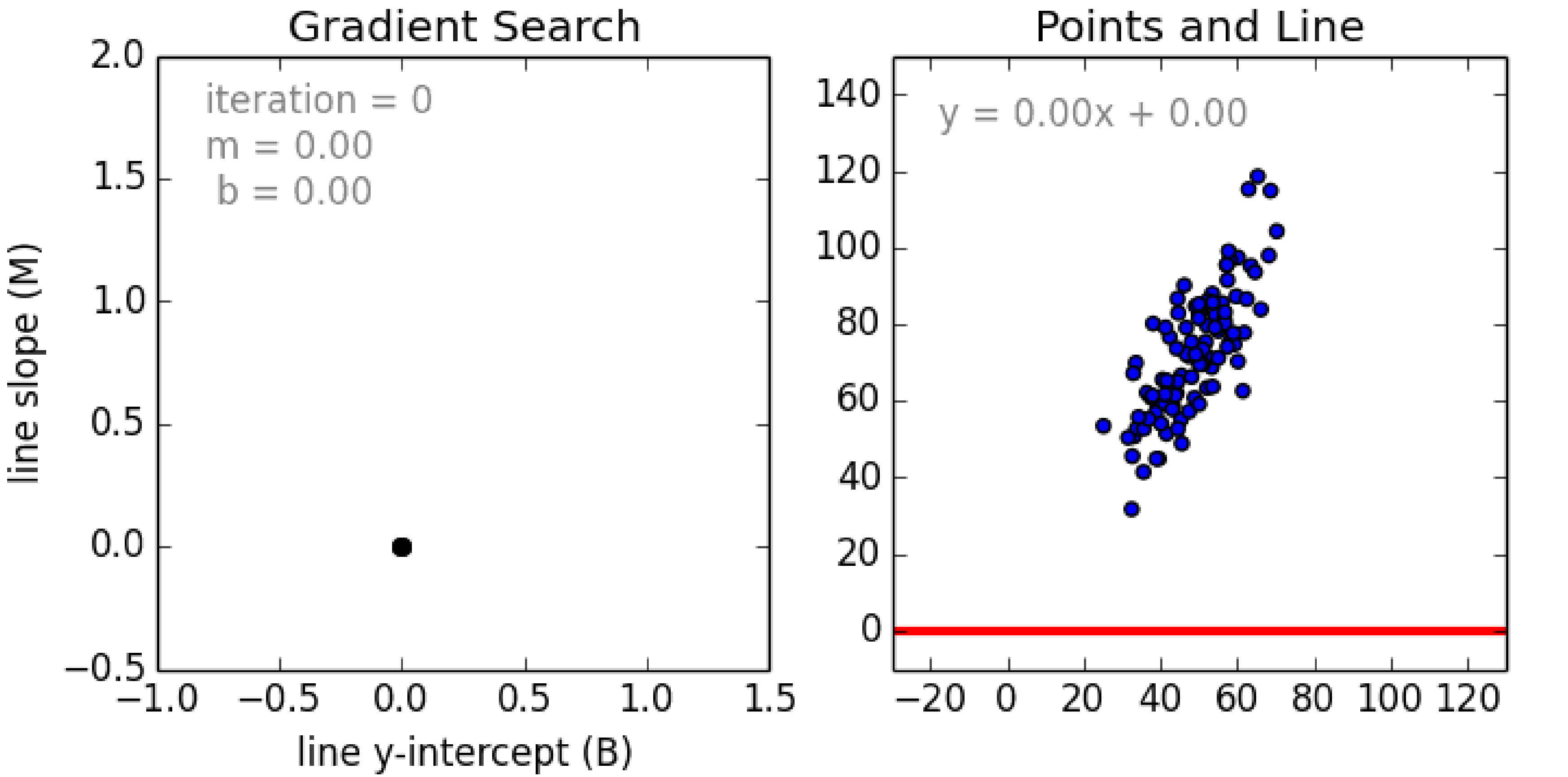
```
b0, b1 = gradient_descent_runner(x, y, initial_b0, initial_b1, learning_rate, num_iterations)
```

Gradient Descent

- Após 100000 iterações, obtemos
  - b0 = 4.247984440219184
  - b1 = 1.3959992655297515
  - error = 110.78631929745077

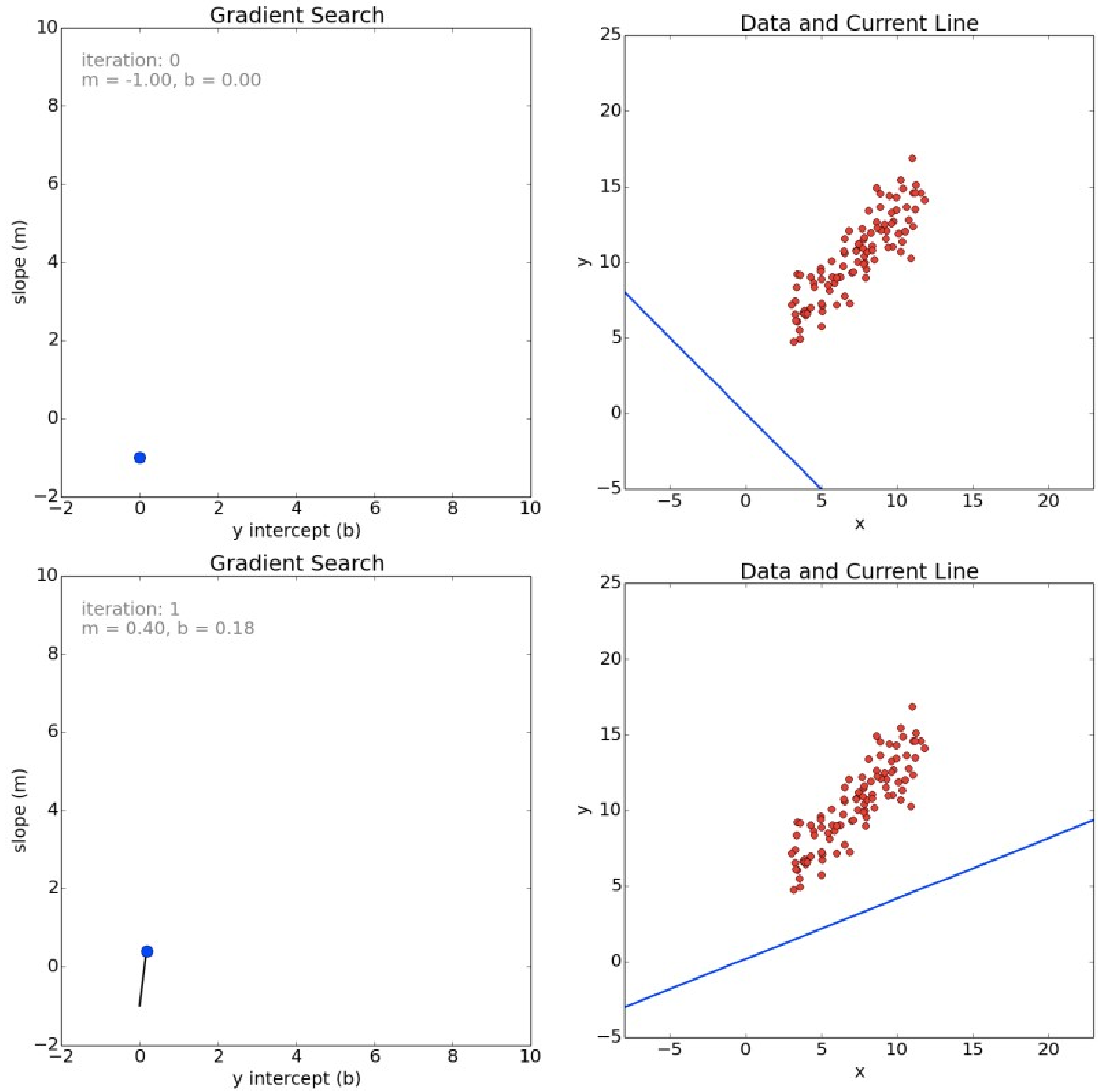
- Linear Regression do Scikit Learn
  - b0 = 7.991020982270399
  - b1 = 1.32243102
  - error = 110.25738346621316

Gradient Descent



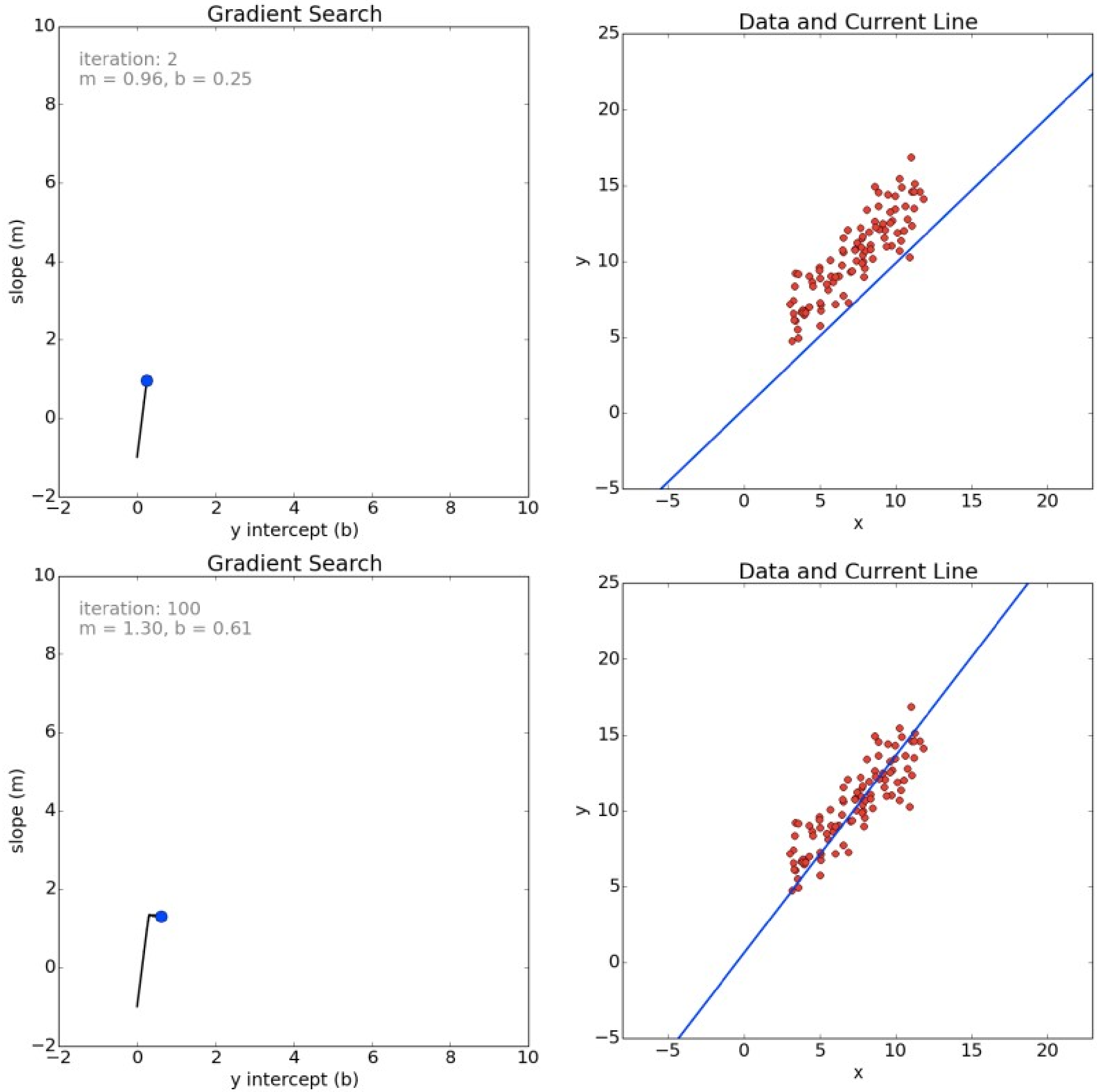
Fonte: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

Gradient Descent



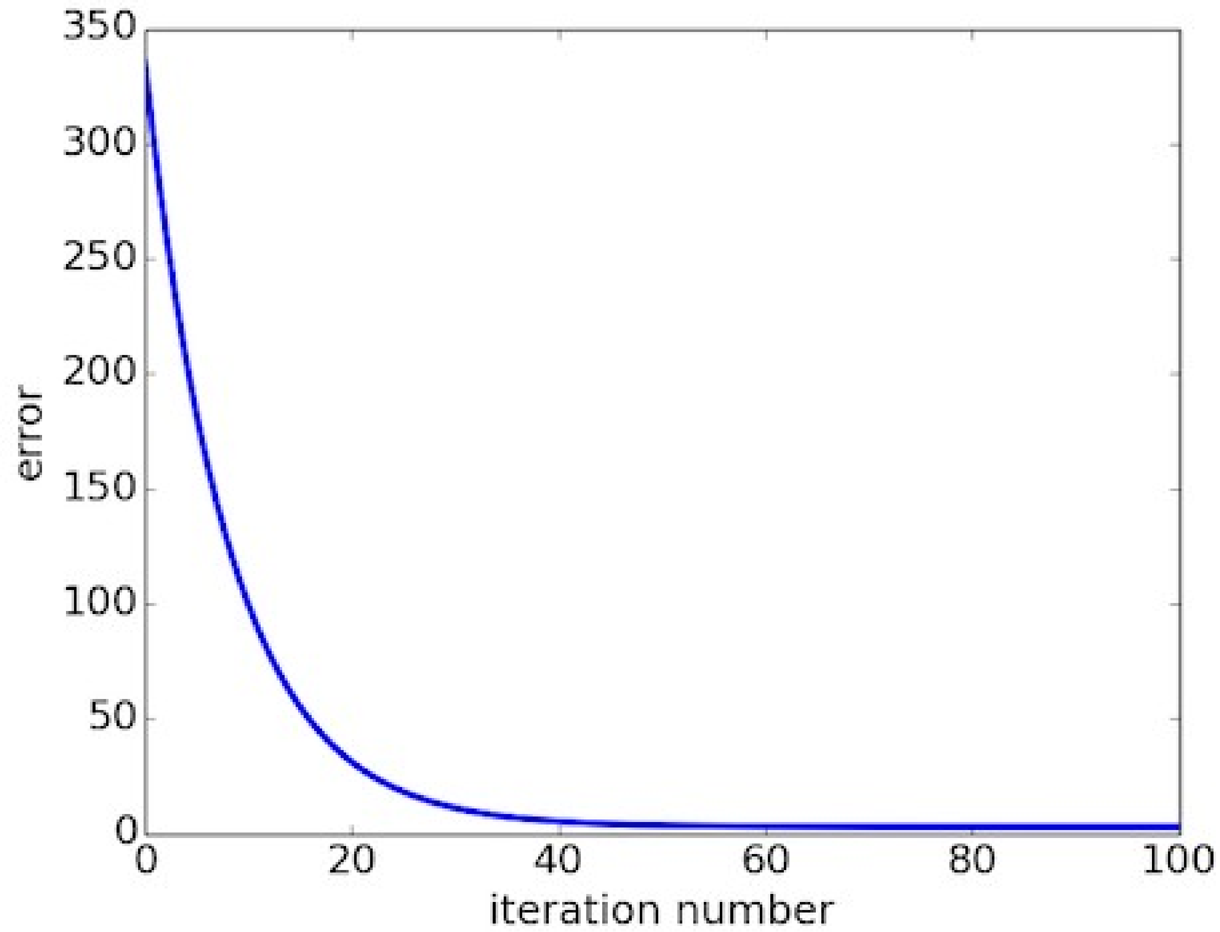
Fonte: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

Gradient Descent



Fonte: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

Gradient Descent



Fonte: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

Multivariate Linear Regression

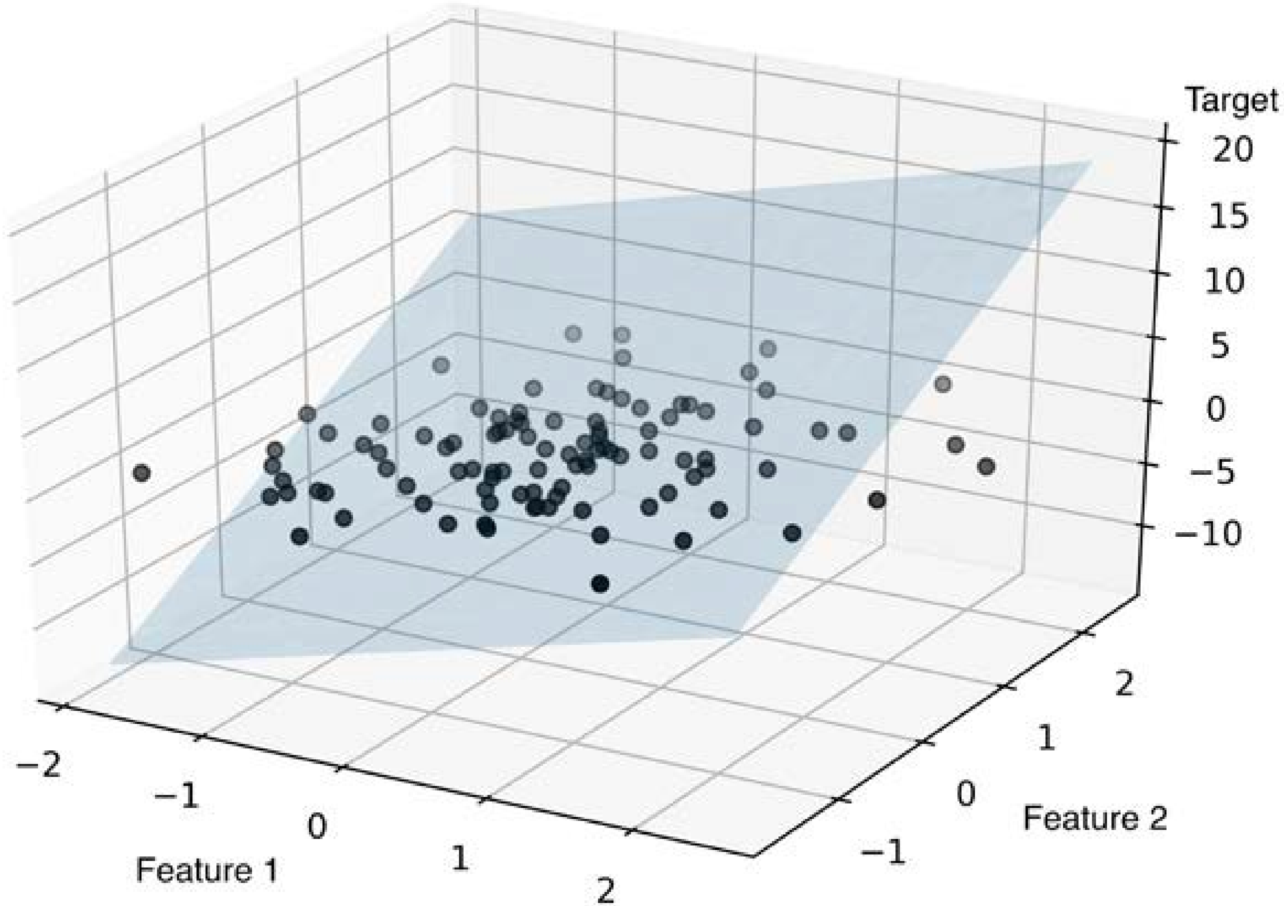
ou Multiple linear regression

$$y = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{i=0}^{m} w_i x_i = w^T x$$

Onde, w_0 é a interseção do eixo y com $x_0=1$.

Multivariate Linear Regression

ou Multiple linear regression



Stochastic Gradient Descent

- Gradient Descent is the process of minimizing a function following the slope or gradient of that function.

- Stochastic gradient descent evaluates and updates the coefficients every iteration to minimize the error of a model on our training data.

$$b = b - \text{learning rate} \times \text{error} \times x$$

Stochastic Gradient Descent

Parâmetros

- Learning Rate
  - Used to limit the amount that each coefficient is corrected each time it is updated.

- Epochs
  - The number of times to run through the training data while updating the coefficients.

$$b_1(t + 1) = b_1(t) - \text{learning rate} \times \text{error}(t) \times x_1(t)$$

$$b_0(t + 1) = b_0(t) - \text{learning rate} \times \text{error}(t)$$

Referências

- https://www.kdnuggets.com/2017/04/simple-understand-gradient-descent-algorithm.html

Obrigado!

Dúvidas, comentários, sugestões?

Regis Pires Magalhães

regismagalhaes@ufc.br

UFC