Regressão Linear

Fabrício Olivetti de França

Universidade Federal do ABC

Tópicos

1. Ordinary Least Square

2. Gradiente Descendente

UFABC

Aprendendo com exemplos

Uma das formas de aprendizado de máquina é definido como um conjunto de dados na forma $\{(\mathbf{x}, y)\}$, com $\mathbf{x} \in \mathbb{R}^d$ um vetor de atributos e $y \in \mathbb{R}$ uma variável alvo, queremos descobrir um mapa entre um certo $\mathbf{x}$ para seu valor y correspondente:

$$f : \mathbf{x} \to y.$$

UFABC

Aprendendo com exemplos

Vamos tomar como exemplo um corretor que quer aprender a avaliar o valor de um imóvel.

Cada imóvel pode ser representado por diversas características como:

- Metragem quadrada
- Vagas na garagem
- Andar
- Bairro
- etc.

Aprendendo com exemplos

Para aprender a avaliar novos imóveis ele investiga pelo classificado do jornal vários exemplos:

m^2	vagas	andar	valor
80	1	2	100.000,00
120	2	5	400.000,00
100	1	10	350.000,00
...	...	...	...

UFABC

Aprendendo com exemplos

O objetivo é aprender:

$$f(m^2, \text{vagas}, \text{andar}) = \text{valor},$$

para quaisquer valores de m^2, vagas e andar.

Isso caracteriza o Aprendizado Supervisionado.

Aprendendo com exemplos

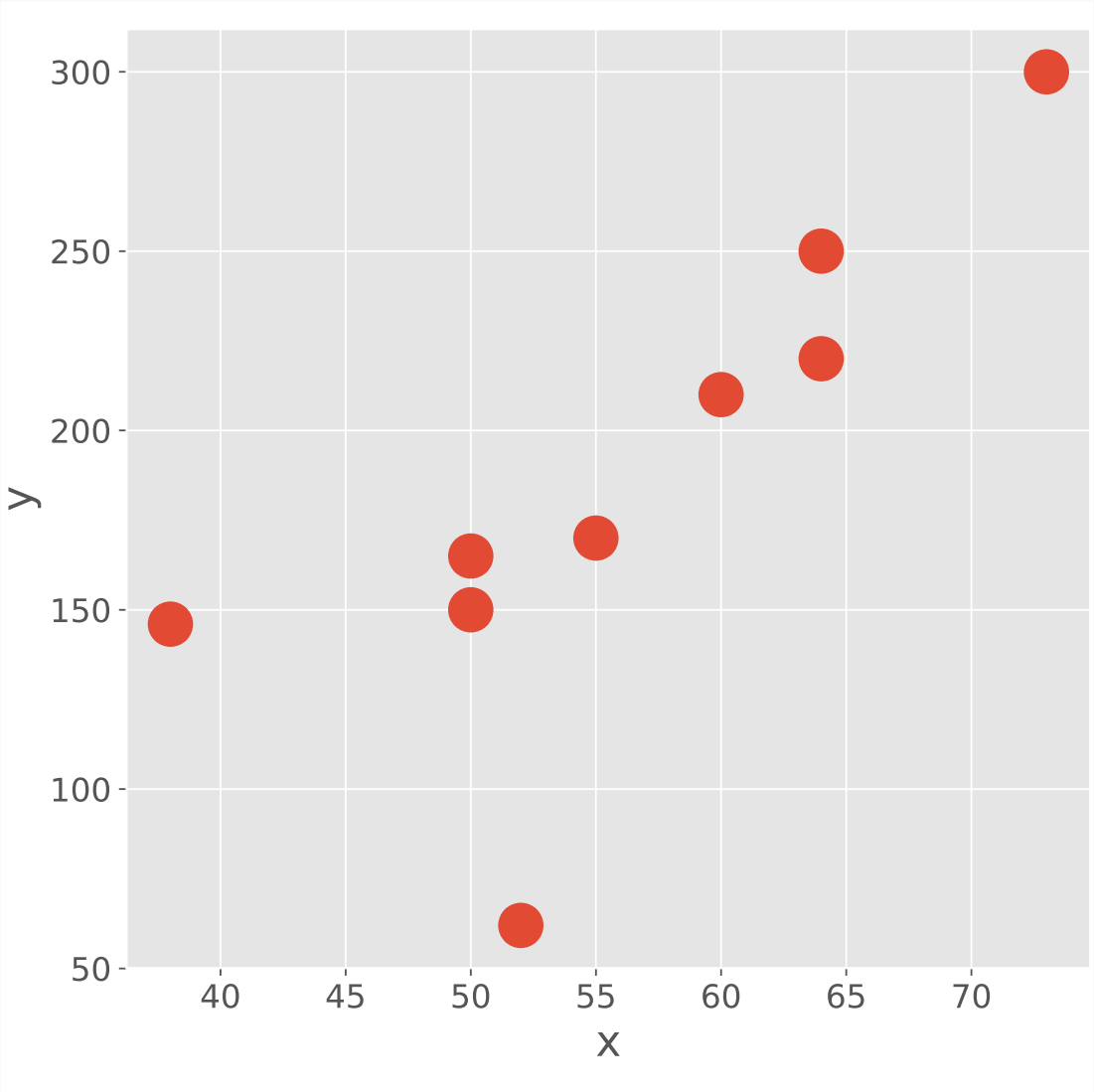
Reduzindo nossa tabela para duas variáveis, temos:

m^2	mil $
52	62
38	146
50	150
50	165
55	170
60	210
64	220
64	250
73	300

UFABC

Aprendendo com exemplos

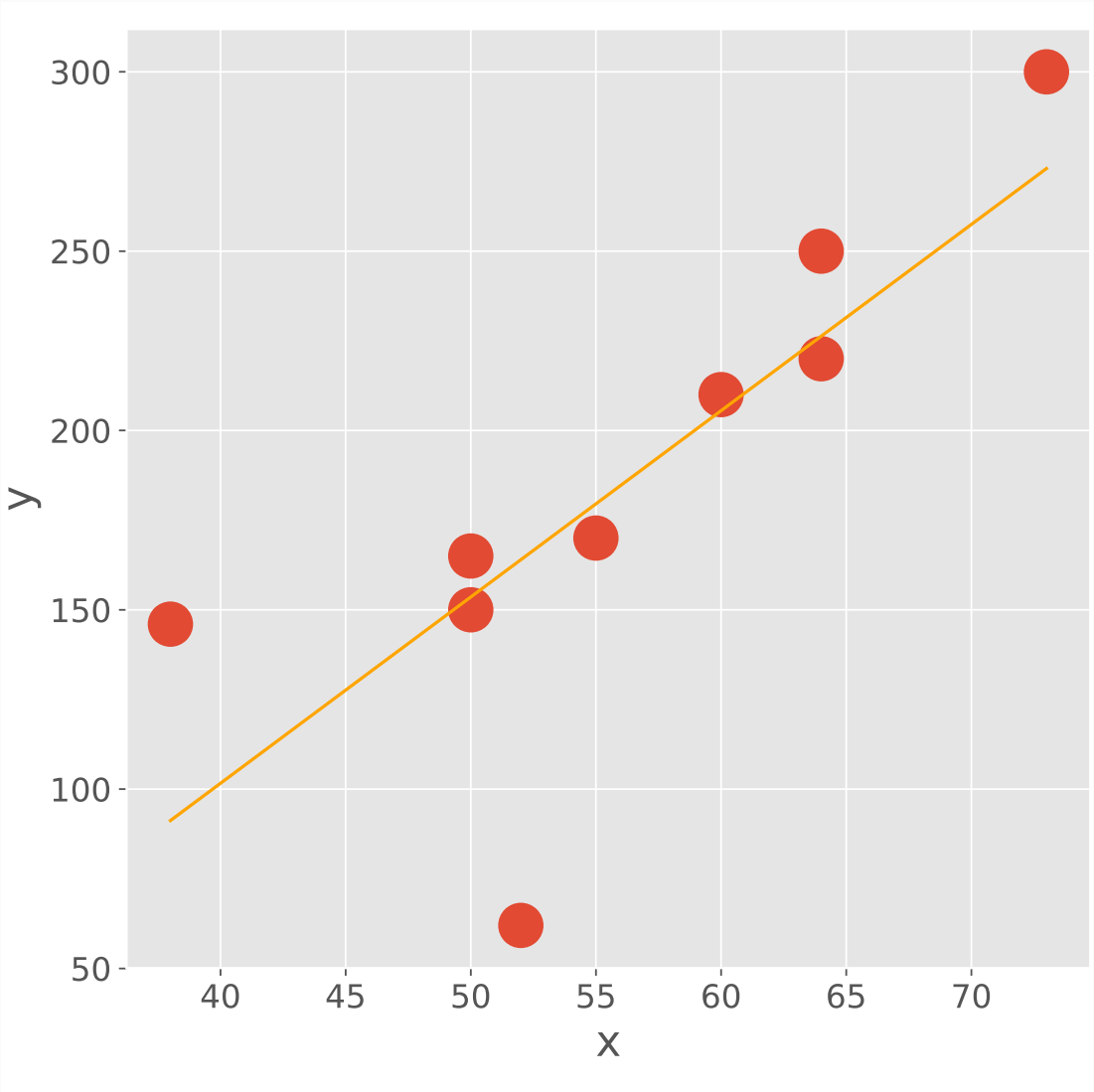
Com duas variáveis é simples verificar a relação visualmente:



UFABC

Aprendendo com exemplos

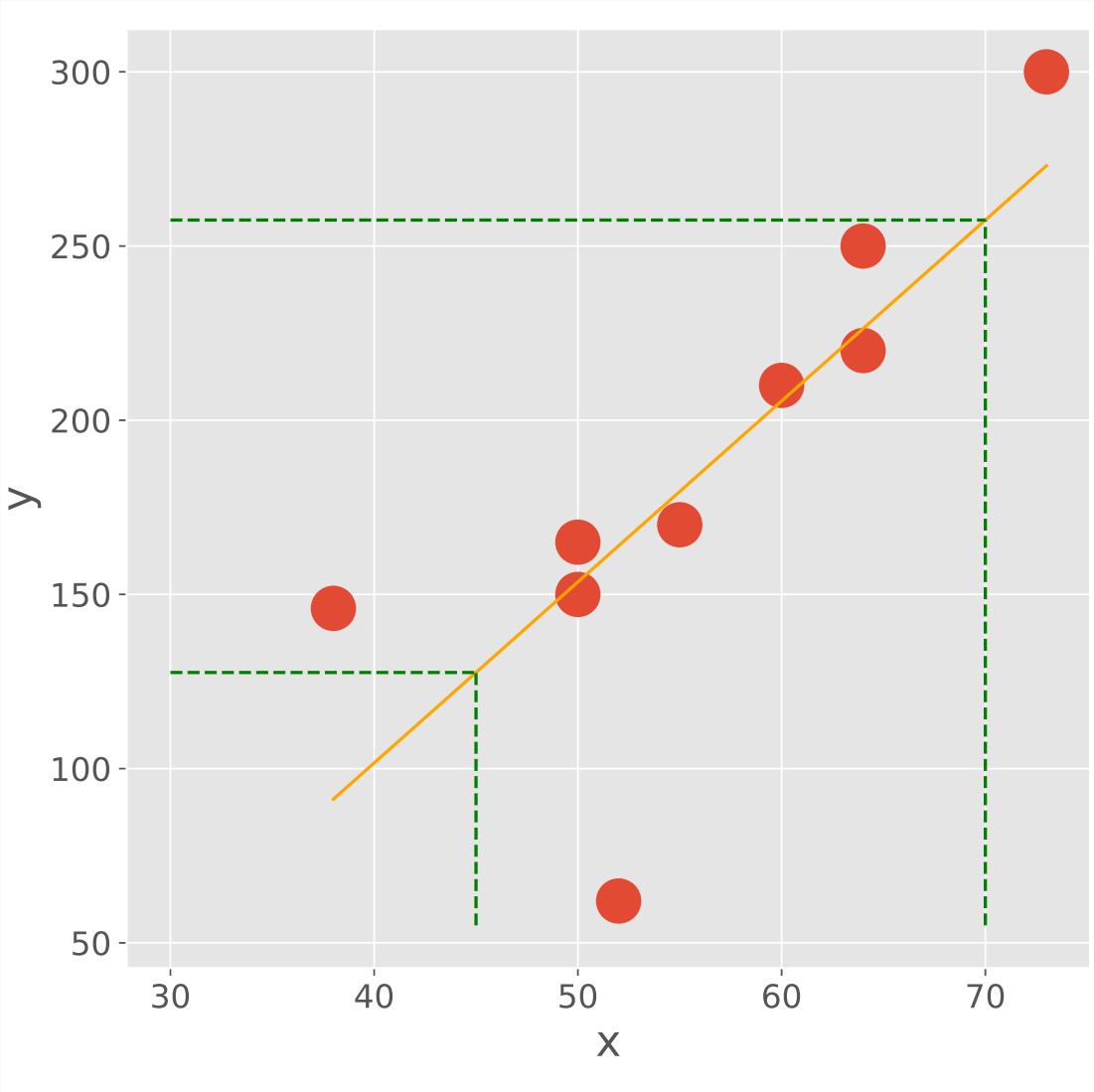
Com duas variáveis é simples verificar a relação visualmente:



UFABC

Aprendendo com exemplos

Com essa reta podemos determinar novos valores:



UFABC

Regressão Linear

Esse método é conhecido como regressão linear.

Ele pode ser usado quando nossas variáveis apresentam uma correlação linear entre elas!

(mas tem alguns truques para permitir correlações não-lineares também)

Regressão Linear

Dado um conjunto com n exemplos $\{(\mathbf{x_i}, y_i)\}$, com $\mathbf{x_i} \in \mathbb{R}^d, y \in \mathbb{R}$.
Queremos encontrar $f : \mathbb{R}^d \to \mathbb{R}$, tal que:

$$f(\mathbf{x_i}) \approx \mathbf{w} \cdot \mathbf{x_i} + b = y_i$$

Regressão Linear

Renomenado nossas variáveis do exemplo, temos:

$$x_{i,1} = m^2 \tag{1}$$
$$x_{i,2} = \text{ vagas} \tag{2}$$
$$x_{i,3} = \text{ andar} \tag{3}$$
$$y_i = \text{ vagas} \tag{4}$$
$$x_i = (x_{i,1}, x_{i,2}, x_{i,3}) \tag{5}$$

UFABC

Regressão Linear

$$f(\mathbf{x_i}) = y_i \tag{6}$$

$$\mathbf{w} \cdot \mathbf{x_i} + b = y_i \tag{7}$$

$$w_1 \cdot x_{i,1} + w_2 \cdot x_{i,2} + w_3 \cdot x_{i,3} + b = y_i \tag{8}$$

UFABC

Regressão Linear

Definindo $w_0 = b$ e $x_{i,0} = 1$ para todo i, podemos escrever:

$$w_0 \cdot x_{i,0} + w_1 \cdot x_{i,1} + w_2 \cdot x_{i,2} + w_3 \cdot x_{i,3} = y_i \tag{9}$$

$$\mathbf{w} \cdot \mathbf{x_i} = y_i, \tag{10}$$

ou seja, y é o produto interno entre $\mathbf{w}$ e $\mathbf{x_i}$.

Regressão Linear

Dados os n exemplos, podemos escrever na forma matricial, $X \in \mathbb{R}^{n \times (d+1)}$, $\mathbf{y} \in \mathbb{R}^{n \times 1}$:

$$X \cdot w = y,$$

com $\mathbf{w} \in \mathbb{R}^{(d+1) \times 1}$.

Esse modelo de regressão é dito *interpretável* pois é fácil determinar o quanto uma mudança em determinado x_j afeta o valor de y.

Ordinary Least Square

Ordinary Least Square

O nosso problema consiste em determinar $\mathbf{w}$ de tal forma a minimizar o erro de aproximação:

$$e(\mathbf{w}) = (y - X \cdot \mathbf{w})^T (y - X \cdot \mathbf{w}).$$

UFABC

Ordinary Least Square

Para determinar a solução, calculamos a derivada e igualamos a zero para encontrar o mínimo local:

$$\frac{\partial e(\mathbf{w})}{\partial \mathbf{w}} = \frac{\partial (y - X \cdot \mathbf{w})^T (y - X \cdot \mathbf{w})}{\partial \mathbf{w}}.$$

Ordinary Least Square

Temos então que:

$$w = (X^T X)^{-1} X^T y,$$

com $(X^T X)^{-1} X^T$ sendo a pseudo-inversa de X.

Ordinary Least Square

Para dimensões grandes de X o custo da multiplicação de matrizes pode se tornar proibitivo.

```
def ols(X,y):
    '''
    X: matriz n x d de exemplos
    y: vetor  n x 1 de valores alvo
    '''
    pseudoInv = np.linalg.inv(X.T @ X)
    xy        = X.T @ y
    return pseudoInv @ xy
```

UFABC

Ordinary Least Square

A multiplicação $X^T X$ pode ser simplificada como a soma dos produtos externos de cada x_i por ele mesmo:

```
from functools import reduce
from operator import add

def ols(X,y):
    '''
    X: matriz n x d de exemplos
    y: vetor  n x 1 de valores alvo
    '''

    xTx       = reduce(add, map(lambda x: np.outer(x,x),X)
    pseudoInv = np.linalg.inv(xTx)
    xy        = X.T @ y
    return pseudoInv @ xy
```

UFABC

Ordinary Least Square - Scikit-Learn

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
lr.fit(X, y)
yi = lr.predict(xi)
```

UFABC

Gradiente Descendente

Gradiente Descendente

Uma outra forma de resolver o problema é utilizando métodos de gradiente para otimização.

Dada uma função $f(x)$ definida e diferenciável em torno de um ponto x_0, sabemos que ela decresce mais rapidamente na direção oposta do gradiente $f(x_0)$.

Gradiente Descendente

Ou seja, fazendo:

$$x^{t+1} = x^t - \alpha \cdot f(x^t).$$

Temos que $f(x^{t+1}) \leq f(x^t)$, para um α pequeno.

UFABC

Gradiente Descendente

Temos então que:

$$f(x^0) \geq f(x^1) \geq f(x^2) \geq f(x^t).$$

UFABC

Gradiente Descendente

Estamos interessados na minimização da aproximação de $f(\mathbf{x})$ com y:

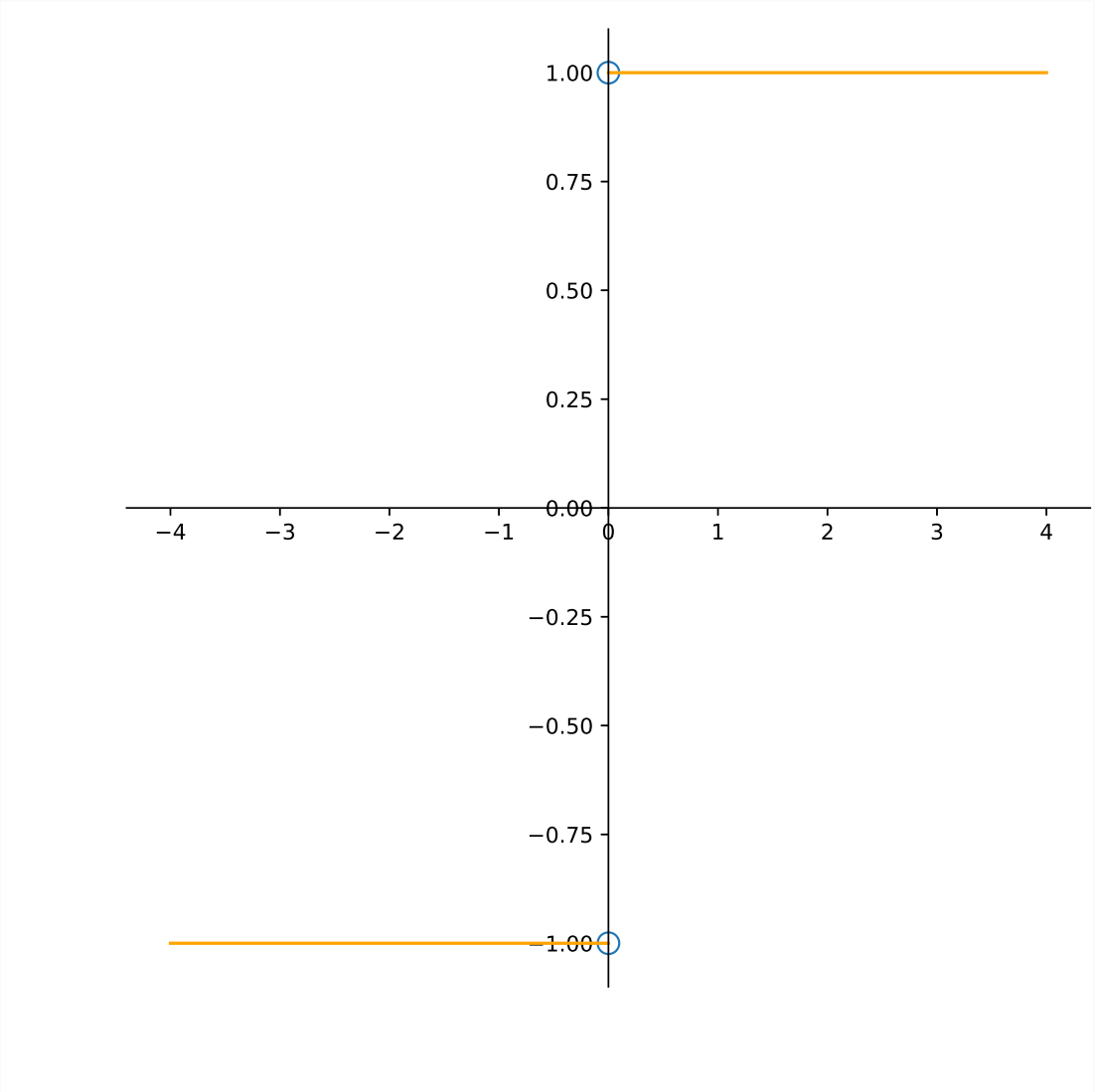
$$e(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \mathbf{x_i} \cdot \mathbf{w}|,$$

por exemplo.

UFABC

Gradiente Descendente

Porém, a derivada da função valor absoluto é indefinida no ponto 0:



UFABC

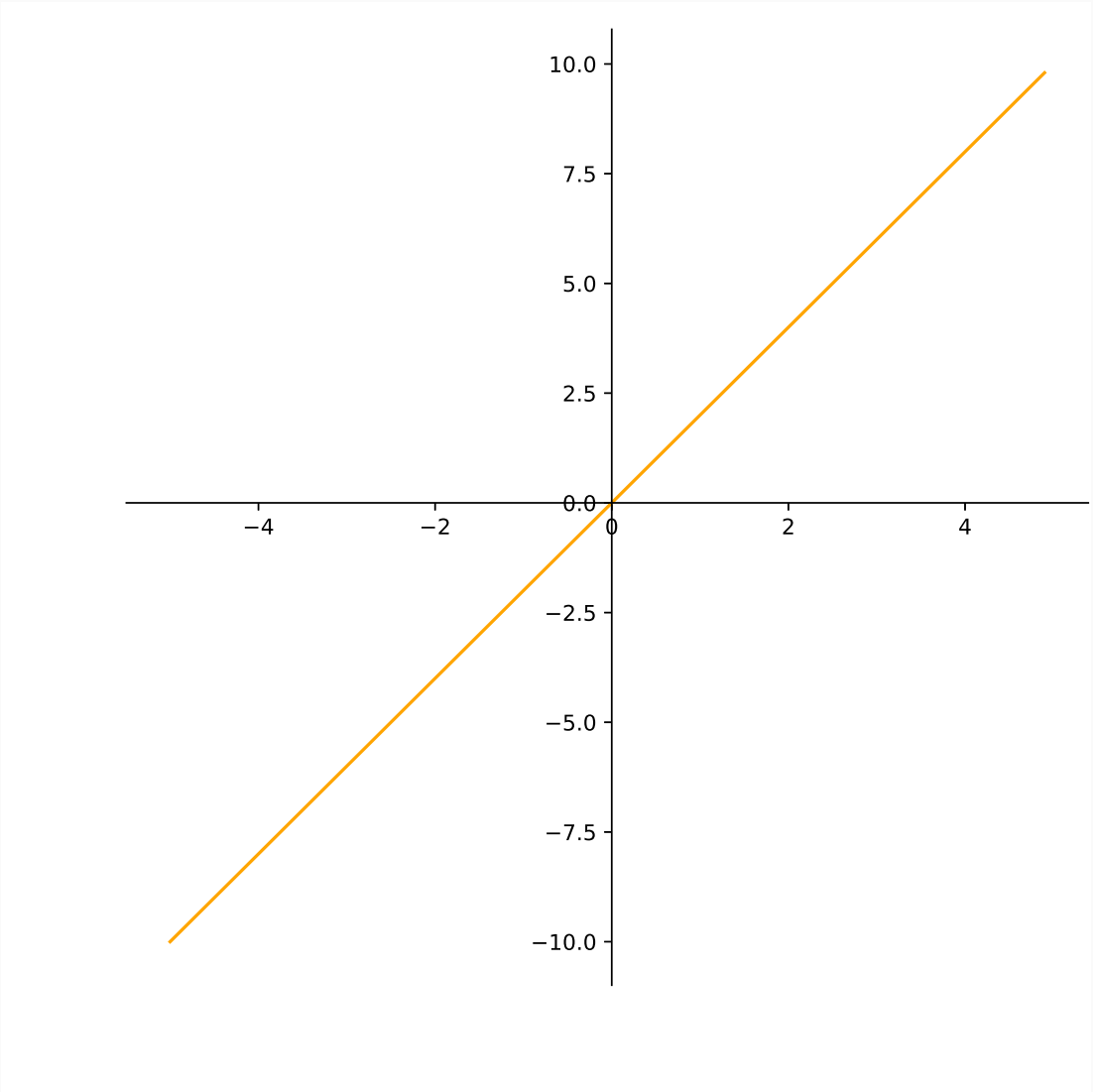
Gradiente Descendente

Por outro lado, a função quadrática apresenta uma única solução ótima, e é bem definida:

$$e(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x_i} \cdot \mathbf{w})^2,$$

UFABC

Gradiente Descendente



UFABC

Gradiente Descendente

Temos então que:

$$\nabla e(\mathbf{w}) = -\frac{2}{n}\sum_{i=1}^{n}(y_i - \mathbf{x_i}\cdot\mathbf{w})\cdot x_i,$$

UFABC

Gradiente Descendente

Ou com o mesmo ótimo:

$$\nabla e(\mathbf{w}) = -\frac{1}{n}\sum_{i=1}^{n}(y_i - \mathbf{x_i}\cdot\mathbf{w})\cdot x_i,$$

UFABC

Gradiente Descendente

O novo valor para $\mathbf{w}$ pode ser calculado como:

$$\mathbf{w^{t+1}} = \mathbf{w^t} + \alpha E((y_i - \mathbf{x_i} \cdot \mathbf{w}) \cdot x_i),$$

UFABC

Gradiente Descendente

O algoritmo pode ser descrito como:

```
def gradDesc(X, y, alpha, max_it):
    '''
    X: matriz n x d de exemplos
    y: vetor  n x 1 de valores alvo
    alpha: taxa de aprendizado
    max_it: máximo número de iterações
    '''
    (n,d) = X.shape
    w     = np.random.normal(size=(1,d))
    for it in range(max_it):
        yhat  = X @ w.T
        erro  = y - yhat
        nabla = np.mean(erro*x, axis=0)
        w    += alpha * nabla
    return w
```

UFABC

Escolha de α

A escolha do valor de passo de atualização de **w** é importante pois um valor muito baixo ocasiona uma demora para atingir o ponto de convergência, por outro lado um valor muito alto faz com que o método do gradiente ultrapasse o ponto de ótimo levando a divergência.

Próxima Aula

Na próxima aula aprenderemos sobre:

- Overfitting
- Treino e Validação

UFABC