Fundamentos Matemáticos e Computacionais de Machine Learning

Especialização em Machine Learning e Big Data

Profa. Dra. Juliana Felix

jufelix16@uel.br

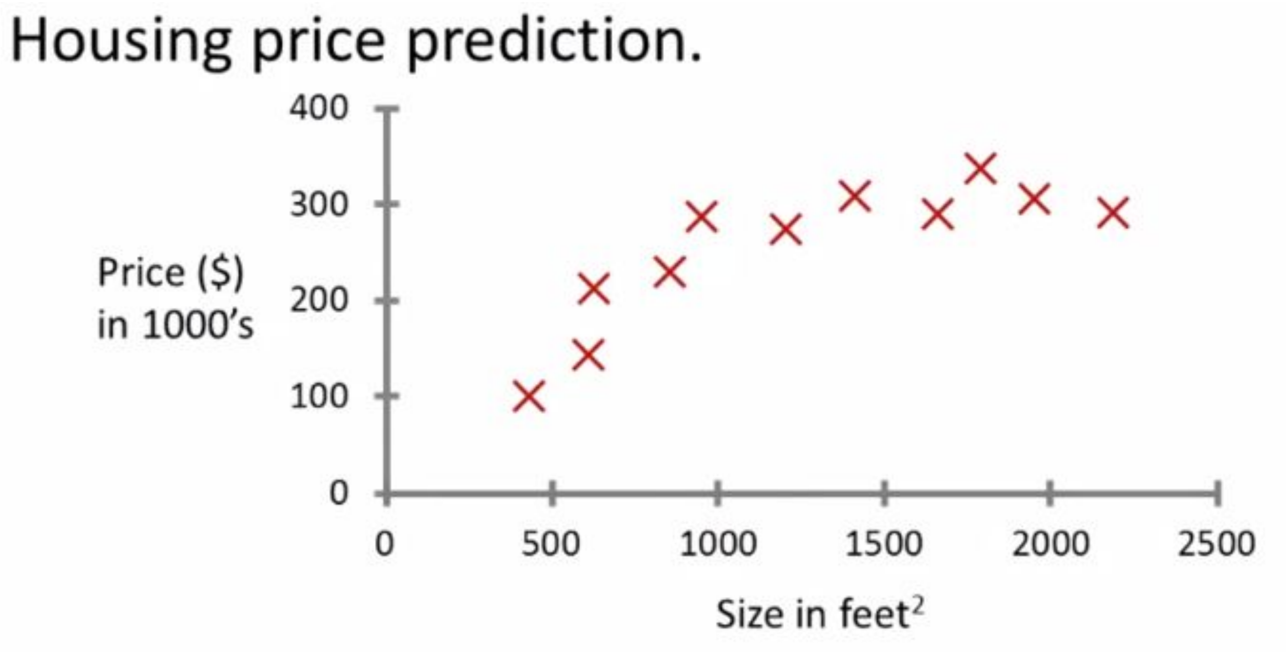
Regressão

Regressão

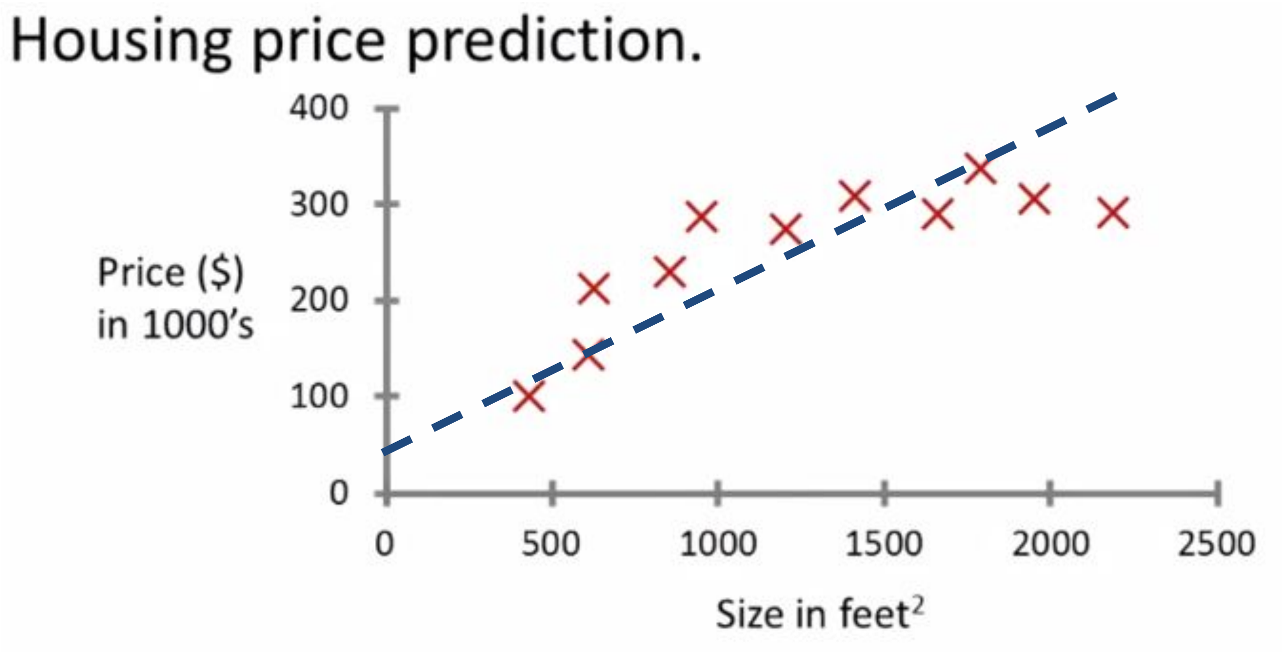
A **regressão linear** é uma técnica de análise de dados que permite prever o valor de dados desconhecidos usando valores de dados relacionados e conhecidos. Para isso modela-se, matematicamente, uma **equação linear** que relaciona:

- uma variável desconhecida, ou dependente, muitas vezes chamada de 'variável de resultado'
- e uma ou mais variáveis independentes, frequentemente chamados de 'preditores', 'covariáveis', 'recursos', ou 'features'

Regressão


Housing price prediction.
Price ($)
in 1000's
400
300
200
100
0
0
500
1000
1500
2000
2500
Size in feet²

Regressão


Housing price prediction.
400
300
200
100
0
Price ($)
in 1000's
0
500
1000
1500
2000
2500
Size in feet²

Regressão

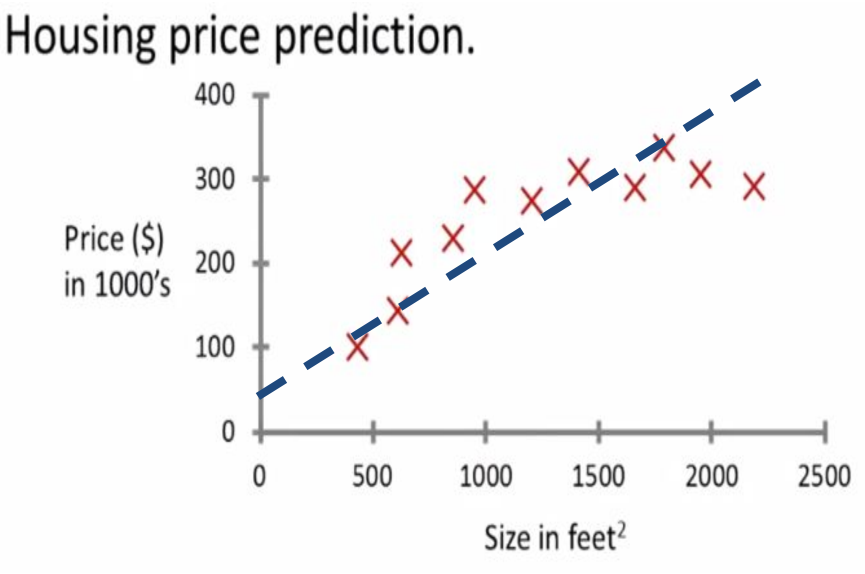
A regressão é um modelo baseado em aprendizado.

- Analizamos os dados
- Supomos que eles seguem algum padrão
- Utilizamos o padrão para predizer dados futuros

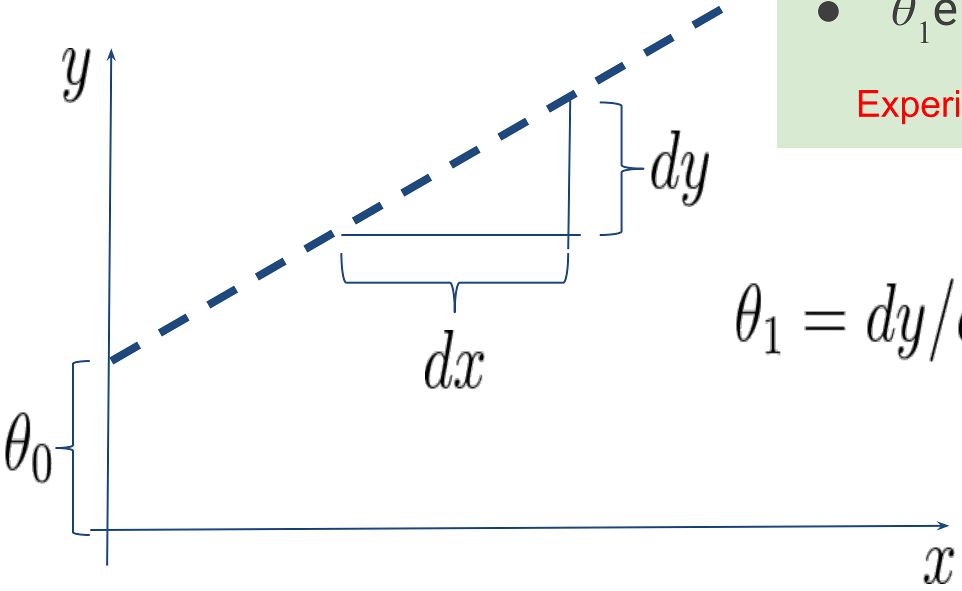
Regressão

Equação da reta:

$$y = \theta_0 + \theta_1 x$$



Regressão



$$\theta_1 = dy/dx$$

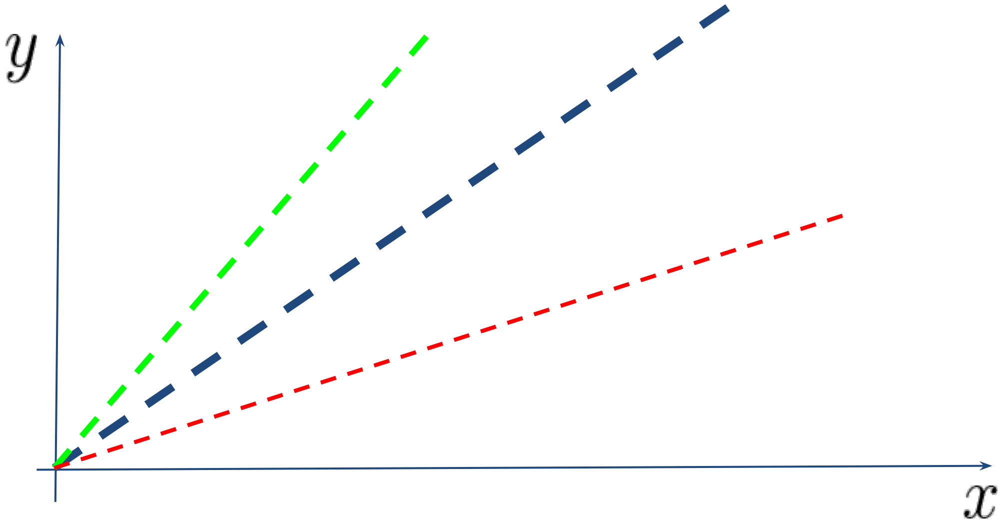
$$y = \theta_0 + \theta_1 x$$

- θ_0 é o deslocamento no eixo y
- θ_1 é a inclinação da reta

Experimente algumas visualizações aqui.

Regressão

- Se θ_0 é zero, a reta passa na origem



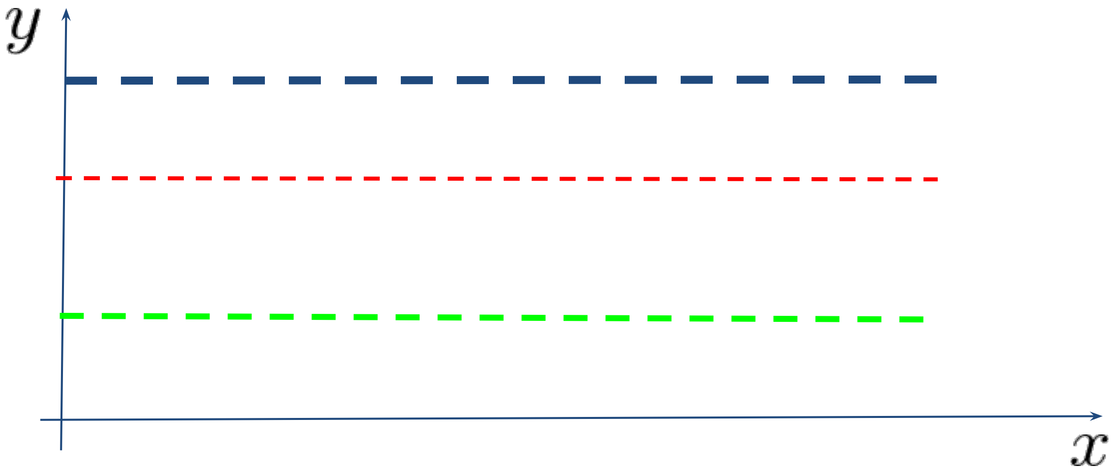
$$\theta_0 = 0 \qquad\qquad y = \theta_1 x$$

Regressão

```
import matplotlib.pyplot as plt
import numpy as np

#coeficiente angular da reta y = mx + b (ou y = theta1*x + theta0)
m = 0.8 #theta1


x = np.linspace(0, 1000, 1000)
#print(x)
y = m * x

plt.plot(x, y, '-r')
plt.xlim(0, 1000)
plt.ylim(0, 2000)
plt.xticks(np.arange(0,1100, step=100))

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado de um lote')
plt.grid()
plt.show()
```

Regressão

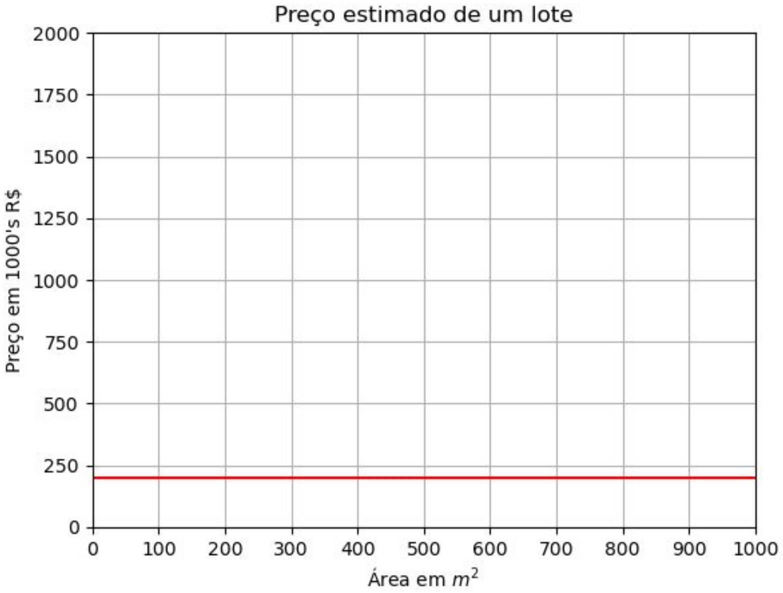
- Se θ_1 é zero, a reta será paralela ao eixo x



$$\theta_1 = 0 \qquad y = \theta_0$$

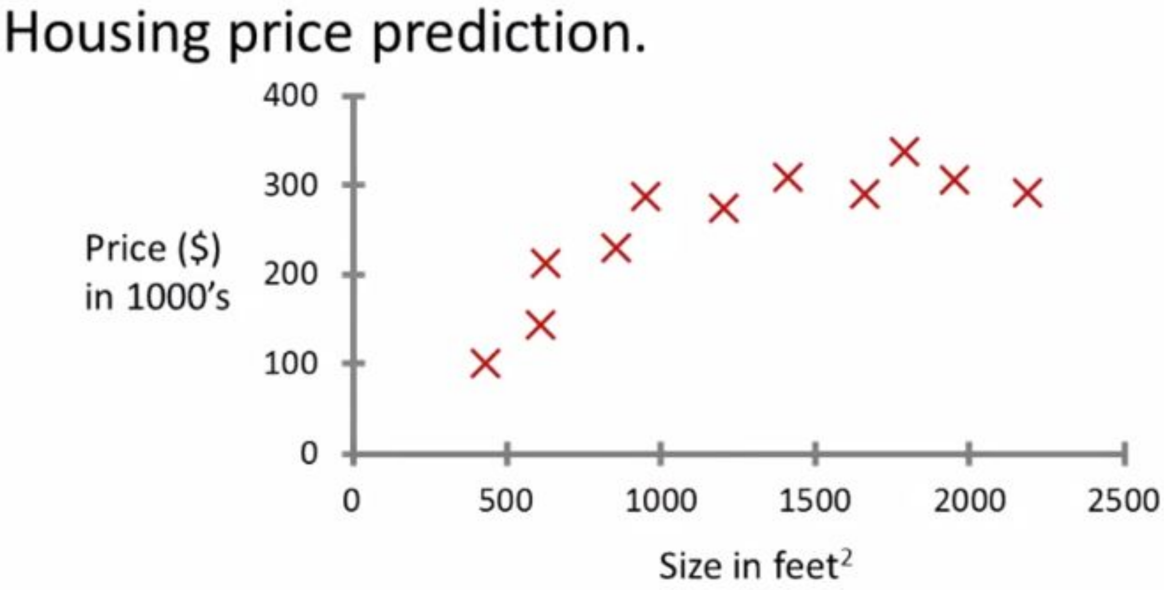
Regressão

```
import matplotlib.pyplot as plt
import numpy as np

#parâmetro para uma reta sem inclinação y= mx + b (ou y = theta1*x + theta0)
b = 200    #theta0

x = np.linspace(0, 1000, 1000)
y = np.ones(1000) * b

plt.plot(x, y, '-r')
plt.xlim(0, 1000)
plt.ylim(0, 2000)
plt.xticks(np.arange(0,1100, step=100))

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado de um lote')
plt.grid()
plt.show()
```



Regressão

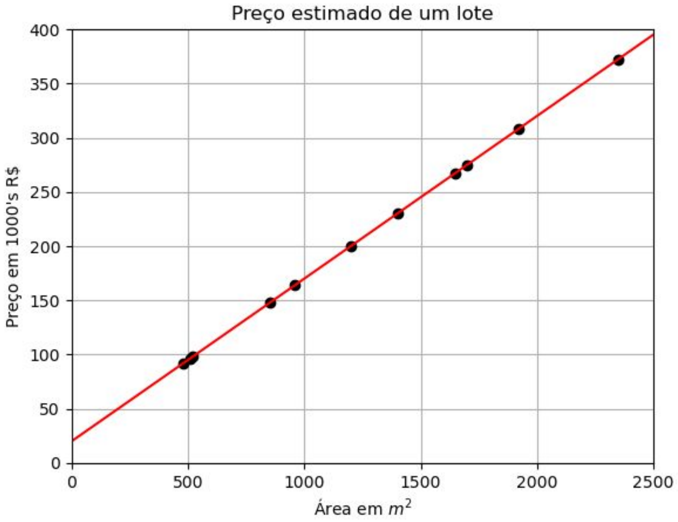
Voltando para o exemplo original… queremos estimar uma reta que melhor se ajuste aos pontos de observação.



Regressão

Se a correlação linear for forte....

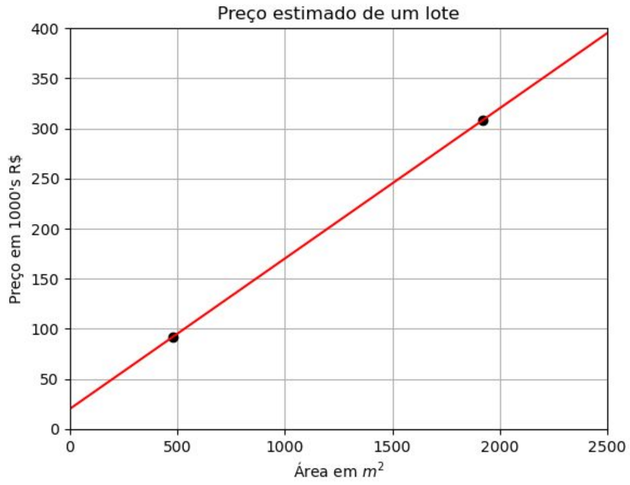
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([ 92, 96.5, 98, 147.5, 164, 200, 230, 267.5, 275, 308, 372.5])

plt.plot(x, y, 'o', color='black'); #plota os pontos no gráfico

m = 0.15 # inclinacao da reta, theta1
b = 20   # deslocamento no eixo y, theta0

x_entrada = np.linspace(0, 2500, 2500)
y_predito = m * x_entrada + b

plt.plot(x_entrada, y_predito, '-r')
plt.xlim(0,2500)
plt.ylim(0,400)

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado de um lote')
plt.grid()
plt.show()
```



Regressão

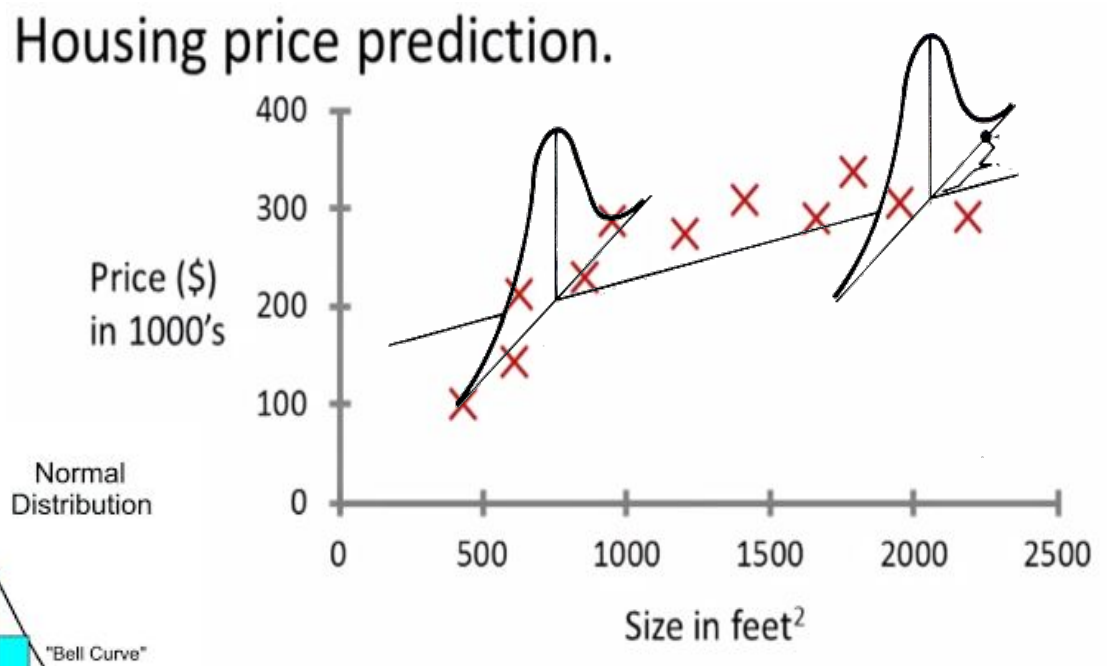
Se a correlação linear for forte…
Apenas 2 pontos quaisquer são suficientes para se encontrar os valores que definem uma reta

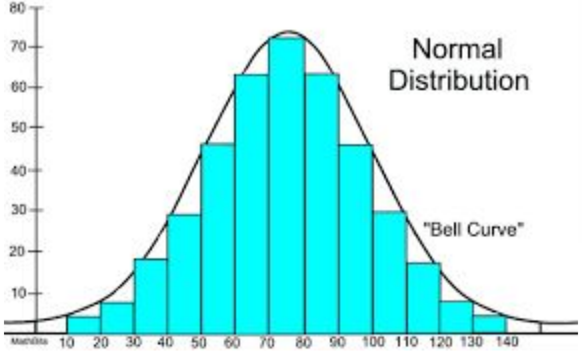
```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([480, 1920])
y = np.array([ 92, 308])

plt.plot(x, y, 'o', color='black'); #plota os pontos no gráfico

m = (y[1] - y[0])/(x[1]-x[0]) # inclinacao da reta, theta1
b = y[1] - m * x[1]    # deslocamento no eixo y, theta0

x_entrada = np.linspace(0, 2500, 2500)
y_predito = m * x_entrada + b

plt.plot(x_entrada, y_predito, '-r')
plt.xlim(0,2500)
plt.ylim(0,400)

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado de um lote')
plt.grid()
plt.show()
```



Regressão

Mas na vida real…





Exercício 1

Considerando os valores x e y fornecidos, tente encontrar uma reta que melhor se ajuste aos dados abaixo:

```
x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([98, 110, 200, 210, 280, 265, 300, 287, 325, 300, 290])
```

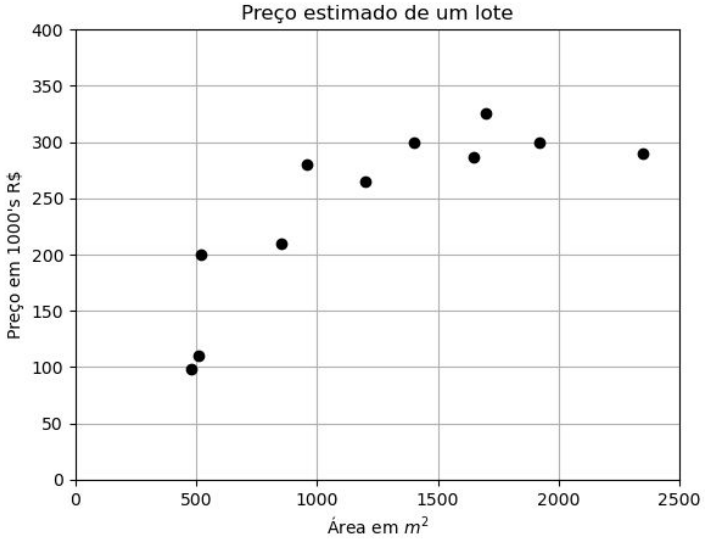
Faça isso utilizando:

- Apenas θ_0 (ou, equivalentemente, b)
- Apenas θ_1 (ou, equivalentemente, m)
- Atribuindo valores para θ_0 e θ_1 (b e m, respectivamente)

Exercício 1

Tente ajustar, manualmente, uma reta que se ajuste aos dados fornecidos!

```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([98, 110, 200, 210, 280, 265, 300, 287, 325, 300, 290])

plt.plot(x, y, 'o', color='black');

plt.xlim(0, 2500)
plt.ylim(0, 400)

plt.xlabel('Área em $m^2$')
plt.ylabel('Preço em 1000\'s R$')
plt.title('Preço estimado de um lote')
plt.grid()
plt.show()
```

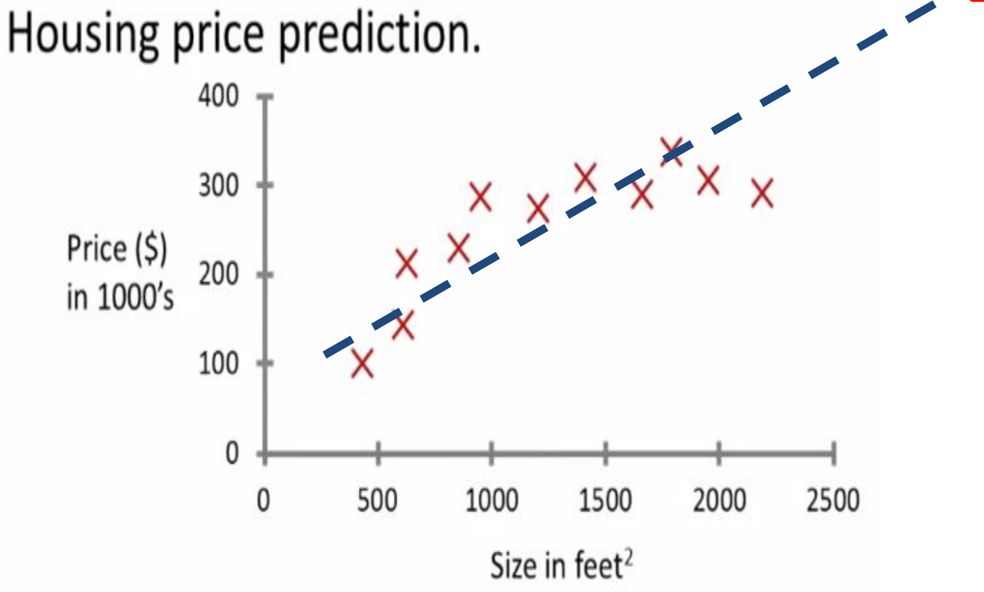


Regressão Linear

Regressão Linear

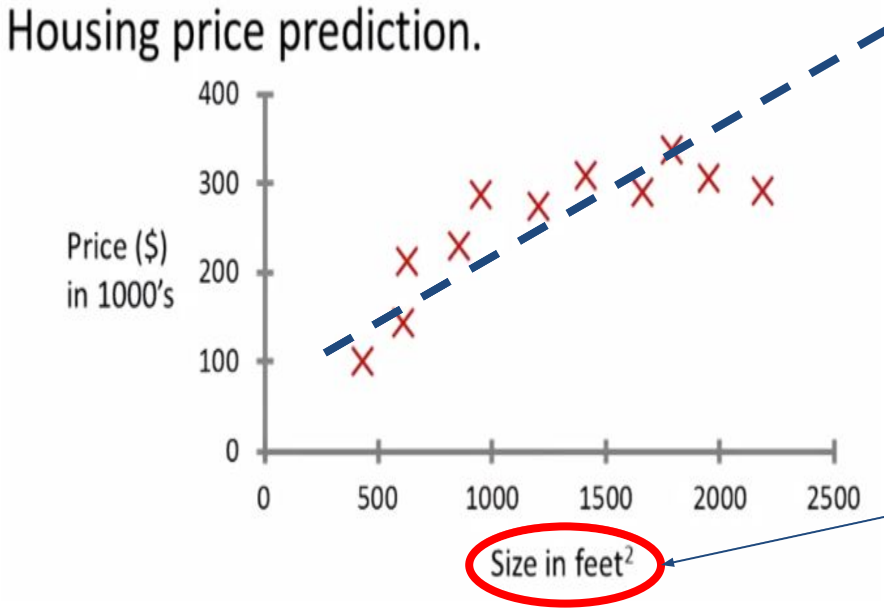
Modelo

$$\boxed{y = \theta_0 + \theta_1 x}$$



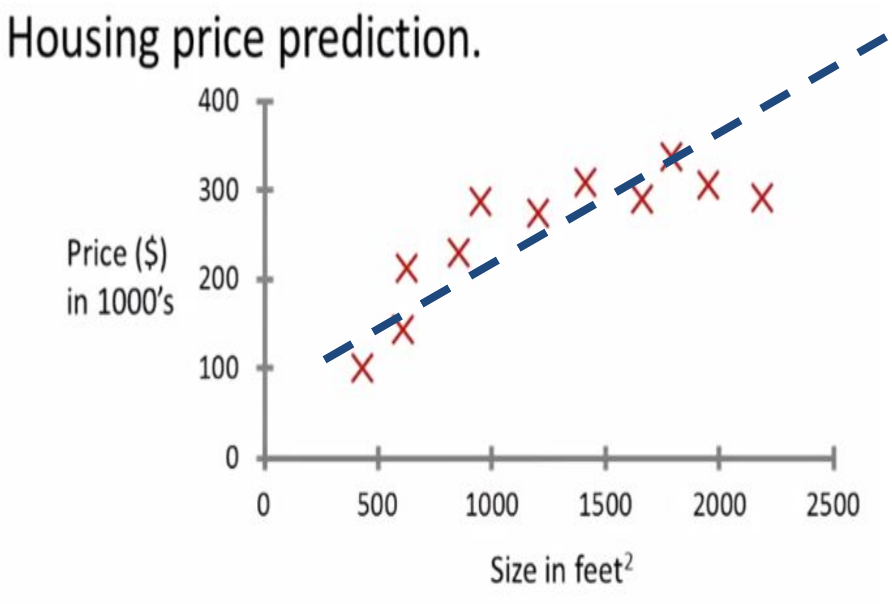
Regressão Linear

$$y = \theta_0 + \theta_1 x$$



Características (features)

Regressão Linear



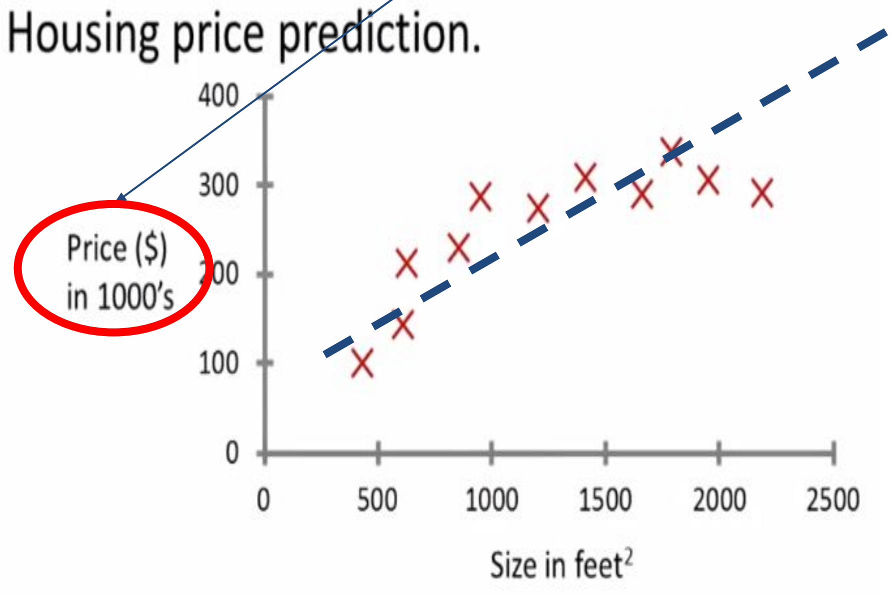
$$y = \theta_0 + \theta_1 x$$

Parâmetros

Regressão Linear

Saída (output)

$$y = \theta_0 + \theta_1 x$$



Exemplo

Dataset

Dataset é um conjunto de dados que combina amostra com

- Valores ou variáveis de **entrada** (features, características) e
- Valores de **saída** (outcome, labels) utilizados no aprendizado supervisionado

Dataset

Exemplo:

Preço de lotes na 'Terra tão tão distante'	
Tamanho do lote (em m^2) - X	Preço do lote (R$) - Y
2104	399.900
1600	329.900
2400	369.000
...	...

Total amostras = 47

Feature (característica)

Outcome (saída)

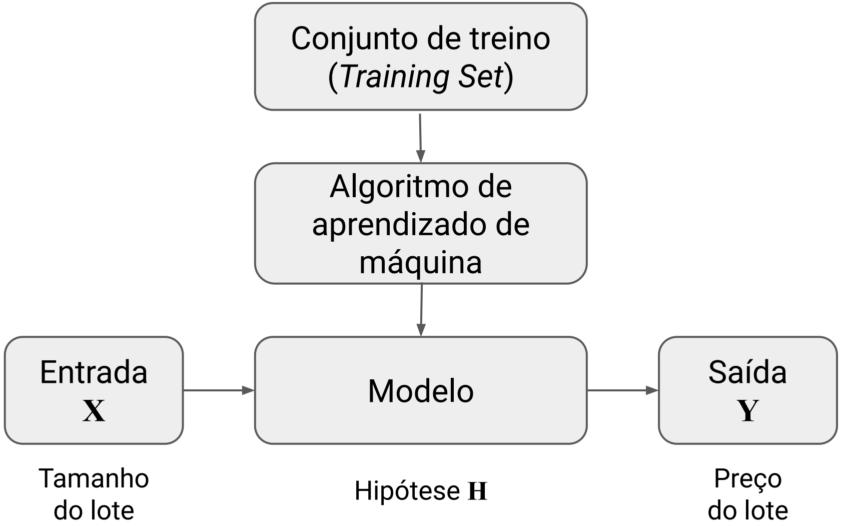
Notação

Podemos pensar no problema anterior como um problema que tem:

- Um total de $\mathbf{m}$ amostras/samples ($\mathbf{m}$ = 47)
- Cada amostra tem 1 única feature/característica (tamanho do lote)
  - Costumamos representar uma variável de entrada por $\mathbf{x}$
- Para cada amostra, temos uma única saída (preço do lote).
  - Costumamos representar uma variável de saída por $\mathbf{y}$
- Cada amostra pode ser representada por um par, ou tupla $\mathbf{(x,y)}$
  - Uma tupla $\mathbf{(x^i, y^i)}$ representa a i-ésima amostra do problema, com $\mathbf{1 \leq i \leq m}$

Processo básico de Machine Learning

A base de qualquer processo de machine learning consiste em mapear um dado de entrada $\mathbf{X}$ em um dado de saída $\mathbf{Y}$.



$$h(x) = \theta_0 + \theta_1 x$$

Desvio

Quando fazemos a predição de um valor, o **desvio** é a diferença entre o **valor esperado** (conhecido) e o **valor predito** pelo modelo construído.

$$\text{desvio}^{i} = \mathbf{Y}^{i} - \mathbf{h}(\mathbf{x}^{i})$$

$$\text{desvio}^{i} = \mathbf{Y}^{i} - \hat{\mathbf{Y}}^{i}$$

MSE

O ***Mean Square Error*** (MSE - Erro Médio Quadrático) é a **média** do **quadrado** dos **erros** obtidos pelo modelo.

$$MSE = \frac{1}{m} \sum_{i=1}^{m} \left(Y^i - h(x^i)\right)^2$$

Notação

- O par (x^1, y^1) refere-se aos dados $(2104, 399900)$.
- No entanto, se a linguagem de programação que estiver utilizando considerar índices iniciados em zero (caso do Python), o par (x^1, y^1) refere-se, na verdade, aos dados $(1600, 329900)$.

Preço de lotes na 'Terra tão tão distante'	
Tamanho do lote (em m^2) - X	Preço do lote (R$) - Y
2104	399.900
1600	329.900
2400	369.000
...	...

MSE

O ***Mean Square Error*** (MSE - Erro Médio Quadrático) é a **média** do **quadrado** dos **erros** obtidos pelo modelo.

$$MSE = \frac{1}{2m}\sum_{i=1}^{m}\left(Y^i - h(x^i)\right)^2$$

Na prática, a média é dividida pela metade (1/2) como uma conveniência para o cálculo do ***gradiente descendente***, método utilizado na regressão linear, que cancelará o termo 1/2.

Em outras palavras, dividir por 1/m ou 1/2m não traz diferenças significativas para o cálculo dos valores analisados.

Leia mais aqui: https://math.stackexchange.com/questions/884887/why-divide-by-2m

MSE

$$MSE = \frac{1}{2m}\sum_{i=1}^{m}\left(Y^i - h(x^i)\right)^2$$

Considerando os seguintes valores preditos, podemos calcular o MSE do modelo.

Preço de lotes na 'Terra tão tão distante'				
x	**y**	**h(x)**	**desvio**	**desvio^2**
2104	399.900	399.800	100	10.000
1600	329.900	339.900	-10.000	10.000.000
2400	369.000	367.000	2.000	4.000.000
			Soma	14.010.000
			MSE	**2.335.000**

Exercício 2

Considerando os valores x e y fornecidos, tente encontrar, manualmente, uma reta que melhor se ajuste aos dados abaixo:

```
x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([98, 110, 200, 210, 280, 265, 300, 287, 325, 300, 290])
```

Faça isso utilizando:

- Apenas θ_0 (ou, equivalentemente, b)
- Apenas θ_1 (ou, equivalentemente, m)
- Atribuindo valores para θ_0 e θ_1 (b e m, respectivamente)

Para cada reta, calcule o respectivo MSE. Plote as retas e o MSE encontrado em todos os casos.

Função Custo

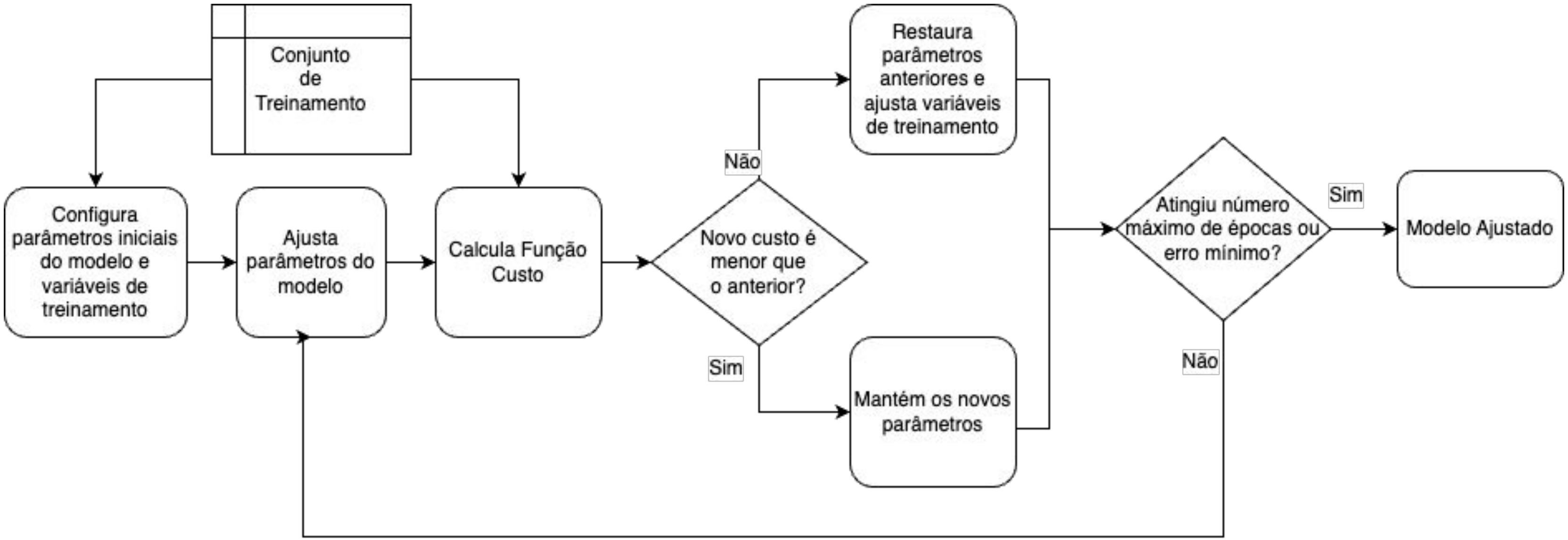
Queremos encontrar θ_0 e θ_1 para os quais os valores preditos serão os mais próximos possíveis dos valores y reais.

Objetivo

$$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1) = \left[\frac{1}{2m} \sum_{i=1}^{m} \left(Y^i - h(x^i) \right)^2 \right]$$

$$h(x) = \theta_0 + \theta_1 x$$

Treinamento do Modelo



Conceitos

- Parâmetros - As variáveis do modelo a serem ajustadas
- Dados de treinamento - Grupo de dados utilizado para ajustar os parâmetros do modelo.
- Variáveis de treinamento - variáveis utilizadas para controlar o algoritmo de treinamento.
  - Step/Passo - tamanho do ajuste que será utilizado para ajustar um parâmetro do modelo.
  - Época - O intervalo durante o qual o modelo é ajustado baseado no dado de treinamento.
  - Épocas - O número de épocas pelo qual o algoritmo de treinamento funcionará.
- Parâmetros anteriores e atuais - valores que representam o modelo antes e depois do ajuste.
- Custo anterior e atual - valores que representam o erro do modelo antes e depois do ajuste.
- Parâmetros anteriores e atuais - valores que representam o modelo antes e depois do ajuste.
- Fator de ajuste - valor que ajuda a determinar o comportamento do algoritmo de treinamento.

Critérios de parada

- Número de épocas
  - O programa para ao atingir o número de épocas definido
- Erro mínimo atingido
  - O programa para ao atingir um erro menor ou igual ao erro mínimo definido
- Sem alterações após **n** épocas
  - O programa para ao perceber que o erro não sofreu alterações após uma quantidade **n** de épocas estabelecidas

Exercício 3

Considerando os valores x e y fornecidos, crie um algoritmo para encontrar, automaticamente, uma reta que melhor se ajuste aos dados abaixo:

```
x = np.array([480, 510, 520, 850, 960, 1200, 1400, 1650, 1700, 1920, 2350])
y = np.array([98, 110, 200, 210, 280, 265, 300, 287, 325, 300, 290])
```

Minimiza o MSE

Faça isso utilizando:

- Apenas θ_0 (ou, equivalentemente, b)
- Apenas θ_1 (ou, equivalentemente, m)
- ~~Atribuindo valores para θ_0 e θ_1 (b e m, respectivamente)~~

Para cada reta, calcule o respectivo MSE. Plote as retas e o MSE encontrado em todos os casos.

Características da Função Custo

- A função custo é o resultado da soma dos desvios ao quadrado.
- Os desvios são obtidos pela diferença entre Y^i e $h(x^i)$.
- Se considerarmos apenas θ_0:

$$J(\theta_0) = \left[\frac{1}{2m}\sum_{i=1}^{m}\left(Y^i - h(x^i)\right)^2\right]$$

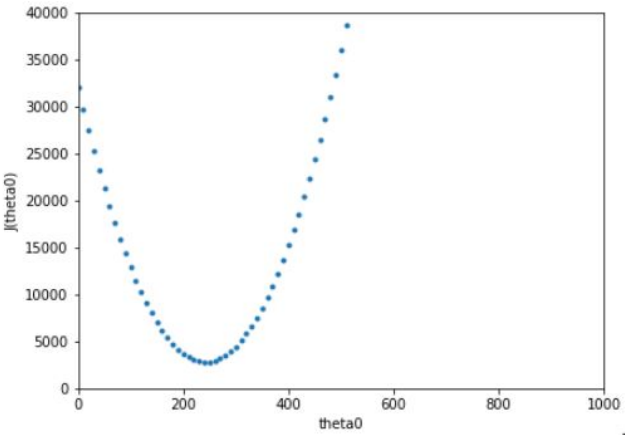
$$J(\theta_0) = (Y - h(x))^2$$

$$J(\theta_0) = Y^2 - 2 \cdot Y \cdot h(x) + h(x)^2$$

Características da Função Custo

- Sabemos que $h(x) = \theta_0 + \theta_1 x$
- Ao simplificar considerando $h(x) = \theta_0$, teremos uma parábola

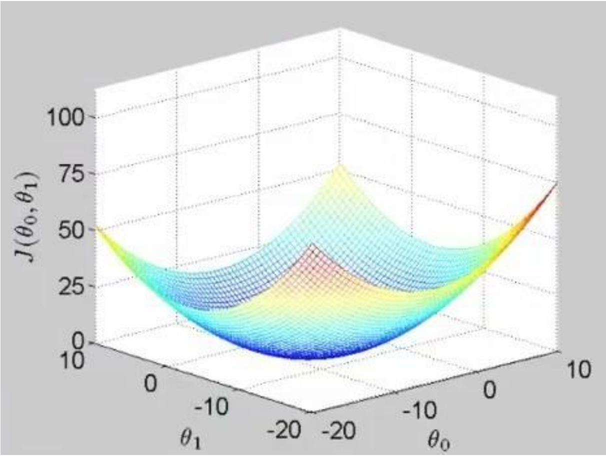
Cost: 29741.954545454544
Theta 0: 10
Epoch: 1

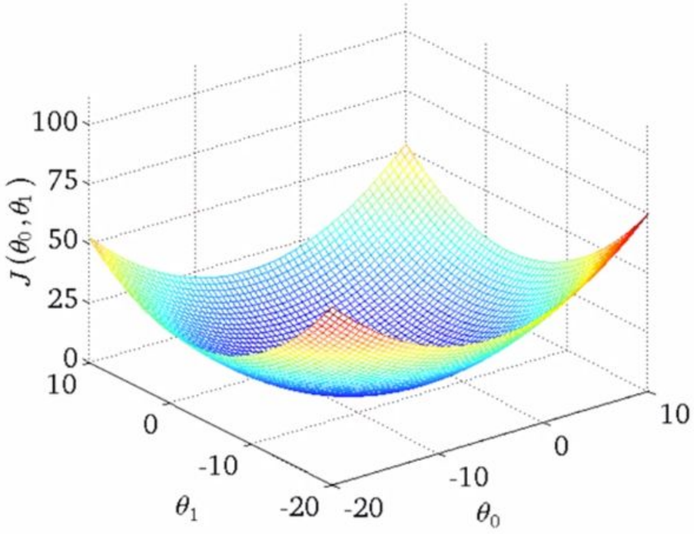


$$J(\theta_0) = Y^2 - 2 \cdot Y \cdot h(x) + h(x)^2$$

Características da Função Custo

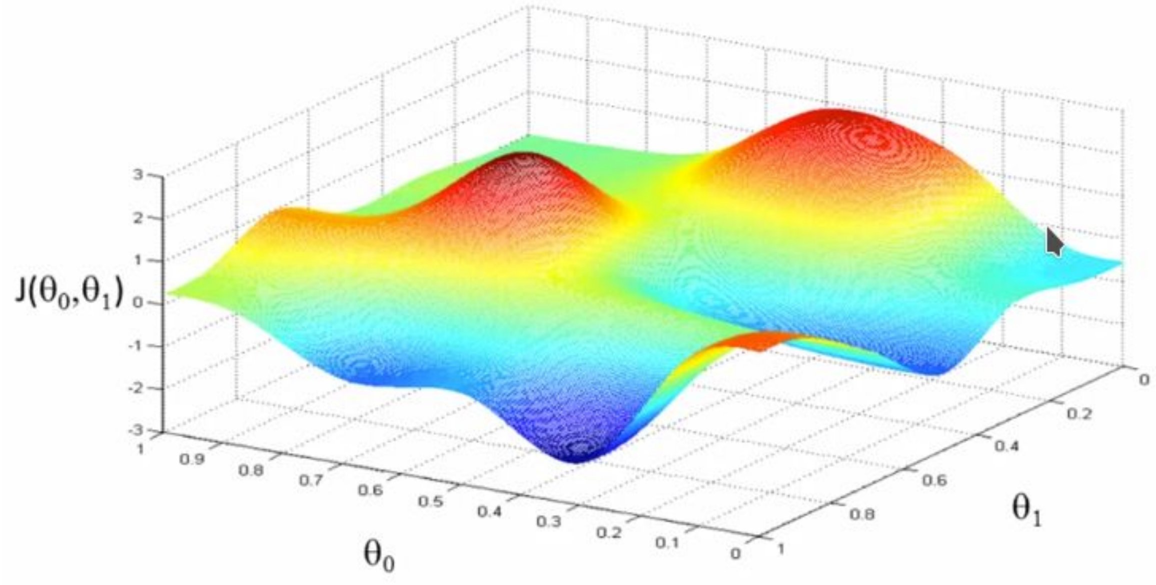
- Sabemos que $h(x) = \theta_0 + \theta_1 x$
- Ao considerar θ_0 e θ_1, teremos uma espaço 3D.





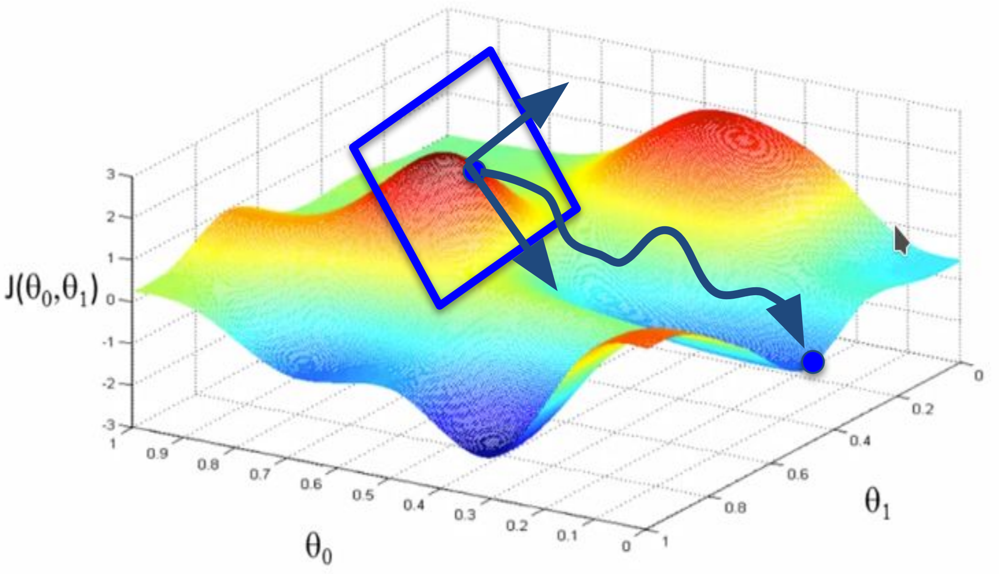
Características da Função Custo

Em certos modelos não lineares, mesmo tendo apenas θ_0 e θ_1, podemos nos deparar com superfícies cada vez mais complexas…



Características da Função Custo

Precisamos de uma maneira para encontrar o mínimo da função correspondente



$$\partial\left[J(\theta_0,\theta_1)\right] = \begin{bmatrix} \frac{\partial}{\partial\theta_0} J(\theta_0,\theta_1) \\ \\ \frac{\partial}{\partial\theta_1} J(\theta_0,\theta_1) \end{bmatrix}$$

Gradiente Descendente

- Algoritmo utilizado para encontrar os valores que minimizam a função custo.
- O algoritmo funciona iterativamente, ajustando os parâmetros em pequenos incrementos para minimizar a função de custo.
- O primeiro passo é calcular o gradiente da função de custo em relação a cada parâmetro.

$$new\theta_0 = \theta_0 - \alpha \cdot \boxed{\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)}$$

$$new\theta_1 = \theta_1 - \alpha \cdot \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

Gradiente Descendente

$$J(\theta_0, \theta_1) = \left[\frac{1}{2m} \sum_{i=1}^{m} \left(Y^i - h(x^i)\right)^2 \right]$$

Calculando-se as derivadas parciais, temos…

$$h(x) = \theta_0 + \theta_1 x$$

$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} [(Y^i - \theta_0 - \theta_1 x^i)]$$

$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} [(Y^i - \theta_0 - \theta_1 x^i) x^i]$$

Gradiente Descendente

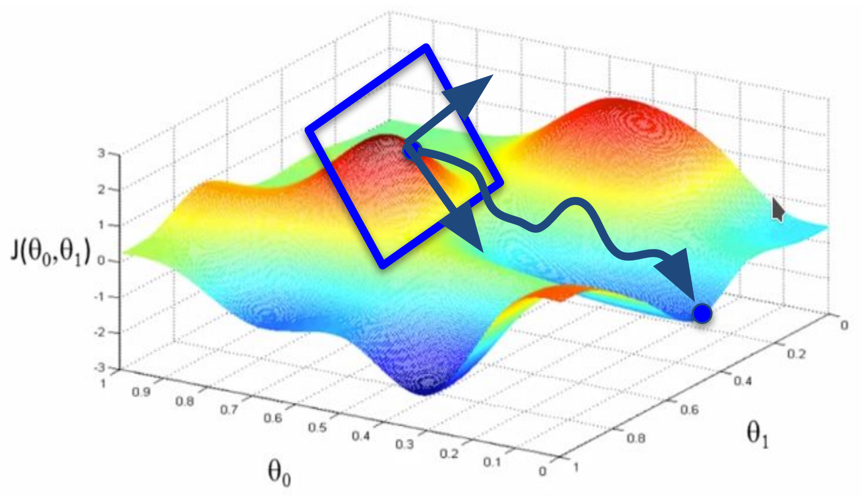
Substituindo...

$$new\theta_o = \theta_0 - \alpha \cdot \left[\frac{1}{m}\sum_{i=1}^{m}[(Y^i - h(x^i))]\right]$$

$$new\theta_1 = \theta_1 - \alpha \cdot \left[\frac{1}{m}\sum_{i=1}^{m}[(Y^i - h(x^i) \cdot x^i)]\right]$$

Gradiente Descendente

- Em seguida, os parâmetros são atualizados movendo-se na direção oposta ao gradiente.
- Esse processo é repetido várias vezes até que uma condição de parada seja atingida.

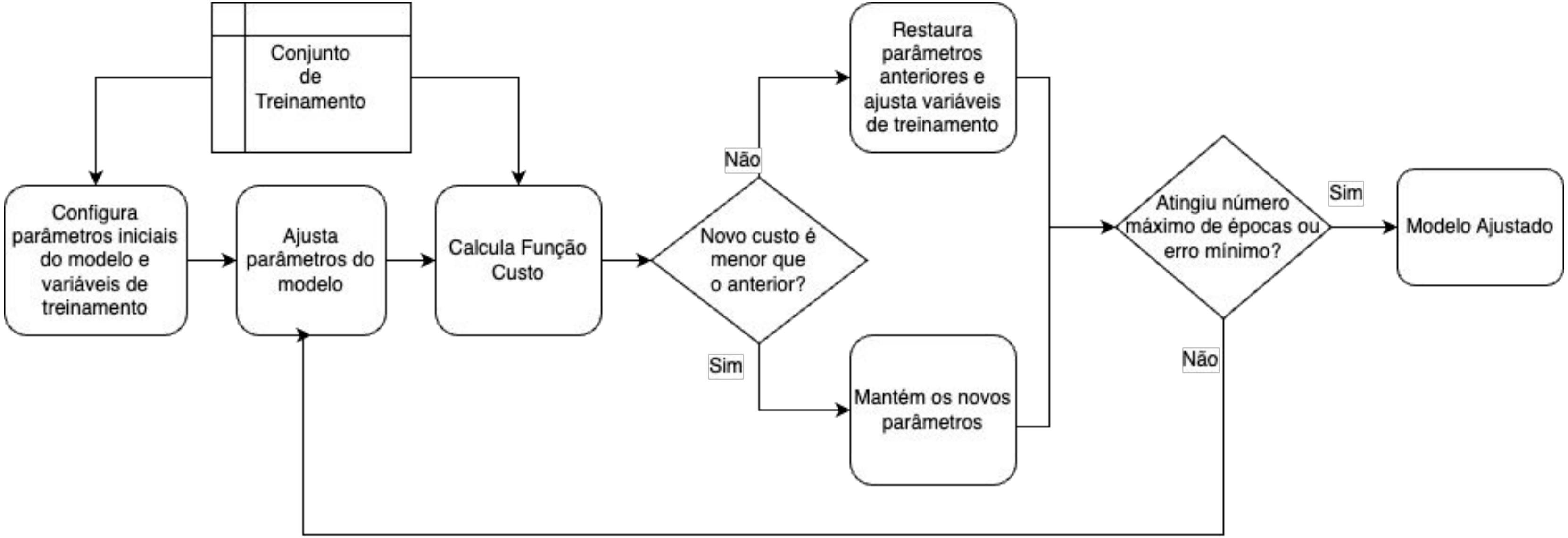


$$new\theta_o = \theta_0 - \alpha \cdot \left[\frac{1}{m}\sum_{i=1}^{m}[(Y^i - h(x^i))]\right]$$

$$new\theta_1 = \theta_1 - \alpha \cdot \left[\frac{1}{m}\sum_{i=1}^{m}[(Y^i - h(x^i) \cdot x^i)]\right]$$

- Alfa (α) é a **taxa de aprendizagem**

Treinamento do Modelo



Leitura recomendada

Calculadora gráfica: Desmos | Calculadora Gráfica

Regressão linear: Explicação sobre o modelo de regressão linear