Mineração de Dados

Regressão

Slides adaptados dos Profs. Jesús Mena-Chalco, Ronaldo Prati e David Correa

Predição/Regressão X Classificação

•Classificação tenta determinar a qual classe um exemplo pertence, baseado em exemplos cujas classes são conhecidas, gerando um modelo que pode ser aplicado a novos exemplos

•O modelo gerado pode estar representado em diferentes maneiras (regras, árvores, grafos, vetores de suporte).

- A saída é a classe em que o novo exemplo é predito.

•A classe em classificação é um atributo **nominal**

•E se o valor que eu quero predizer é **numérico**, ao invés de um conjunto nominal de valores?

•Então o nosso problema passa a ser de **regressão** ao invés de classificação.

Predição/Regressão

- Regressão gera uma fórmula para prever um valor numérico a partir dos dados.

- Um caso especial é prever a probabilidade de um exemplo pertencer a uma classe (e.g. regressão logística)

- O caso geral, no entanto, é **estimar/prever valores** de um atributo numérico diretamente **a partir da fórmula** a novos exemplos

- Também existem adaptações de algoritmos de classificação para o problema da regressão, como as Árvores de Regressão

Predição/Regressão

•Por exemplo, ao invés de prever se o tempo amanhã vai ser “quente”, “ameno”, “frio” ou “gelado”, podemos prever diretamente a temperatura

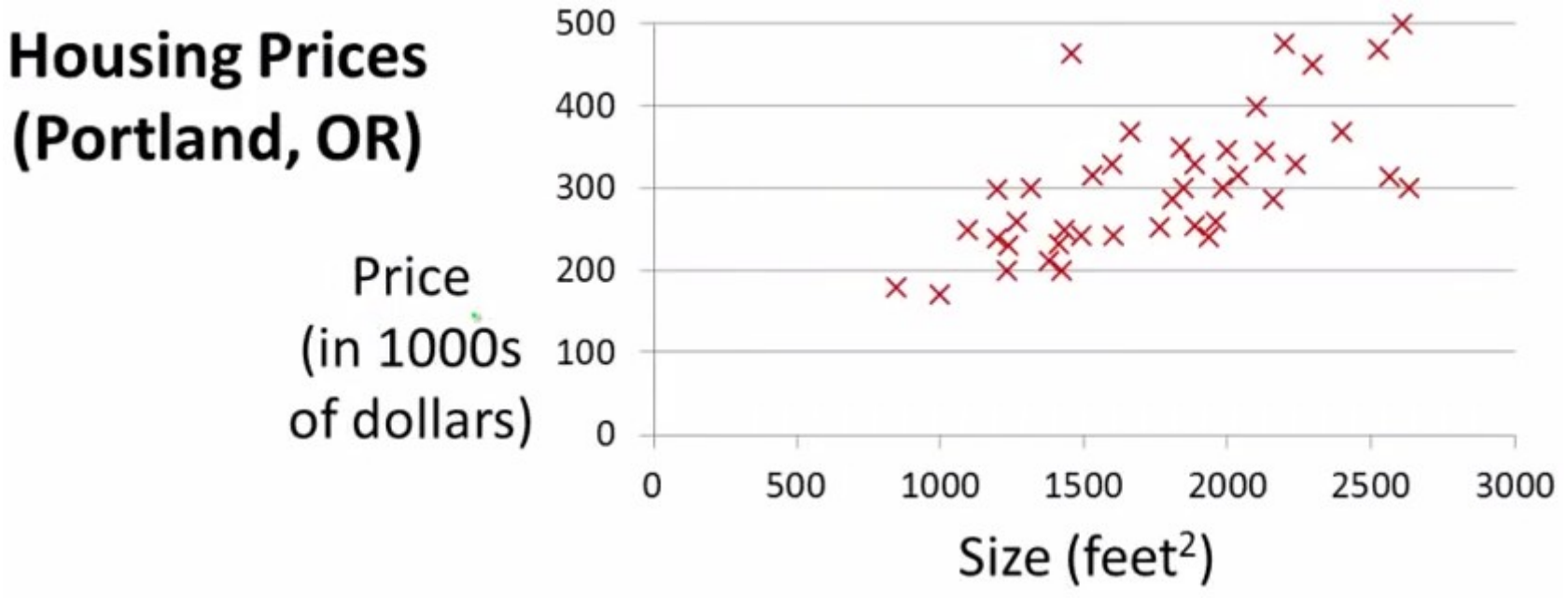
- 28,1 graus
- 7,3 graus
- mesmo que 28,1 ou 7,3 nunca tenham aparecido no conjunto de treinamento

•Ou o “nível de stress” de uma estrutura em diferentes condições, preço de um imóvel de acordo com suas características, o número de segundos até o próximo ônibus aparecer, ou o número de gols que a seleção irá marcar no próximo jogo, ou … qualquer outro valor numérico que você queira prever

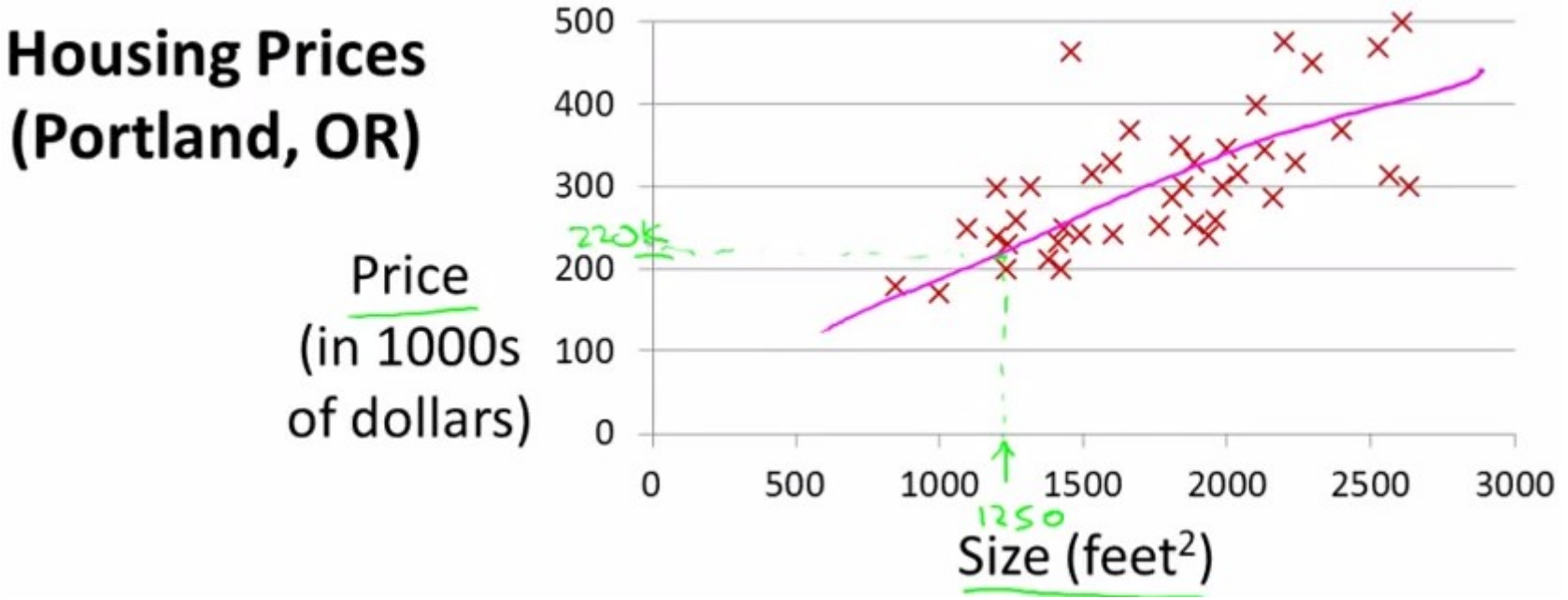
Regressão linear com uma variável

(A) Representação do modelo

Representação do modelo

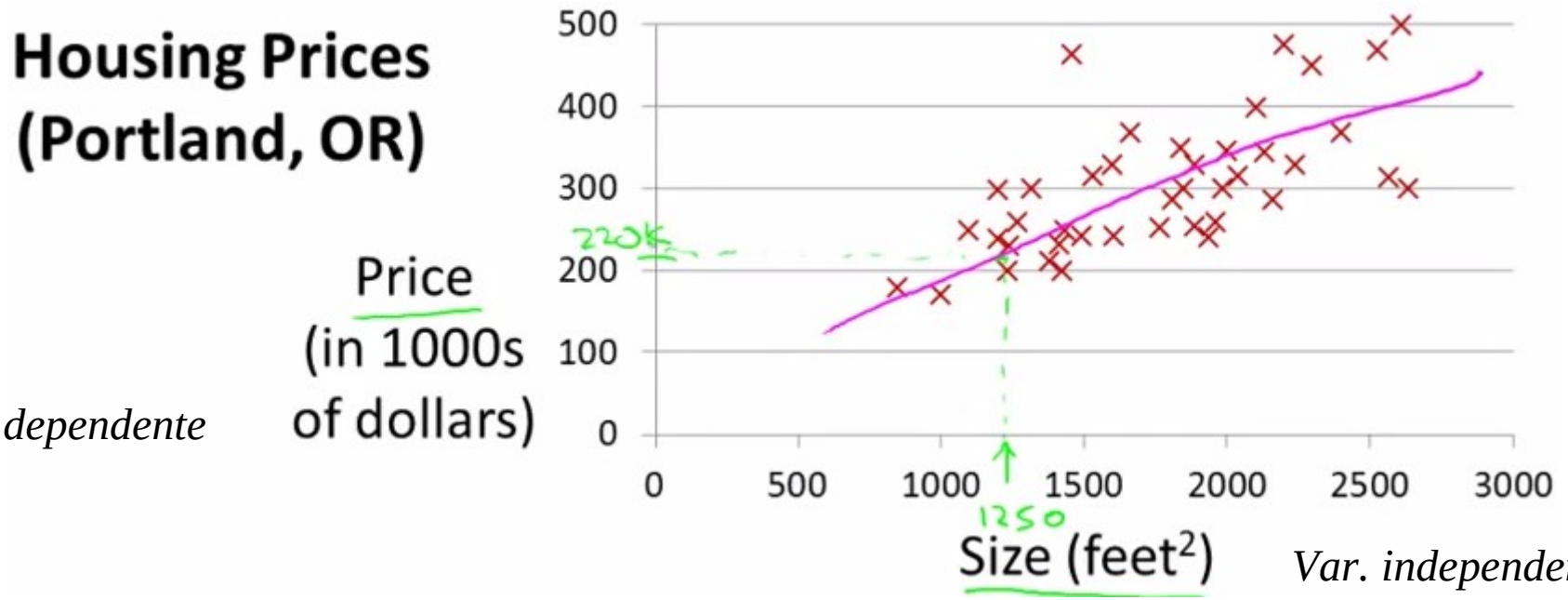


Representação do modelo



Pode se ajustar um modelo (uma reta)

Representação do modelo



Var. dependente

Var. independente

- Este é um exemplo de **Aprendizado Supervisionado**.
- Dando a “resposta correta” para cada um dos dados coletados.

- Este é um exemplo de um **Problema de Regressão** (inferência de relação)
- Usado para Inferir uma relação entre uma var. independente e outra(s) dependente(s).

Representação do modelo

Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Notação:

- **m**: número de exemplos (amostras) de treinamento.
- **x'**s: variáveis de entrada / atributos / características
- **y'**s: variáveis de saída / variáveis “alvos” (*target*)

Representação do modelo

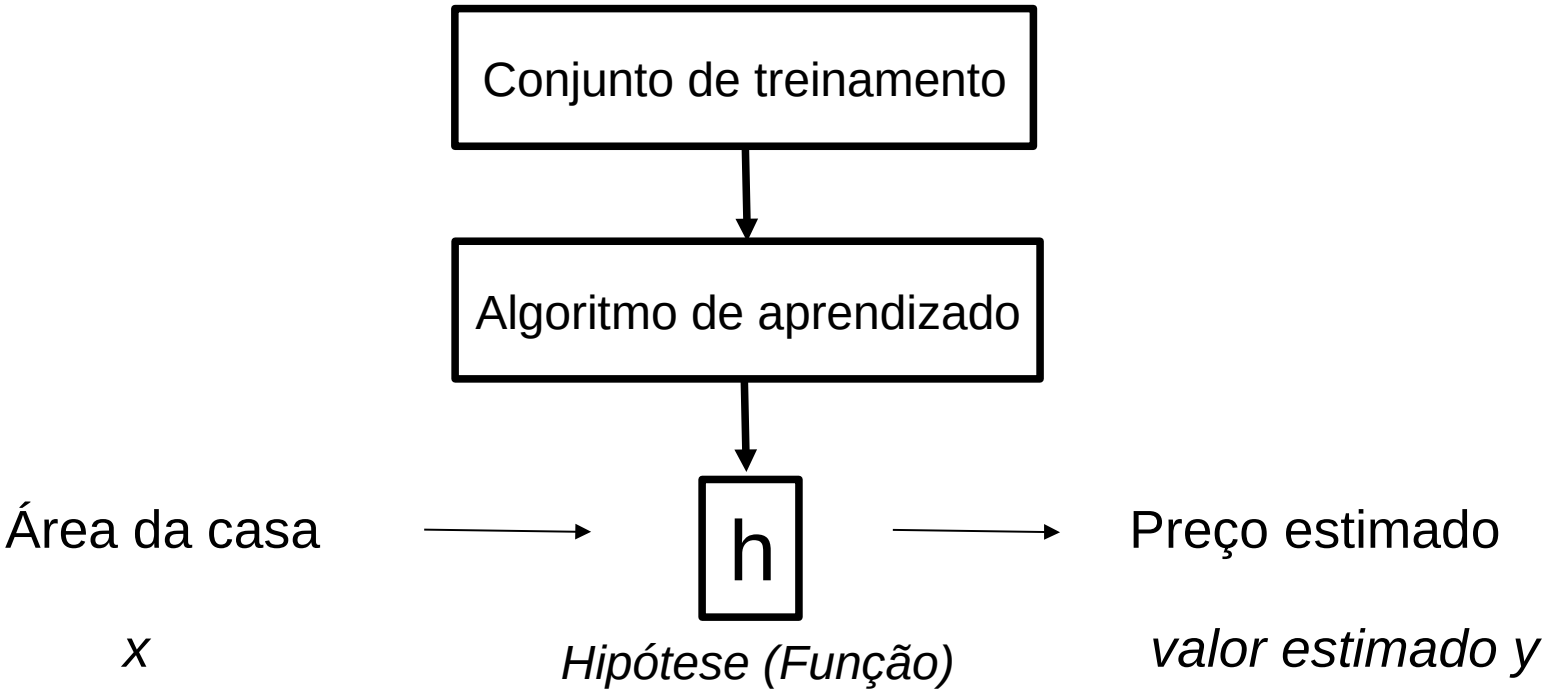
Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

- (x, y): denota uma amostra qualquer do conjunto de treinamento.
- $(x^{(i)}, y^{(i)})$: denota uma amostra específica (i-ésima amostra). Onde **i** é o índice.

$$y^{(3)} = 315$$

Representação do modelo



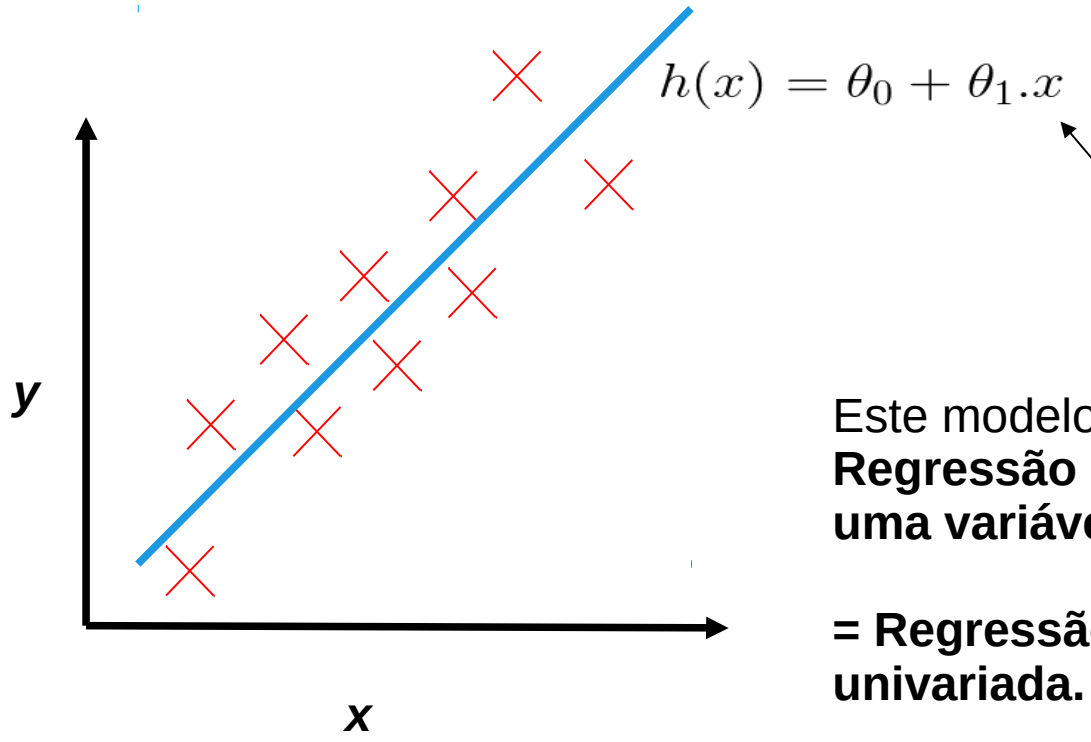
h* é uma função que mapeia *x* a *y

Representação do modelo

Como representar ***h***?

$$h_\theta(x) = \theta_0 + \theta_1.x$$

$$h(x) = \theta_0 + \theta_1.x \qquad \leftarrow \textit{forma abreviada}$$



Este modelo é uma:
Regressão linear com uma variável

= Regressão linear univariada.

Regressão linear

- Expressa a 'classe' (atributo alvo) como uma combinação linear dos outros atributos com alguns pesos, p.ex:

$$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

em que θ_i é um peso, e x_i é um atributo

- O valor previsto para um exemplo é calculado com base nos valores dos atributos do exemplo.
  - Então é necessário **aprender os pesos que minimizam o erro** entre o valor real e o valor previsto (**função custo**) nos exemplos de treinamento.

Regressão linear com uma variável

(B) Função custo

Função custo

- A função custo nos permitirá definir a melhor função utilizada para aproximar a linha reta para o conjunto de treinamento.

Conjunto de treinamento:

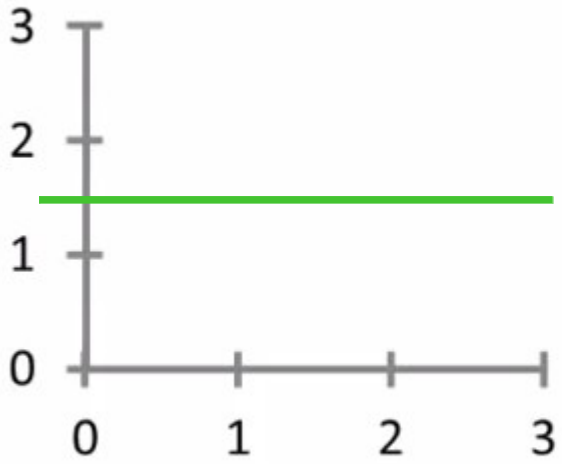
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Hipotese: $h_\theta(x) = \theta_0 + \theta_1.x$

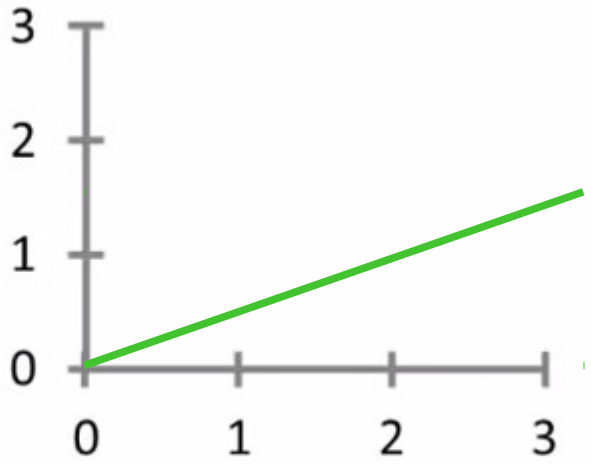
θ_i são os parâmetros (pesos). Como definir esses parâmetros?

Função custo

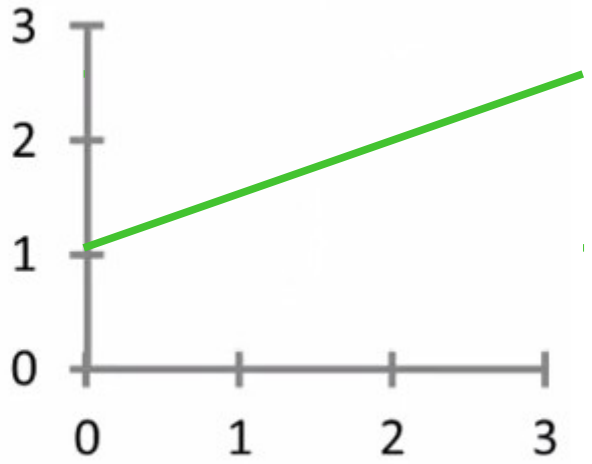
$h_\theta(x) = \theta_0 + \theta_1 x$



$\theta_0 = 1.5$
$\theta_1 = 0$



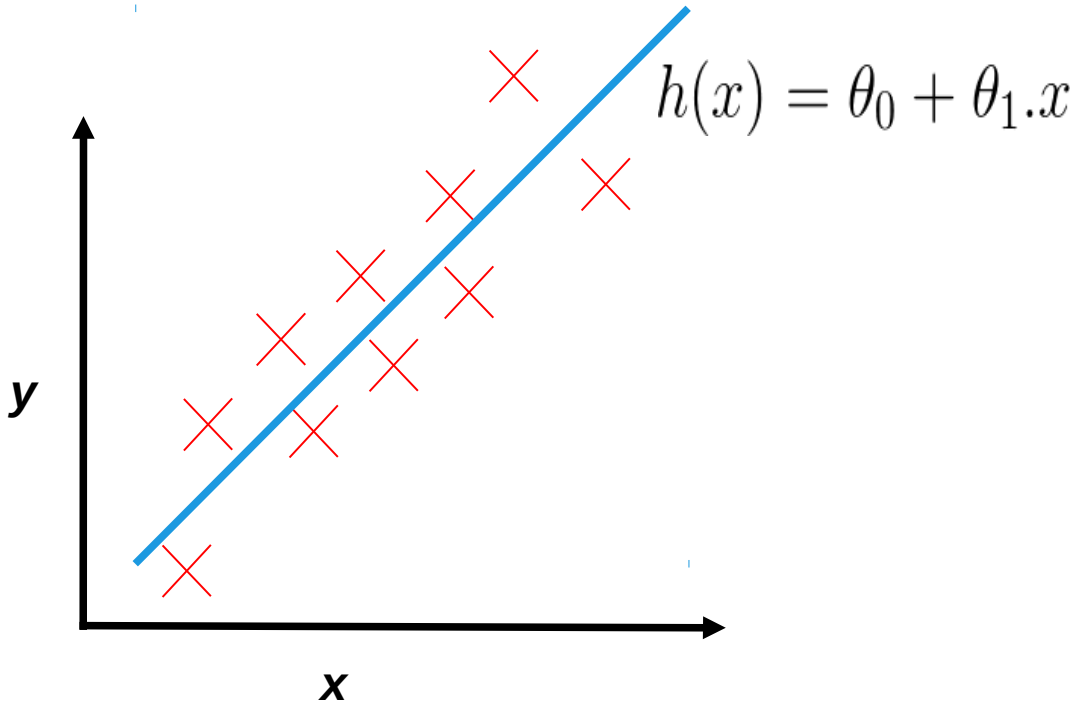
$\theta_0 = 0$
$\theta_1 = 0.5$



$\theta_0 = 1$
$\theta_1 = 0.5$

$h(x) = 1.5$ $\quad$ $h(x) = 0.5x$ $\quad$ $h(x) = 1 + 0.5x$

Função custo



- Como escolher os parâmetros (θ_0 e θ_1) que permitam ajustar a melhor reta aos dados do conjunto de treinamento?
  - **IDEIA:** Escolher θ_0 e θ_1 tal que **h(x)** **seja o mais próximo** dos valores de **y** de acordo com o **conjunto de treinamento** **(x,y)**.

Função custo

Minimizar θ_0 e θ_1

$$h_\theta(x^{(i)}) - y^{(i)}$$

$$\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\sum_{i=1}^{m} |h_\theta(x^{(i)}) - y^{(i)}|$$

Função custo

Minimizar θ_0 e θ_1

$$h_\theta(x^{(i)}) - y^{(i)}$$

$$\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})$$

$$\sum_{i=1}^{m}|h_\theta(x^{(i)}) - y^{(i)}|$$

$$\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$$

Função custo

Função custo: $$J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Objetivo: Descobrir θ_0 e θ_1 que minimize $J(\theta_0,\theta_1)$

Essa função custo também é conhecida como Erro Quadrático Médio (EQM).

Função custo – intuição

Hipótese: $h_\theta(x) = \theta_0 + \theta_1.x$

Parâmetros: θ_0, θ_1

Função custo: $J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Objetivo: Descobrir θ_0 e θ_1 que minimize $J(\theta_0, \theta_1)$

Função custo – intuição

Hipótese simplificada:

$$\theta_0 = 0$$

$$h_\theta(x) = \theta_1 x$$

Parâmetros:

$$\theta_1$$

Função custo:

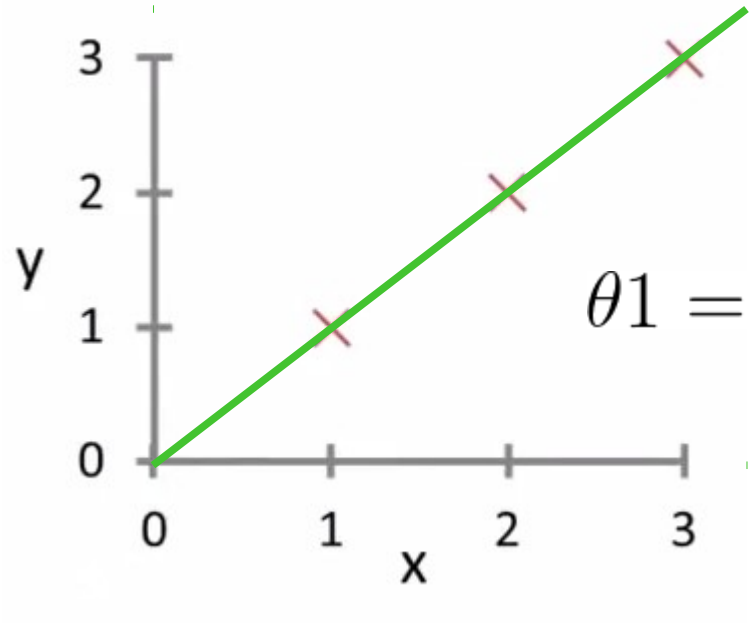
$$J(\theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_1 x^{(i)} - y^{(i)})^2$$

Objetivo: Descobrir θ_1 que minimize $J(\theta_1)$

Função custo – intuição

$h_\theta(x)$

(para um $\theta 1$, é uma função de x)

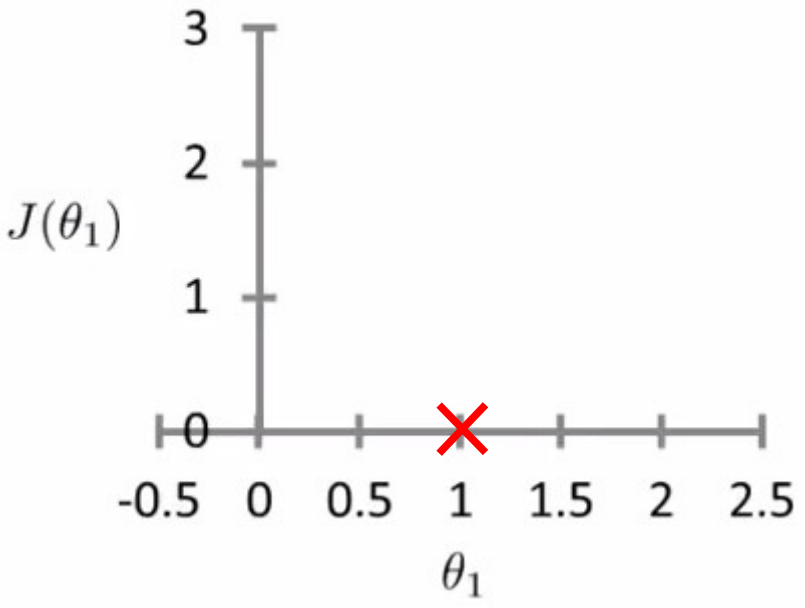


$$J(\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$= \frac{1}{m}(0^2 + 0^2 + 0^2)$$

$J(\theta 1)$

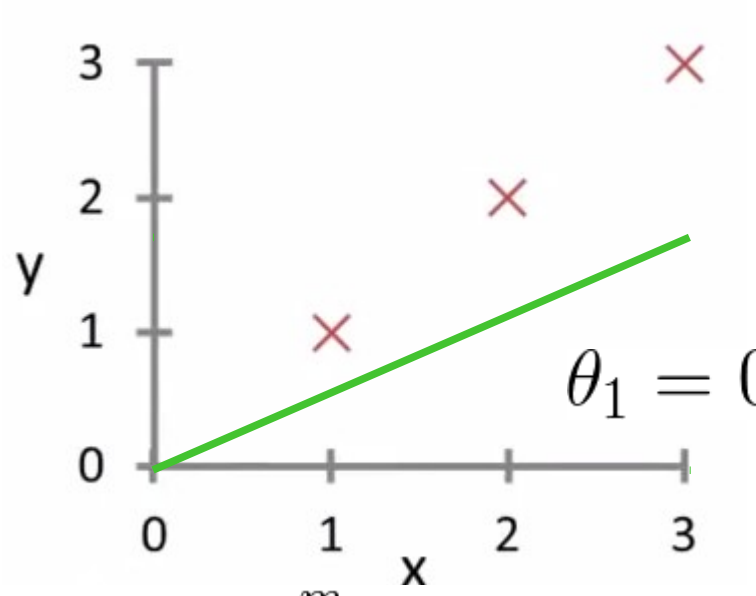
(função com parâmetro $\theta 1$)



Função custo – intuição

$h_\theta(x)$

(para um $\theta 1$, é uma função de x)

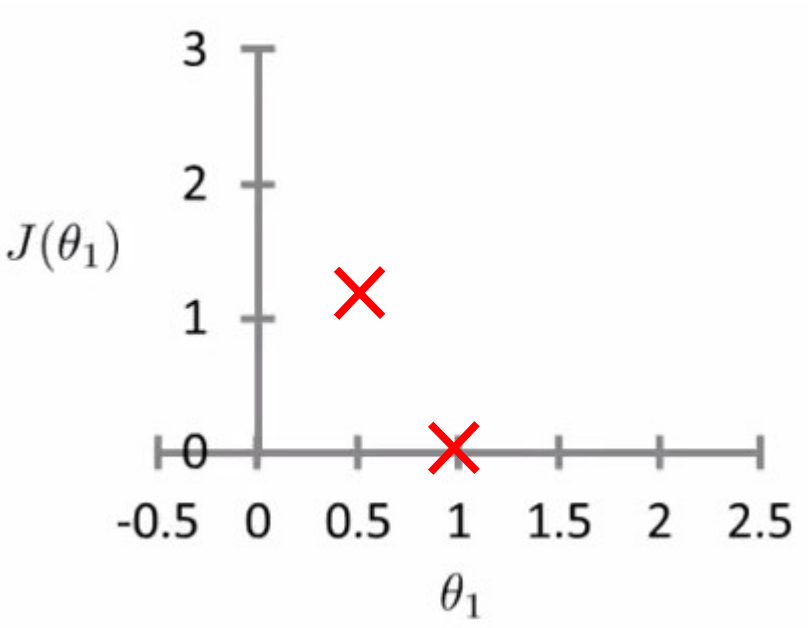


$$J(\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$J(0.5) = \frac{1}{3}[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2] \approx 1.17$$

$J(\theta 1)$

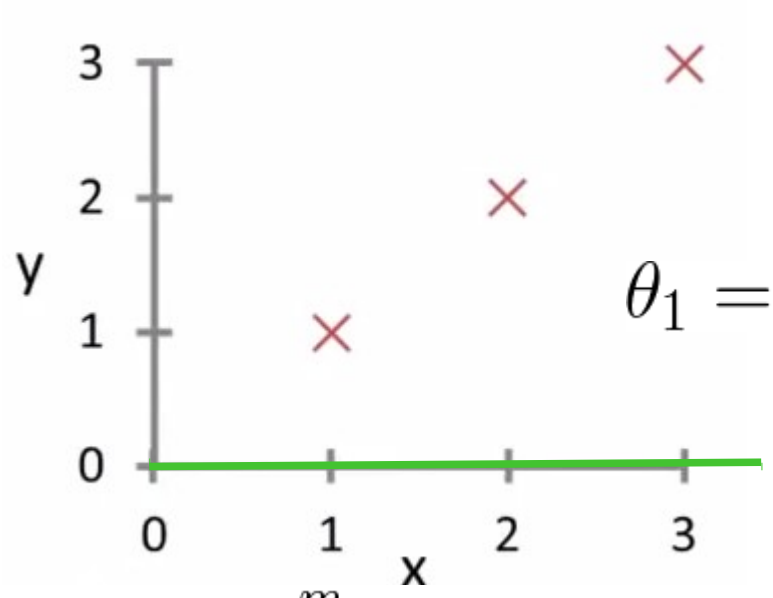
(função com parâmetro $\theta 1$)



Função custo – intuição

$h_\theta(x)$

(para um $\theta 1$, é uma função de x)



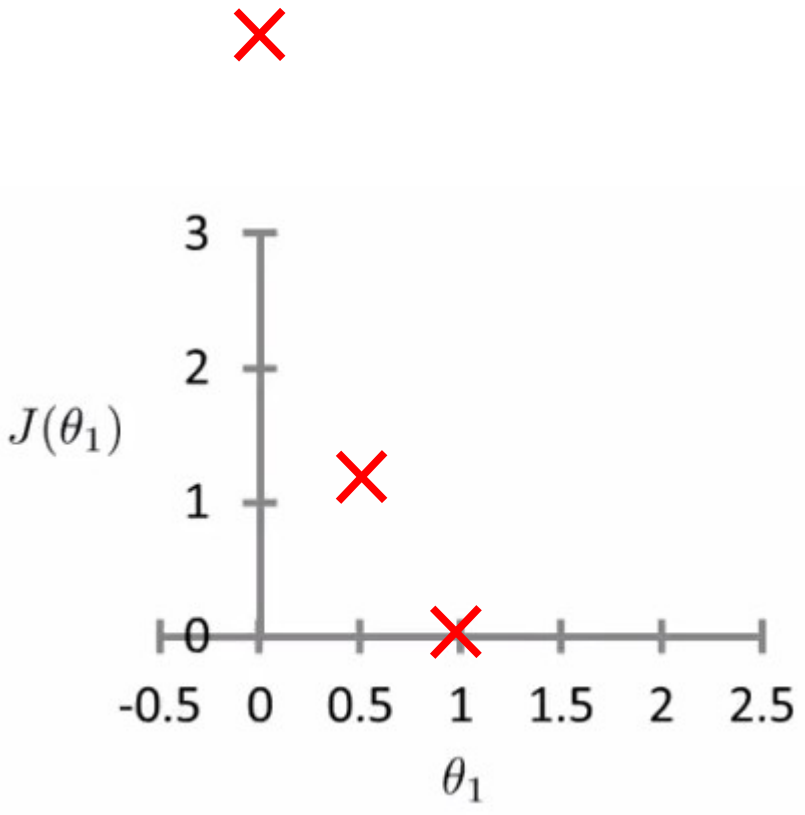
$\theta_1 = 0$

$$J(\theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\theta_1 x^{(i)} - y^{(i)})^2$$

$$J(0.5) = \frac{1}{3}(1^2 + 2^2 + 3^2) \approx 4.67$$

$J(\theta 1)$

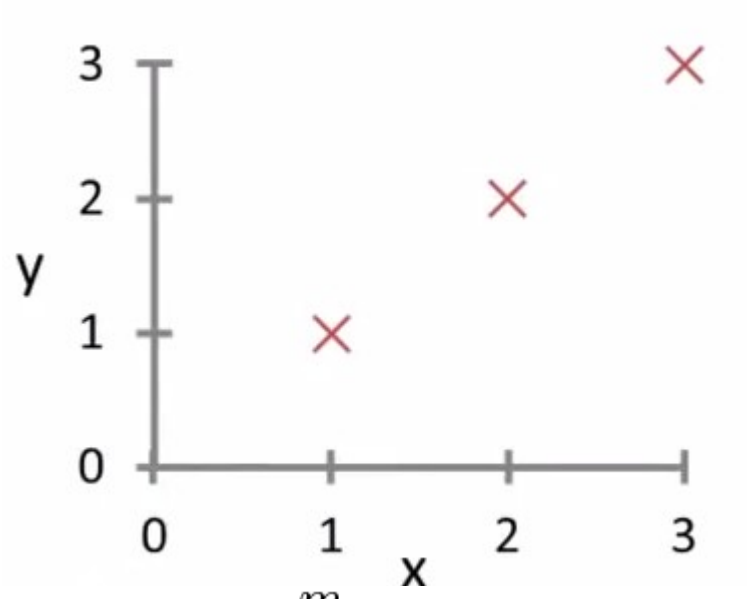
(função com parâmetro $\theta 1$)



Função custo – intuição

$h_\theta(x)$

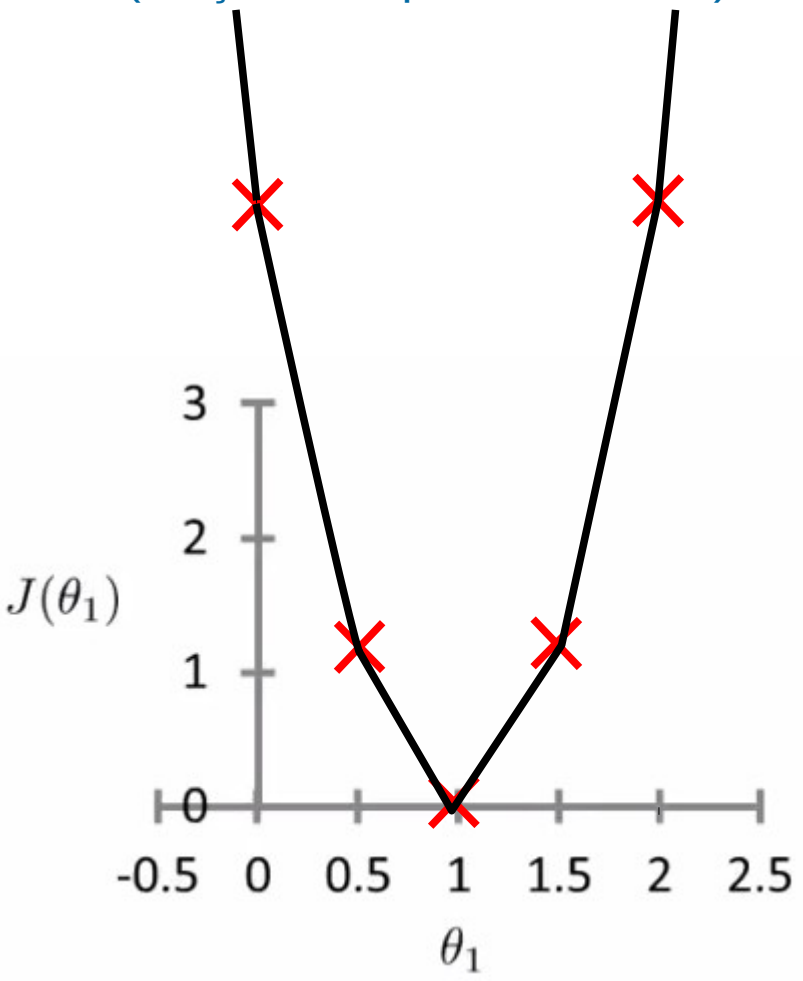
(para um $\theta 1$, é uma função de x)



$$J(\theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_1 x^{(i)} - y^{(i)})^2$$

$J(\theta 1)$

(função com parâmetro $\theta 1$)



Como buscar os parametros θ?

- Como buscar os parâmetros θ que minimizam a função custo?
  - **Função Gradiente Descendente**

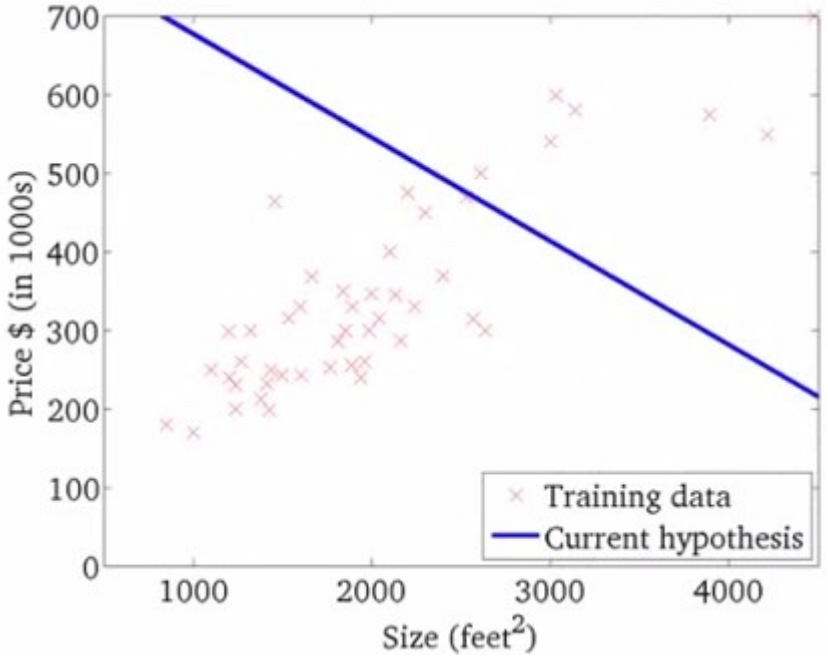
Função custo

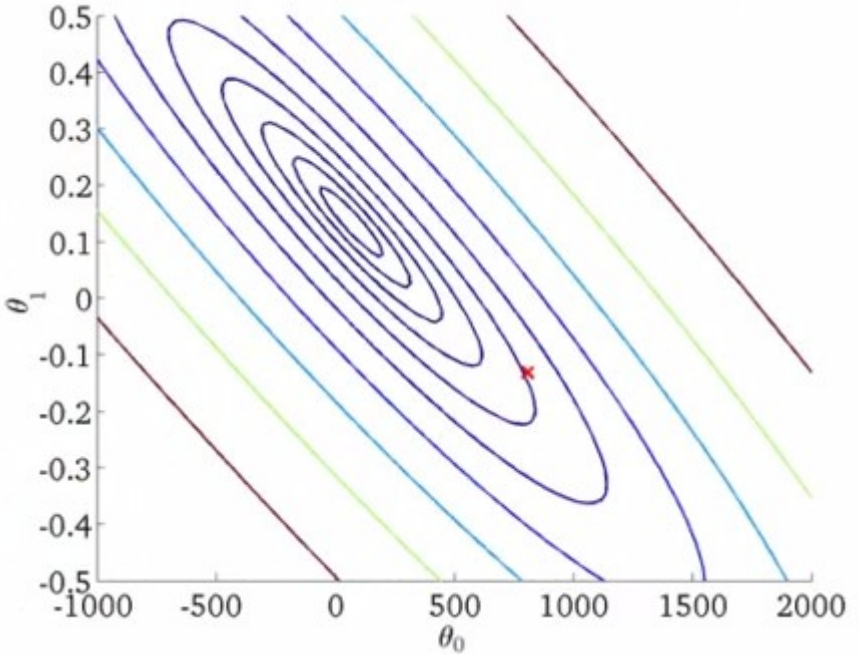
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





bacia

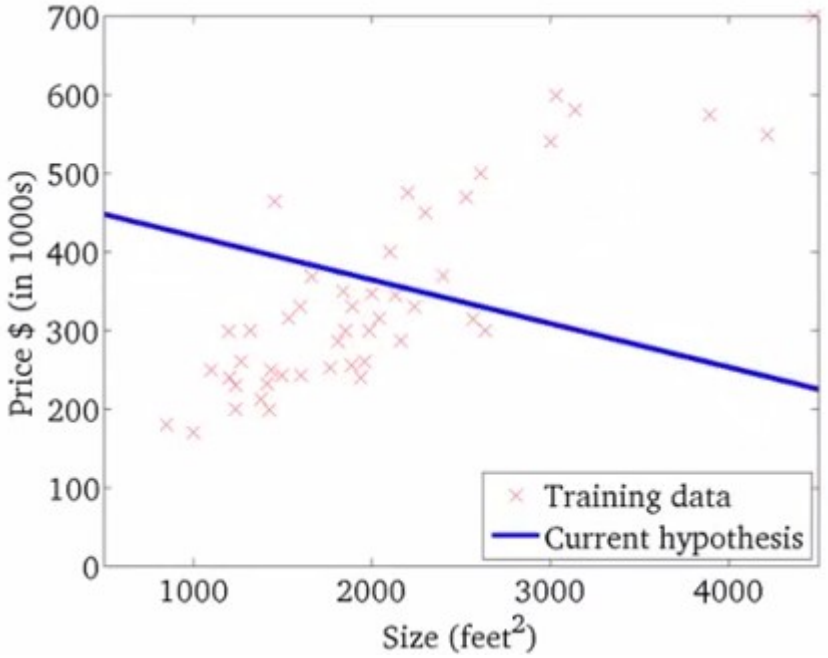
Função custo

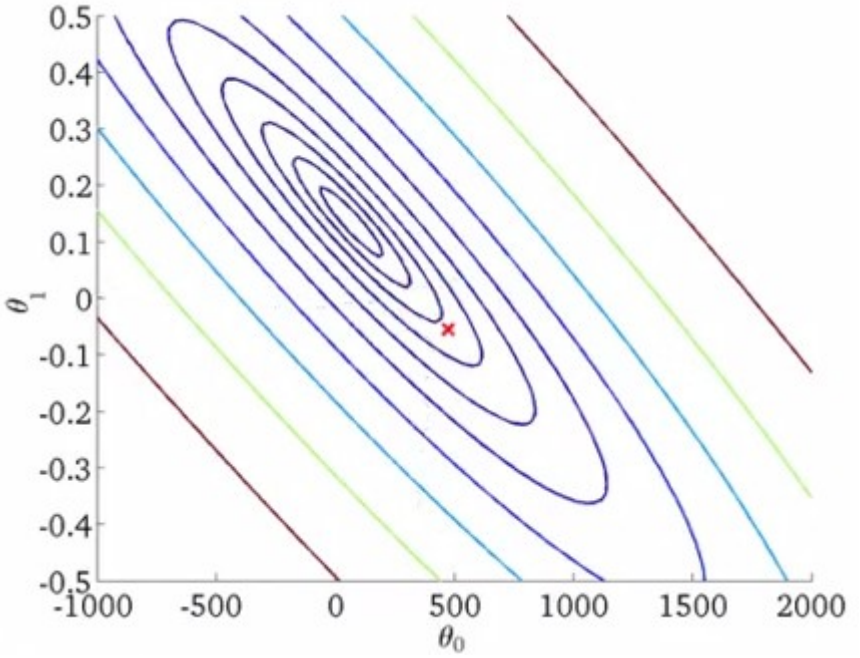
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





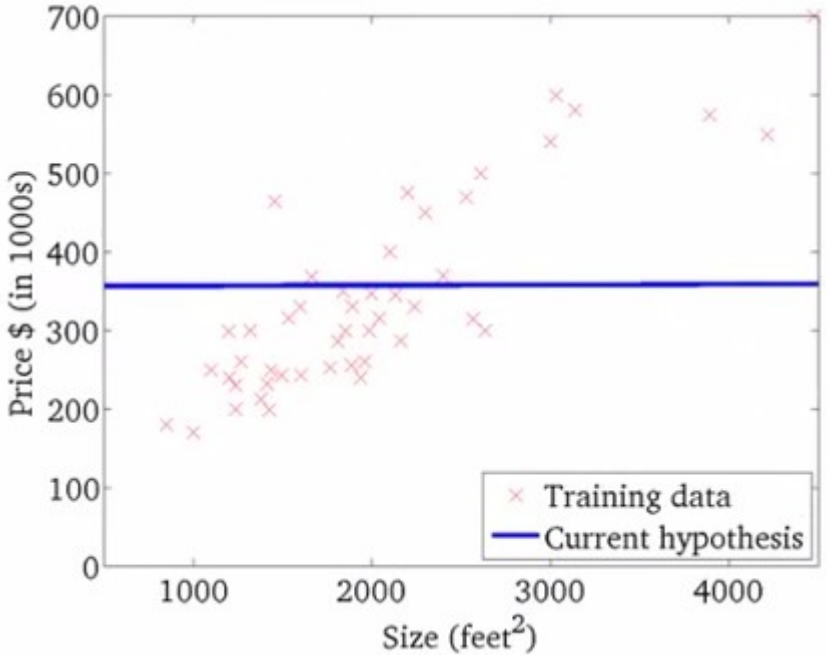
Função custo

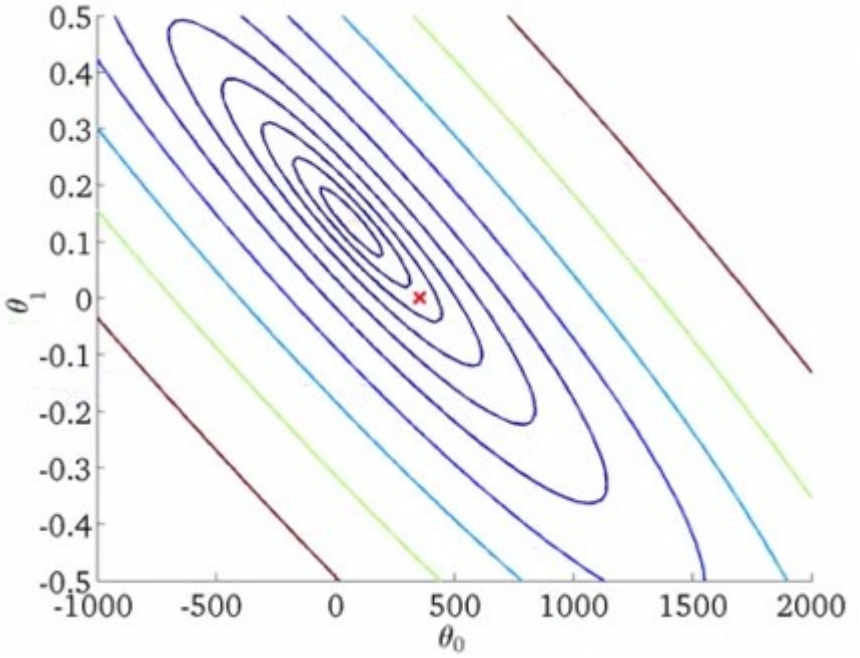
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





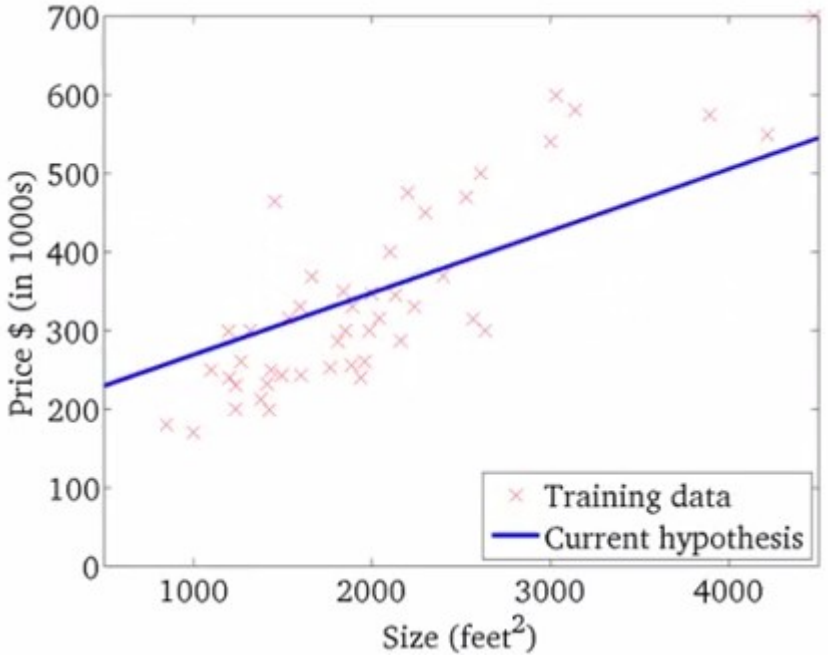
Função custo

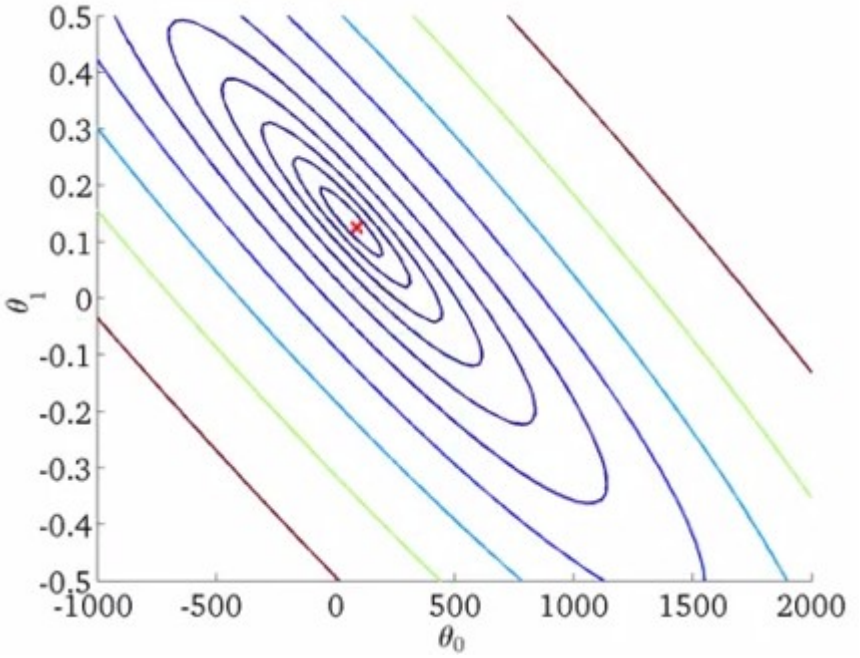
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





Gradiente descendente

- Temos uma função $J(\theta_0,\theta_1)$

- Desejamos descobrir θ_0 e θ_1 que minimize $J(\theta_0,\theta_1)$

- **Ideia:**
  - Iniciar com alguns valores para θ_0 e θ_1
  - Modifique os valores de θ_0 e θ_1 para reduzir $J(\theta_0,\theta_1)$ até atingir um valor mínimo.

Gradiente descendente

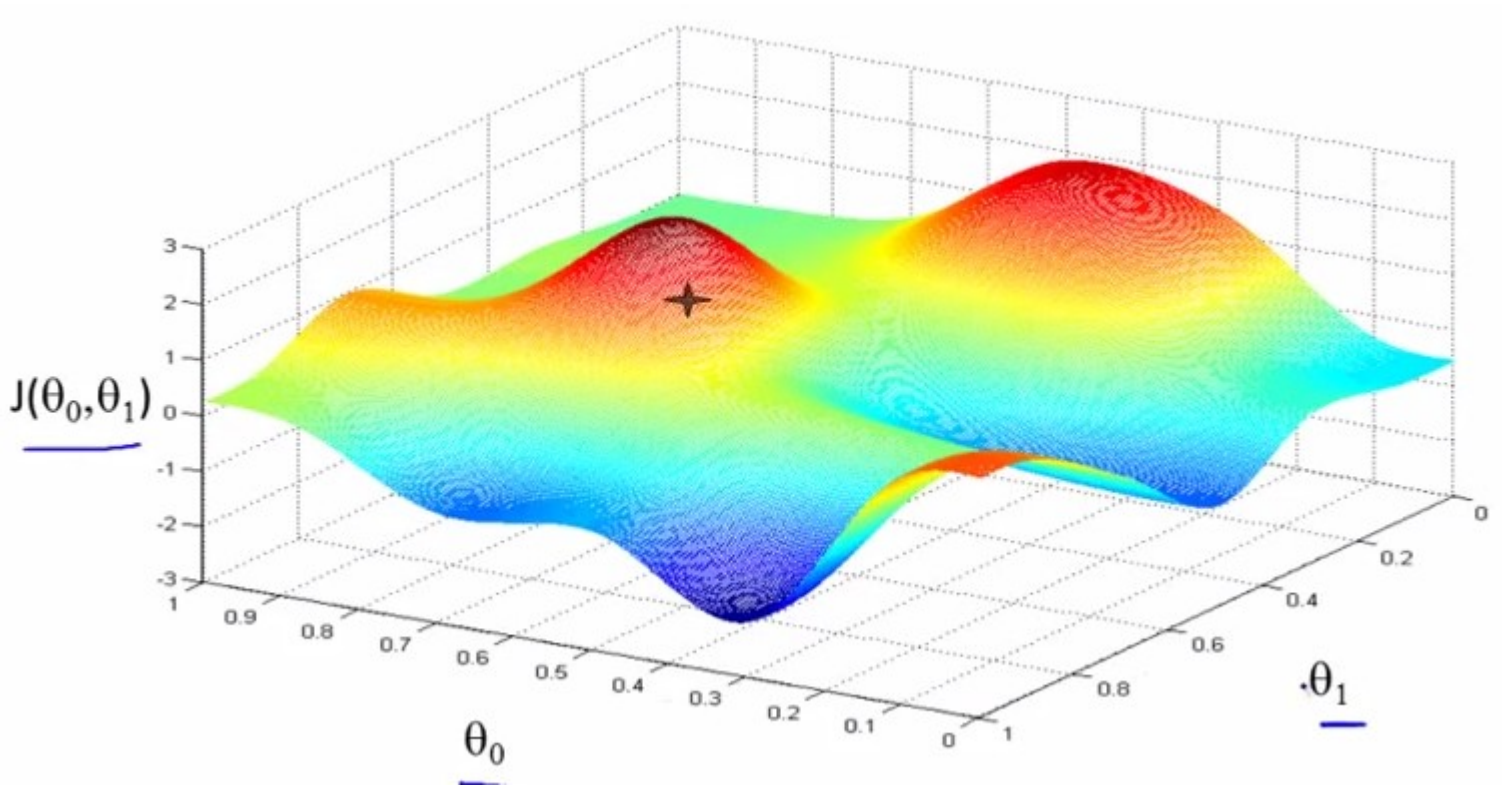
- Generalização para n dimensões
- Temos uma função $J(\theta_0,\theta_1,\ldots,\theta_n)$

- Desejamos descobrir $\theta_0,\theta_1,\ldots,\theta_n$ que minimize $J(\theta_0,\theta_1,\ldots,\theta_n)$

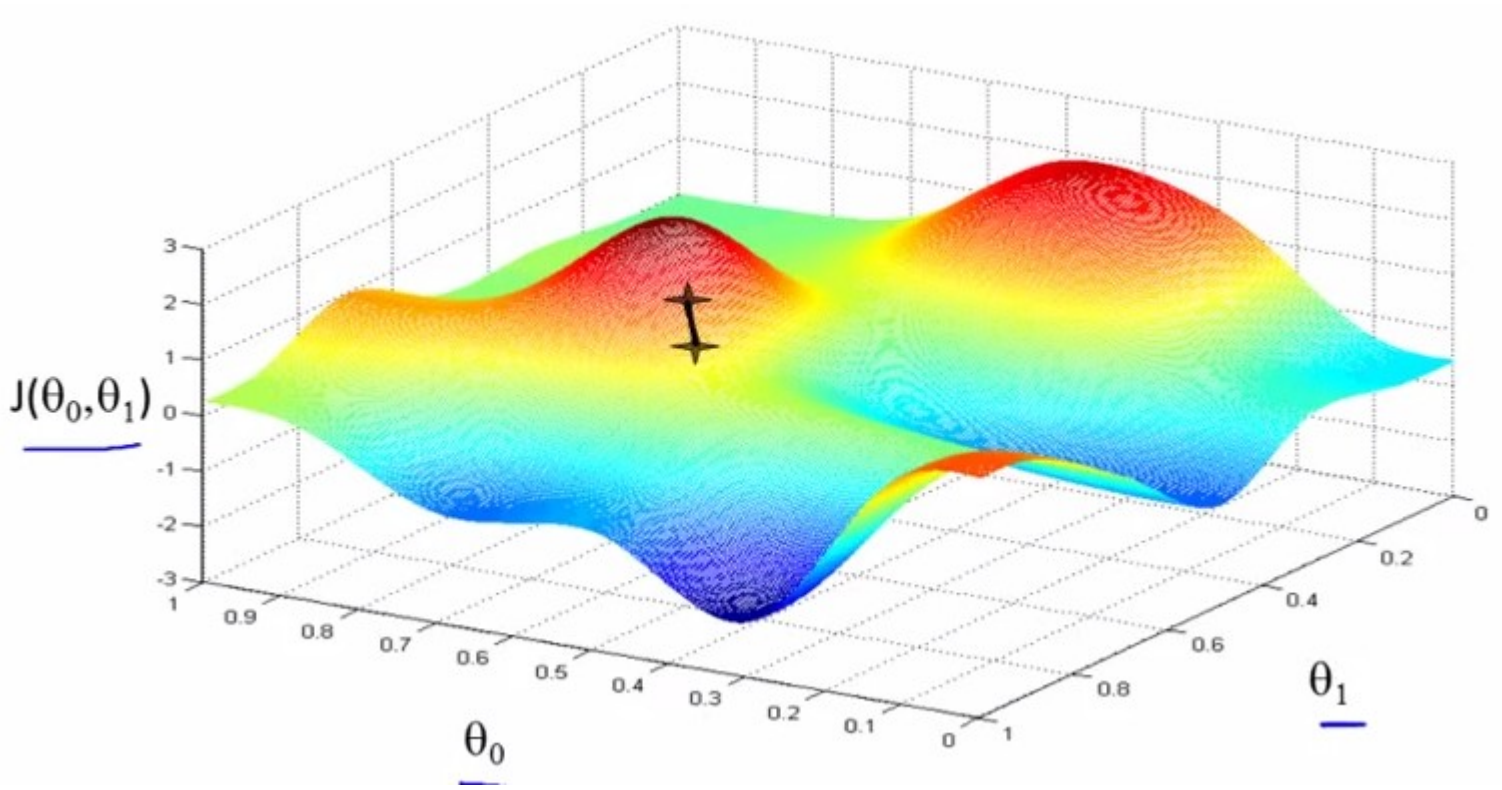
- **Ideia:**

  - Iniciar com alguns valores para $\theta_0,\theta_1,\ldots,\theta_n$
  - Modifique os valores de $\theta_0,\theta_1,\ldots,\theta_n$ para reduzir $J(\theta_0,\theta_1,\ldots,\theta_n)$ até atingir um valor mínimo.

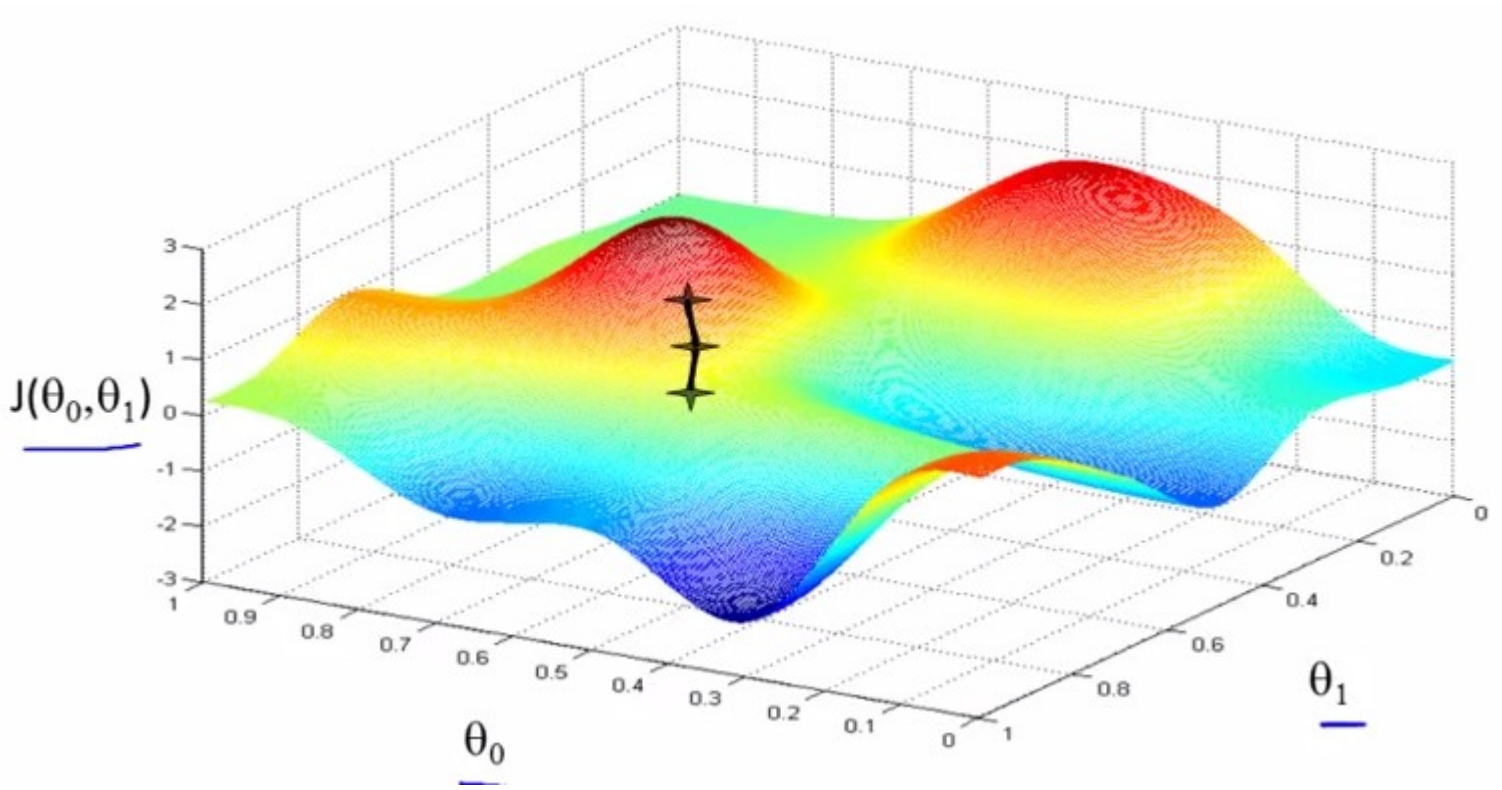
Gradiente descendente



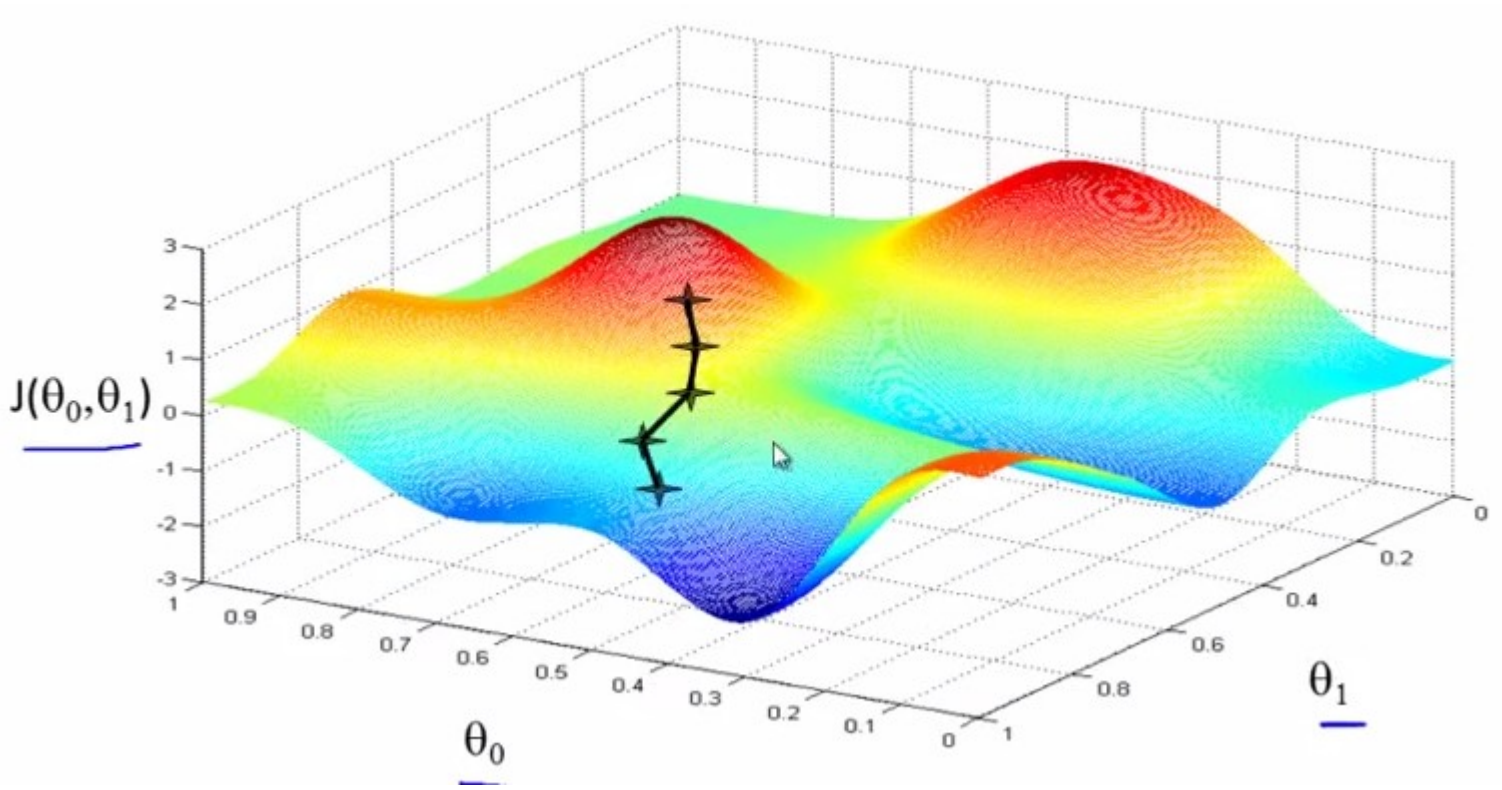
Gradiente descendente



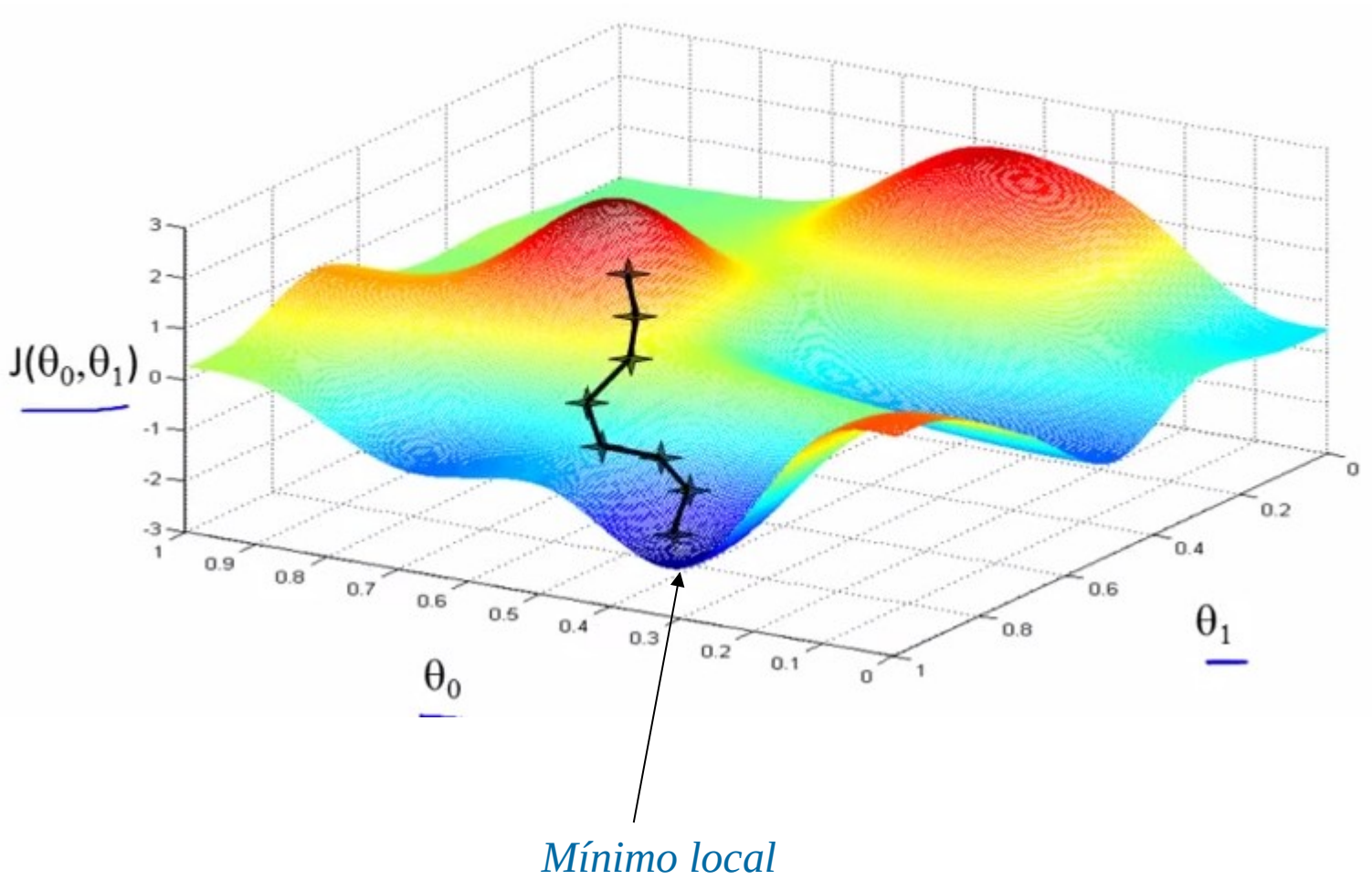
Gradiente descendente



Gradiente descendente

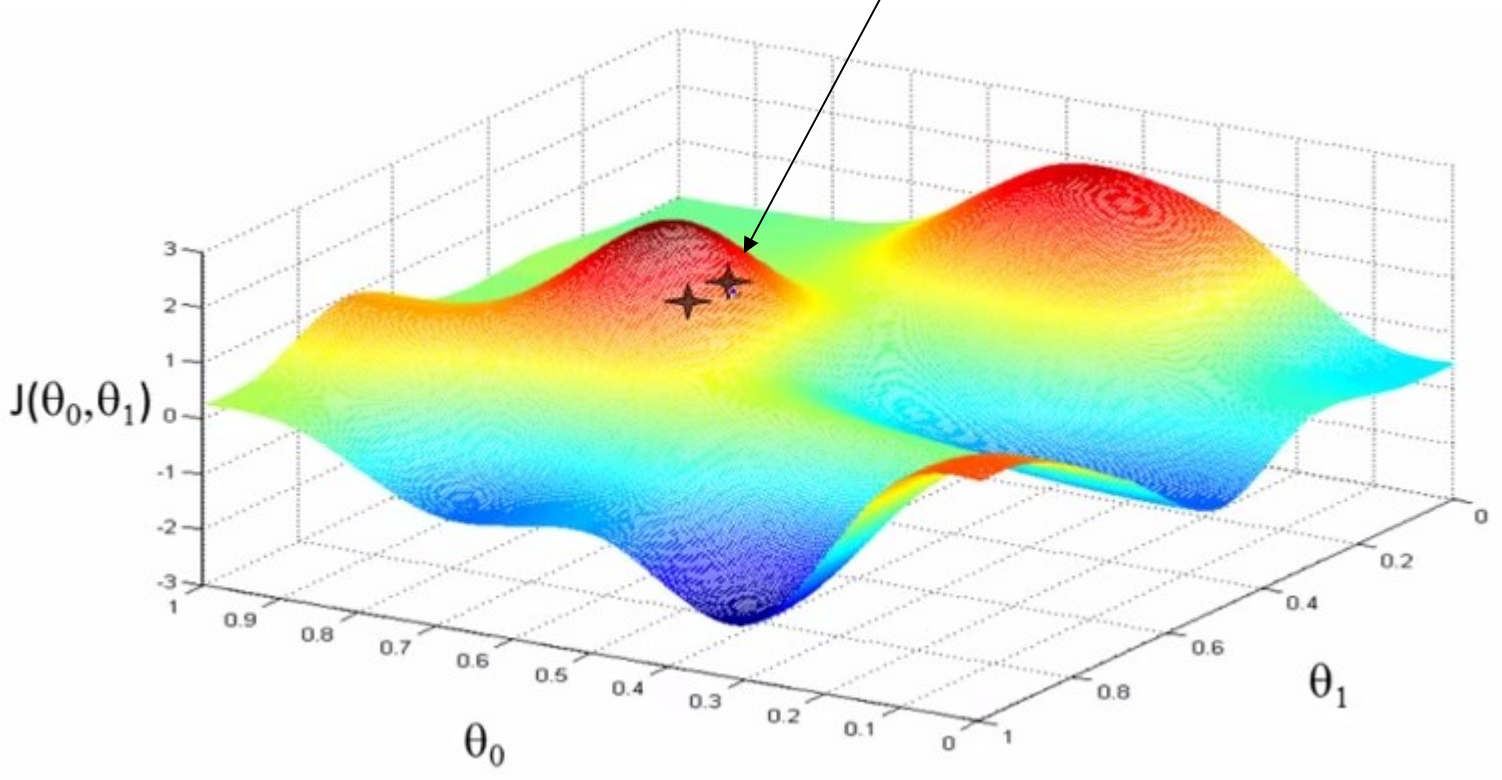


Gradiente descendente

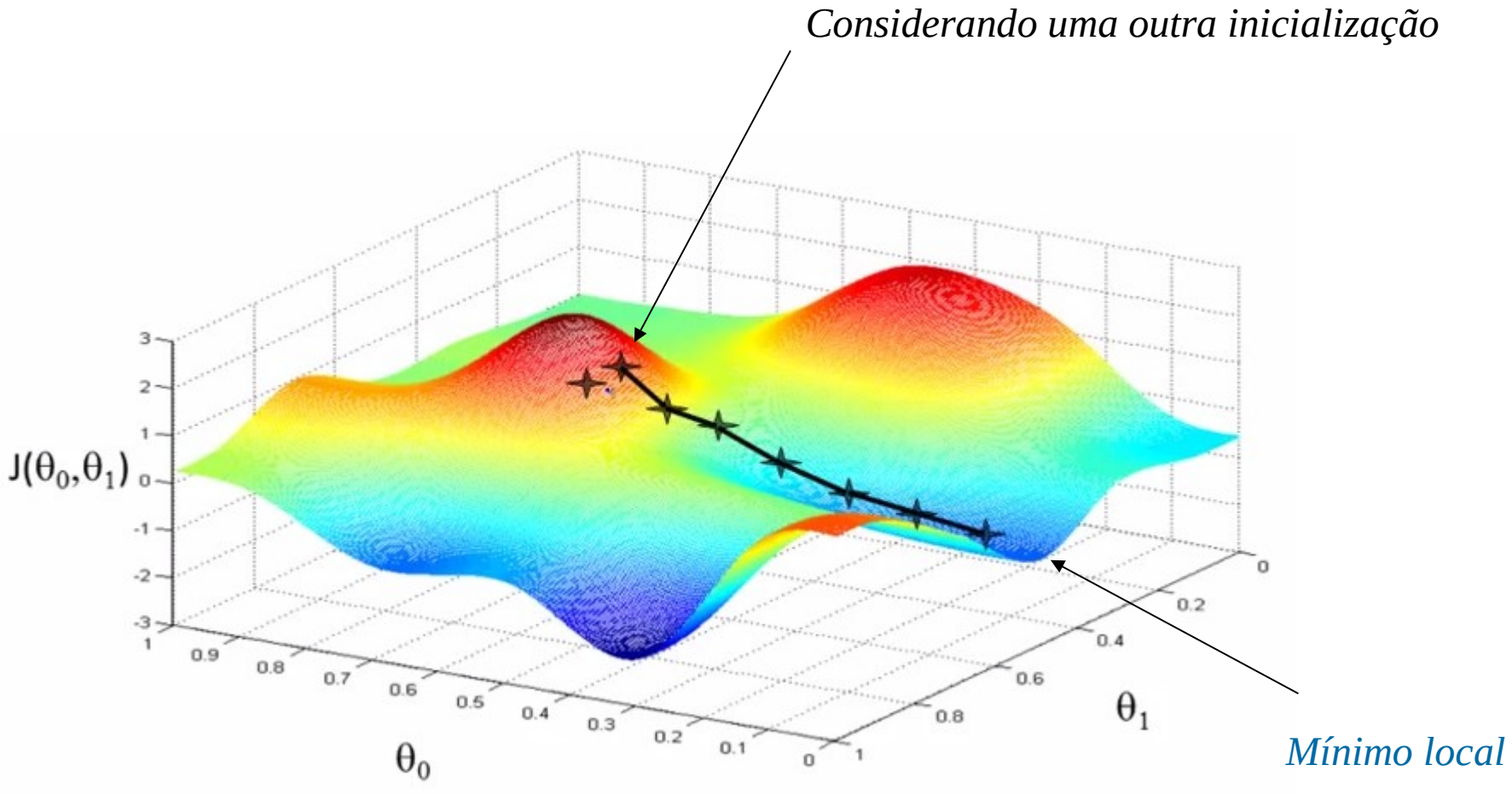


Gradiente descendente

Considerando uma outra inicialização



Gradiente descendente



Algoritmo gradiente descendente

- Repetir até a convergência:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta 1) \text{para j=0 e j=1}$$

- onde α é uma constante que indica a taxa de aprendizagem
  - controla o tamanho do passo (step)

- A implementação correta considera atualização simultânea:

$$\begin{aligned} \text{temp0} &:= \theta_0 - \alpha \tfrac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ \text{temp1} &:= \theta_1 - \alpha \tfrac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\ \theta_0 &:= \text{temp0} \\ \theta_1 &:= \text{temp1} \end{aligned}$$

Atribuição correta

$$\begin{aligned} \text{temp0} &:= \theta_0 - \alpha \tfrac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\ \theta_0 &:= \text{temp0} \\ \text{temp1} &:= \theta_1 - \alpha \tfrac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\ \theta_1 &:= \text{temp1} \end{aligned}$$

Atribuição incorreta

Algoritmo gradiente descendente

Repetir até a convergência:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta 1)$$ para j=0 e j=1

- Nos próximos exemplos apenas consideraremos a versão simplificada da função custo:

$$\text{Minimizar } \theta_1 \text{ de } J(\theta_1)$$

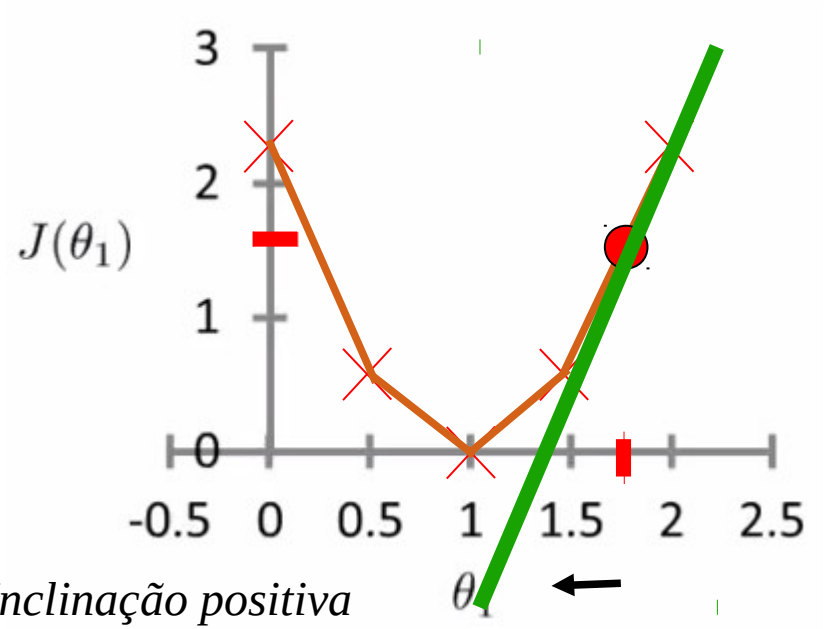
$$\theta_1 \in \mathbb{R}$$

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

Algoritmo gradiente descendente

$J(\theta 1)$

(função com parâmetro $\theta 1$)



Inclinação positiva

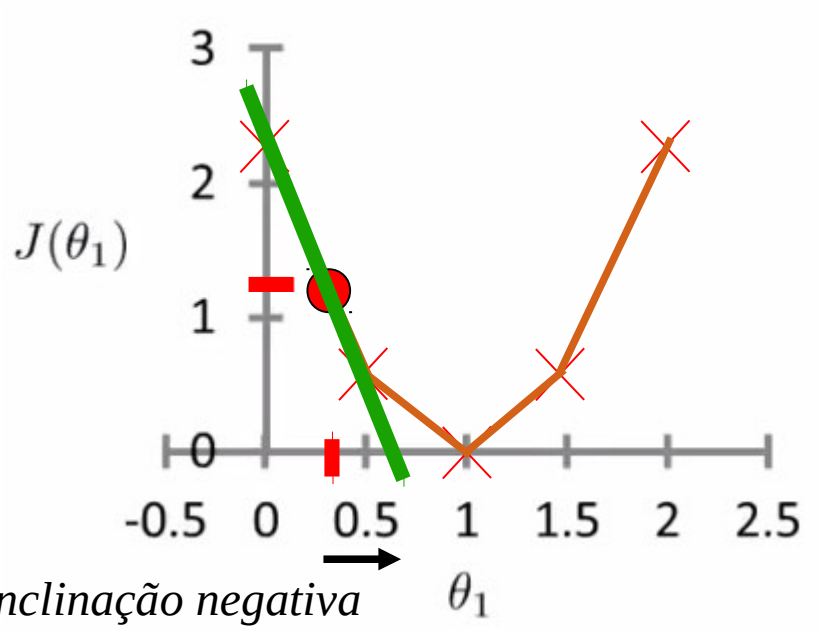
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

$$\theta_1 := \theta_1 - \alpha(\text{umValorPositivo})$$

Algoritmo gradiente descendente

$J(\theta 1)$

(função com parâmetro $\theta 1$)



Inclinação negativa

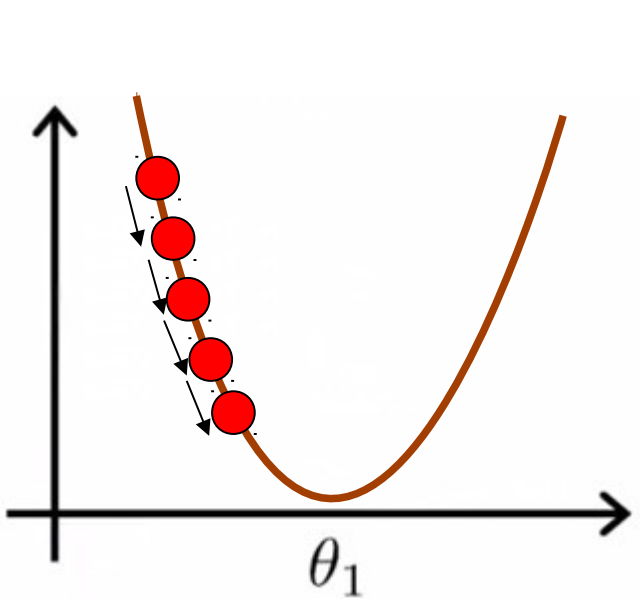
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

$$\theta_1 := \theta_1 - \alpha(\text{umValorNegativo})$$

Algoritmo gradiente descendente

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

•Se α for **muito pequeno** o gradiente descendente pode ser **lento**.

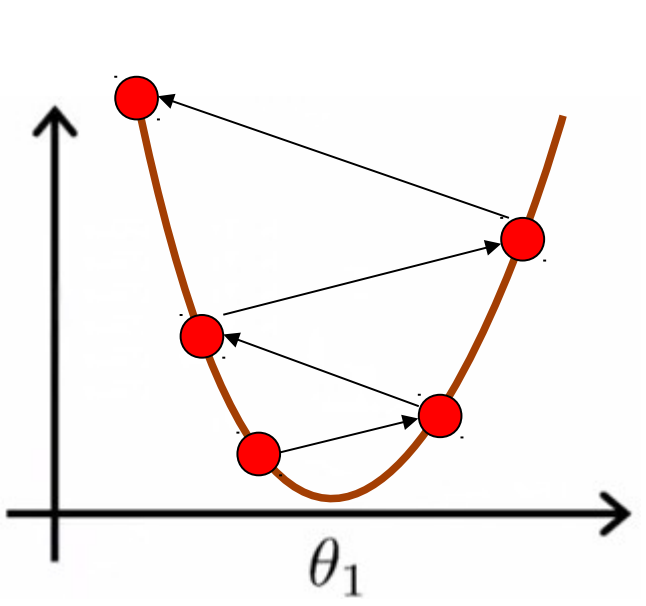


Algoritmo gradiente descendente

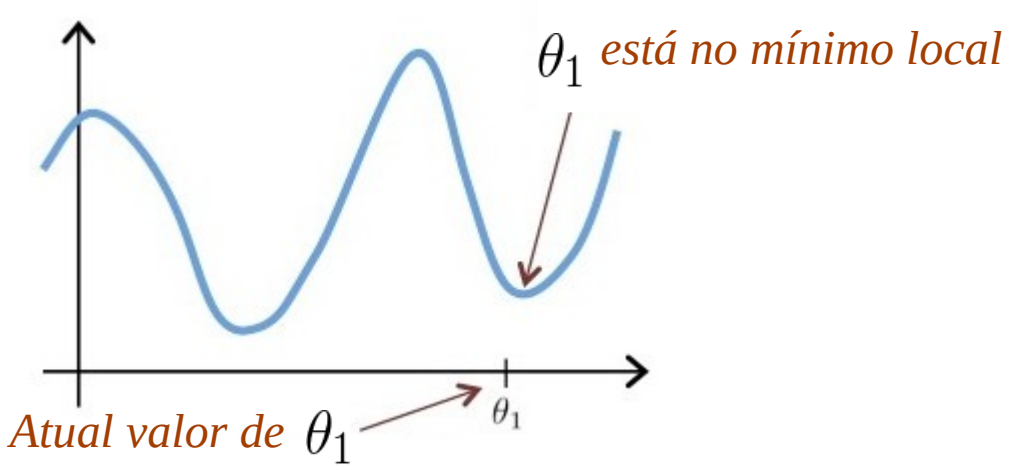
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

•Se α for **muito grande** o gradiente descendente pode **ultrapassar o mínimo**

•Pode falhar na convergência, ou até pode **divergir**.



Suponha θ_1 que é um ótimo local de $J(\theta_1)$, como apresentado na figura.



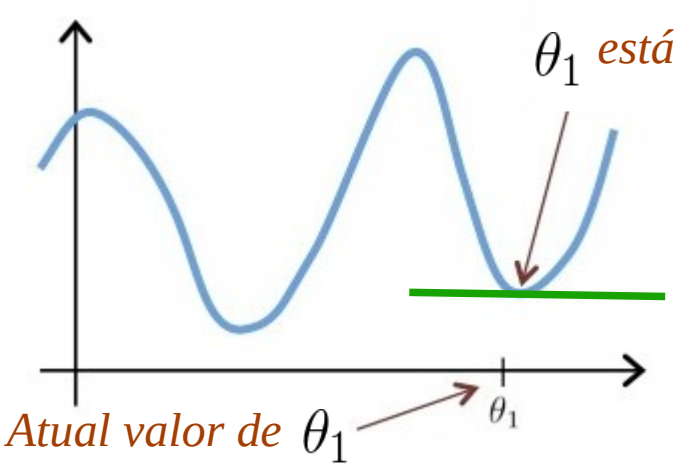
O seguinte passo do gradiente descendente: $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$

(A) Diminuirá o valor de θ_1.

(B) Deixará θ_1 sem modificação.

(C) Moverá θ_1 em uma direção aleatória.

(D) Moverá θ_1 na direção do mínimo global de $J(\theta 1)$.



$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

$$\theta_1 := \theta_1 - \alpha(0)$$

Resposta Correta (B)

Gradiente descendente + Regressão linear

- **Algoritmo gradiente descencente**

Repetir até a convergência:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta 1)$$

para j=0 e j=1

- **Modelo de regressão linear**

$$h_\theta(x) = \theta_0 + \theta_1 . x$$

$$J(\theta_0, \theta_1) = \frac{1}{2n} \sum_{i=1}^{n} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

•

Gradiente descendente + Regressão linear

•

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta 1) = \frac{\partial}{\partial \theta_j} . \frac{1}{2m} \sum \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{\partial}{\partial \theta_j} . \frac{1}{2m} \sum \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2$$

$$\frac{\partial (x^n)}{\partial x} = n.x^{(n-1)} \frac{\partial x}{\partial x} = n.x^{(n-1)}$$

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) . x^{(i)}$$

Gradiente descendente + Regressão linear

- **Algoritmo gradiente descendente**

Repetir até a convergência:

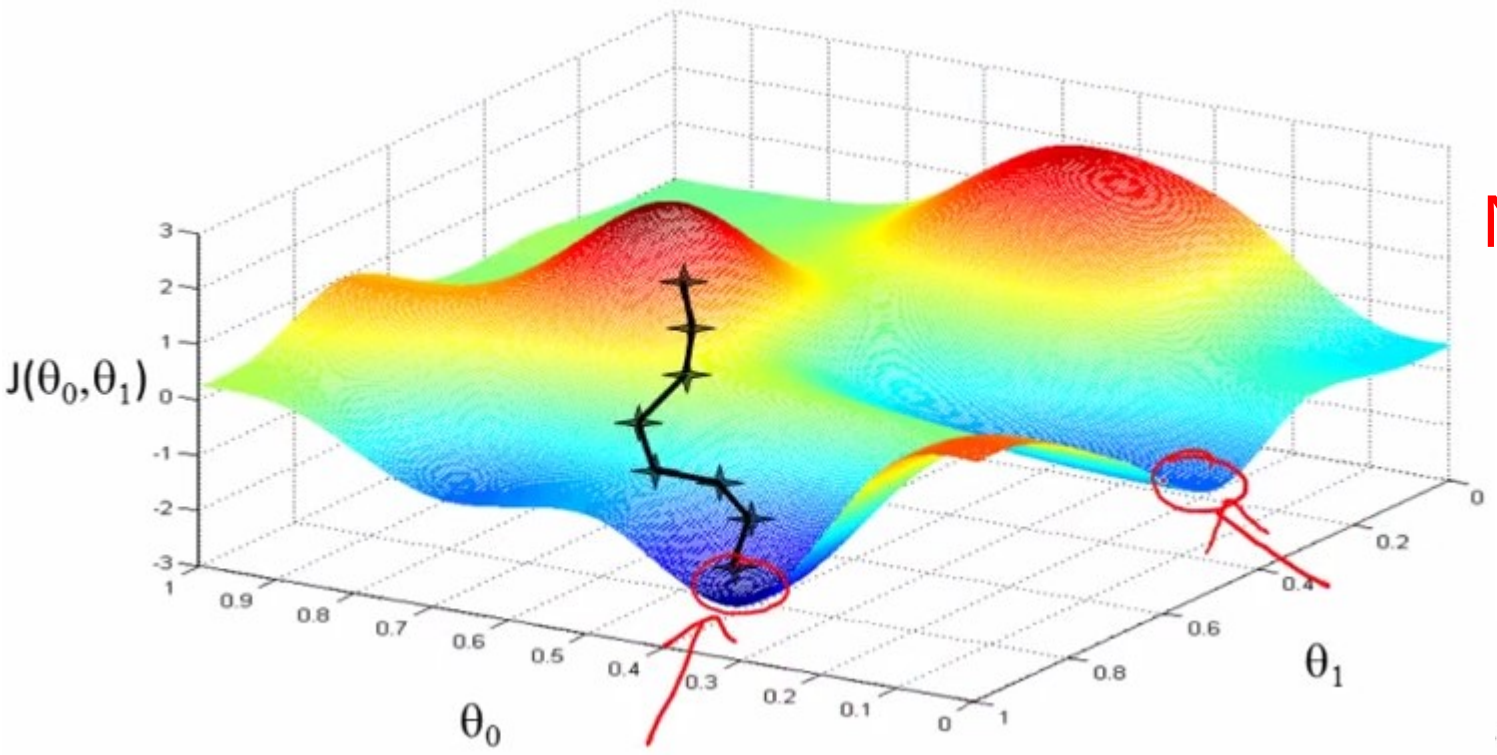
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta 1)$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) . x^{(i)}$$

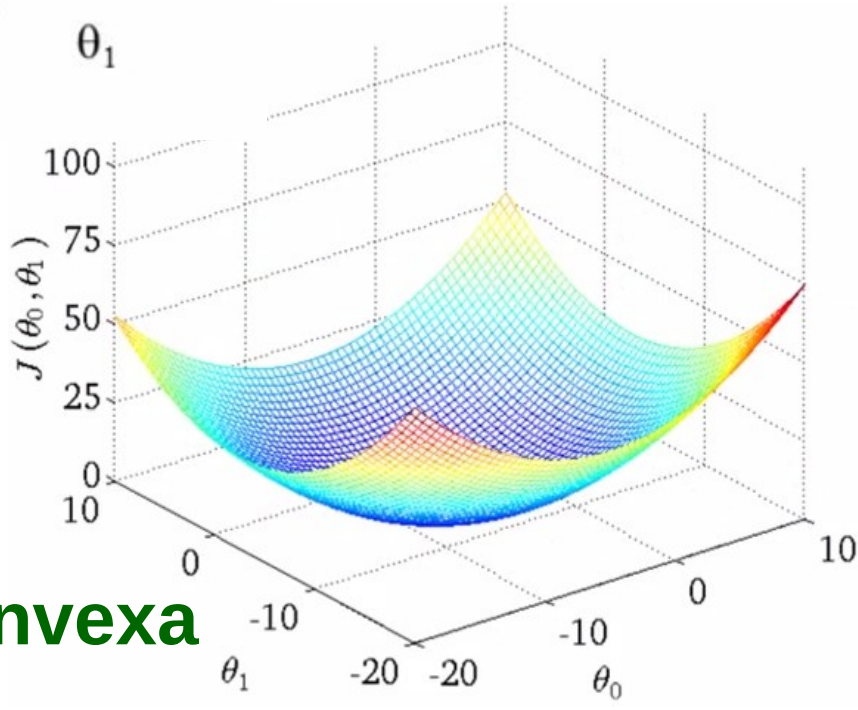
Repetir simultaneamente

Gradiente descendente + Regressão linear



Não-convexa

- *A função custo para uma **regressão linear**, sempre será uma função na forma de **parabola**.*
  - → ***função convexa***
  - → ***gradiente descendente garante o ótimo global***



Convexa

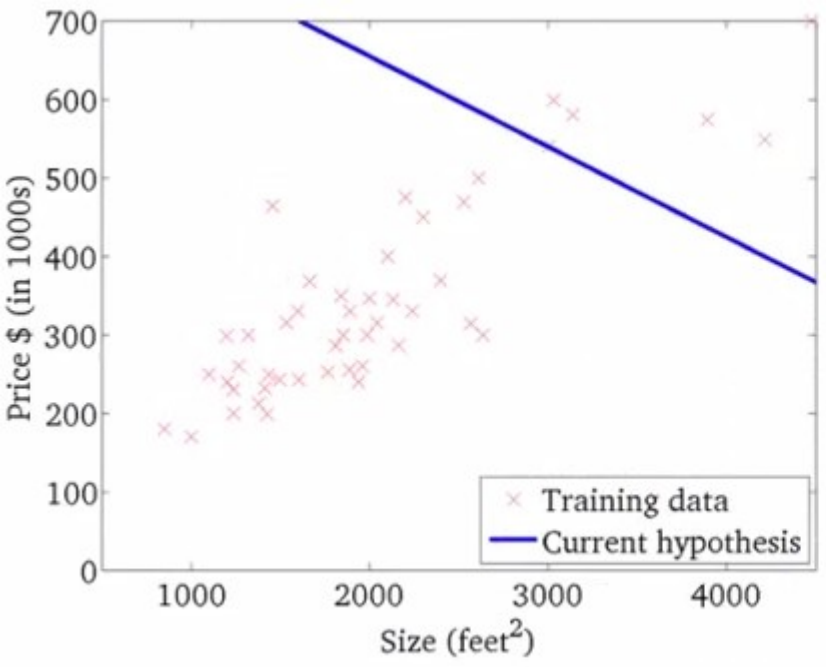
Gradiente descendente + Regressão linear

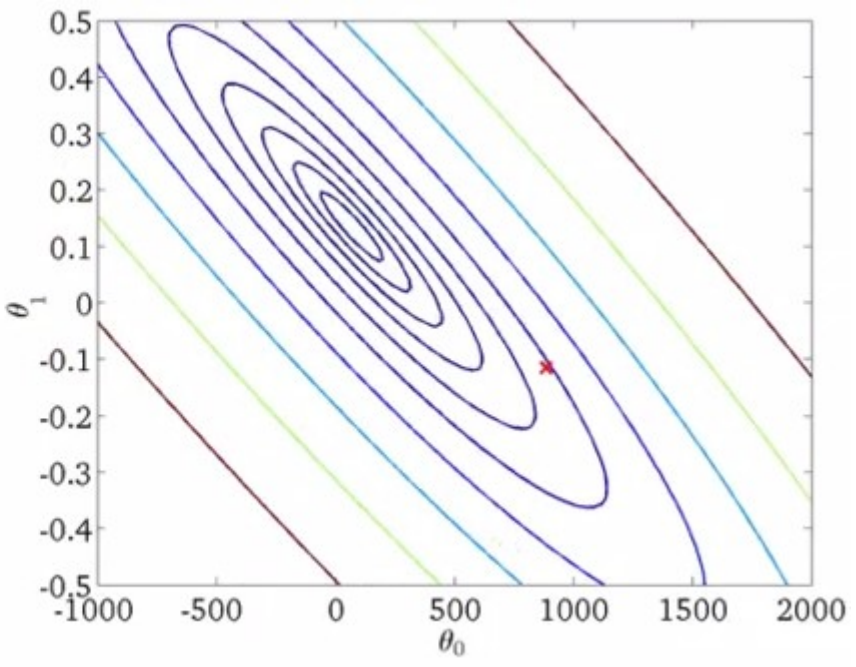
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





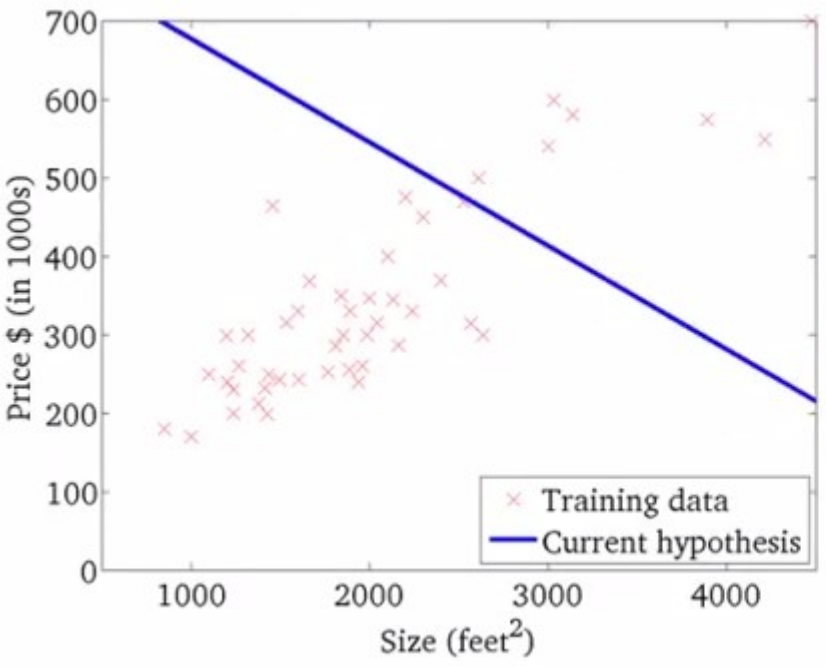
Gradiente descendente + Regressão linear

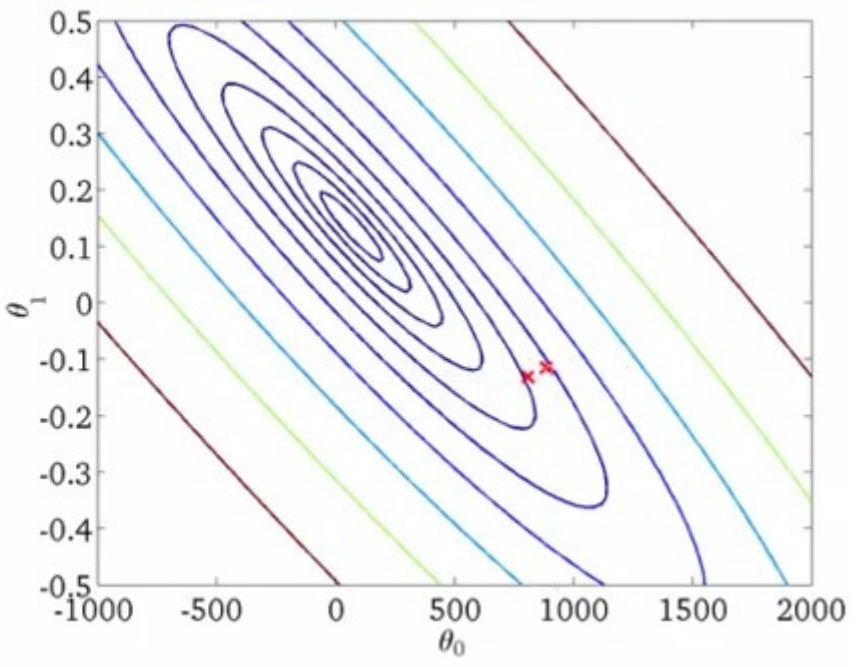
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





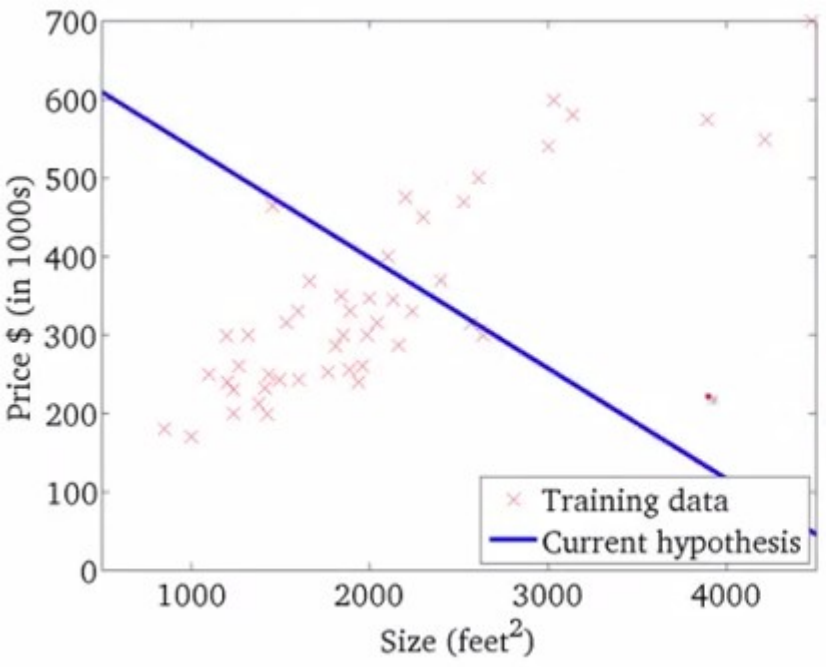
Gradiente descendente + Regressão linear

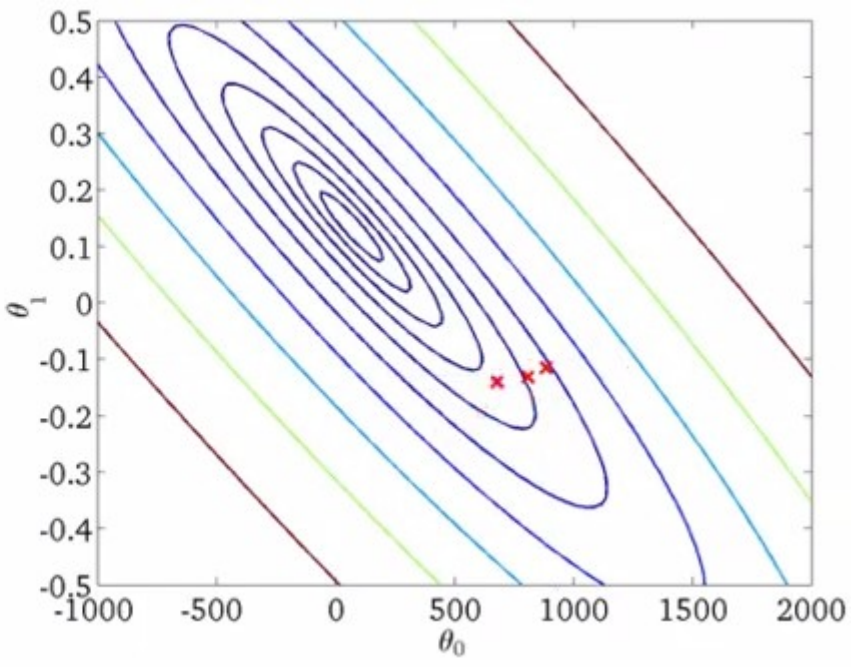
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





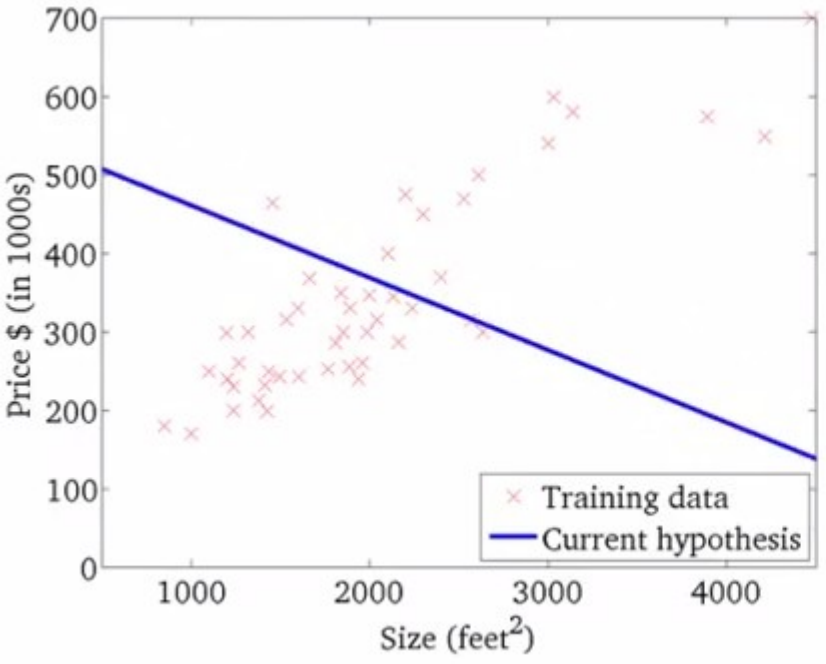
Gradiente descendente + Regressão linear

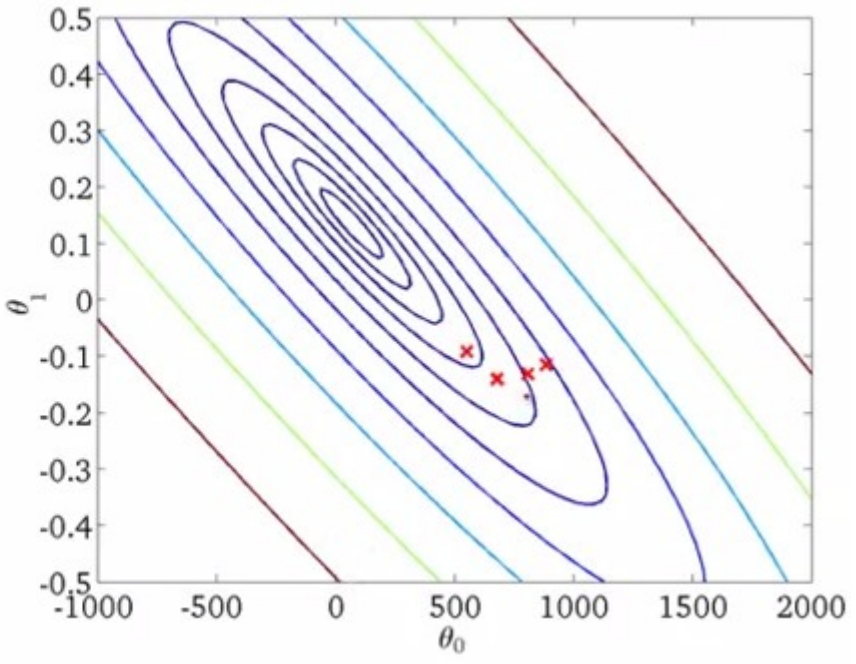
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





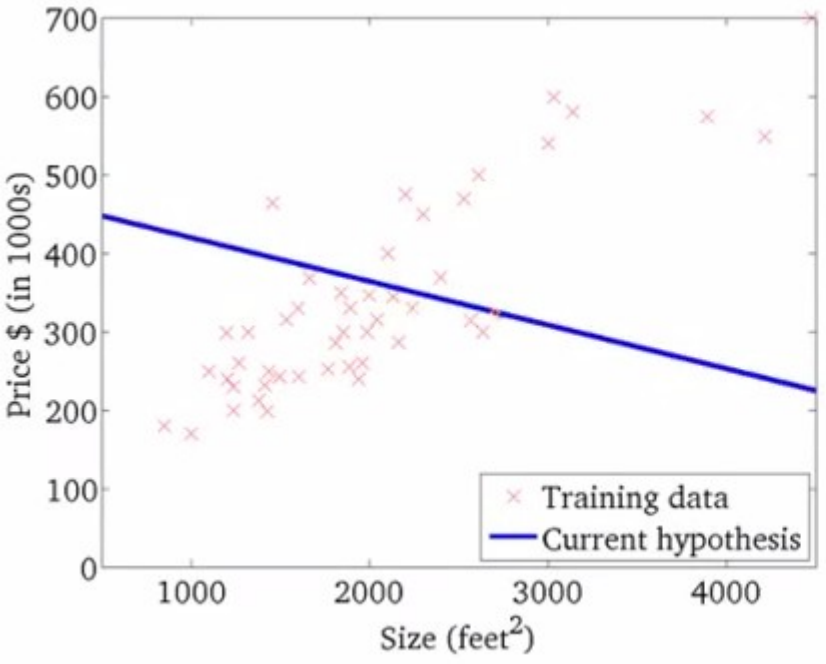
Gradiente descendente + Regressão linear

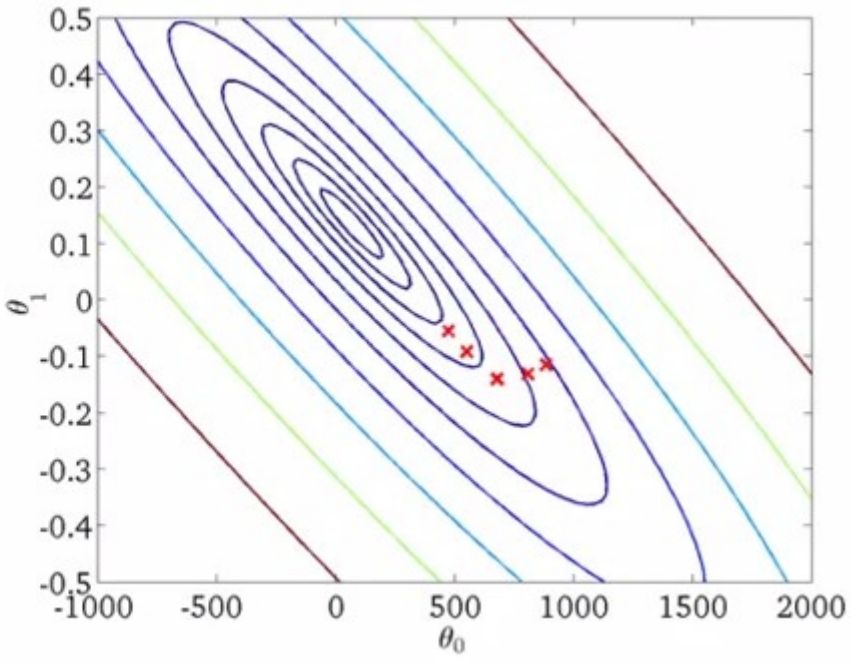
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





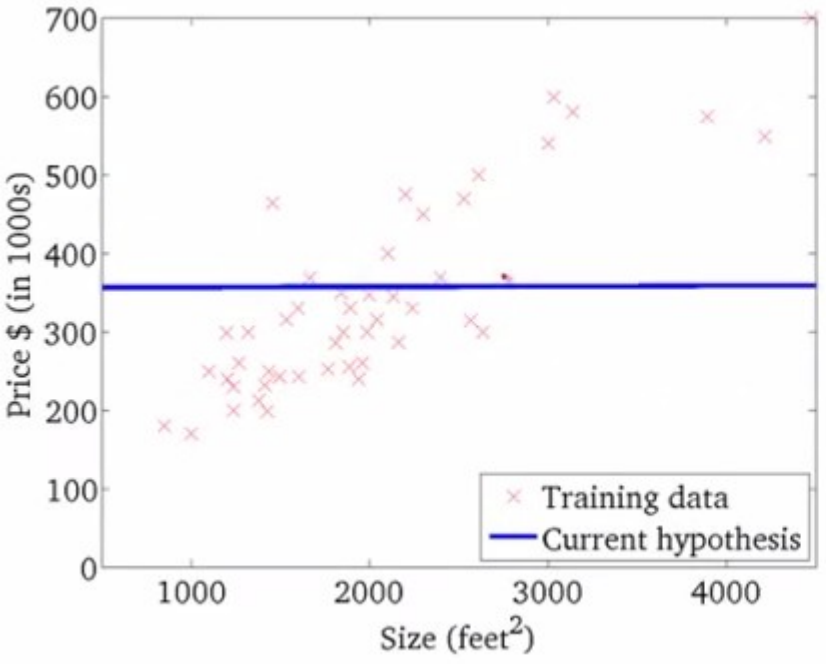
Gradiente descendente + Regressão linear

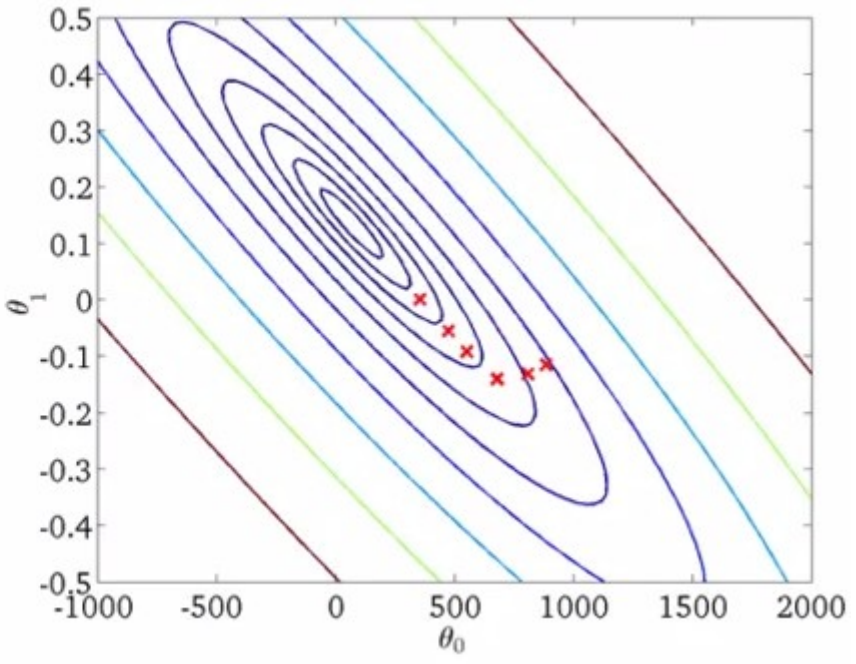
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





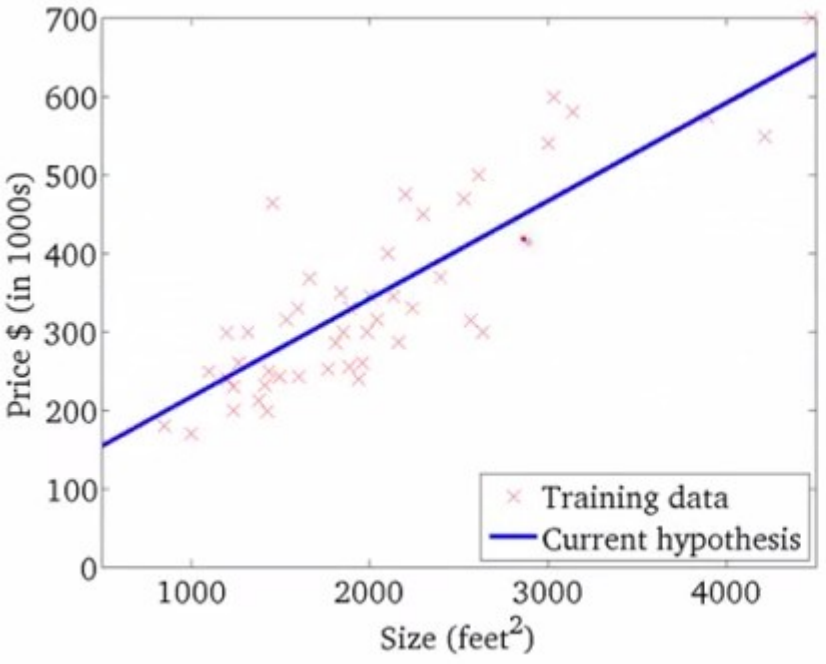
Gradiente descendente + Regressão linear

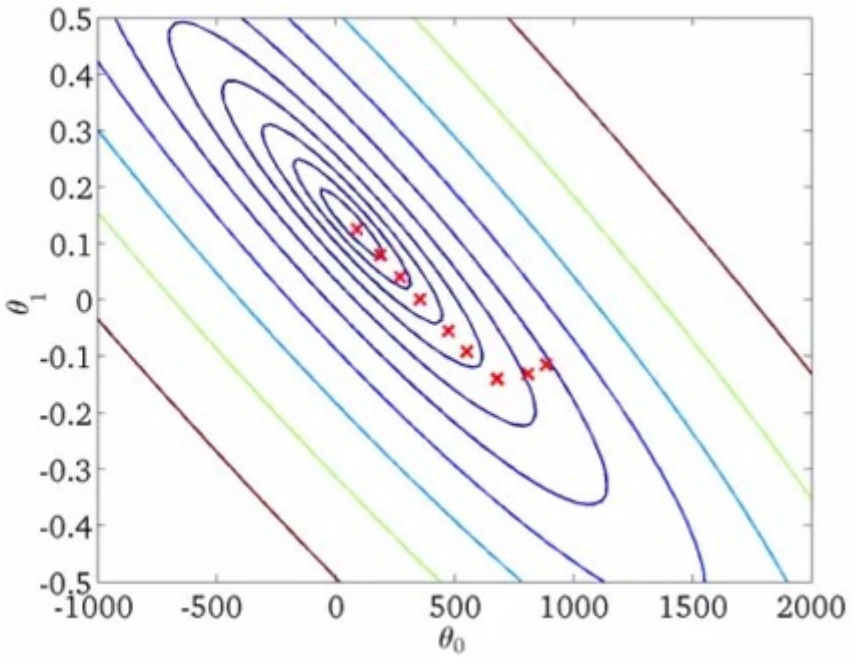
$h_\theta(x)$

(para θ_0, θ_1 , é uma função de x)

$J(\theta_0, \theta_1)$

(função com parâmetro θ_0, θ_1)





Outra interpretação

Maximum Likelihood and Least Squares (1)

Assume observations from a deterministic function with added Gaussian noise:

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \quad \text{where} \quad p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1})$$

which is the same as saying,

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

Given observed inputs, $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and targets, $\mathbf{t} = [t_1, \ldots, t_N]^{\mathrm{T}}$, we obtain the likelihood function

$$p(\mathbf{t}|\mathbf{X}, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}).$$

Maximum Likelihood and Least Squares (2)

Taking the logarithm, we get

$$\begin{aligned}\ln p(\mathbf{t}|\mathbf{w},\beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n),\beta^{-1}) \\ &= \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi) - \beta E_D(\mathbf{w})\end{aligned}$$

where

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2$$

is the sum-of-squares error.

Maximum Likelihood and Least Squares (3)

Computing the gradient and setting it to zero yields

$$\nabla_{\mathbf{w}} \ln p(\mathbf{t}|\mathbf{w}, \beta) = \beta \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n) \right\} \boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} = \mathbf{0}.$$

Solving for w, we get

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t}$$

The Moore-Penrose pseudo-inverse, $\boldsymbol{\Phi}^{\dagger}$.

where

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}.$$

Regularized Least Squares (1)

Consider the error function:

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Data term + Regularization term

With the sum-of-squares error function and a quadratic regularizer, we get

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$$

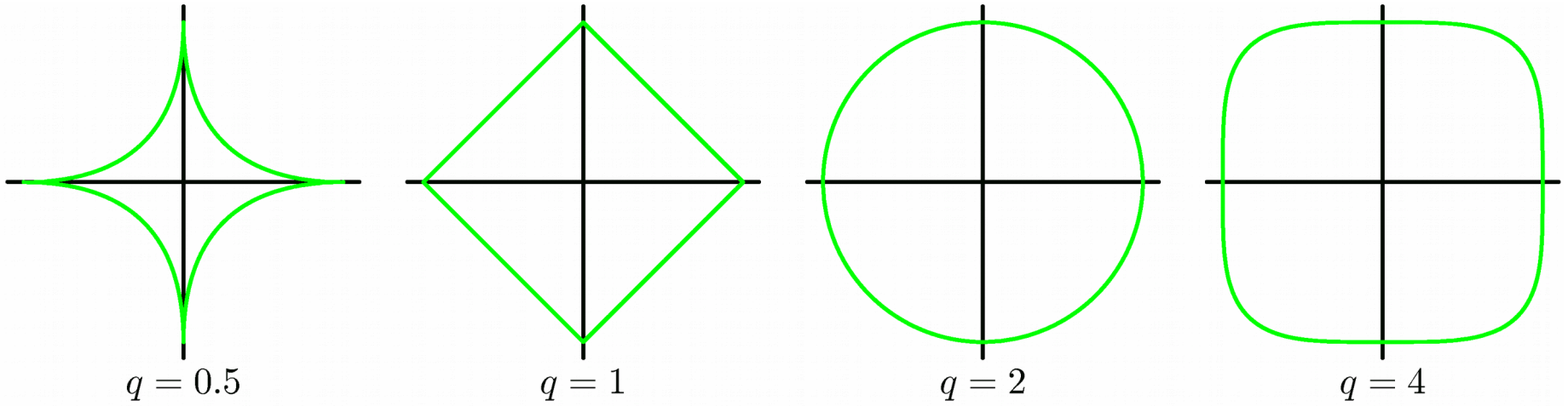
which is minimized by

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$$

Regularized Least Squares (2)

With a more general regularizer, we have

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q$$

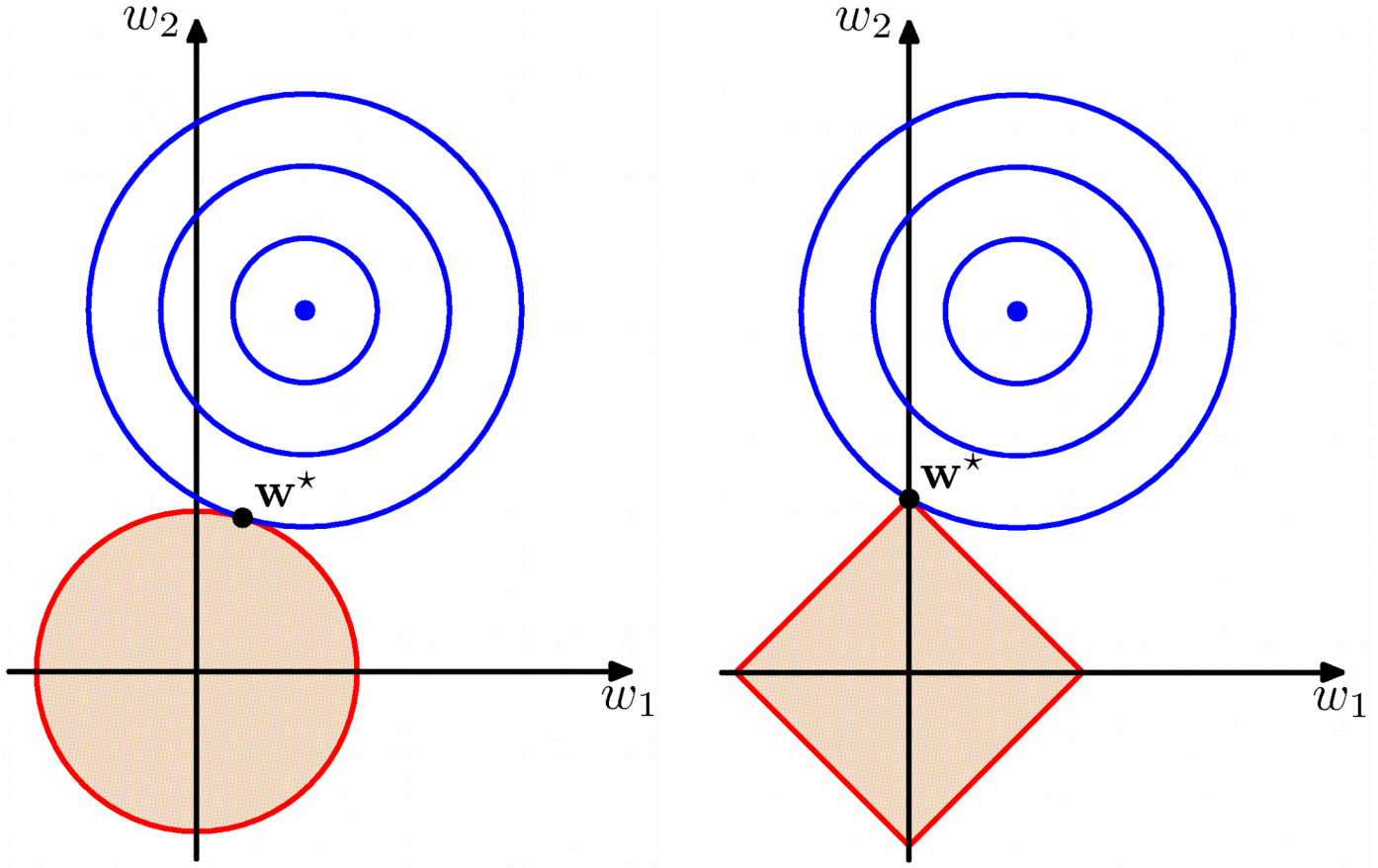


Lasso Quadratic

Slides de: C. Bishop. Pattern Recognition and Machine Learning

Regularized Least Squares (3)

Lasso tends to generate sparser solutions than a quadratic regularizer.



Slides de: C. Bishop. Pattern Recognition and Machine Learning

Outros tipos de regressão

Regressão não-linear

- Podemos aplicar uma função para cada atributo ao invés de multiplicar pelo peso
  - Na verdade, regressão linear é um caso particular da regressão não-linear
    - $y = c + f_1(x_1)+f_2(x_2)+\ldots+f_n(x_3)$
    - Funções podem ser quadrado, log, raiz, módulo, exponencial, etc…
- Determinar as funções apropriadas é um problema!
- Splines, modelos lineares generalizados etc

Regressão por árvores

- Um problema com regressão linear é que a maioria dos conjuntos de dados é não-linear.
  - A formulação não-linear também é problemática, pois não é trivial definir a função que iremos usar!

- Árvores de regressão
  - **Ideia:** Usar uma estrutura de árvore (dividir e conquistar) para dividir os exemplos de tal maneira que eles possam ser melhor modelados linearmente pelos exemplos que estão no nó “folha”.
  - Dessa maneira, os ramos são similiares a árvore de decisão, mas ao invés da classe no nó folha, temos que prever um valor numérico.

Regressão por árvores

- Árvore de regressão: a folha contém um valor numérico, geralmente a média dos exemplos que caem nela.

- **Model Trees:** As folhas contém uma regressão (geralmente linear) para prever o valor dos exemplos que caem nela.

- A árvore de regressão é uma model tree “constante”.

Regressão por árvores

- Sabemos que a árvore a ser construída deve ter um modelo linear em cada folha e um nó de testes em cada nó não folha.

- Para criar o nó de teste, é preciso determinar qual atributo devemos usar, e onde fazer a divisão.

- Witten (p245) propõe o uso do Standard Deviation Reduction
  - trata o desvio padrão do atributo meta como uma medida de erro no nó e maximiza a redução desse valor a cada divisão.

Regressão por árvores

- Os valores preditos pela árvore nos extremos geralmente contém grandes discrepâncias, porque geralmente elas são construídas usando apenas um pequeno subconjunto dos dados.
- É possível fazer um ajuste fino dos valores “interpolando” duas folhas vizinhas de um mesmo ramo, ajustando os valores vizinhos
- A poda pode ser também realizada usando a diferença entre os modelos da folha e de seu nó pai
  - Podemos estimar o erro em cada folha, e ponderar com o número de exemplos em cada folha comparado com seus ancestrais.
  - Ponderar pelo número de exemplos (peso menor para valores estimados com poucos exemplos)
  - Se o erro estimado for menor no nó pai, a folha pode ser podada

Regressão por árvores

- Alguma árvores de regressão/model trees:
  - **CHAID** (Chi-Squared Automatic Interaction Detector). 1980.
  - **CART** (Classification And Regression Tree). 1984.
  - **M5** Quinlan's model (o mesmo que propôs o C4.5). 1992.

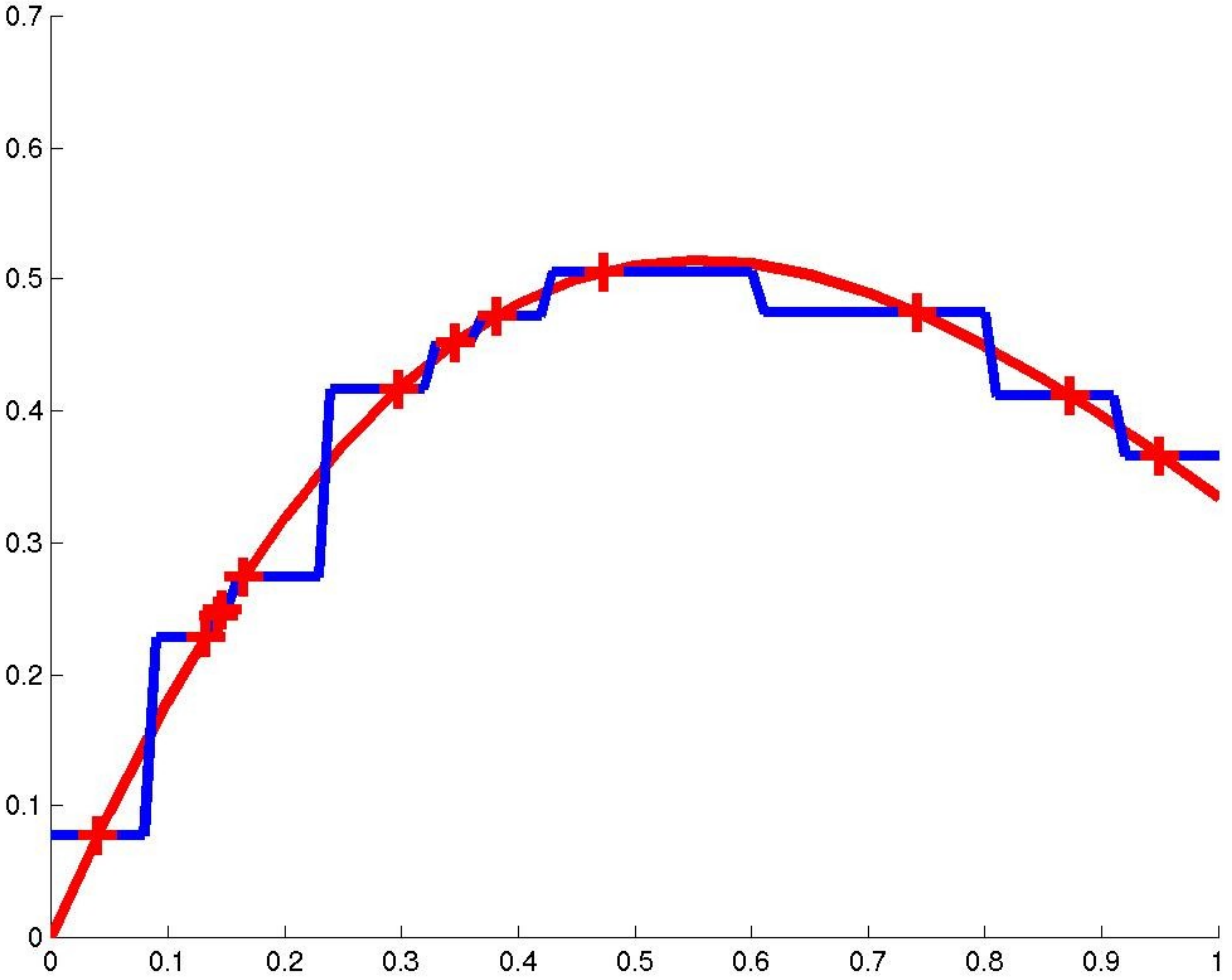
KNN

- Calcula a média dos k-vizinhos mais próximos para estimar o valor

  - Assuma os seguintes exemplos de treinamento
    $(x^{(1)},1);(x^{(2)},1);(x^{(3)},0)$
  - Desejamos classificar $x^{(4)}$ usando K=2 vizinhos mais próximos, sabendo que
    - $dist(x^{(1)},x^{(4)}) = 5$
    - $dist(x^{(2)},x^{(4)}) = 2$
    - $dist(x^{(3)},x^{(4)}) = 4$
    - Portanto $x^{(4)}$ está mais próximo a $x^{(2)}$ e $x^{(3)}$
    - Calculando a média, obtém-se (1+0)/2 = 0.5

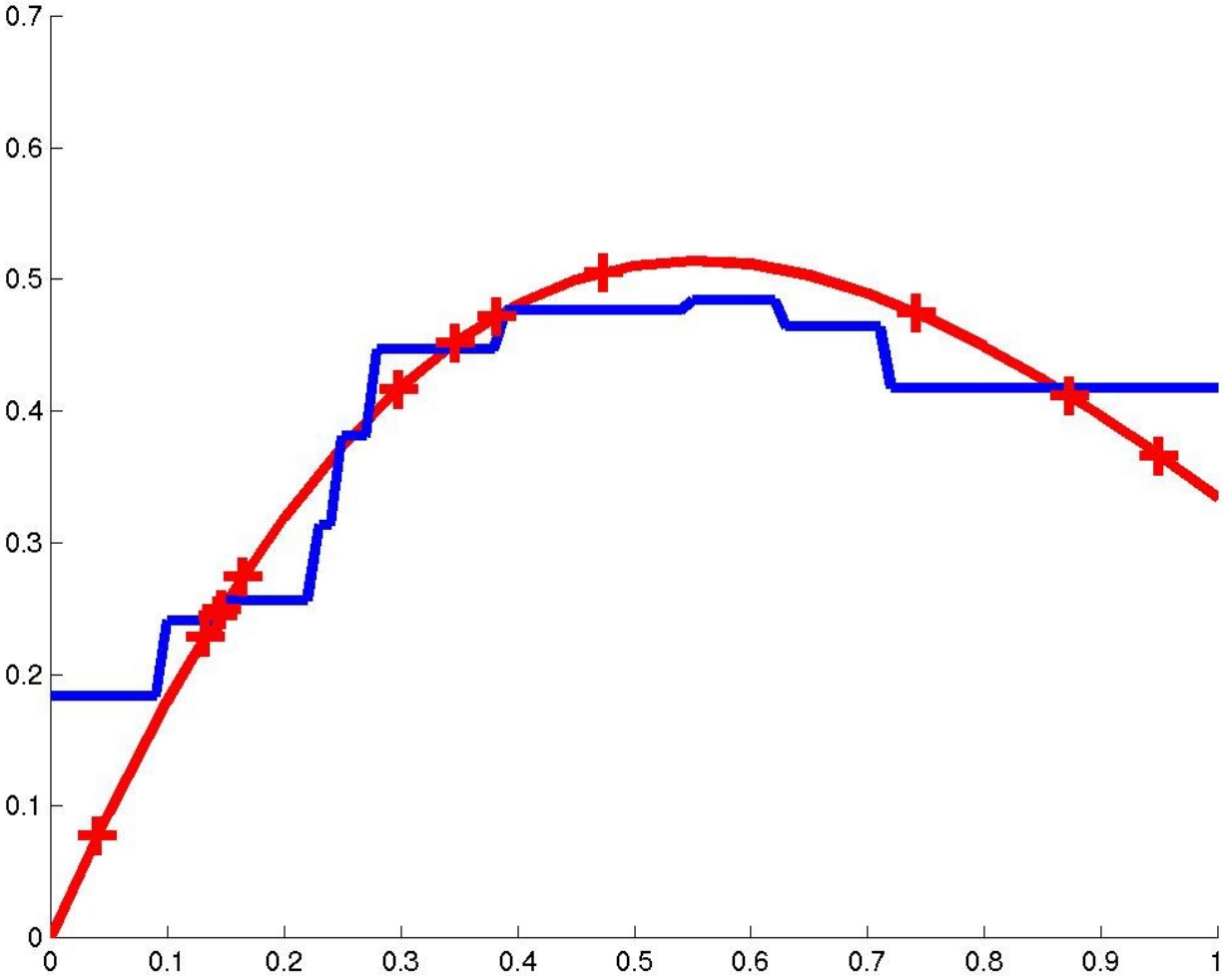
kNN

- k = 1



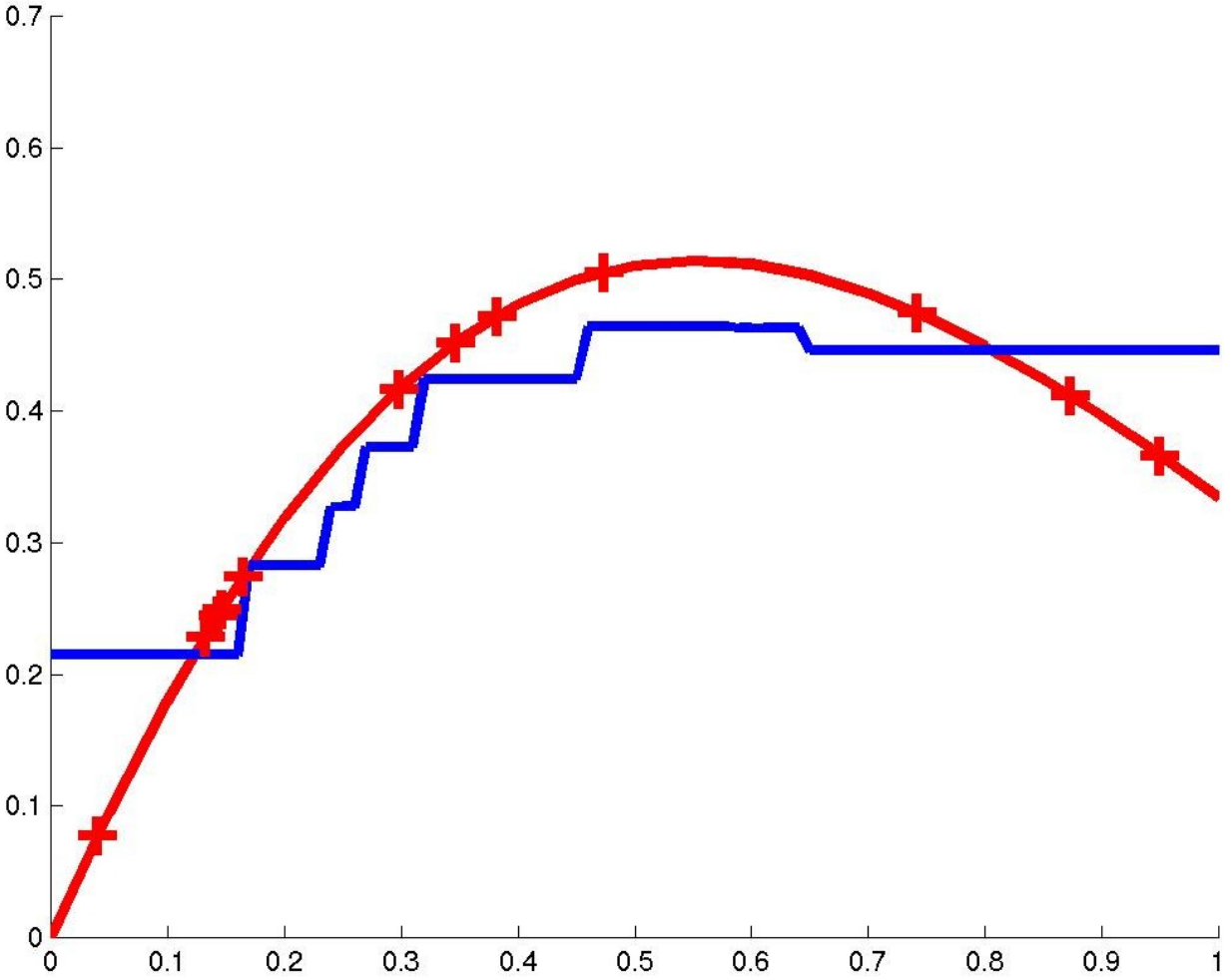
kNN

- k = 3



kNN

- k = 5



Analisando o modelo

- Existem várias abordagens para avaliar os erros obtidos por um modelo de regressão
- Um gráfico que costuma ser útil consiste no gráfico de dispersão de valores preditos vs reais
- Os valores dos coeficientes de um modelo linear podem dar uma estimativa de importância para variáveis/funções base
  - Sob algumas condições

Regressão isotônica

- Modelo não-paramétrico
  - Assume que a função de regressão é não-decrescente

- Utilizada para calibração de classificadores
  - Temos um modelo que costuma acertar qual a classe de um objeto
  - Mas, esse modelo tem estimativas ruins do valor da probabilidade *a posteriori*

Regressão isotônica

- Uma forma de gerar a função é o Pair-Adjacent Violators (PAV)

Algorithm 1. PAV algorithm for estimating posterior probabilities from uncalibrated model predictions.

1 Input: training set (f_i, y_i) sorted according to f_i

2 Initialize $\hat{m}_{i,i} = y_i$, $w_{i,i} = 1$

3 While $\exists\ i\ s.t.\ \hat{m}_{k,i-1} \geq \hat{m}_{i,l}$

Set $w_{k,l} = w_{k,i-1} + w_{i,l}$

Set $\hat{m}_{k,l} = (w_{k,i-1}\hat{m}_{k,i-1} + w_{i,l}\hat{m}_{i,l})/w_{k,l}$

Replace $\hat{m}_{k,i-1}$ and $\hat{m}_{i,l}$ with $\hat{m}_{k,l}$

4 Output the stepwise constant function:

$\hat{m}(f) = \hat{m}_{i,j}$, for $f_i < f \leq f_j$

Regressão isotônica - Exemplo

#	Score	Prob Inicial	Iteração 1	Iteração 2	Iteração 3	Iteração 4	Iteração 5	Final
1	0,9	1	1					
2	0,8	1	1					
3	0,7	0	0					
4	0,6	1	1					
5	0,55	1	1					
6	0,5	1	1					
7	0,45	0	0					
8	0,4	1	1					
9	0,35	1	1					
10	0,3	0	0					
11	0,27	1	1					
12	0,2	0	0					
13	0,18	0	1/2					
14	0,1	1	1/2					
15	0,02	0	0					

Regressão isotônica - Exemplo

#	Score	Prob Inicial	Iteração 1	Iteração 2	Iteração 3	Iteração 4	Iteração 5	Final
1	0,9	1	1	1	1	1	1	1
2	0,8	1	1	1	1	1	1	1
3	0,7	0	0	0	0	0	0	3/4
4	0,6	1	1	1	1	1	1	3/4
5	0,55	1	1	1	1	1	1	3/4
6	0,5	1	1	1	1	1	1	3/4
7	0,45	0	0	0	0	1/2	2/3	2/3
8	0,4	1	1	1	1	1/2	2/3	2/3
9	0,35	1	1	1	1	1	2/3	2/3
10	0,3	0	0	0	1/2	1/2	1/2	1/2
11	0,27	1	1	1	1/2	1/2	1/2	1/2
12	0,2	0	0	1/3	1/3	1/3	1/3	1/3
13	0,18	0	1/2	1/3	1/3	1/3	1/3	1/3
14	0,1	1	1/2	1/3	1/3	1/3	1/3	1/3
15	0,02	0	0	0	0	0	0	0

Referências

C. Bishop. 2006. Pattern Recognition and Machine Learning. Capítulo 3.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02). ACM, New York, NY, USA, 694-699.