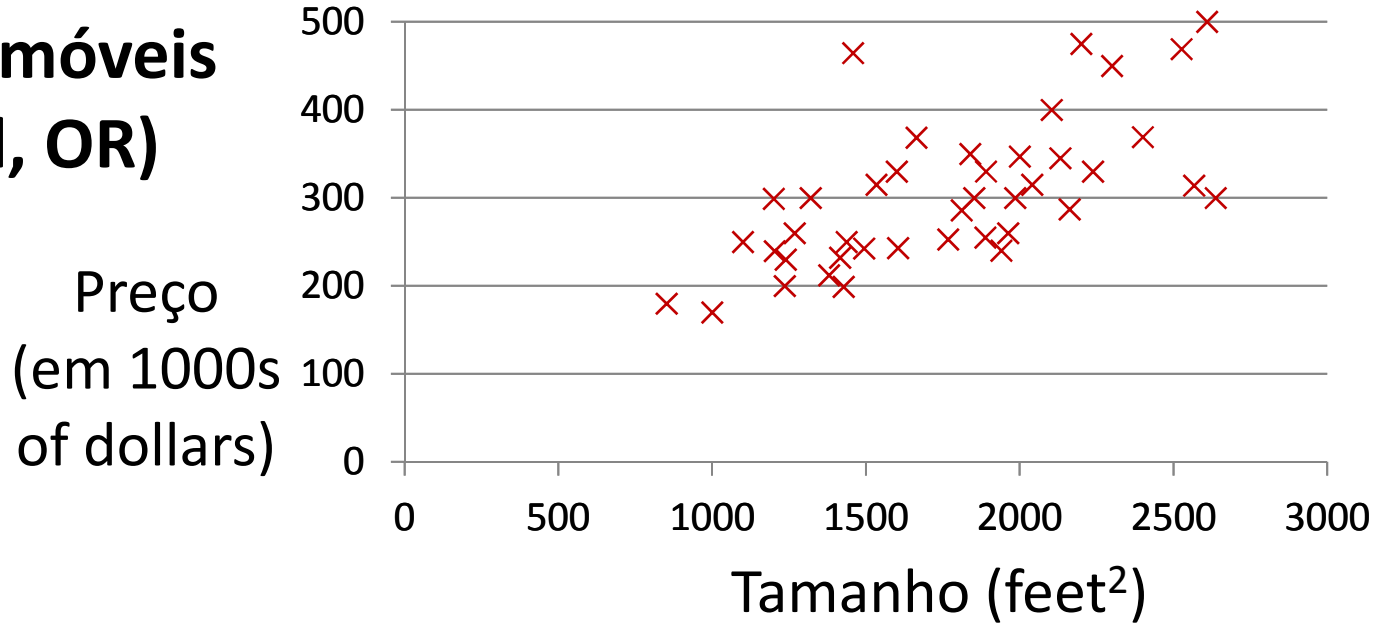
INTRODUÇÃO AO ESTUDO DAS REDES NEURAIS ARTIFICIAS

Laboratório de Conexionismo e Ciências Cognitivas L3C
Grupo SICRES
INE - UFSC

Regressão Linear com Uma Variável

Preço dos Imóveis (Portland, OR)



Aprendizado Supervisionado	Problema de Regressão
Fornecer a "resposta correta" para cada exemplo visto nos dados.	Predizer um valor contínuo de saída

Conjunto de treinamento Para o preço dos imóveis (Portland, OR)

Tamanaho em feet² (x)	**Preço ($) em 1000's (y)**
2104	460
1416	232
1534	315
852	178
…	…

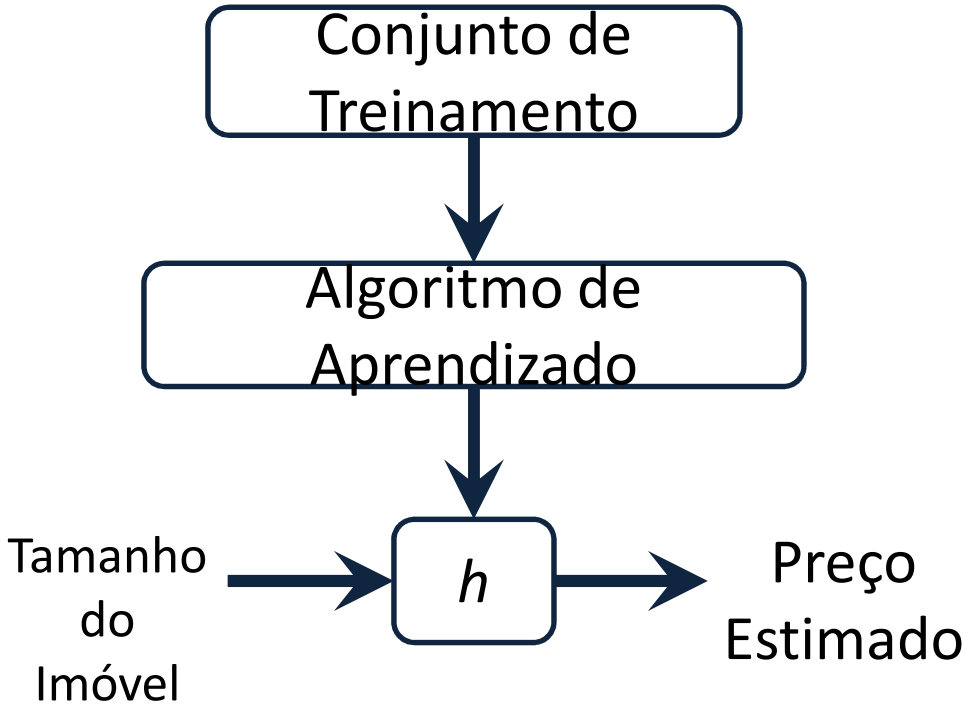
Notation:

m = Number of training examples

x's = "input" variable / features

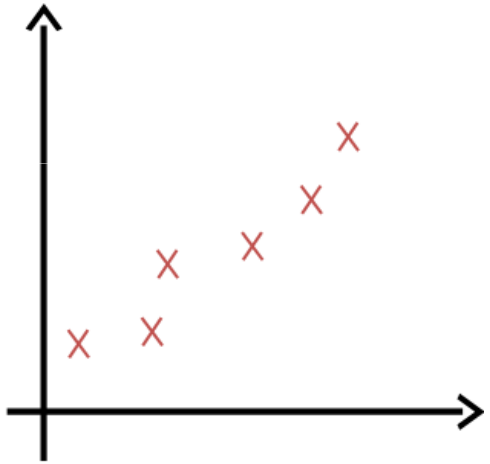
y's = "output" variable / "target" variable

Introdução à Regressão Linear com Uma Variável – a questão do Modelo



Como representar a hipótese h?

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



Regressão Linear com uma variável.
Regressão Linear Univariável.

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

Parâmetros: θ_0, θ_1

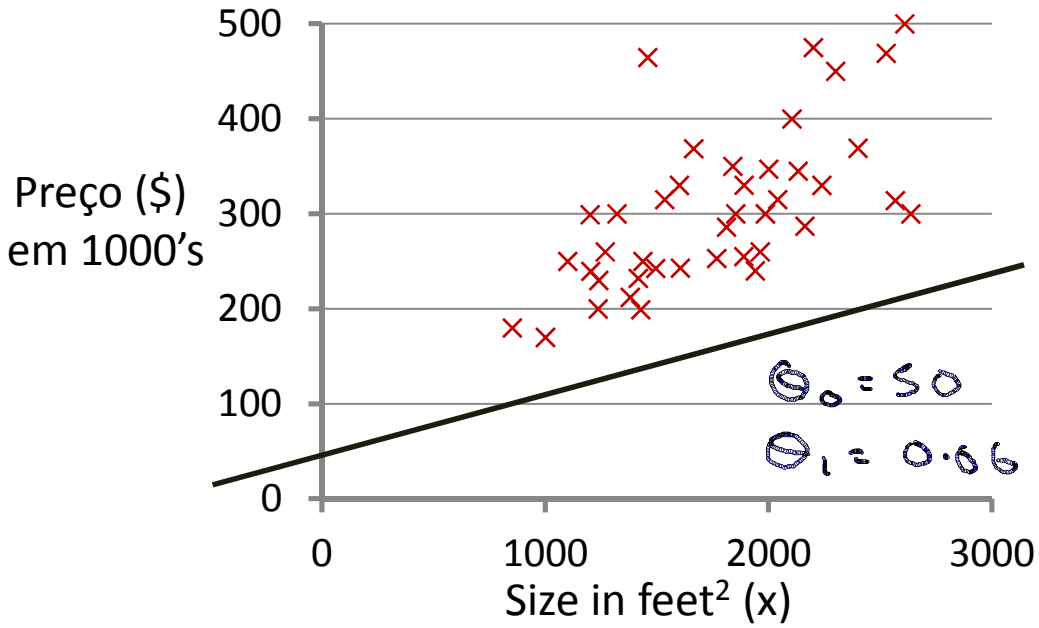
Função de Custo: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Objetivo: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Introdução à Regressão Linear com Uma Variável – a questão do Custo

$h_\theta(x)$

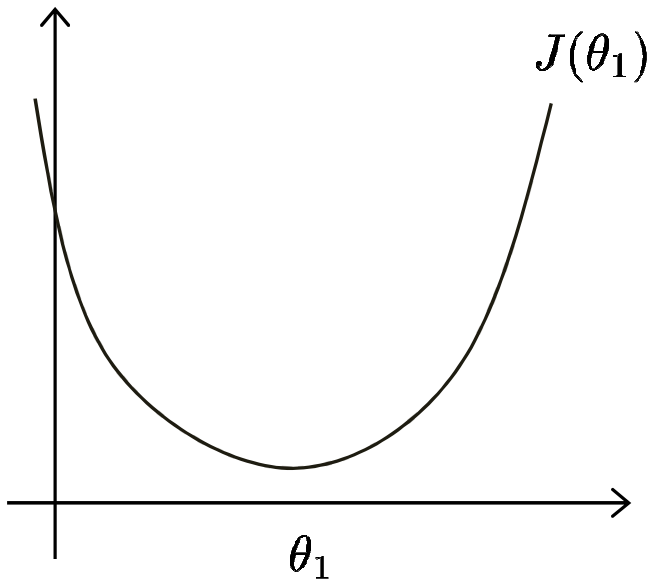
(for fixed θ_0, θ_1, this is a function of x)



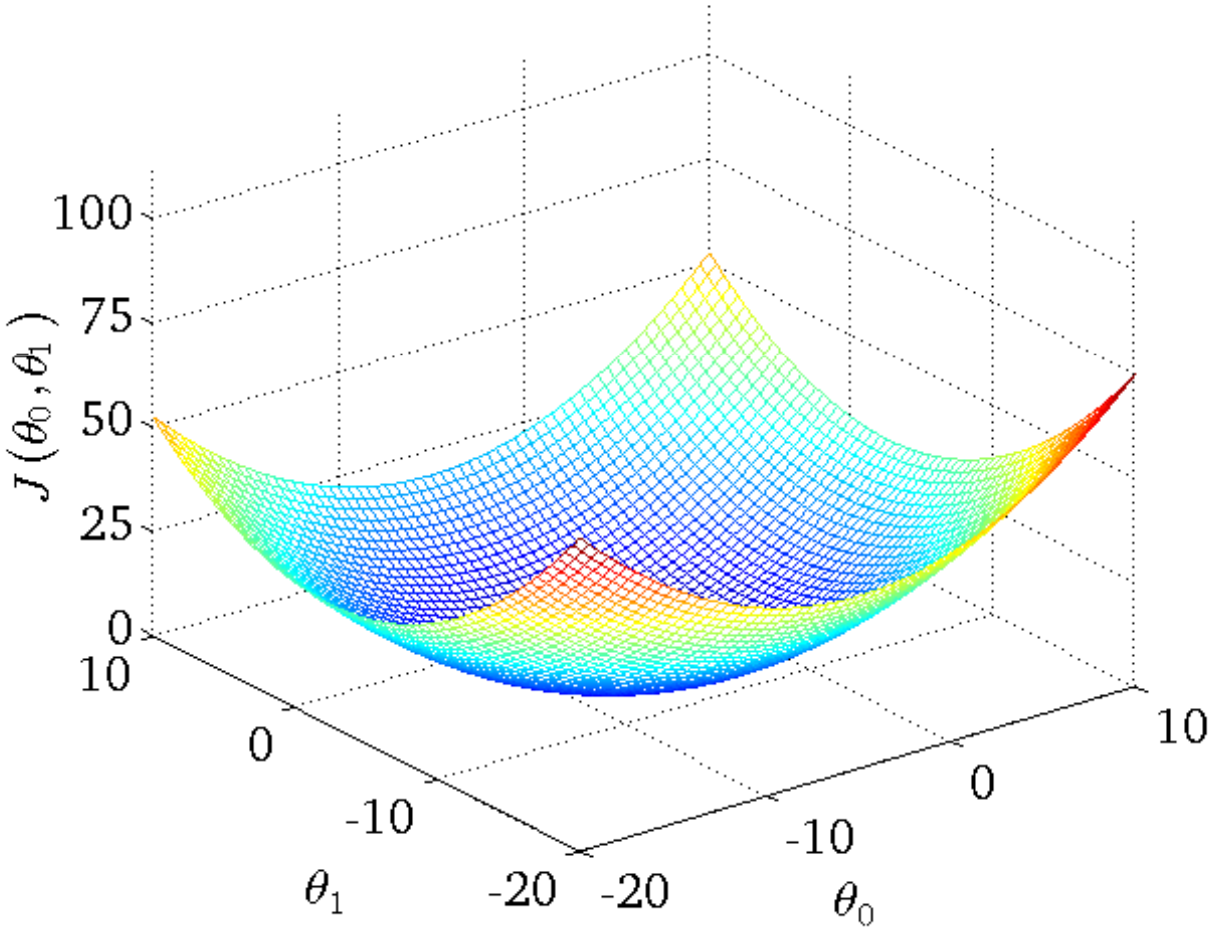
$$h_\theta(x) = 50 + 0.06x$$

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



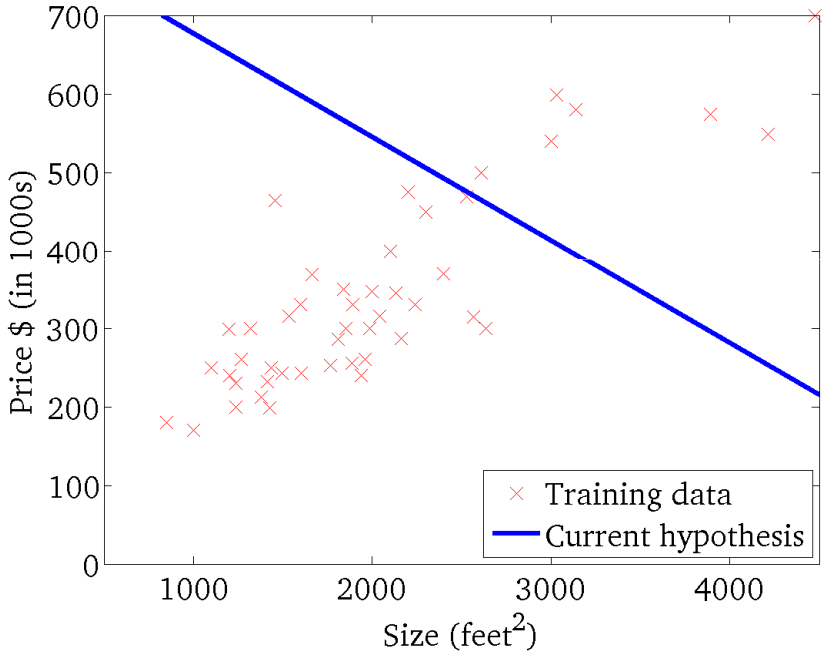
Introdução à Regressão Linear com Uma Variável – a questão do Custo



Introdução à Regressão Linear com Uma Variável – a questão do Custo

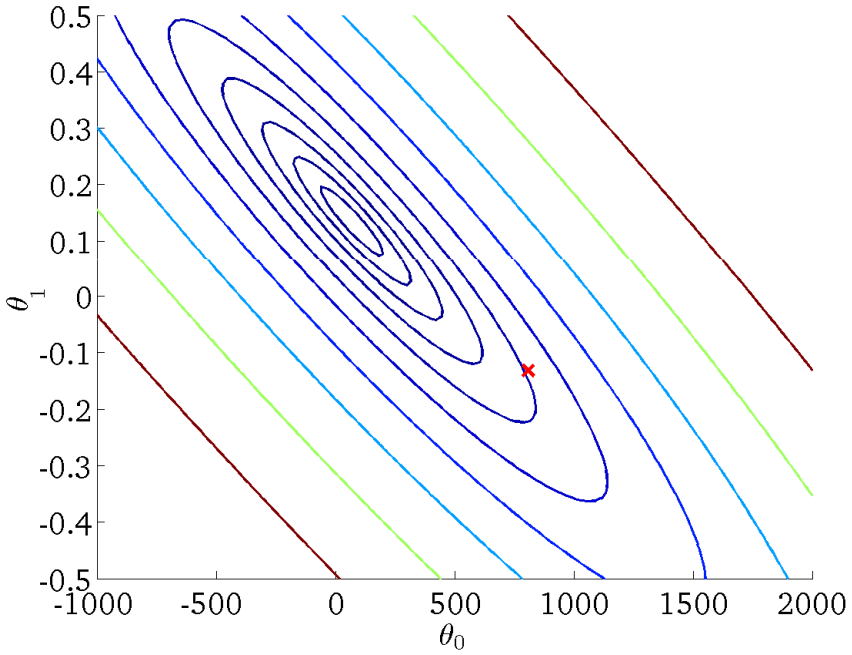
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Introdução à Regressão Linear com Uma Variável – Minimizando a Função de Custo

O MÉTODO DOS MÍNINOS QUADRÁTICOS

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

Esta é a função que queremos minimizar

$$h_\Theta(X) = X.\Theta \qquad \Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 & x^1 \\ 1 & x^2 \\ 1 & x^2 \\ \dots & \dots \\ 1 & x^m \end{bmatrix}$$

$$e = Y - h_\Theta(X) = Y - X.\Theta$$

$$J(\Theta) = e^2$$

Na forma vetorial ela fica assim

$$e^2 = e^T.e = (Y - X\Theta)^T.(Y - X\Theta)$$
$$e^2 = Y^T Y - Y^T X\Theta - \Theta^T X^T Y - \Theta^T X^T X\Theta$$
$$\frac{\partial J}{\partial \Theta} = -2X^T Y + 2X^T X\Theta = 0$$
$$-X^T Y + X^T X\Theta = 0$$
$$X^T X\Theta = X^T Y$$
$$\Theta = (X^T X)^{-1}.X^T Y$$

Esta é a fórmula analítica para calcular os valores de Θ que minimizam a função de custo

Introdução à Regressão Linear com Uma Variável – Minimizando a Função de Custo

O MÉTODO DO GRADIENTE DESCENDENTE

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

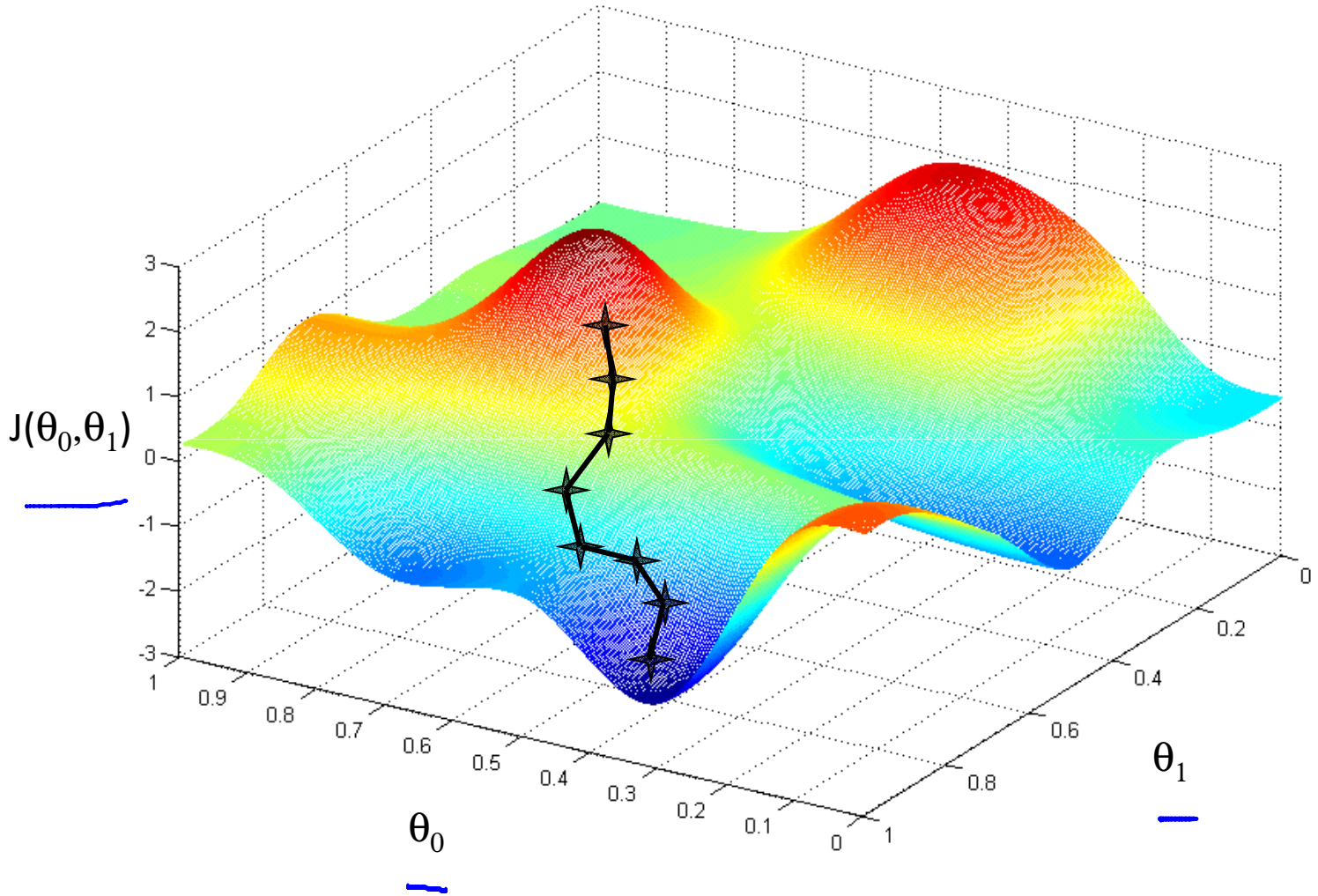
Esta é a função que queremos minimizar

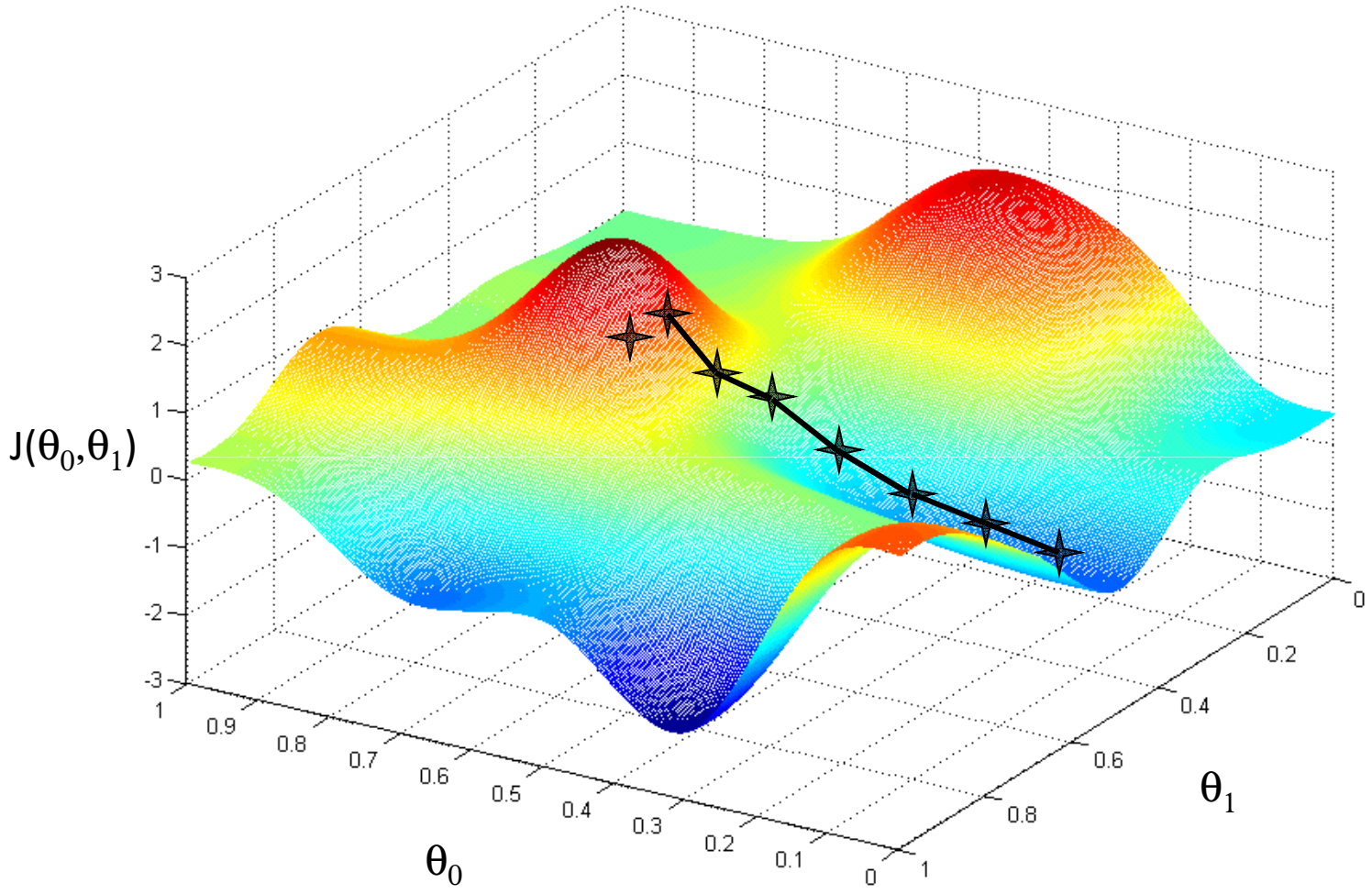
Dada uma função $J(\theta_0, \theta_1)$

Queremos $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Roteiro:

- Inicie com valores (aleatório ou não) θ_0, θ_1
- Permaneça alterando θ_0, θ_1 para reduzir $J(\theta_0, \theta_1)$ até que “esperançosamente” alcancemos um mínimo


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

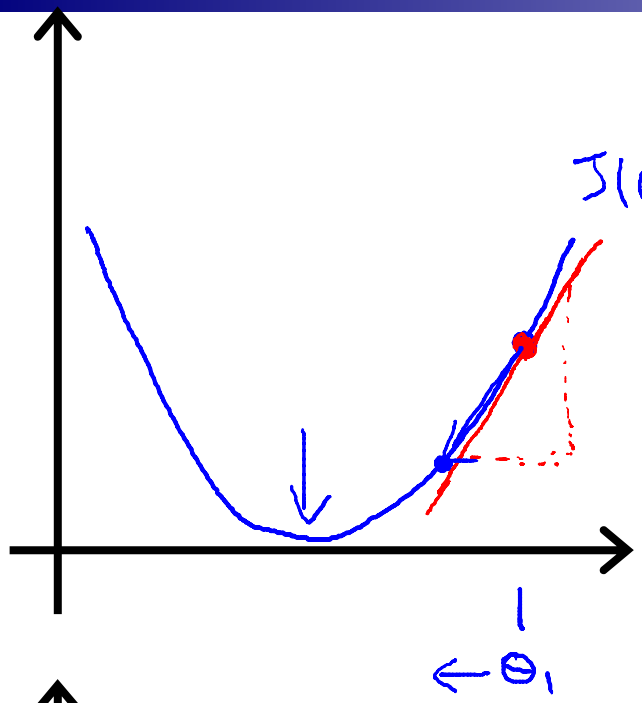
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$ (simultaneously update $j = 0$ and $j = 1$)

}

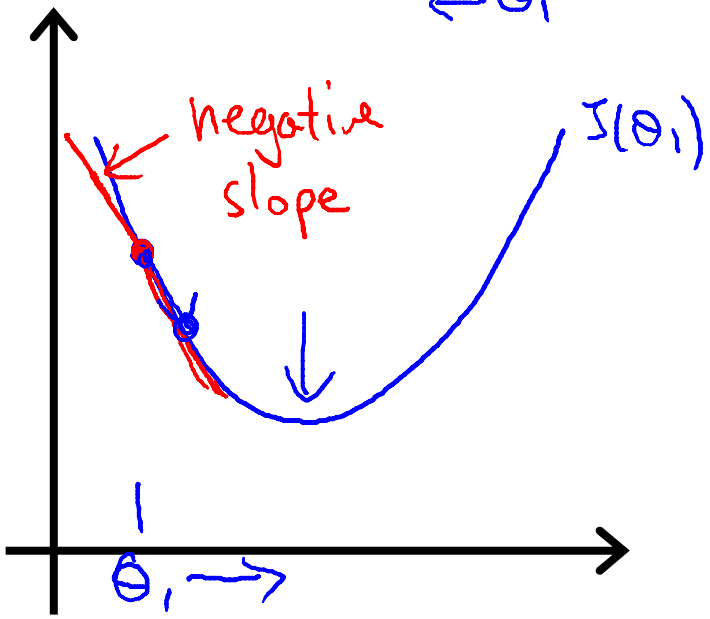


$(\theta_1 \in \mathbb{R})$

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

$\alpha \leftarrow$ $\frac{d}{d\theta_1} J(\theta_1) \geq 0$

$\theta_1 := \theta_1 - \alpha \cdot (\text{positive number})$

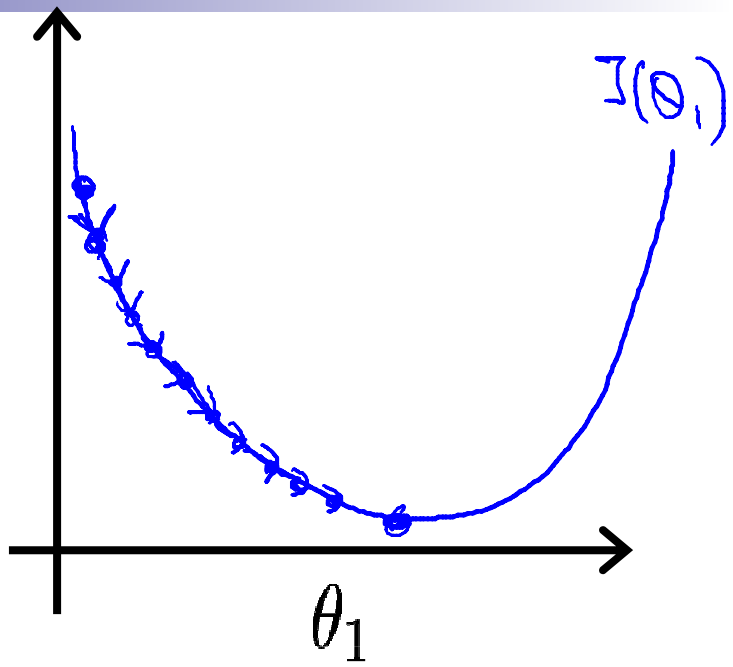


$\frac{d}{d\theta_1} J(\theta_1) \leq 0$

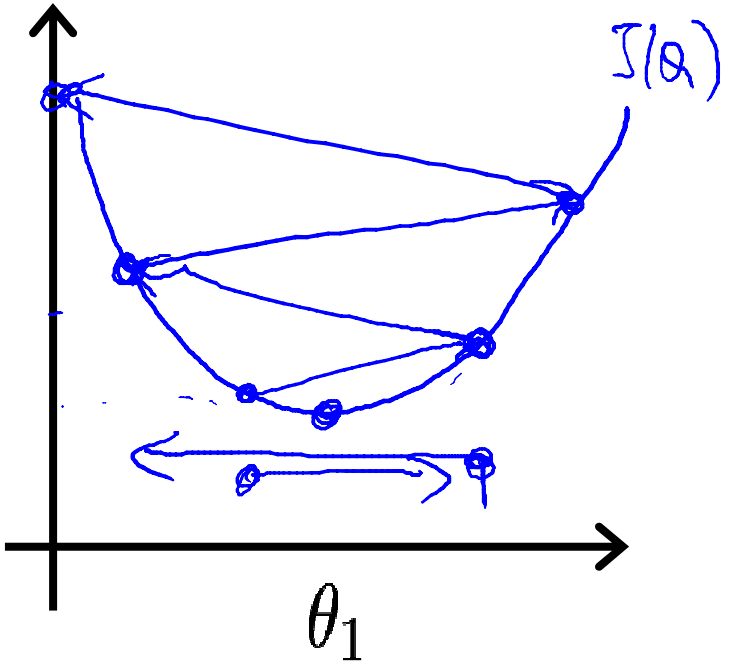
$\theta_1 := \theta_1 - \alpha (\text{negative number})$

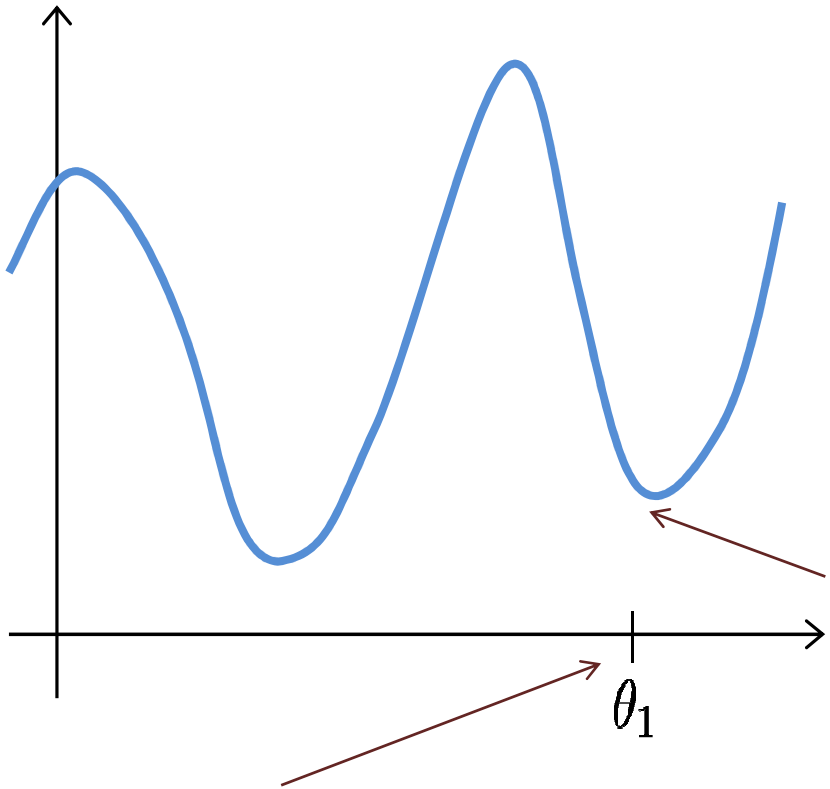
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



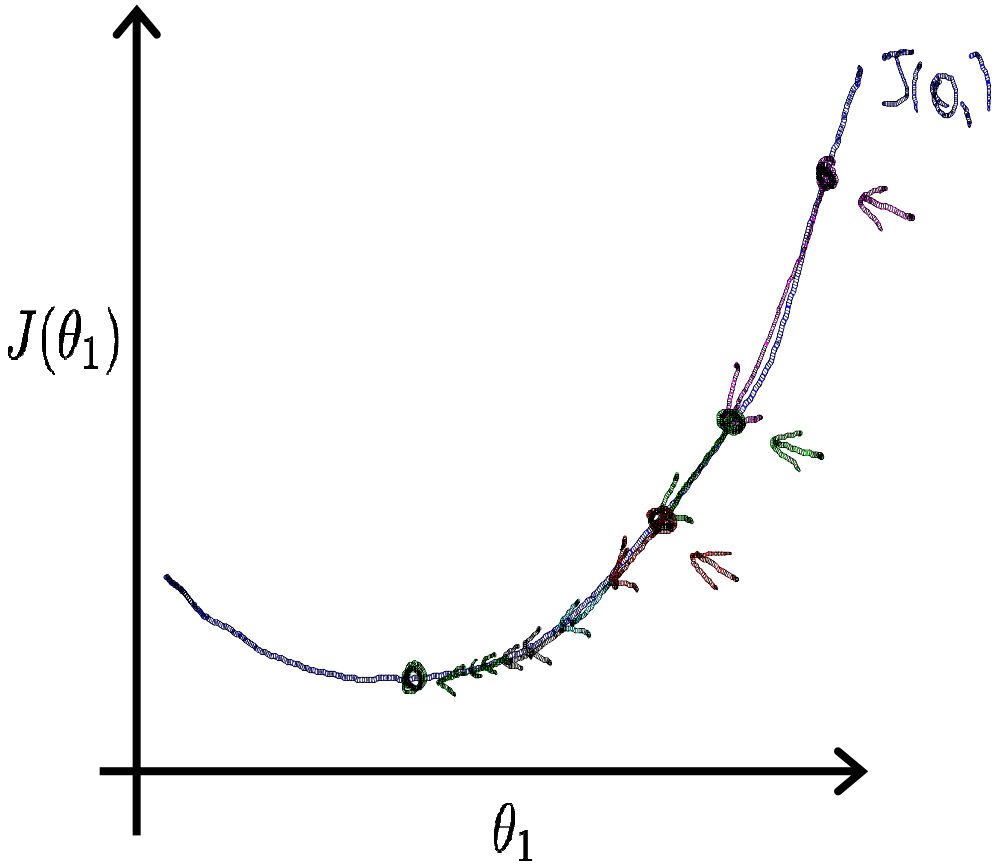


$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Gradient descent can converge to a local minimum, even with the learning rate α fixed.

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

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.



Gradient descent algorithm

$$\text{repeat until convergence } \{$$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$(\text{for } j = 1 \text{ and } j = 0)$$

$$\}$$

Linear Regression Model

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

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

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

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

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

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

Gradient descent algorithm

$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

repeat until convergence {

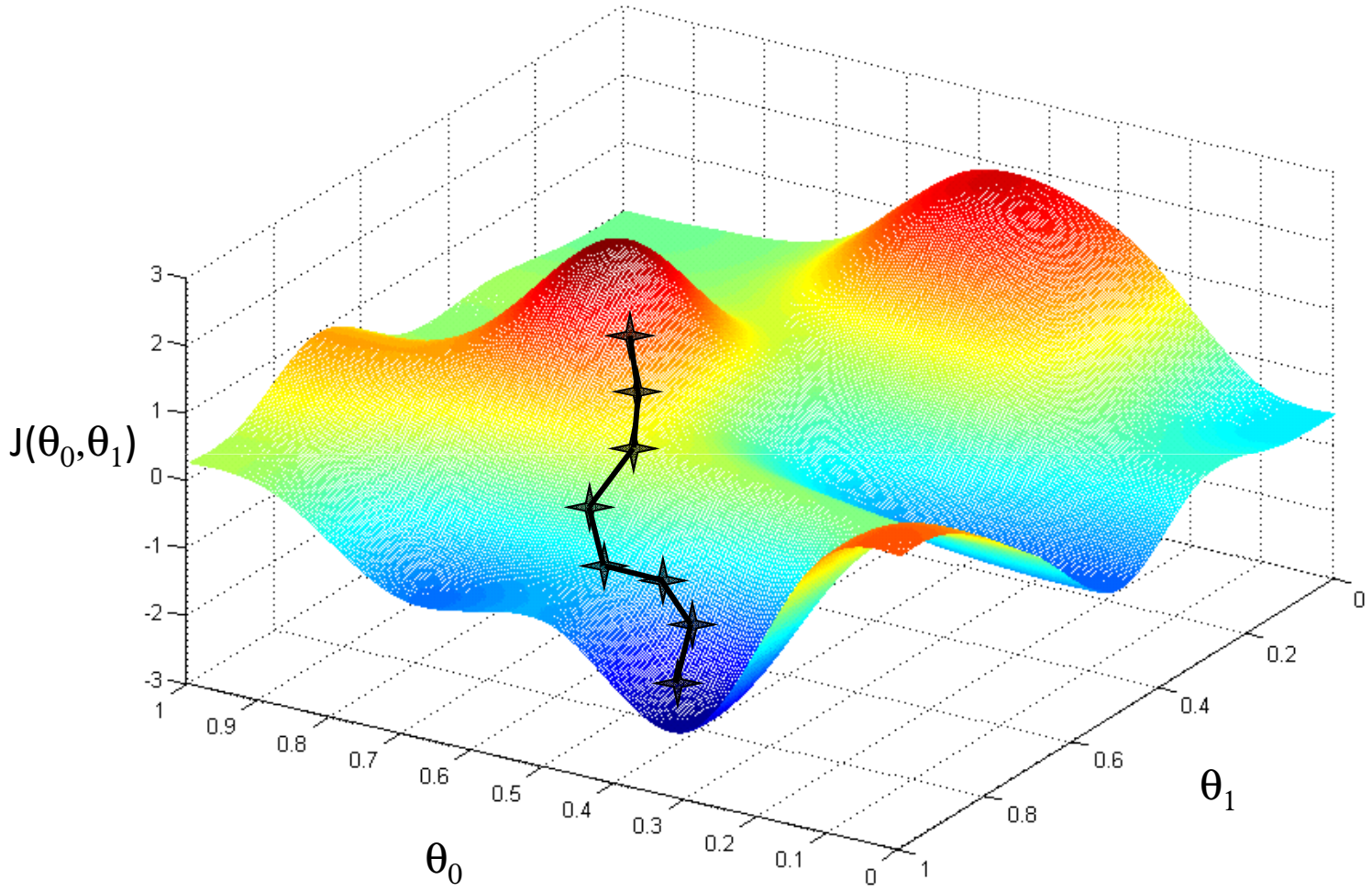
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

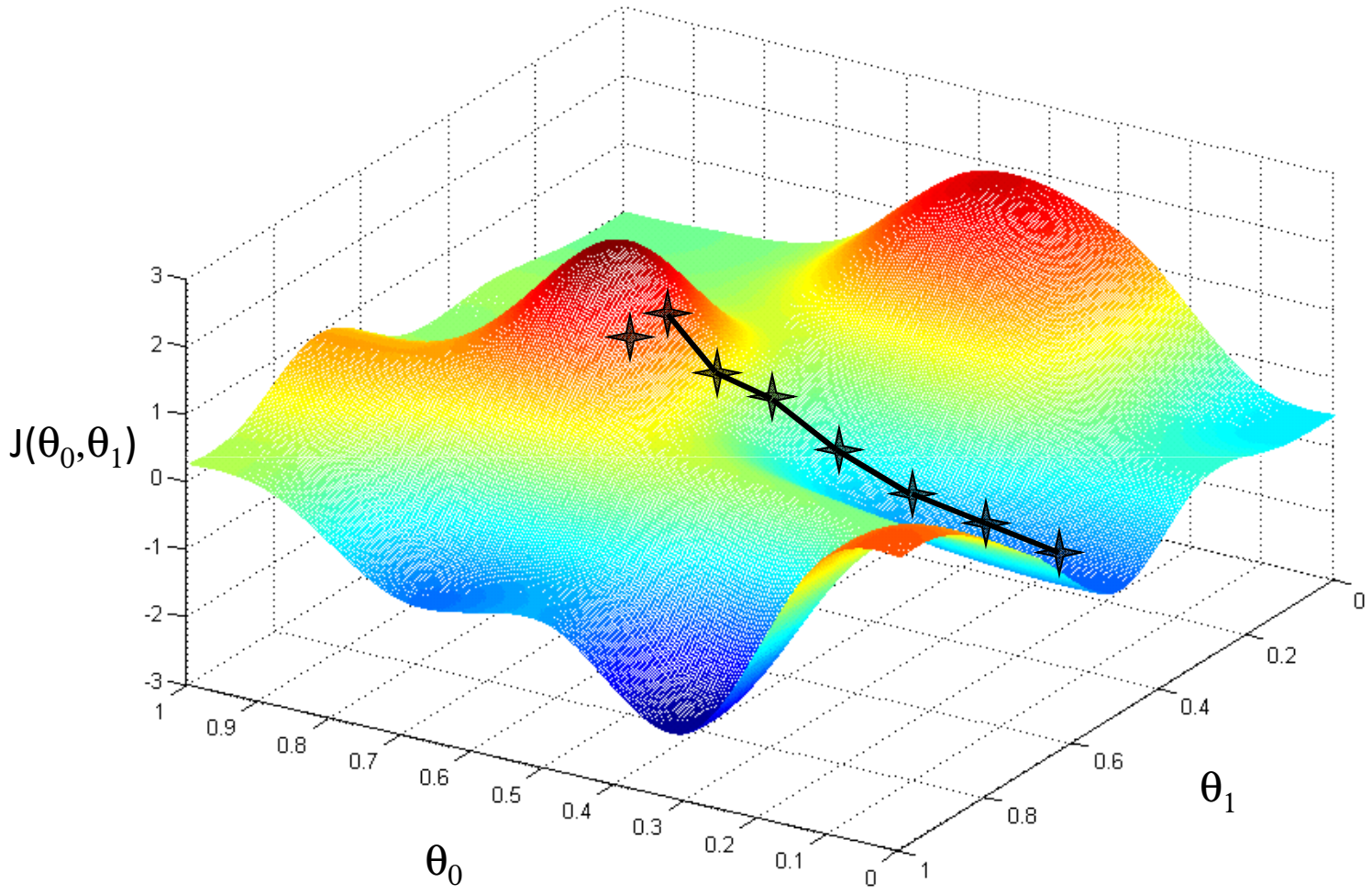
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)}$$

}

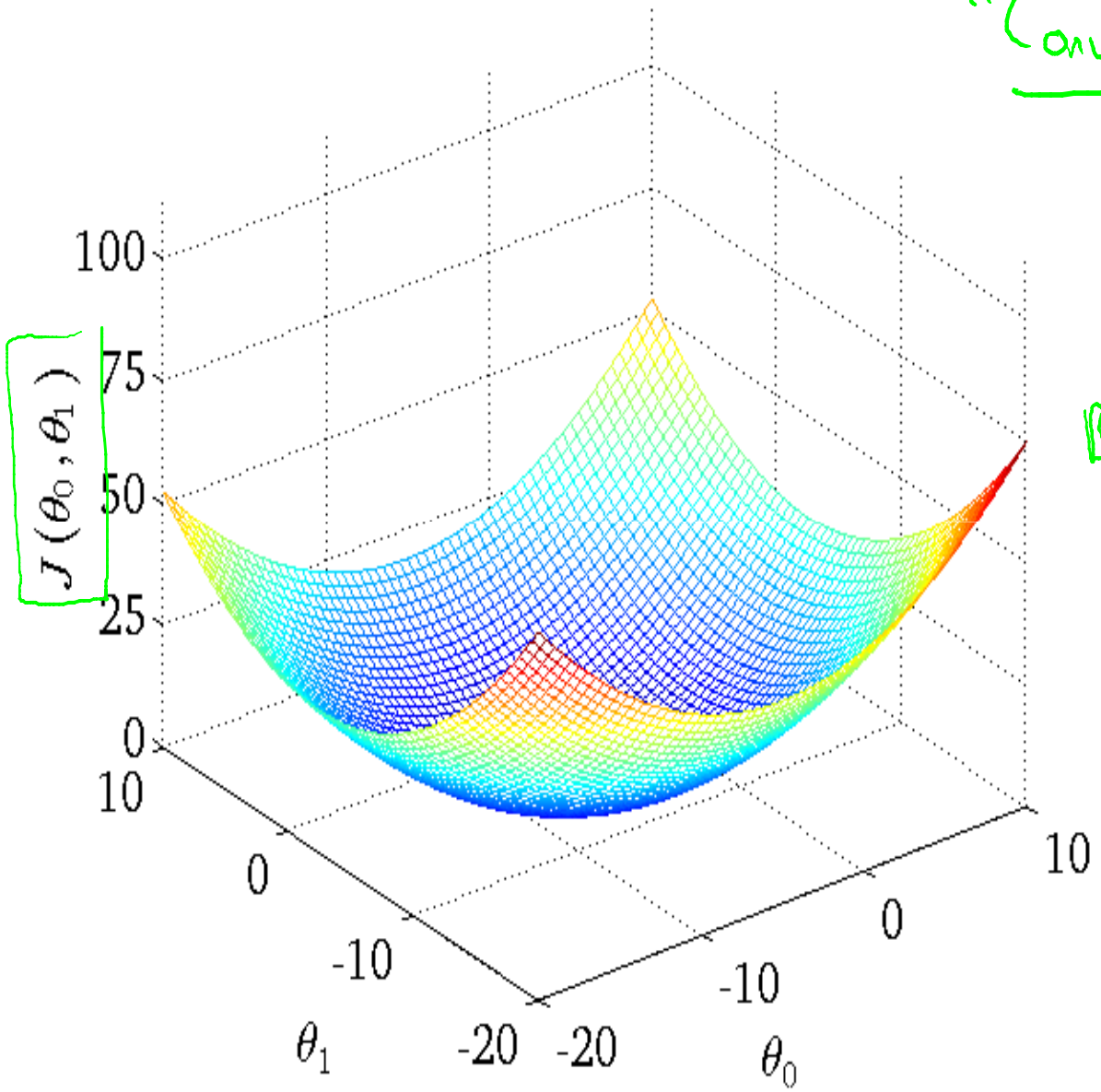
update θ_0 and θ_1 simultaneously

$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

"Convex function"



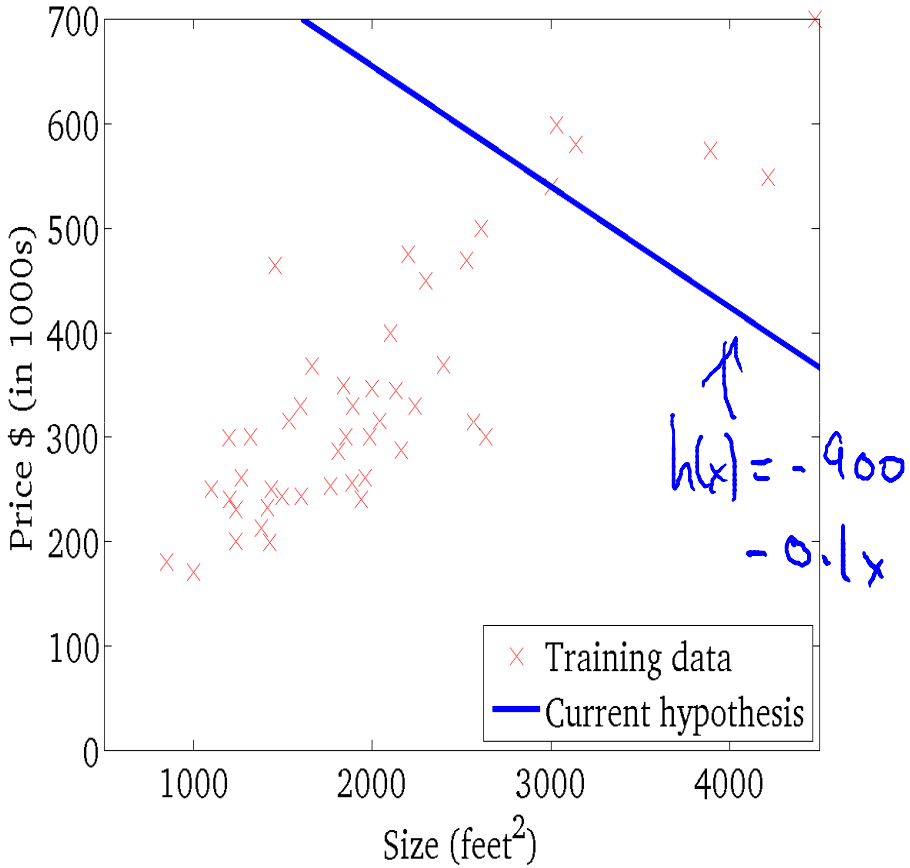
Bowl-shaped

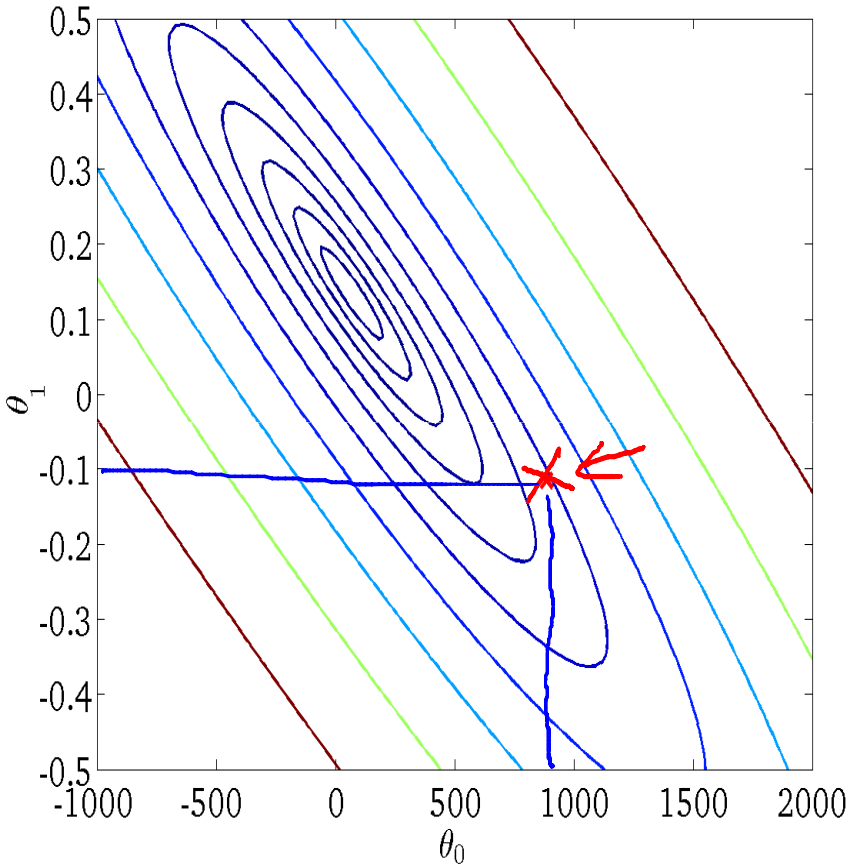
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)

$J(\theta_0, \theta_1)$

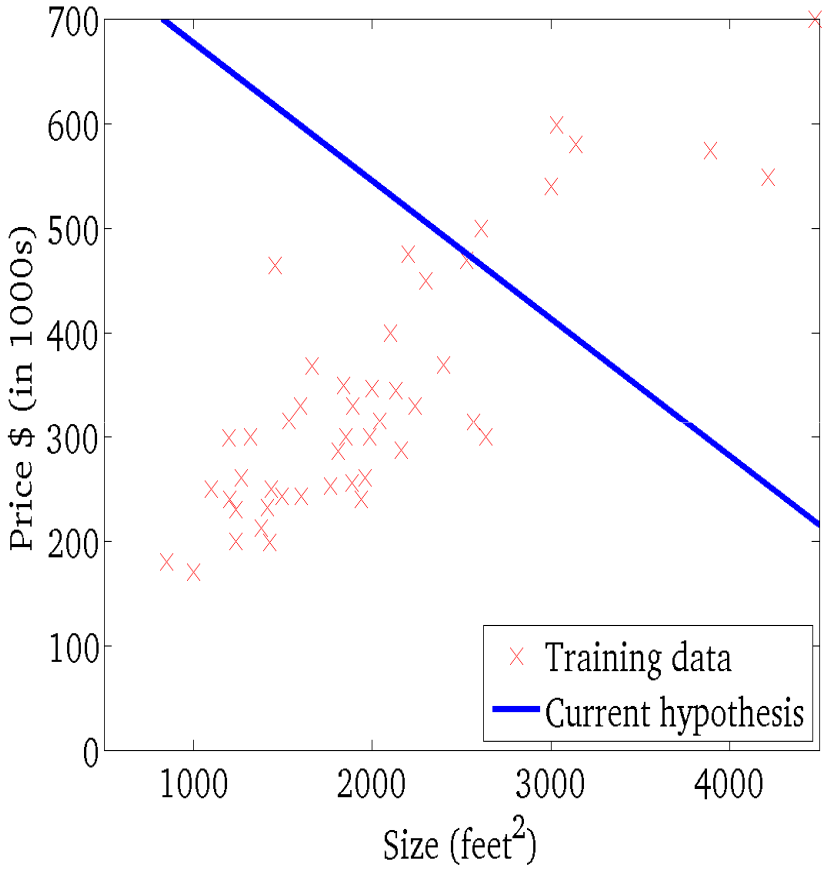
(function of the parameters θ_0, θ_1)





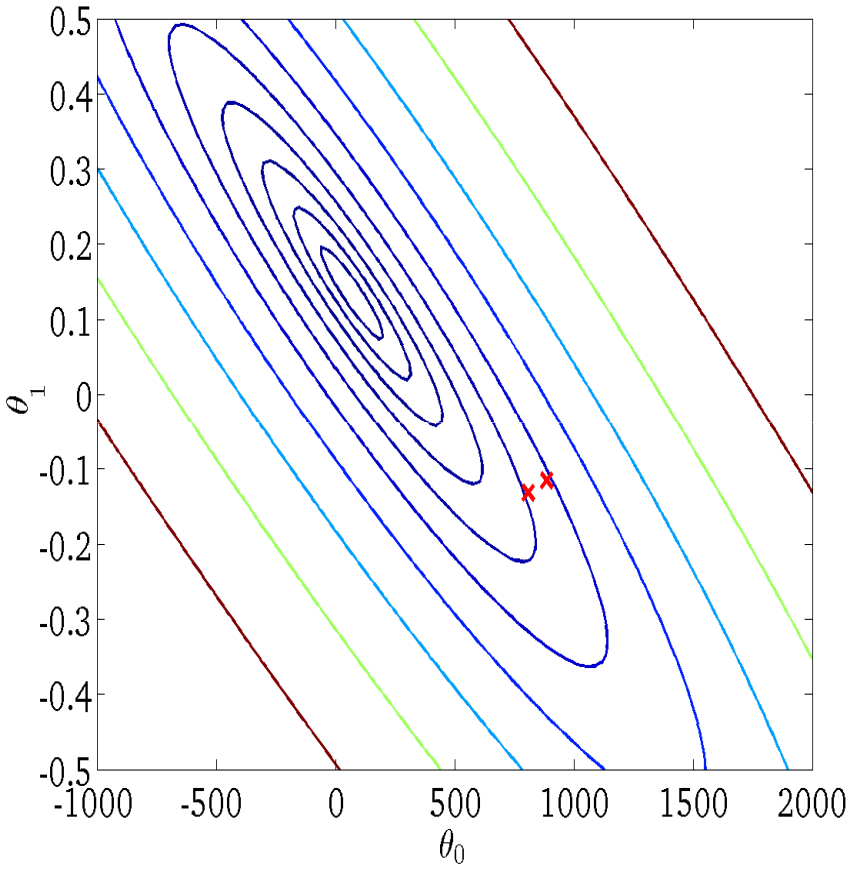
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



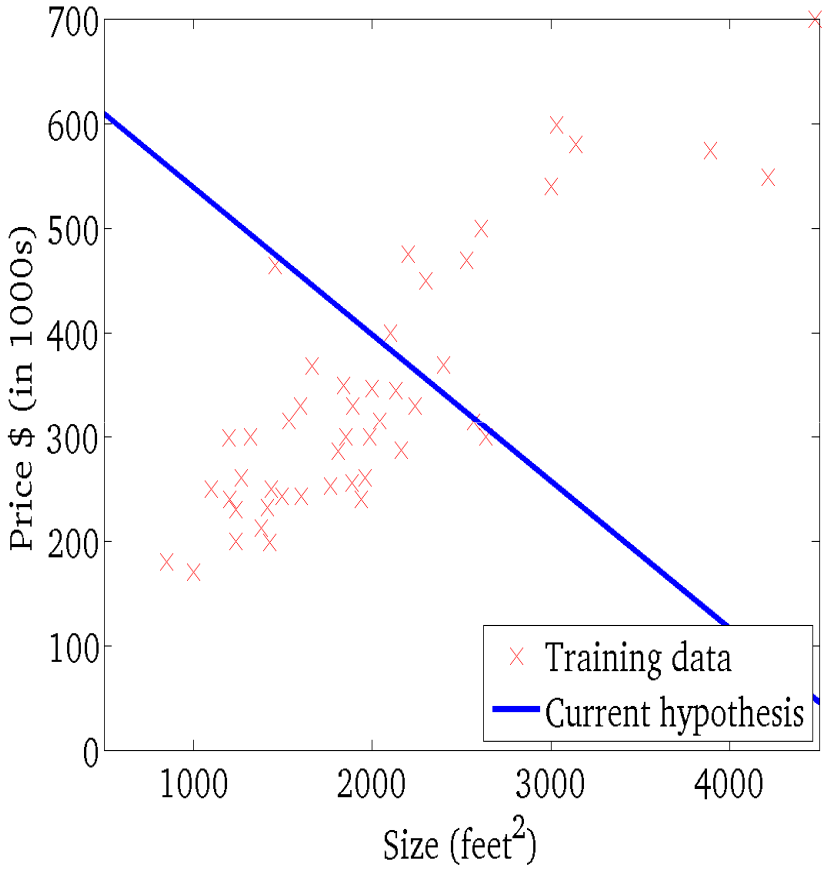
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



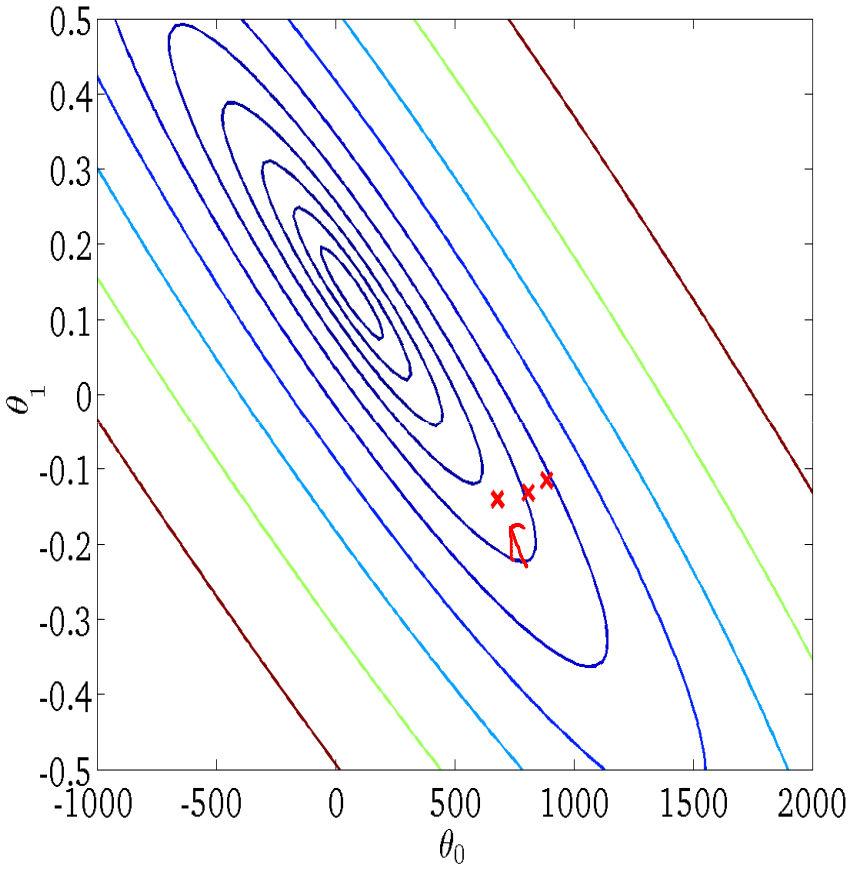
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



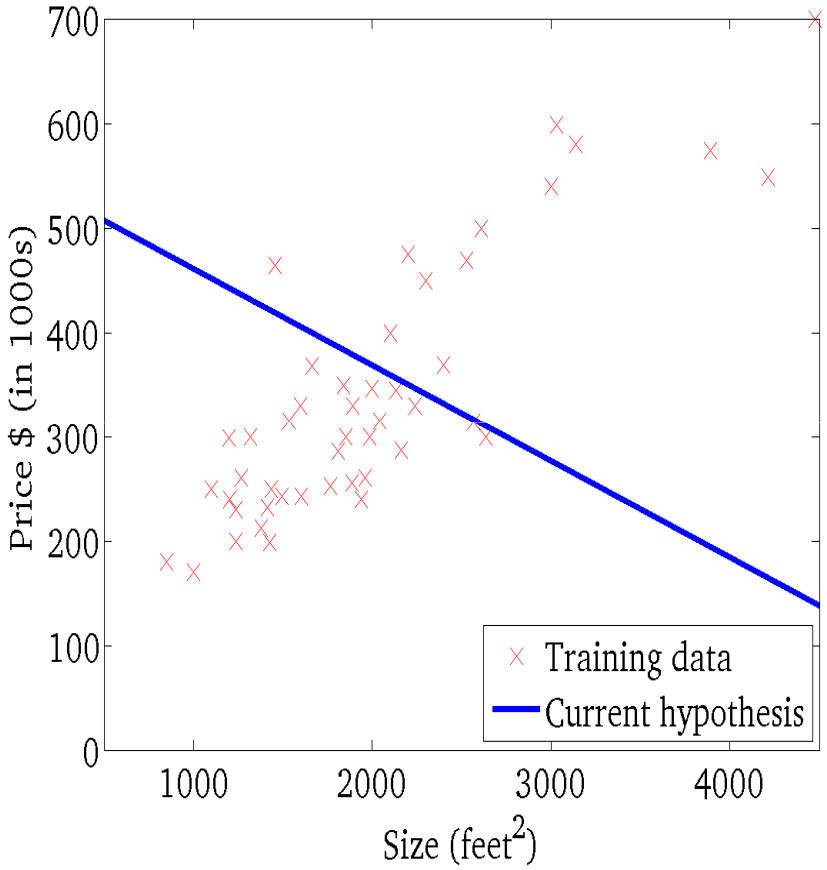
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



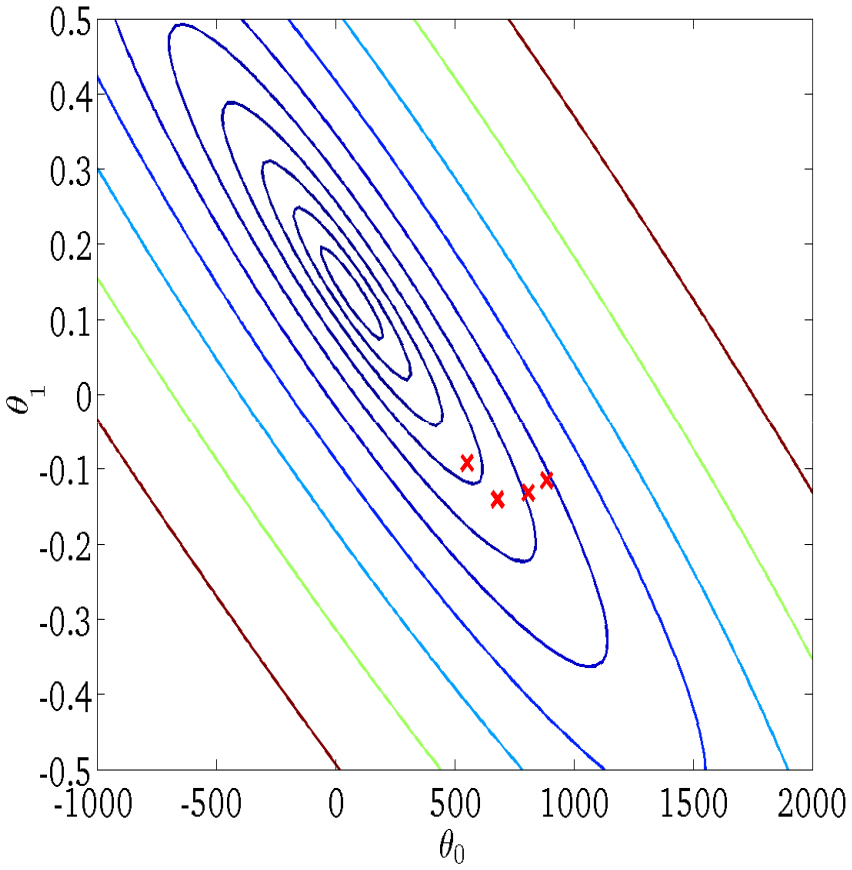
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



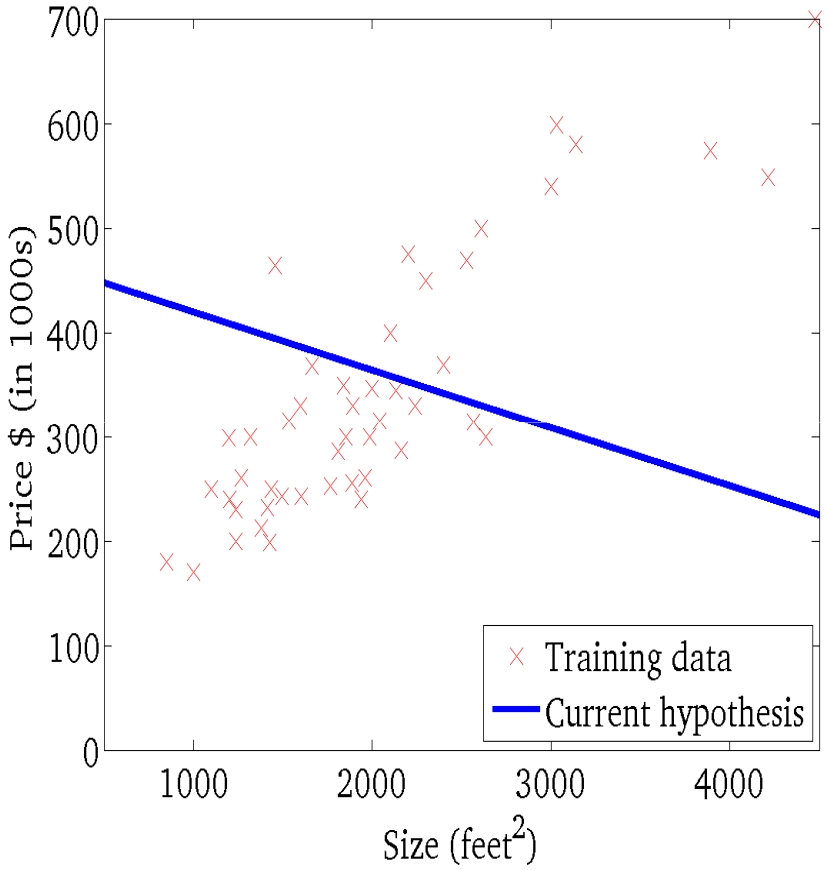
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



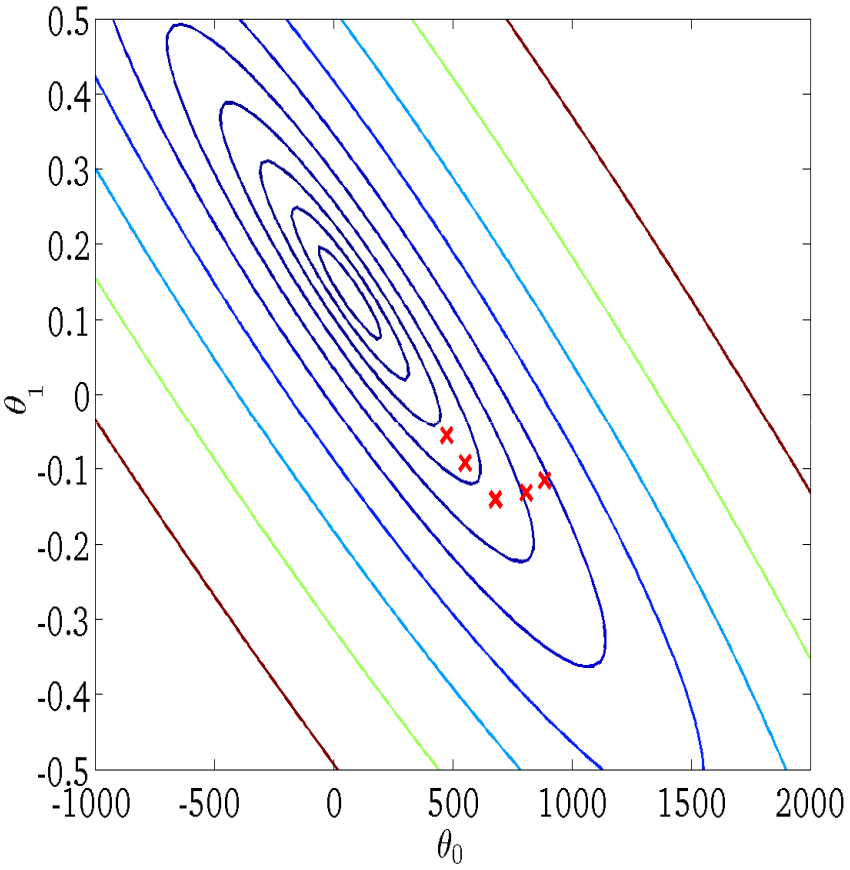
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



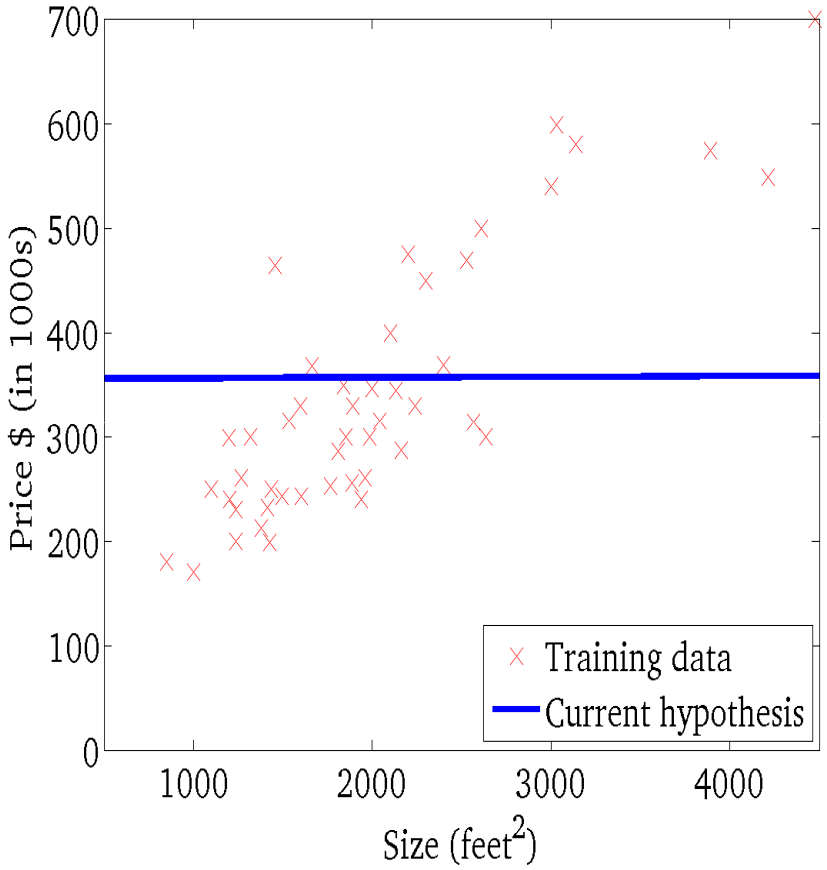
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



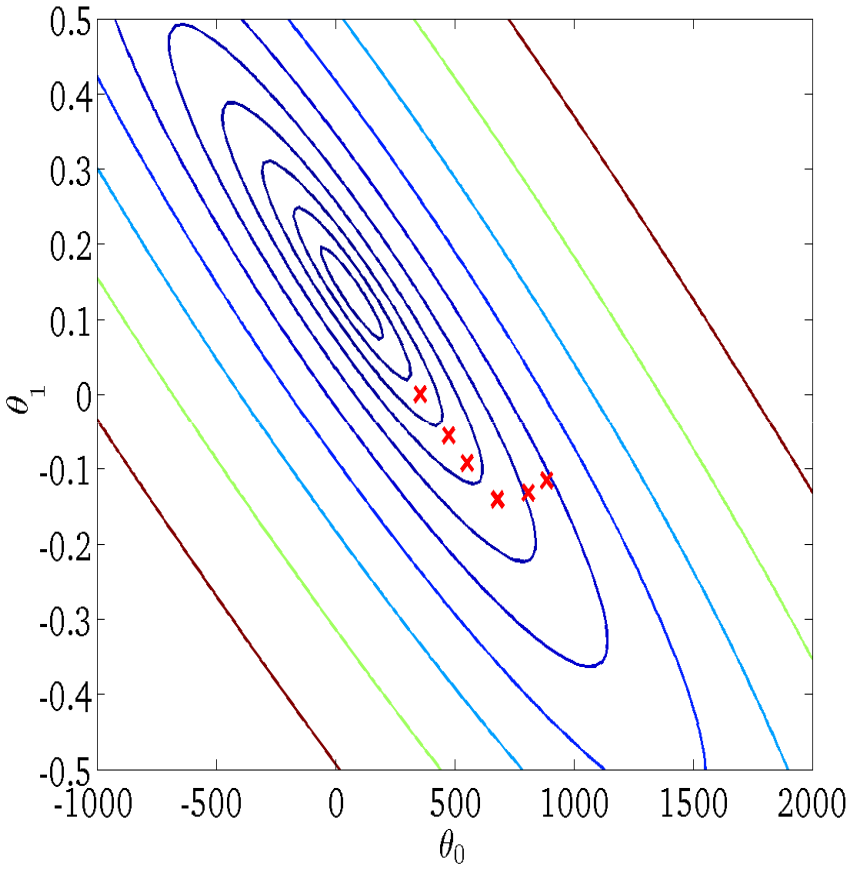
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



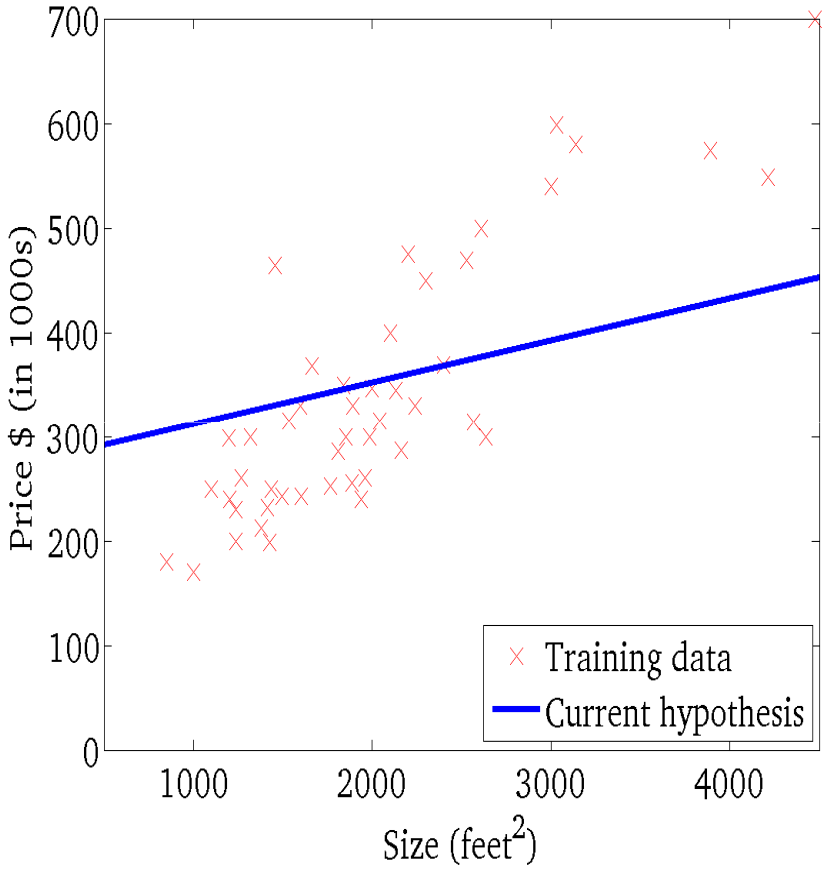
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



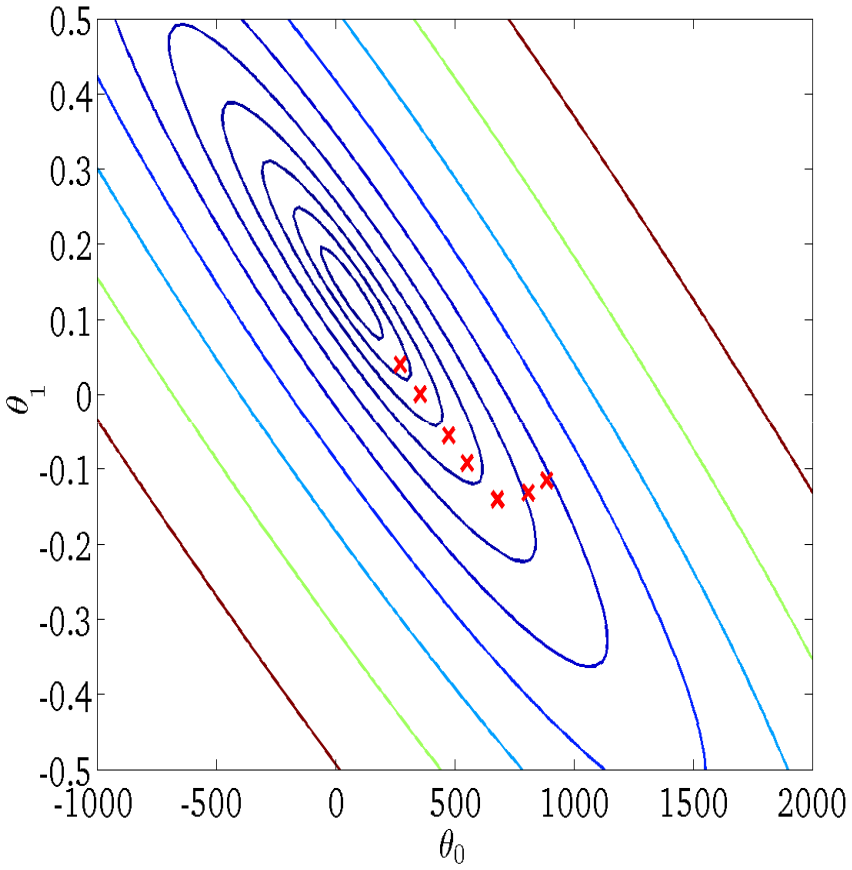
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



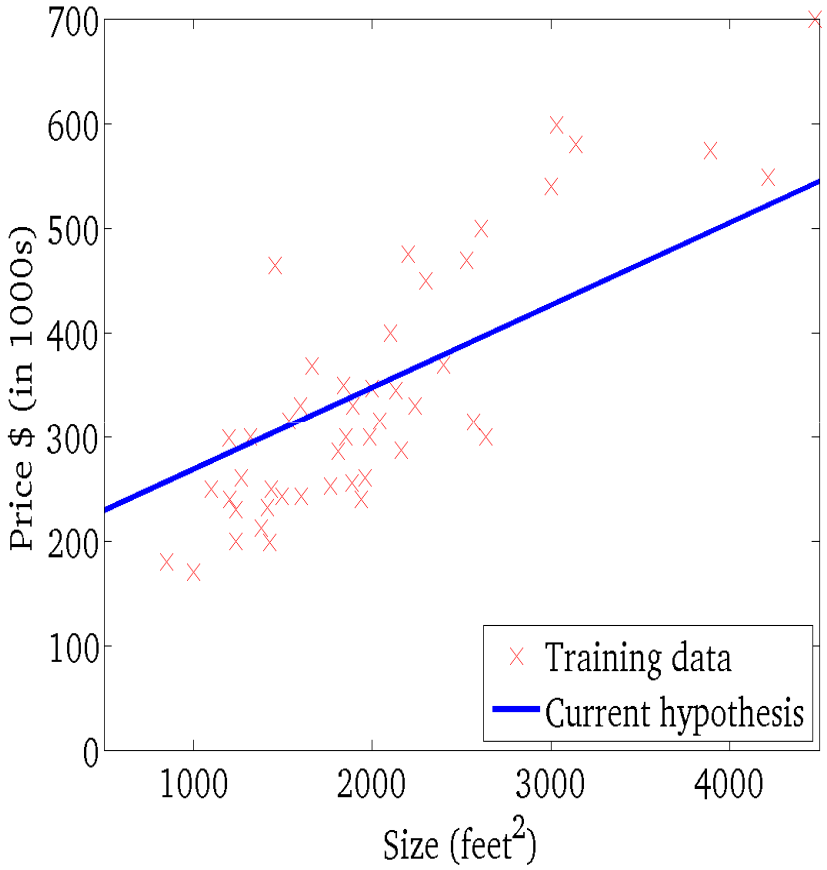
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



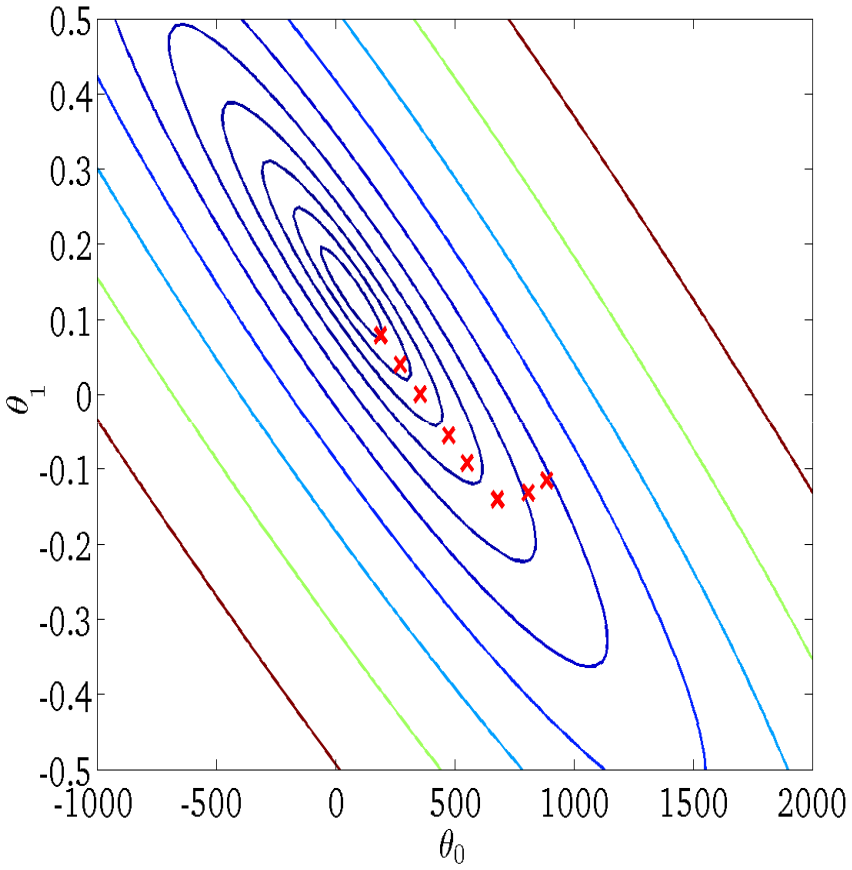
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



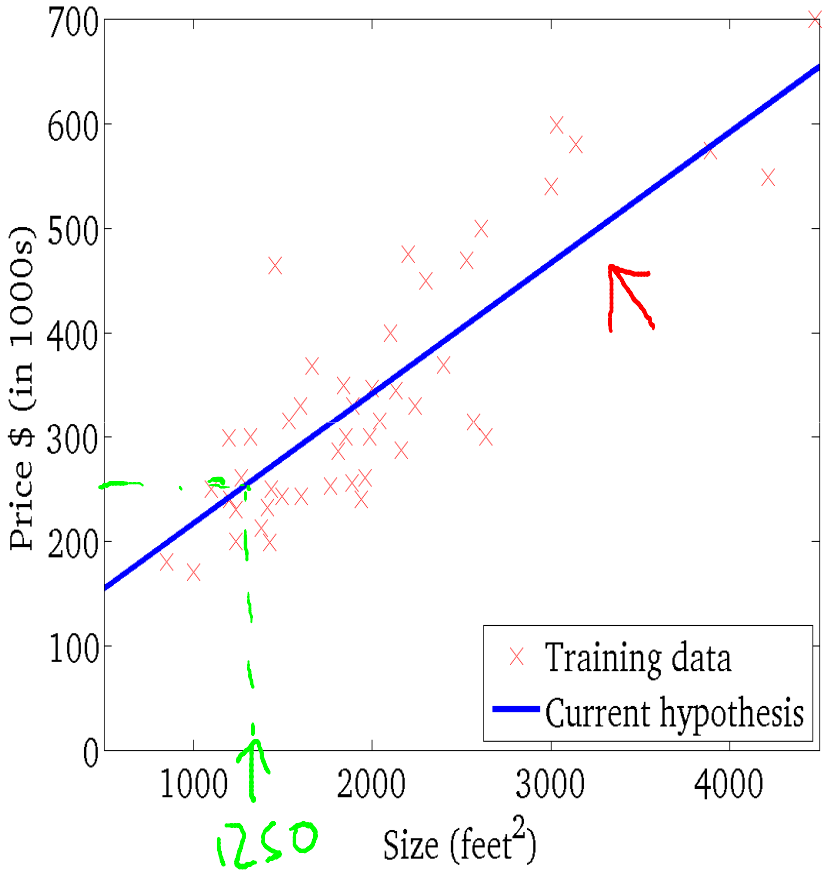
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



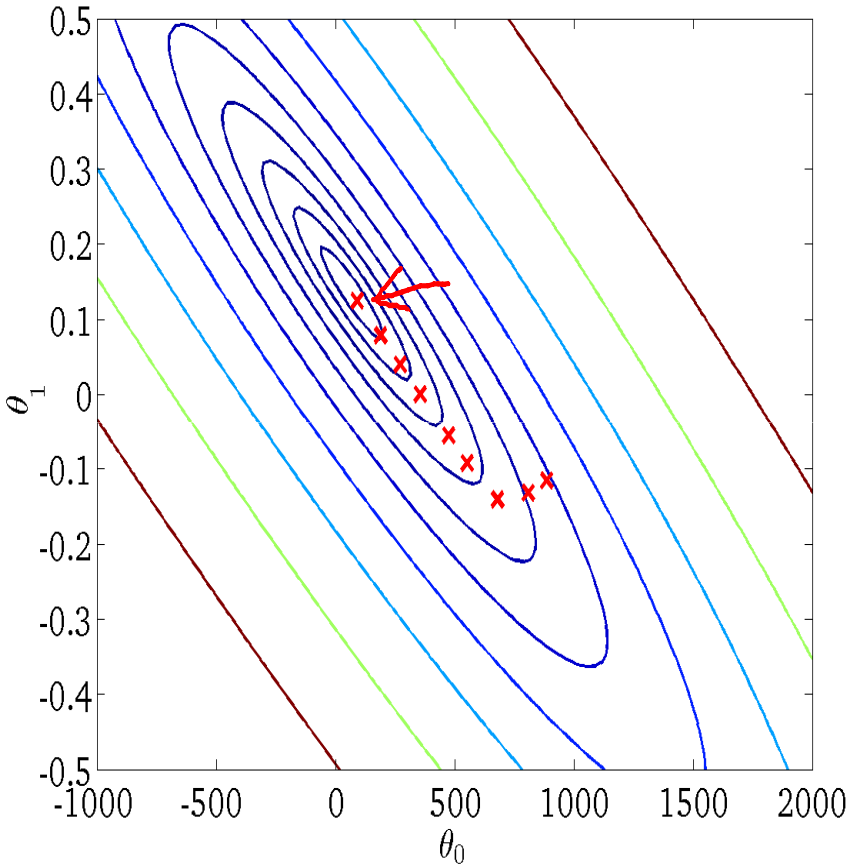
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Conjunto de treinamento Para o preço dos imóveis (Portland, OR)

Multiplas Características (variáveis).

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
…	…	…	…	…

Hipótese:

Anterior (para uma variável): $h_\theta(x) = \theta_0 + \theta_1 x$

→ Agora: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Por conveniência de notação, definimos $x_0 = 1$ $(x_0^{(i)} = 1)$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$h_\theta(x) = \underbrace{\theta_0 x_0}_{=1} + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$= \boxed{\theta^T x}.$$

θ^T: $(n+1) \times 1$ matrix

$\theta^T x$

x

Regressão Linear Multivariada.

Hipótese: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parâmetros: $\theta_0, \theta_1, \ldots, \theta_n$

Função de Custo:

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

Gradiente descendente:

Repeat $\{$

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

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Gradiente Descendente

Anterior (n=1):

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial\theta_0}J(\theta)}$$

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

(simultaneously update θ_0, θ_1)

$\}$

Novo algoritmo: $(n \geq 1)$

Repeat $\{$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$$

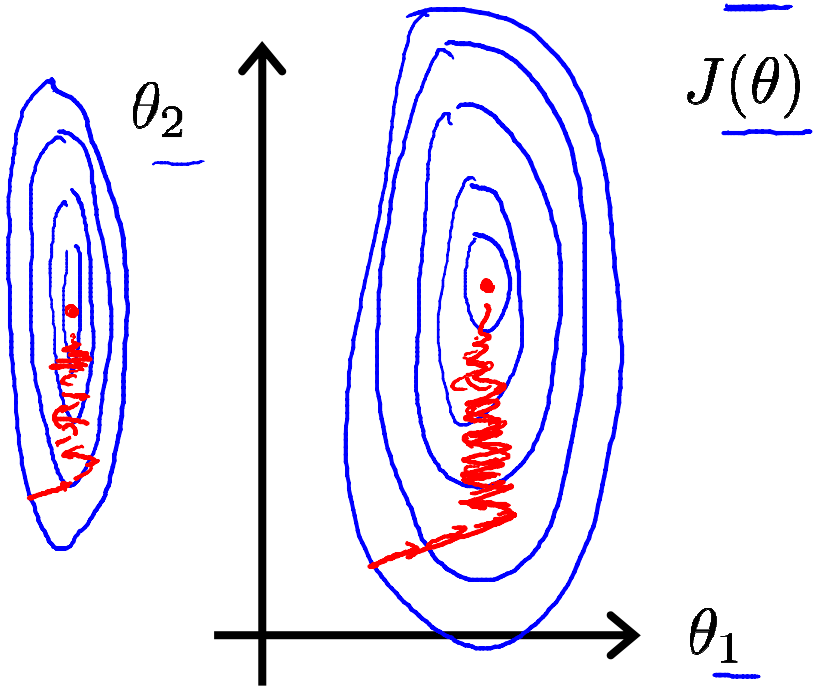
$\ldots$

Escalonamento (Normalização) das Variáveis

Ideia: Garantir que as variáveis estejam na mesma escala.

E.g. x_1 = size (0-2000 feet^2)

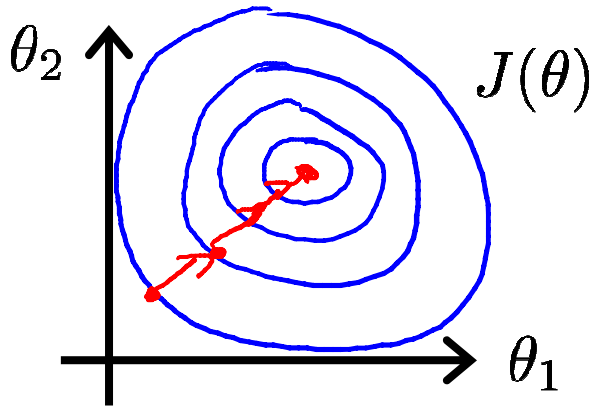
x_2 = number of bedrooms (1-5)



$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$

$0 \le x_1 \le 1$ $\quad 0 \le x_2 \le 1$



Escalonamento (Normalização) das Variáveis

Normalização linear entre ymin e ymax. $-1 \leq x_i \leq 1$

$$y = (ymax - ymin) * (x - xmin)/(xmax - xmin) + ymin$$

Normalização para media 0 e desvio padrão 1.

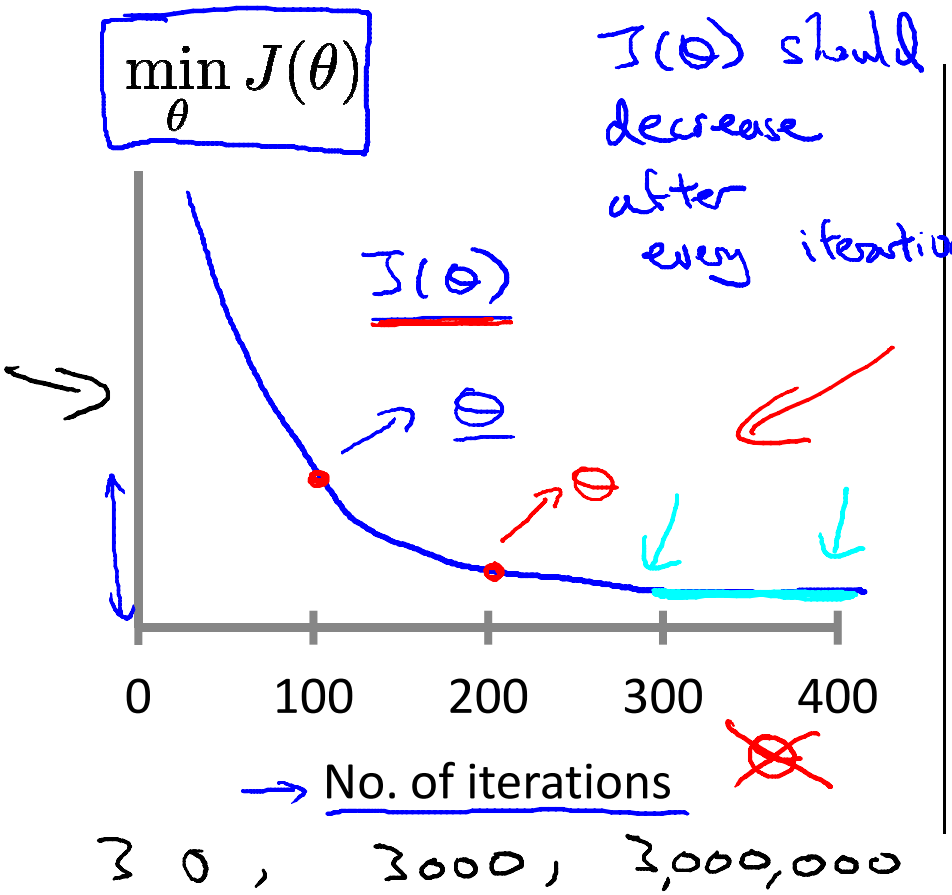
$$y = (x - xmean) * (ystd/xstd) + ymean$$

Gradiente descendente

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

- "Debugging": How to make sure gradient descent is working correctly.
- How to choose learning rate α.

Making sure gradient descent is working correctly.

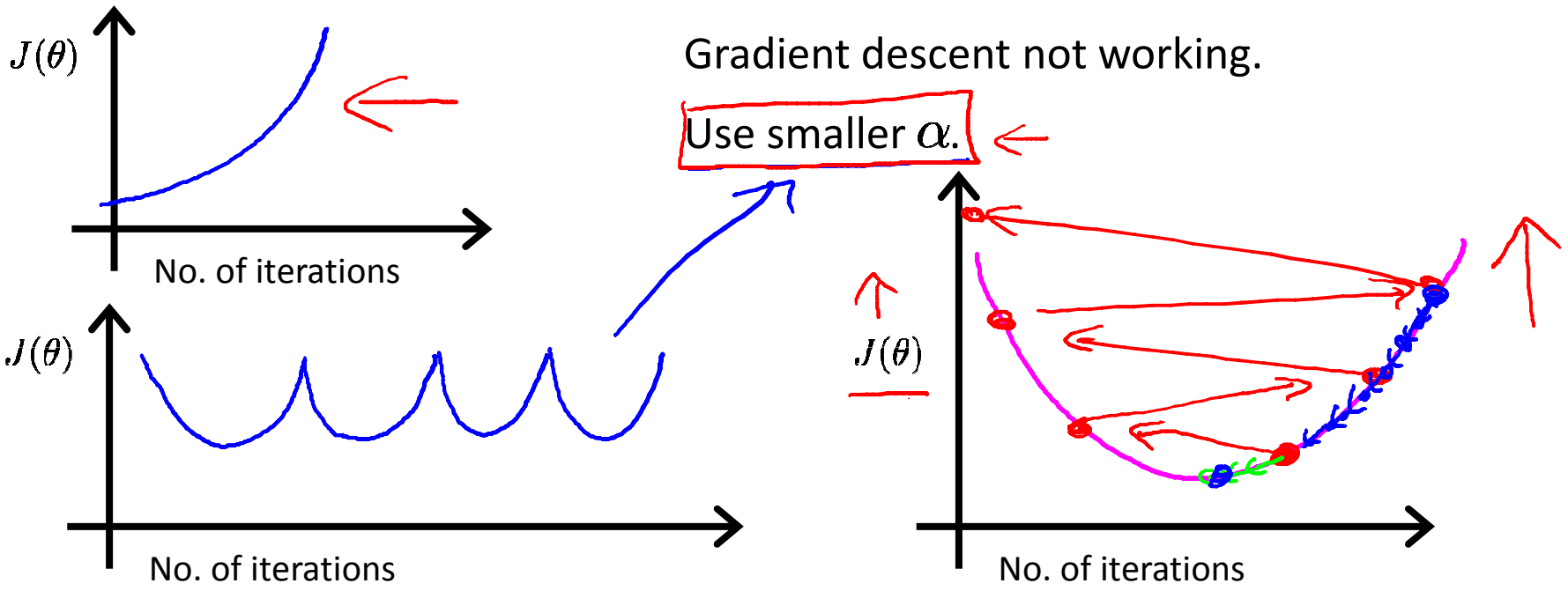


→ Example automatic convergence test:

→ Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.

ε

Making sure gradient descent is working correctly.



- For sufficiently small α, $J(\theta)$ should decrease on every iteration.
- But if α is too small, gradient descent can be slow to converge.

Summary:

- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

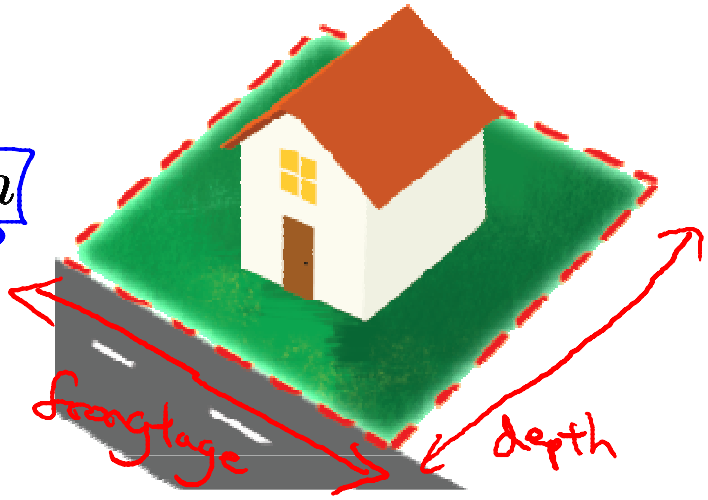
To choose α, try

$$\ldots, 0.001, \quad , 0.01, \quad , 0.1, \quad , 1, \ldots$$

Housing prices prediction

$$h_\theta(x) = \theta_0 + \theta_1 \times frontage + \theta_2 \times depth$$

x_1 x_2



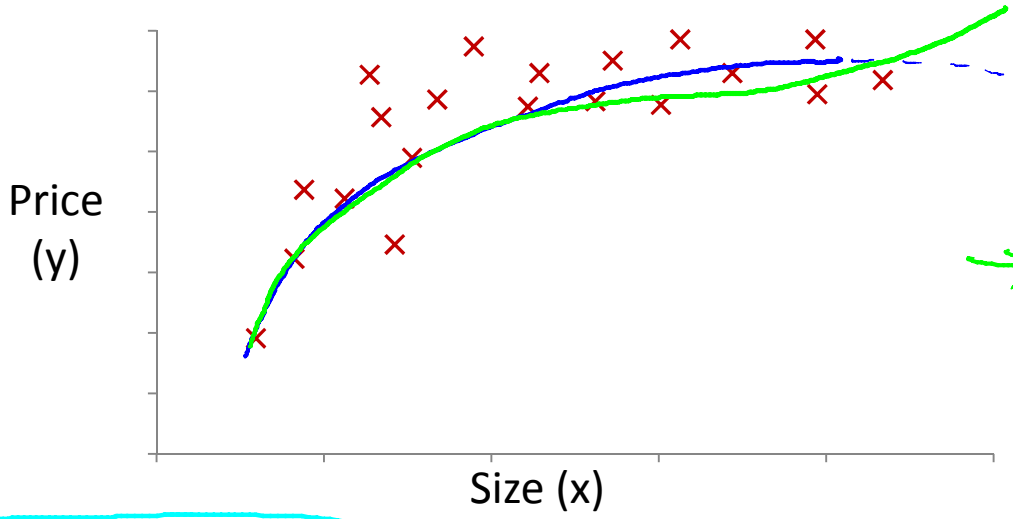
Area

x = frontage * depth

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

↑ land area

Regressão Polinomial



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$

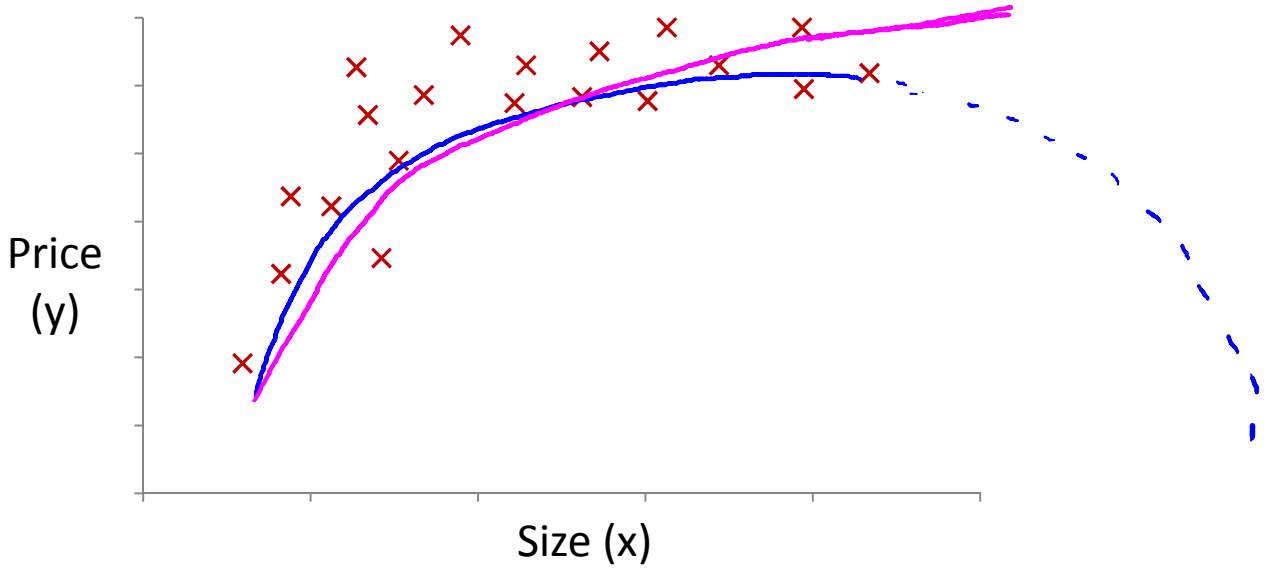
$$x_1 = (size)$$
$$x_2 = (size)^2$$
$$x_3 = (size)^3$$

Size: 1 - 1000

$Size^2$: 1 - 1000,000

$Size^3$: 1 - 10^9

Outras alternativas para “linearização” das variáveis



$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\sqrt{(size)}$$

INTRODUÇÃO AO ESTUDO DAS REDES NEURAIS ARTIFICIAS

Laboratório de Conexionismo e Ciências Cognitivas L3C
Grupo SICRES
INE - UFSC

Regressão Logística

Classification

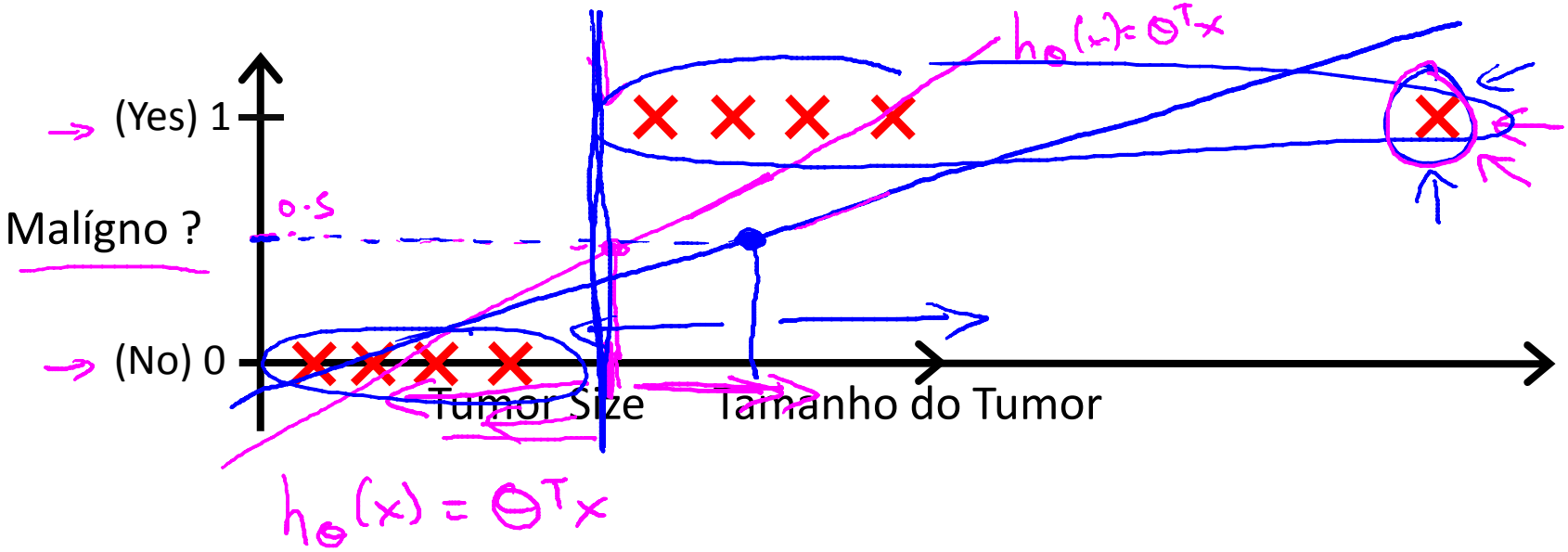
- Email: Spam / Not Spam?
- Trasnsações Online: Fraudulenta (Yes / No)?
- Tumor: Malígno / Benígno ?

$y \in \{0, 1\}$

0: “Negativo” (e.g., tumor benígno)

1: “Positivo” (e.g., tumor malígno)

$y \in \{0, 1, 2, 3\}$



→ Saída do classificador $h_\theta(x)$ para 0.5:

→ If $h_\theta(x) \geq 0.5$, “y = 1”

If $h_\theta(x) < 0.5$, “y = 0”

Classificação: $y = 0$ or 1

$h_\theta(x)$ Pode ser > 1 or < 0

Regressão Logística: $0 \leq h_\theta(x) \leq 1$

Classification

Regressão Logística

Representação da Hipótese

Modelo de Regressão Logística

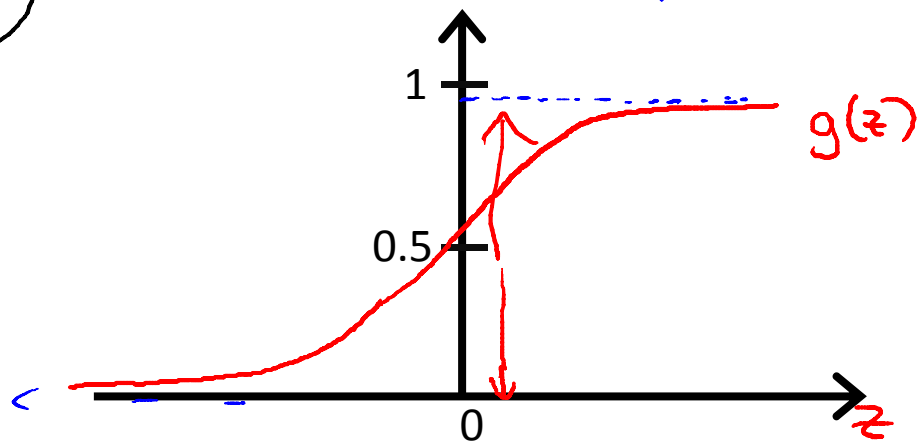
Queremos $0 \leq h_\theta(x) \leq 1$

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$

$z = \theta^T x$

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$



- Função Sigmoide
- Função Logística

Parametros θ.

$$h_\theta(x) = g(\theta^T x) \qquad g(z) = \frac{1}{1+e^{-z}}$$

Interpretação da Saída da Hipótese

$h_\theta(x)$ = probabilidade estimada de y = 1 , entrada x

Examplo: Se $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$h_\theta(x) = 0.7$$

70% de chance do tumor ser benígno

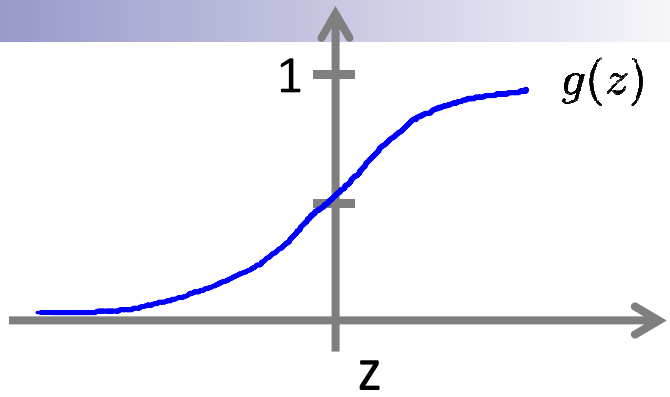
"probabilidade de y = 1, dado x, parametrizado por θ"

$$P(y = 0|x;\theta) + P(y = 1|x;\theta) = 1$$
$$P(y = 0|x;\theta) = 1 - P(y = 1|x;\theta)$$

Regressão Logística

$$h_\theta(x) = g(\theta^T x)$$

$$g(z) = \frac{1}{1+e^{-z}}$$



Suponha prever "$y = 1$" se $h_\theta(x) \geq 0.5$

$\theta^T x \geq 0$

$g(z) \geq 0.5$ when $z \geq 0$

$h_\theta(x) = g(\theta^T x)$

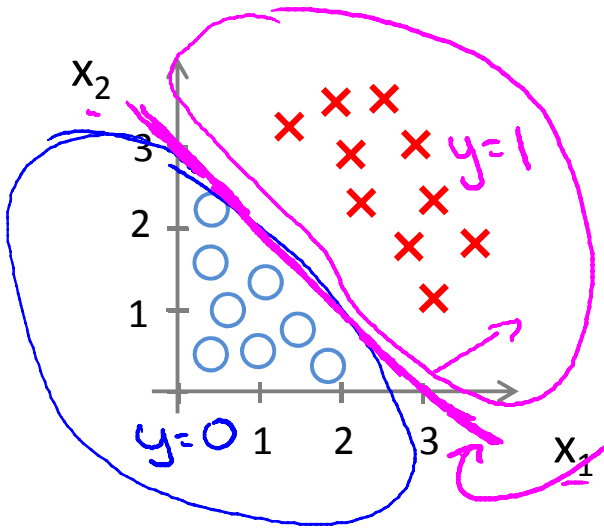
prever "$y = 0$" se $h_\theta(x) < 0.5$

$\theta^T x < 0$

$g(z) < 0.5$ when $z < 0$

Fronteira de Decisão

$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

Decision boundary

Prever "$y = 1$" se $-3 + x_1 + x_2 \geq 0$

$\theta^T x$

x_1, x_2

$h_\theta(x) = 0.5$

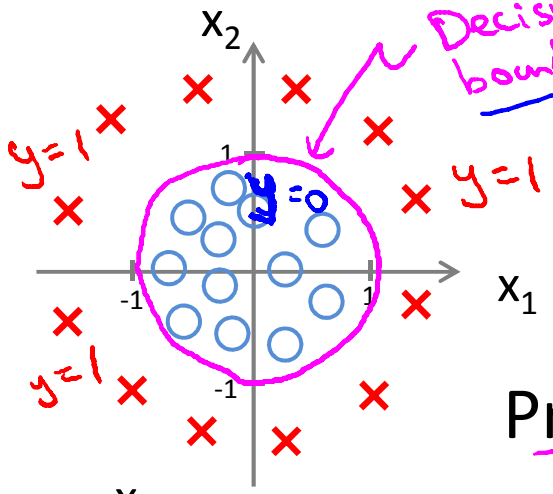
$x_1 + x_2 = 3$

$x_1 + x_2 \geq 3$

$x_1 + x_2 < 3$

$y = 0$

Fronteiras de Decisão Não-lineares



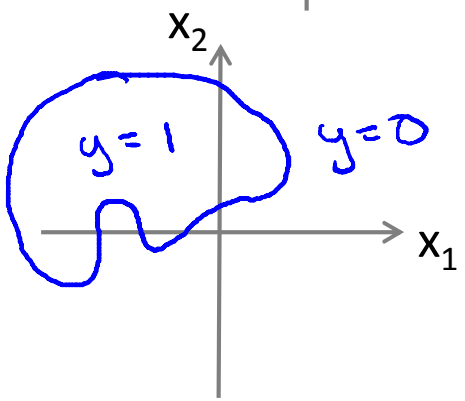
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Prever "$y = 1$" se $-1 + x_1^2 + x_2^2 \geq 0$

$x_1^2 + x_2^2 = 1$

$x_1^2 + x_2^2 \geq 1$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \ldots)$$

Função Custo

Conjunto de Treino: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m exemplos $\quad x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \quad x_0 = 1, y \in \{0, 1\}$

$\mathbb{R}^{n+1}$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

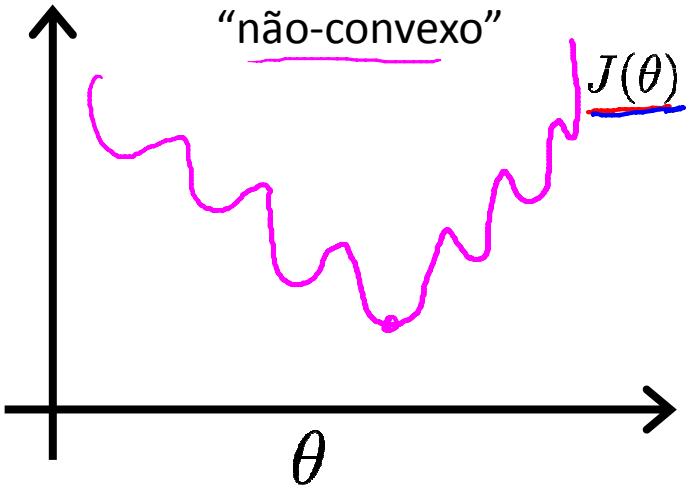
Como escolher o parâmetro θ ?

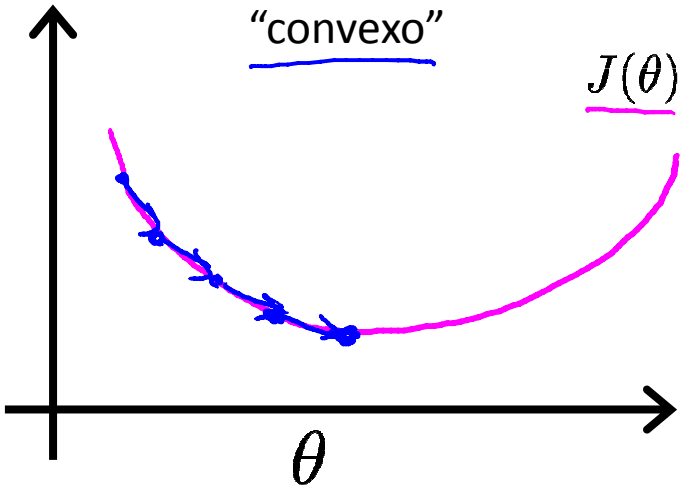
Função Custo

Regressão Linear:

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

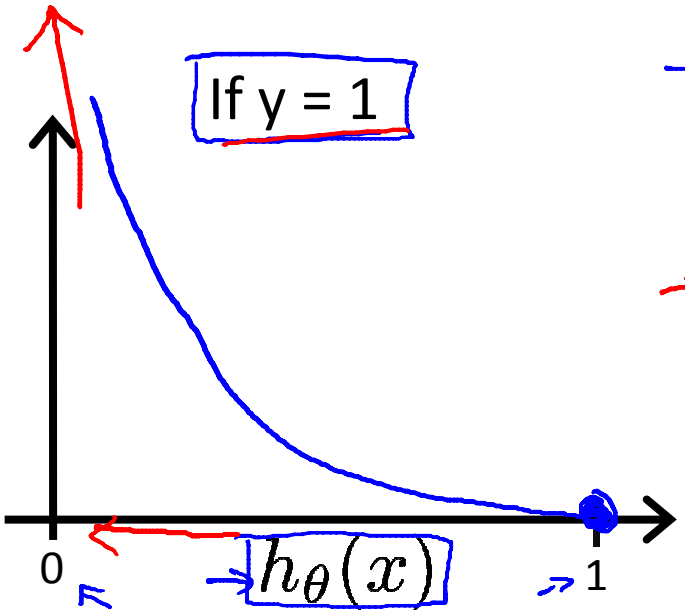
$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$





Função Custo da Regressão Logística

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



$\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$

But as $h_\theta(x) \to 0$

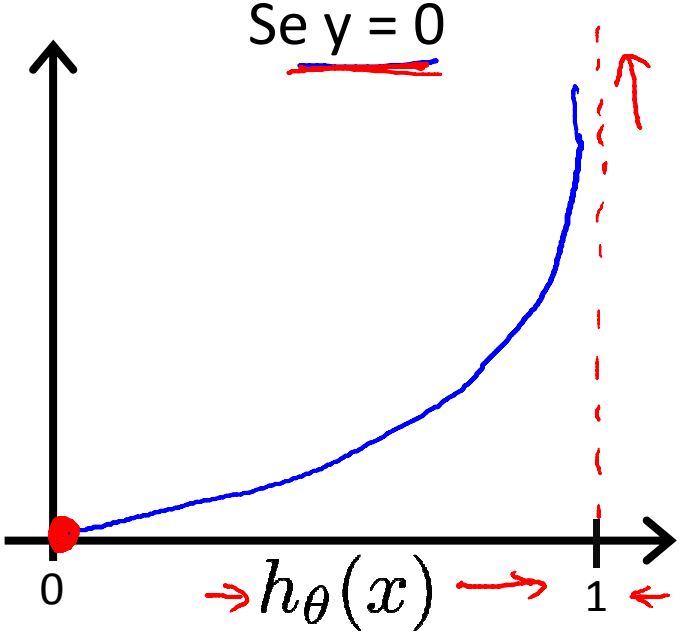
$Cost \to \infty$

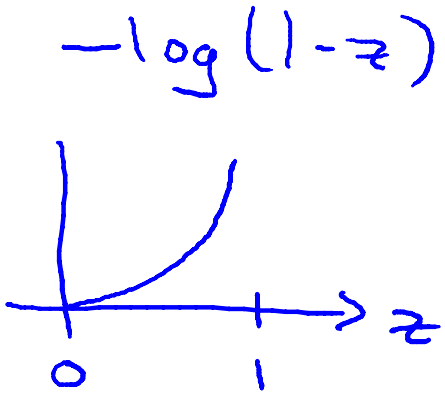
Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Função Custo da Regressão Logística

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Se y = 0





Regressão Logística

Função custo simplificada e gradiende descendente

Função custo da regressão logística

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

If $y = 1$: $\text{Cost}(h_\theta(x), y) = -\log h_\theta(x)$

If $y = 0$: $\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x))$

Função custo da regressão logística

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Ajustar o parâmetro θ :

$$\min_\theta J(\theta)$$

Get θ

Fazer previsão dado um novo x :

Saída $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$p(y=1 \mid x;\theta)$

Gradiente Descendente

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Desejamos $\min_\theta J(\theta)$:

Repeat $\{$

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

$\}$

(Atualiza simultaneamente θ_j)

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Desejamos $\min_\theta J(\theta)$:

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$ for $i = 0$ to n

Repeat $\{$

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

(atualiza simultaneamente θ_j)

$\}$

$h_\theta(x) = \theta^T x$

$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$

Algoritmo parece idêntico à regressão linear!

Regressão Logística

Otimização Avançada

Algoritmo de Otimização

Função custo $J(\theta)$. Desejamos $\min_\theta J(\theta)$.

Dado θ, podemos computar

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Gradiente descendente:

Repeat $\{$

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

$\}$

Algoritmo de Otimização

Dado θ, podemos computar

- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Algoritmos de Otimização:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Vantagens:

- Sem atribuição manual de α
- Sempre é mais rápido que o gradient descendente.

Desvantagens:

- Mais complexo

Exemplo:

$\min_{\theta} J(\theta)$

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ $\theta_1 = 5, \theta_2 = 5.$

$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

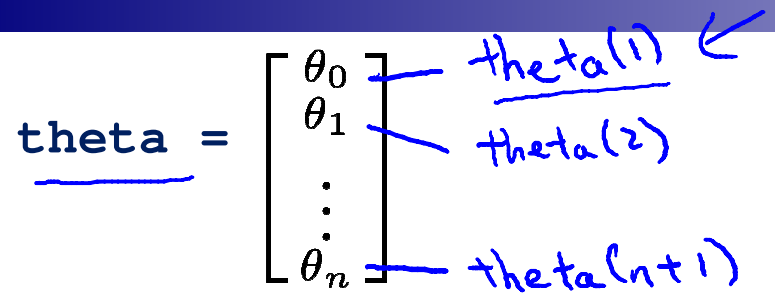
$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

```
function [jVal, gradient]
            = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradiente = zeros(2,1);
  gradiente(1) = 2*(theta(1)-5);
  gradiente(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
      = fminunc(@costFunction, initialTheta, options);
```

$\theta \in \mathbb{R}^d \quad d \geq 2.$




```
function [jVal, gradient] = costFunction(theta)
```

```
  jVal = [código a computar J(θ)] ;
```

```
gradiente(1) = [código a computar ∂/∂θ0 J(θ)] ;
```

```
gradiente(2) = [código a computar ∂/∂θ1 J(θ)] ;
```

```
  ⋮
```

```
gradiente(n+1) = [code to compute ∂/∂θn J(θ)] ;
```

Regressão Logística

Classificação multiclasse: Um-vs-Todos

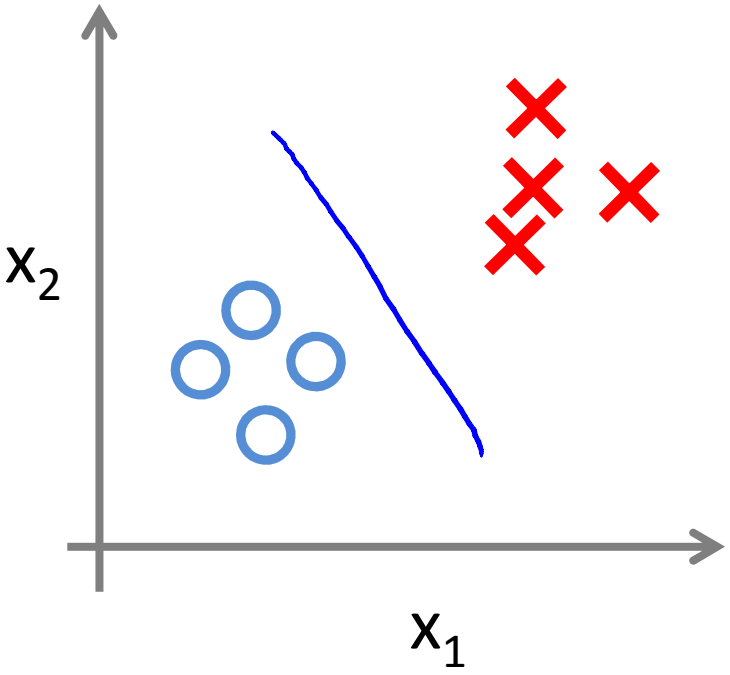
Classificação Multiclasse

Filtro de email: Trabalho, Amigos, Família

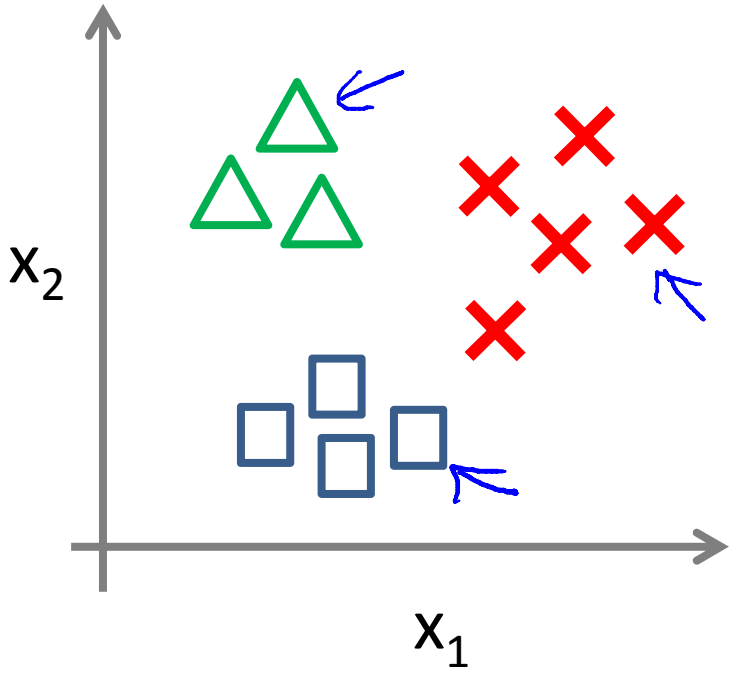
Diagramas médicos: Doente, Resfriado, Gripado

Tempo: Ensoladarp, Nublado, Chuvoso, Neve

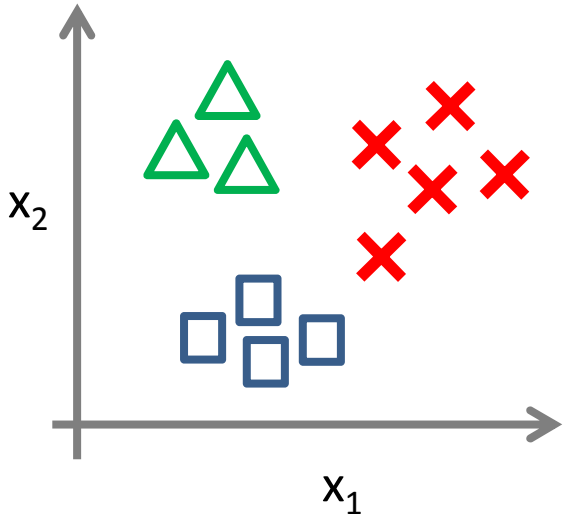
Classificação binária:



Classificação multiclasse



Um-vs-Todos(um-vs-resto):

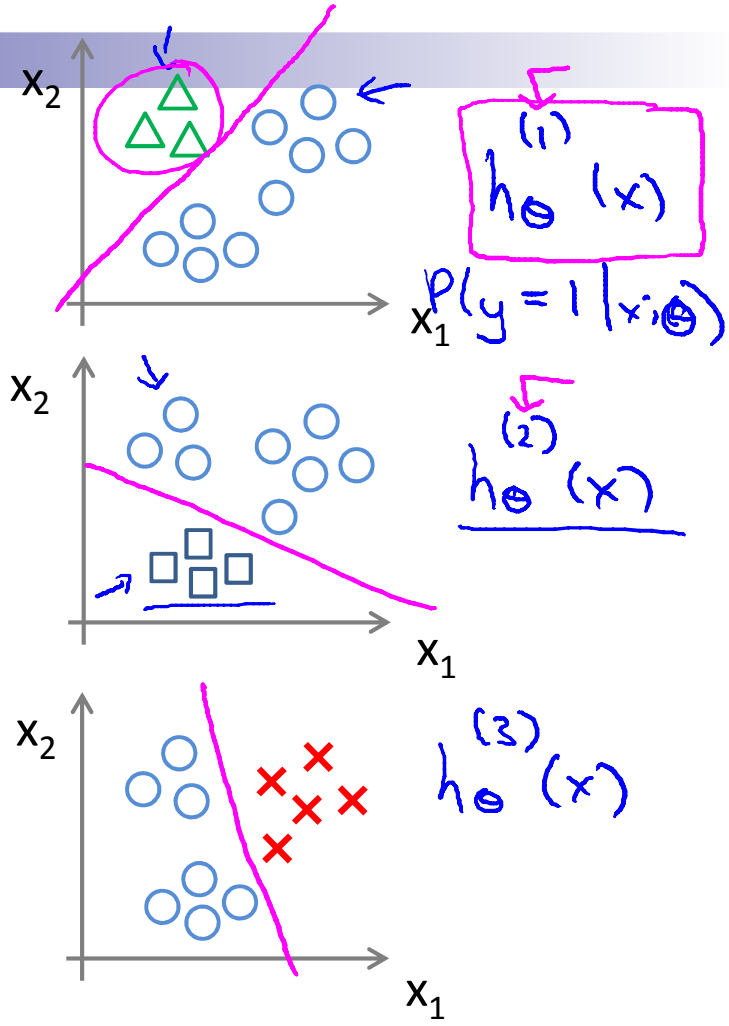


Classe 1: △ ←

Classe 2: □ ←

Classe 3: ✕ ←

$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1, 2, 3)$$



Um-vs-Todos

Treinar um classificador de regressão logística $h_\theta^{(i)}(x)$ para cada classe i para prever a probabilidade de $y = i$

Em uma nova entrada x, encontrar a classe i que maximiza

$$\max_i h_\theta^{(i)}(x)$$