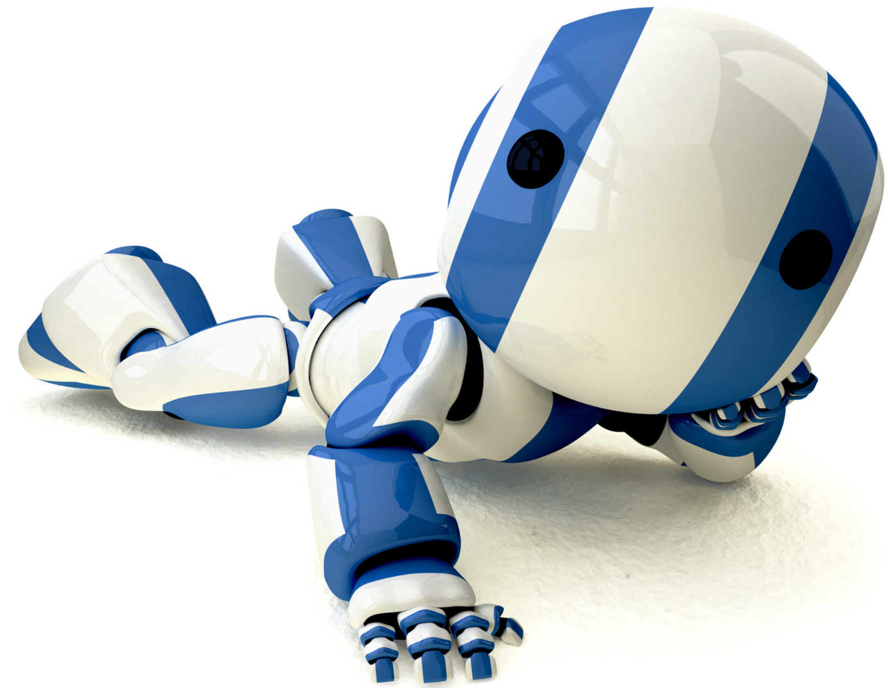
PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA

Aprendizado de Máquina



Paradigma baseado em Otimização
Parte I: Regressão Linear e Logística

Prof. Me. Otávio Parraga

MALTA

Machine Learning Theory and Applications Lab

Aula de Hoje

- Revisão de Álgebra
- Regressão Linear
  - Gradiente Descendente
  - Solução Analítica
- Regressão Logística

Aula de Hoje

- Revisão de Álgebra
- Regressão Linear
  - Gradiente Descendente
  - Solução Analítica
- Regressão Logística

Revisão de Álgebra

- Matrizes:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1k} \\ 1 & x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_k \end{bmatrix}, \qquad \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

Revisão de Álgebra

- Matrizes:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}^T \longrightarrow \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

Revisão de Álgebra

- Utilizaremos uma matriz de pesos
- $w = \theta$
- Podemos chama-los de w (literatura de redes neurais)
- Podemos chama-los de θ (literatura de regressão)
- Podem aparecer outros termos para representar o mesmo: β

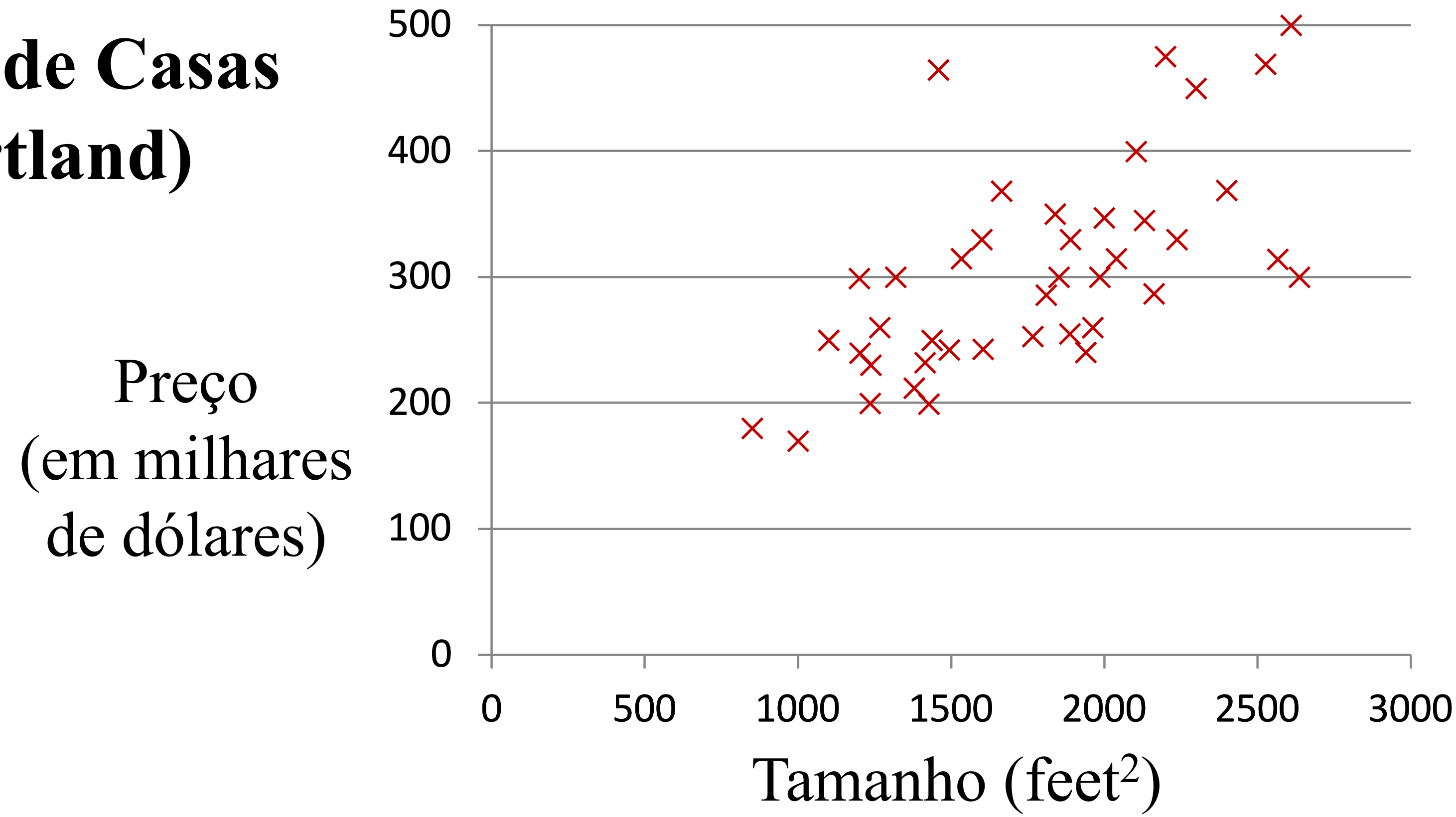
Revisão de Álgebra

- *Para poder multiplicar matrizes:* $A^{NxM}\, B^{MxN}$
- ${A^T}^T = A$
- $(AB)^T = B^T A^T$
- $(A+B)^T = A^T + B^T$
- $\nabla_x(a^T x) = \nabla_x(x^T a) = a$
- $\nabla_x(x^T A x) = 2Ax$ (Caso A seja simétrica)

Aula de Hoje

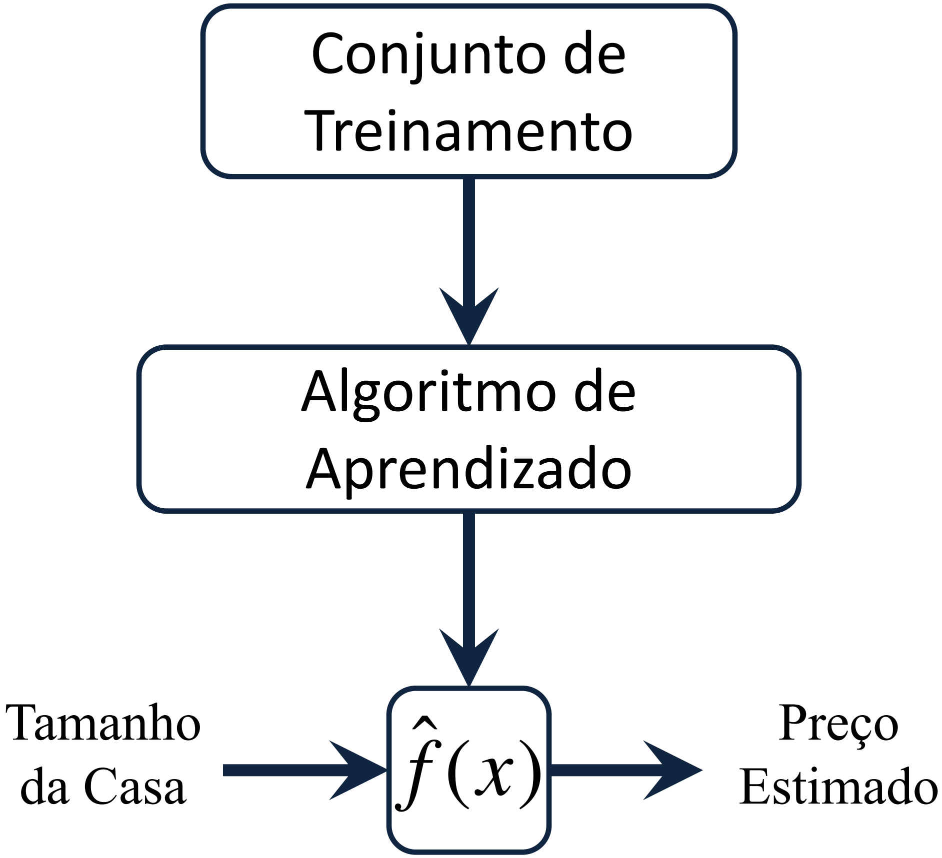
- Revisão de Álgebra
- Regressão Linear
  – Gradiente Descendente
  – Solução Analítica
- Regressão Logística

Preços de Casas (Portland)



Conjunto de Treinamento (Portland)

Tamanho em feet^2 (x)	Preço ($) em milhares ($f(x)$)
2104	460
1416	232
1534	315
852	178
...	...



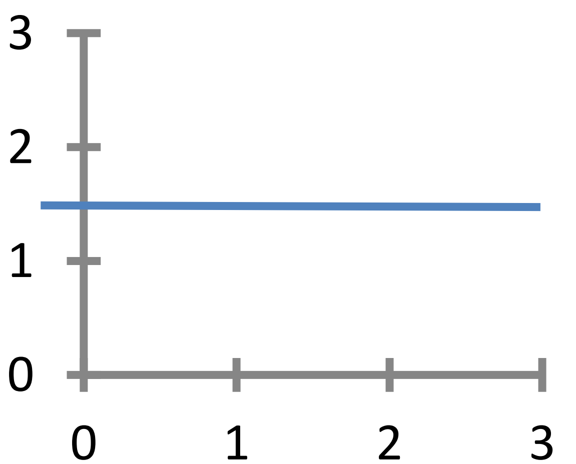
Regressão Linear Univariada

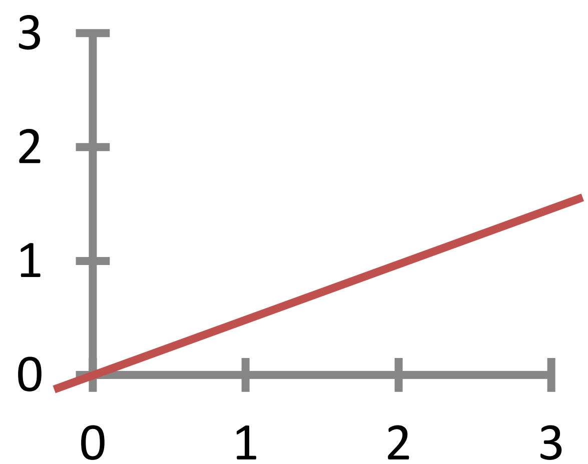
$$\hat{f}(x) = w_0 + w_1 x$$

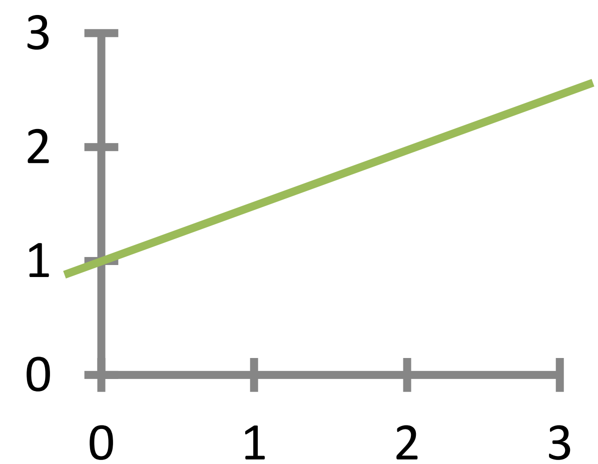
A ideia:

Ajustar uma reta ao conjunto de dados

$$\hat{f}\,(x) = w_0 + w_1 x$$







$w_0 = 1.5$
$w_1 = 0$

$$\boxed{\hat{f}(x) = 1.5 + 0x}$$

$w_0 = 0$
$w_1 = 0.5$

$$\boxed{\hat{f}(x) = 0 + 0.5x}$$

$w_0 = 1$
$w_1 = 0.5$

$$\boxed{\hat{f}(x) = 1 + 0.5x}$$

A ideia

- Como comparar nossa predição com o alvo?

$$\hat{f}(x) = w_0 + w_1 x$$

A ideia

- Como comparar nossa predição com o alvo?

$$\hat{f}(x) = w_0 + w_1 x$$

$$\hat{f}(x) - f(x)$$

- Como comparar nossa predição com o alvo?

$$\hat{f}\,(x) = w_0 + w_1 x$$

$$\hat{f}\,(x) - f(x)$$

- Para considerar todas as instâncias da base

$$\sum_{i=1}^{N} \hat{f}\,(x) - f(x)$$

- Problema! Valores negativos e positivos se anulam...

- Como comparar nossa predição com o alvo?

$$\sum_{i=1}^{N}(\hat{f}\,(x) - f(x))^2$$

- Também chamada de *Sum of Squared Errors (SSE)* ou *Sum of Squared Residuals (SSR)*

$$SSR = \sum_{i=1}^{N}(\hat{f}\,(x) - f(x))^2$$

Função de Custo

$$J(w_0, w_1) = \sum_{i=1}^{N} (\hat{f}\,(x) - f(x))^2$$

- Determina o quão boa é uma reta para o conjunto de dados
- Onde está o *w* na função?

$$J(w_0, w_1) = \sum_{i=1}^{N} ((w_0 + w_1 x) - f(x))^2$$

Função de Custo

$$J(w_0, w_1) = \sum_{i=1}^{N} ((w_0 + w_1 x) - f(x))^2$$

- Gostaríamos que ela fosse máxima ou mínima?

Função de Custo

$$J(w_0, w_1) = \sum_{i=1}^{N} ((w_0 + w_1 x) - f(x))^2$$

- Gostaríamos que ela fosse máxima ou mínima?
- Quanto menor o erro, melhor, então:

$$\min_{w_0, w_1} J(w_0, w_1)$$

Modelo: $\hat{f}\,(x) = w_0 + w_1 x$

Parâmetros: w_0, w_1

Função de Custo: $J(w_0, w_1) = \sum_{i=1}^{N}((w_0 + w_1 x) - f(x))^2$

Objetivo: $\min_{w_0, w_1} J(w_0, w_1)$

Modelo: $\hat{f}\,(x) = w_0 + w_1 x$

Parâmetros: w_0, w_1

Função de Custo: $J(w_0, w_1) = \sum_{i=1}^{N}((w_0 + w_1 x) - f(x))^2$

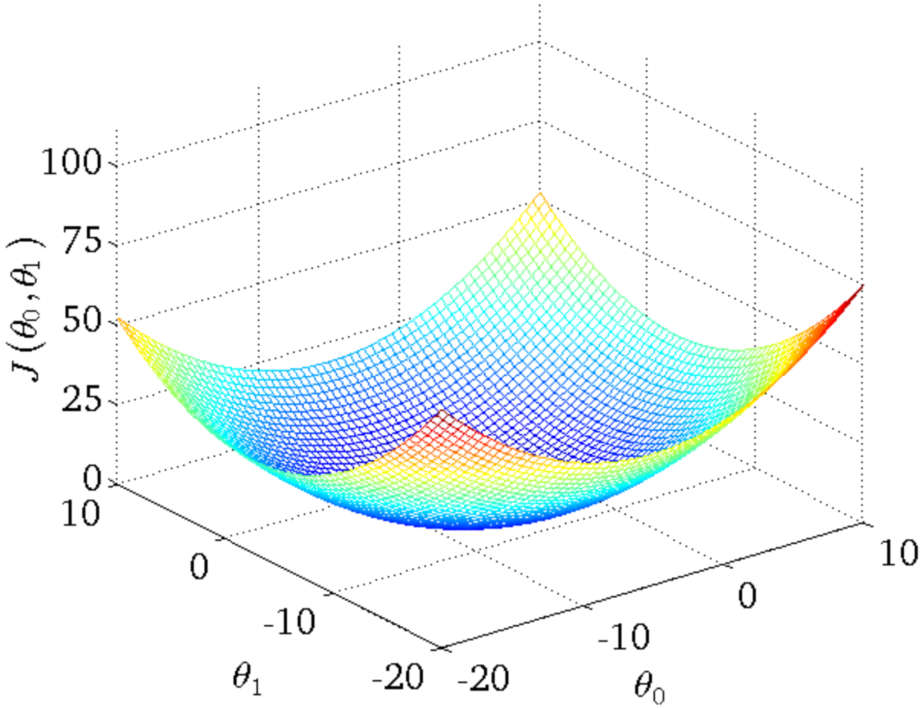
Objetivo: $\min\limits_{w_0, w_1} J(w_0, w_1)$

Como minimizamos?

Como minimizar $J(w_0, w_1)$?

- Conseguimos saber como nossa função $J(w_0, w_1)$ se parece

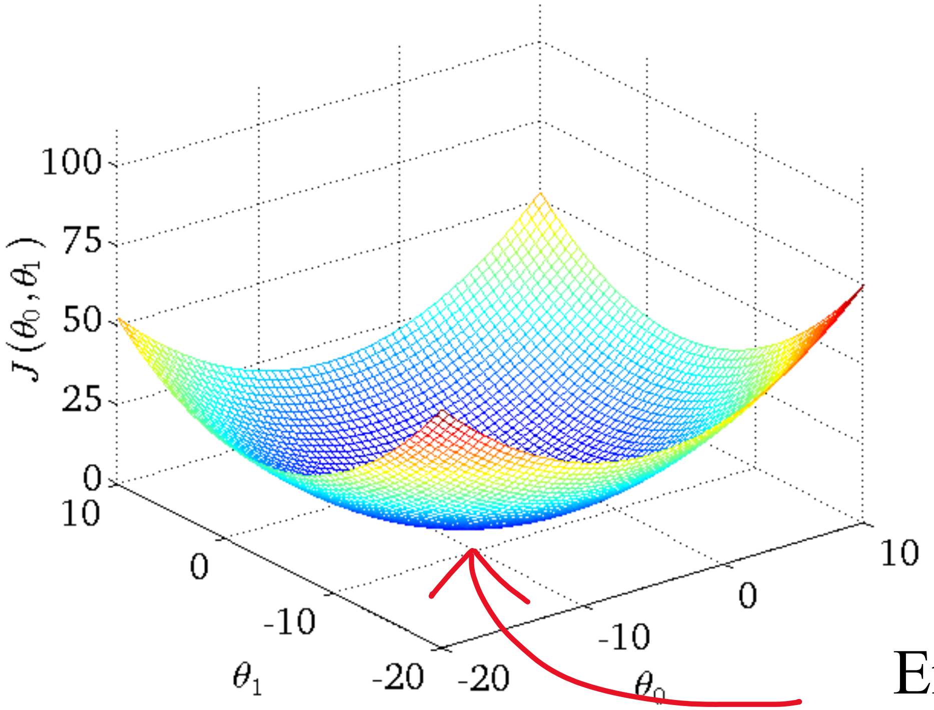
Lembrando
$w = \theta$



Como minimizar $J(w_0, w_1)$?

- Conseguimos saber como nossa função $J(w_0, w_1)$ se parece

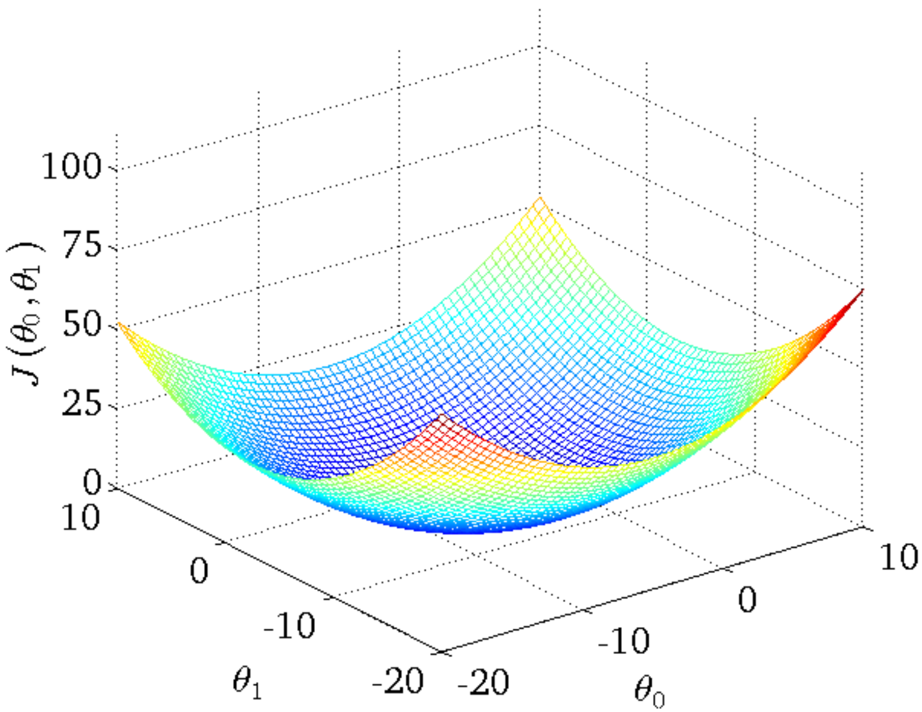
$Lembrando$
$w = \theta$



Erro mínimo

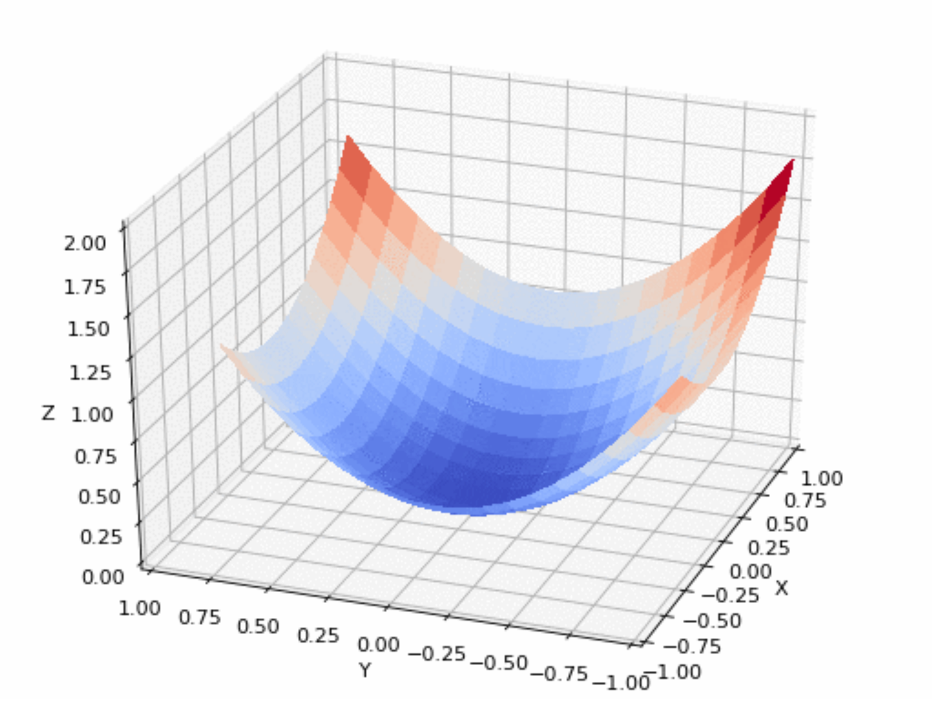
Como minimizar $J(w_0, w_1)$?

- Como chegamos lá? Duas alternativa
  - Gradiente Descendente
  - Solução Analítica



Gradiente Descendente

- Método iterativo para “andar pela função”

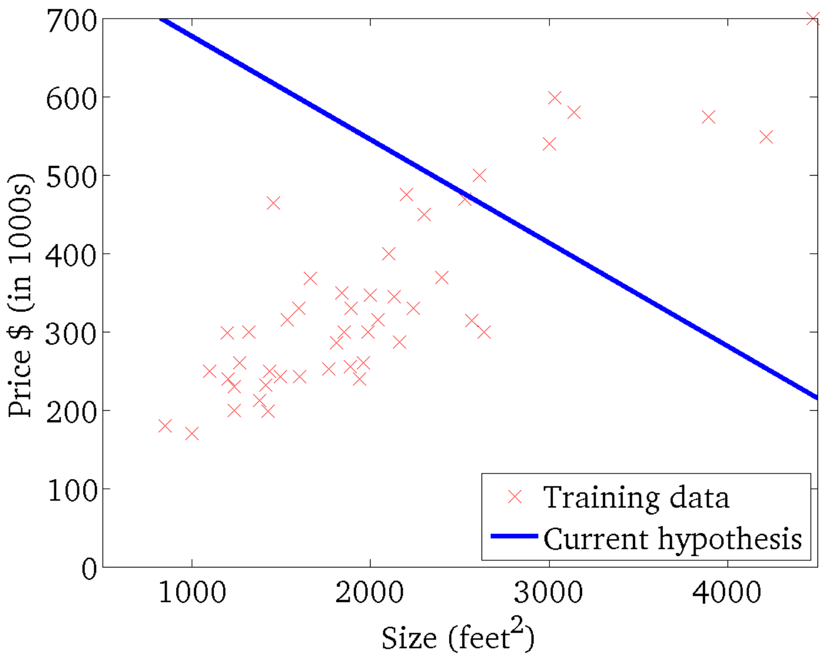


Gradiente Descendente

- (Proto) Algoritmo:
  - Inicialize com algum valor para w_0, w_1
  - Modifique incrementalmente w_0, w_1 para reduzir $J(w_0, w_1)$ até que possivelmente tenha sido minimizada
  - Algoritmo iterativo
  - Ponderado por uma Taxa de Aprendizado

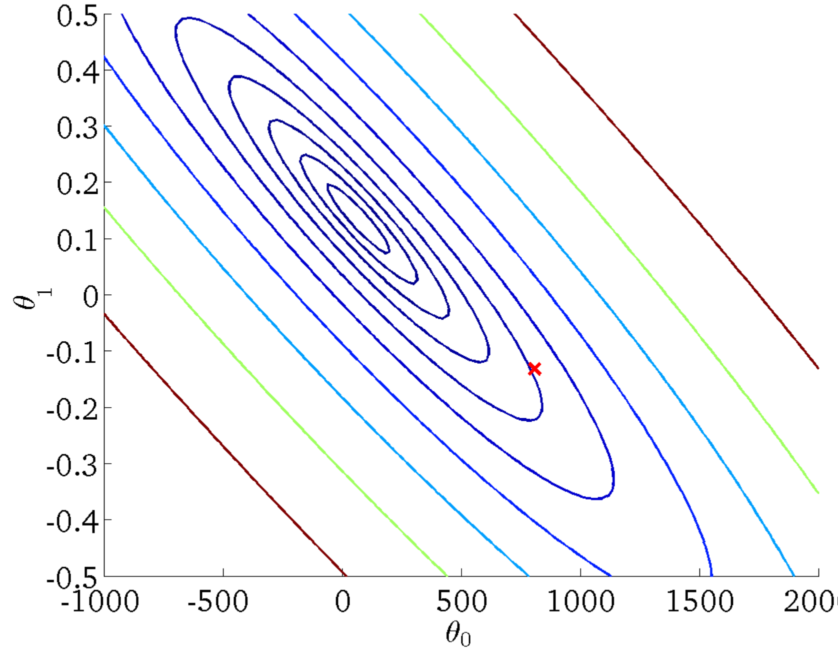
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de *x)*



$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)

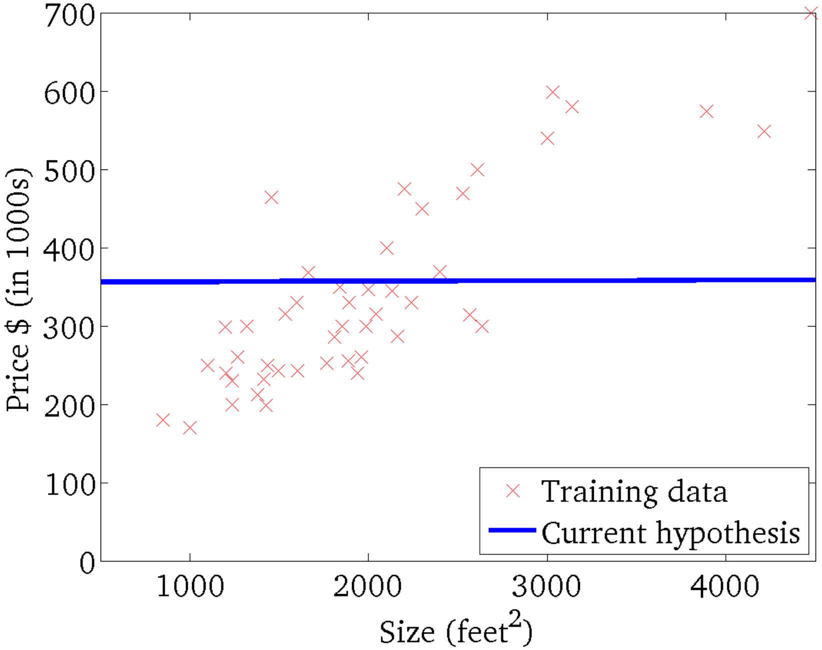


$$\theta_0 = 800$$

$$\theta_1 = -0.15$$

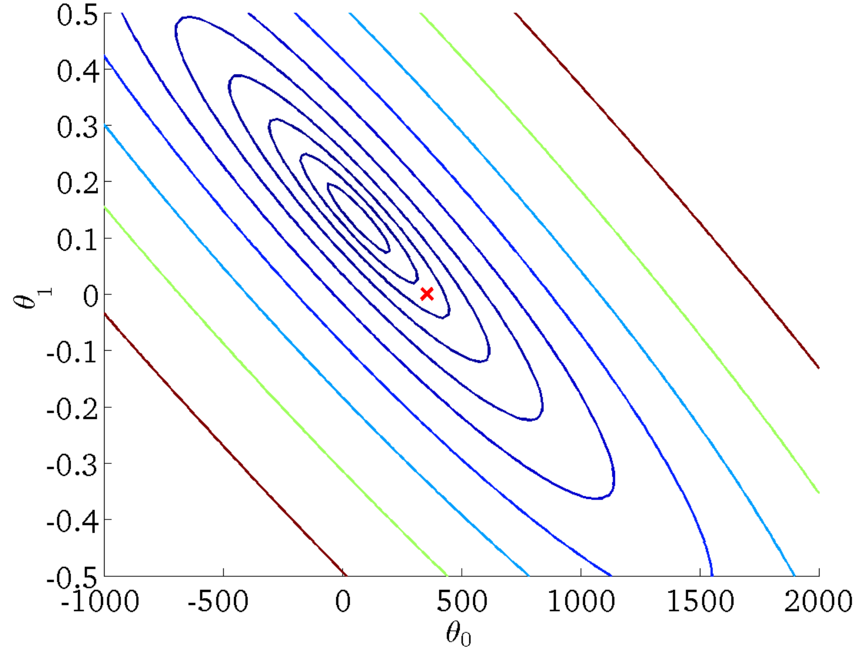
$\hat{f}(x)$

(Para valores fixos de θ_0, θ_1 , é uma função de x)



$J(\theta_0, \theta_1)$

(Função dos parâmetros θ_0, θ_1)



$$\theta_0 = 360$$

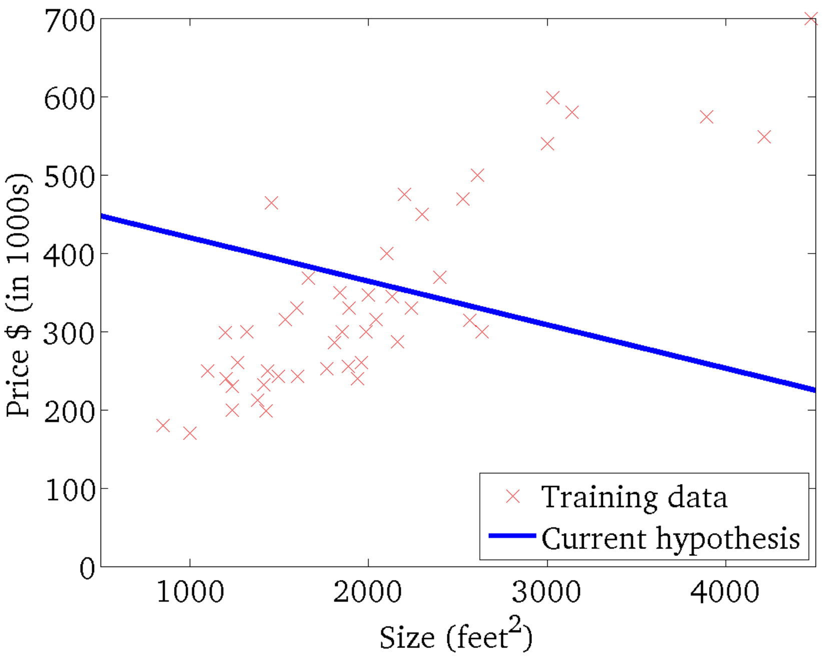
$$\theta_1 = 0$$

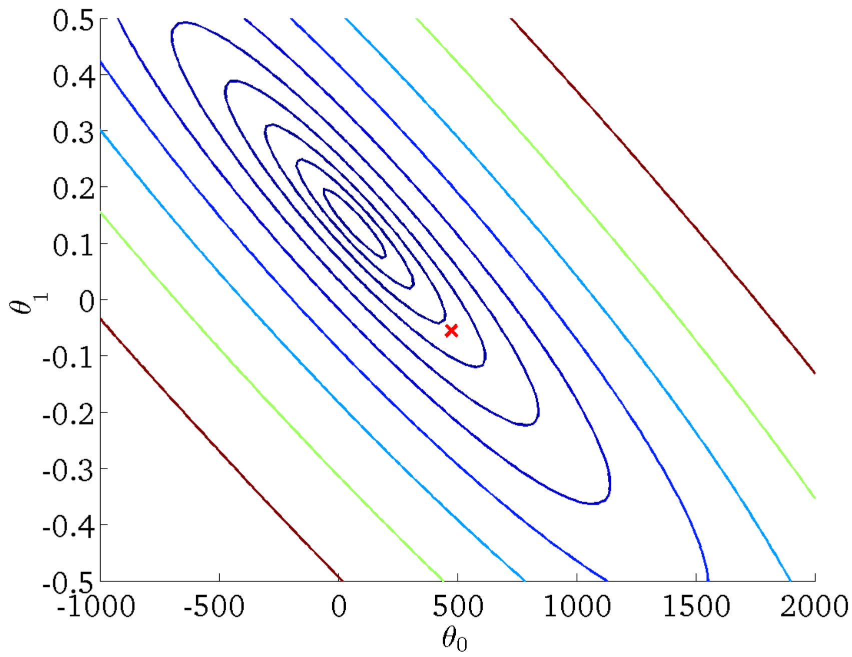
$\hat{f}(x)$

(Para valores fixos de θ_0, θ_1 , é uma função de *x)*

$J(\theta_0, \theta_1)$

(Função dos parâmetros θ_0, θ_1)



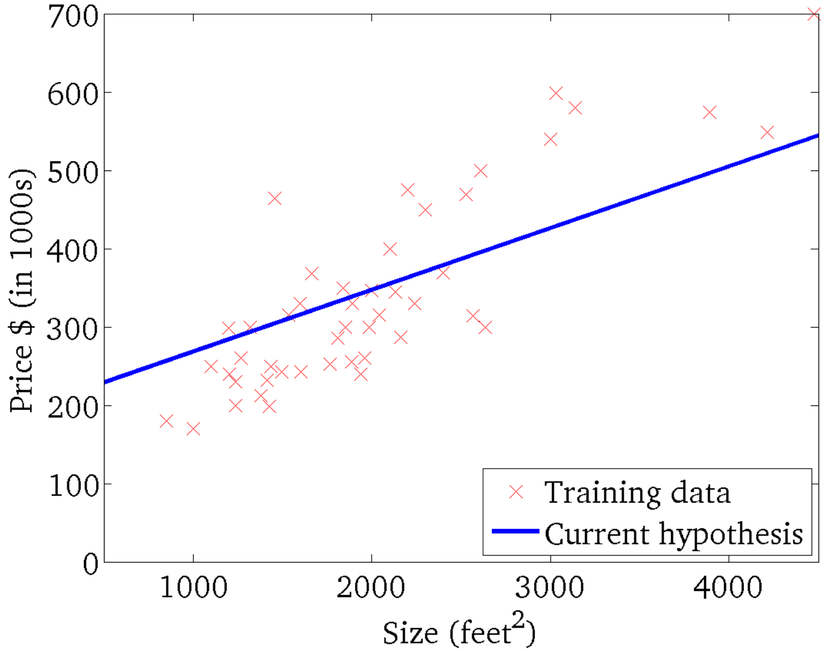


$$\theta_0 = 500$$

$$\theta_1 = -0.025$$

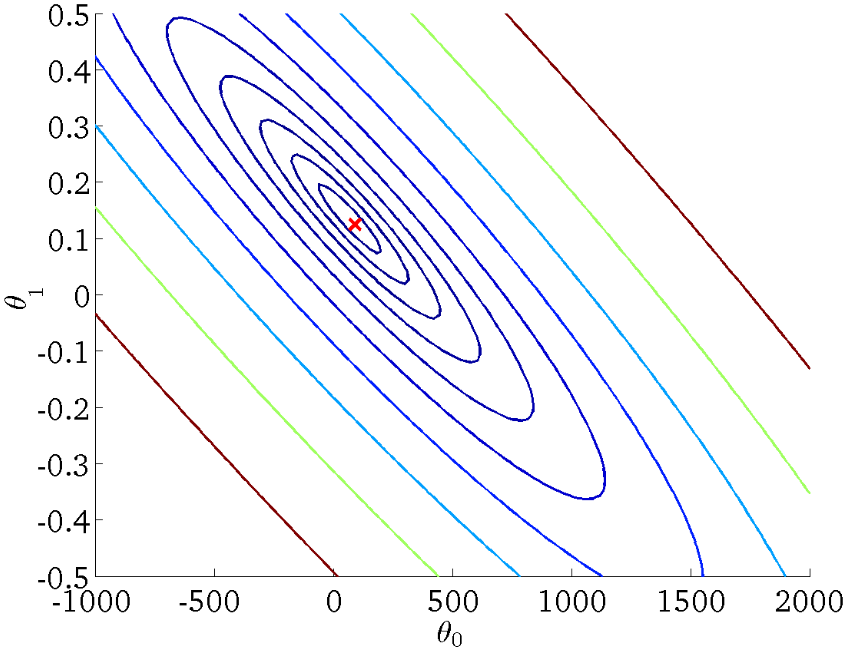
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de x)



$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)

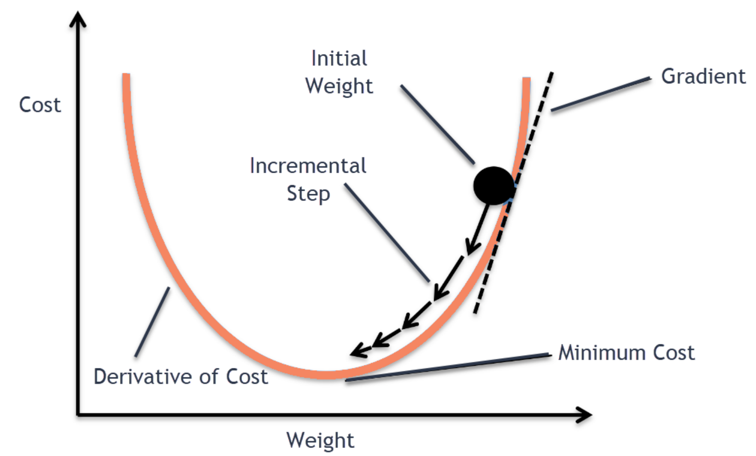


$$\theta_0 = 230$$

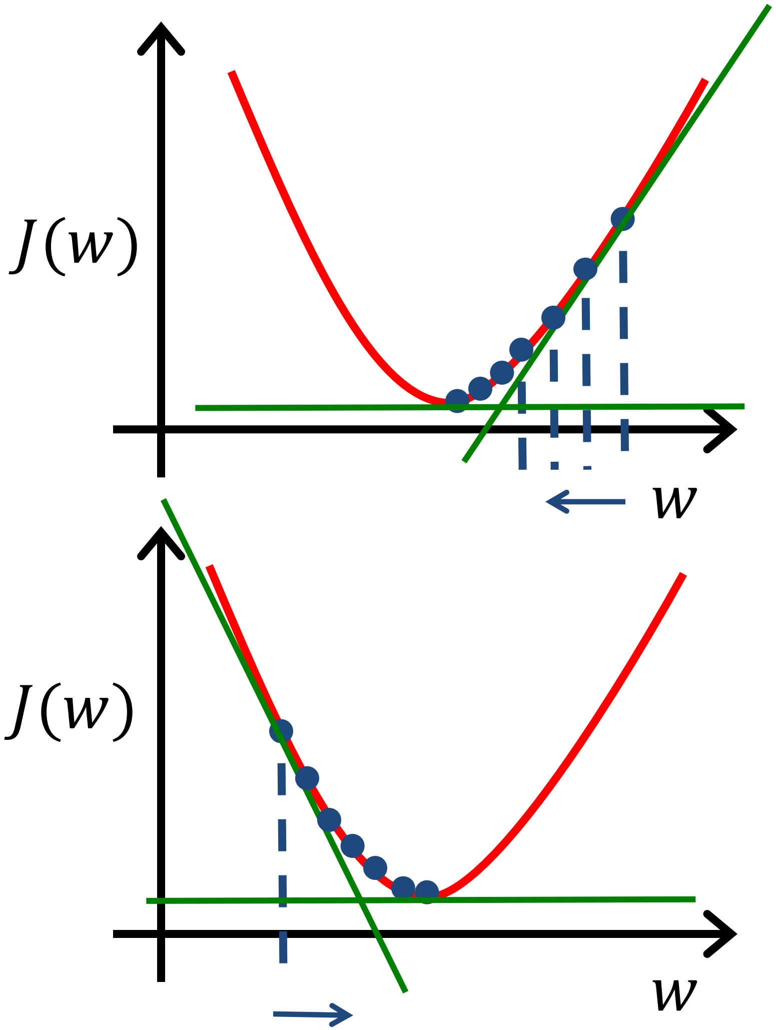
$$\theta_1 = 0.13$$

Descida de Gradiente *(Batch Gradient Descent)*

- $w = \theta$
- $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - a\nabla_{\boldsymbol{w_t}} loss$
- Utilizamos a derivada da função de custo
- $\boldsymbol{w}$ inicializado com valores aleatórios
- a : *Learning Rate* (taxa de aprendizado) (0,1]



Entendendo o papel da derivada



Por simplicidade, assuma a existência de uma função de custo com apenas um parâmetro ,w, $J(w)$

$$w := w - \alpha \boxed{\frac{d}{dw} J(w)}$$

$$w := w - \alpha \text{ (num positivo)}$$

$$w := w - \alpha \text{(num negativo)}$$

Tamanho do passo α (taxa de aprendizado)

Adequada (pequena)

Ruim (grande)

Descida de Gradiente

- Vamos considerar os dados como a matriz X e w como a matriz de pesos.
- Para poder ter todos os pesos no mesmo vetor, vamos fazer um truque de notação (*notation trick*):

$$x = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \qquad x = \begin{bmatrix} 1 & x_{11} & x_{12} \\ 1 & x_{21} & x_{22} \end{bmatrix}$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} \qquad \text{Preds} = \begin{bmatrix} w_0 1 + w_1 x_{11} + w_2 x_{12} \\ w_0 1 + w_1 x_{21} + w_2 x_{22} \\ \end{bmatrix}$$

Descida de Gradiente

- Vamos considerar os dados como a matriz X e w como a matriz de pesos.
- Nosso objetivo vai ser derivar a soma dos quadrados dos resíduos e ajustar os pesos conforme a derivada

$$J(w) = \sum_{i=1}^{N} (\hat{f}\,(x) - f\,(x))^2$$

Descida de Gradiente

1. $J(w) = \sum_{i=1}^{N} \left(\hat{f}\,(x) - f(x)\right)^2$
2. $J(w) = \left(\hat{f}\,(x) - f(x)\right)^T \left(\hat{f}\,(x) - f(x)\right)$
3. $J(w) = \hat{f}\,(x)^T \hat{f}\,(x) - 2\hat{f}\,(x)^T f(x) + f\,(x)^T f(x)$
4. $J(w) = (Xw)^T (Xw) - 2(Xw)^T f(x) + f\,(x)^T f(x)$

Descida de Gradiente

1. $J(w) = \sum_{i=1}^{N}\left(\hat{f}\,(x) - f(x)\right)^2$
2. $J(w) = \left(\hat{f}\,(x) - f(x)\right)^T\left(\hat{f}\,(x) - f(x)\right)$
3. $J(w) = \hat{f}\,(x)^T\hat{f}\,(x) - 2\hat{f}\,(x)^T f(x) + f\,(x)^T f(x)$
4. $J(w) = (Xw)^T(Xw) - 2(Xw)^T f(x) + f\,(x)^T f(x)$

$$\frac{\partial J(w)}{\partial w} = \frac{\partial((Xw)^T(Xw) - 2(Xw)^T f(x) + f\,(x)^T f(x))}{\partial w}$$

Descida de Gradiente

1. $$\frac{\partial J(w)}{\partial w} = \frac{\partial((Xw)^T(Xw)-2(Xw)^T f(x)+f\ (x)^T f(x))}{\partial w}$$

2. $$\frac{\partial J(w)}{\partial w} = \frac{\partial((Xw)^T(Xw)-2(Xw)^T f(x)}{\partial w}$$

3. $$\frac{\partial J(w)}{\partial w} = \frac{\partial((Xw)^T(Xw))}{\partial w} - \frac{\partial(2(Xw)^T f(x))}{\partial w}$$

4. $$\frac{\partial J(w)}{\partial w} = 2X^T Xw\ - 2X^T f(x)$$

5. $$\frac{\partial J(w)}{\partial w} = 2X^T(Xw\ - f(x))$$

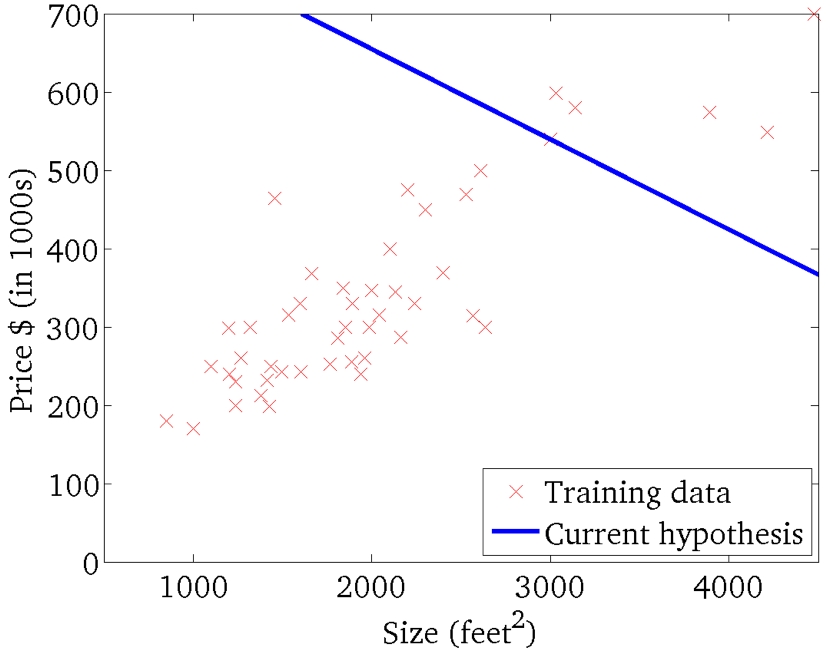
Regra para atualizar os pesos

- Gradiente:
- $\nabla_{\boldsymbol{w_t}} loss = X^T(\hat{y} - y)$

- Regra de Atualização:
- $w_{t+1} = w_t - a(\nabla_{\boldsymbol{w_t}} loss)$

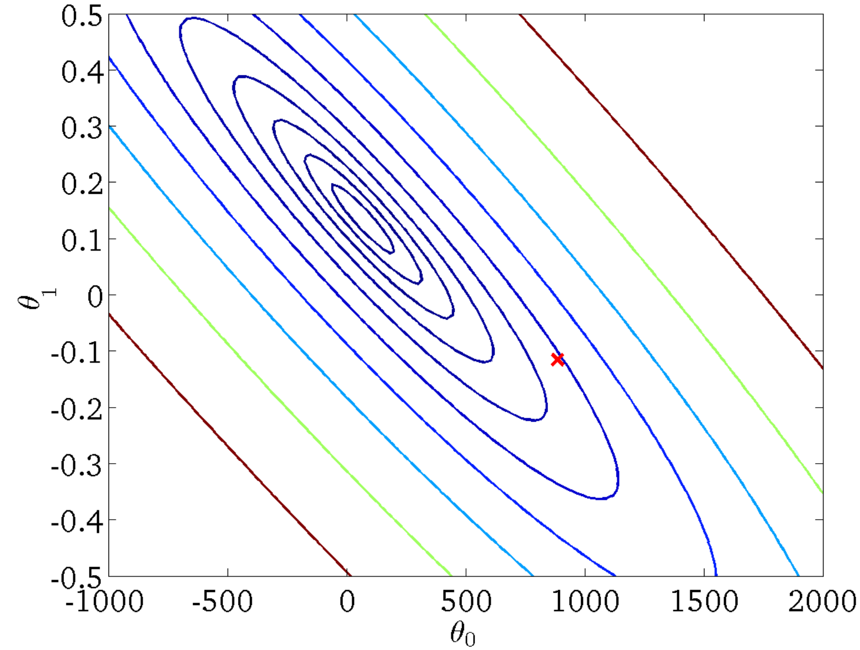
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de x)



$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)



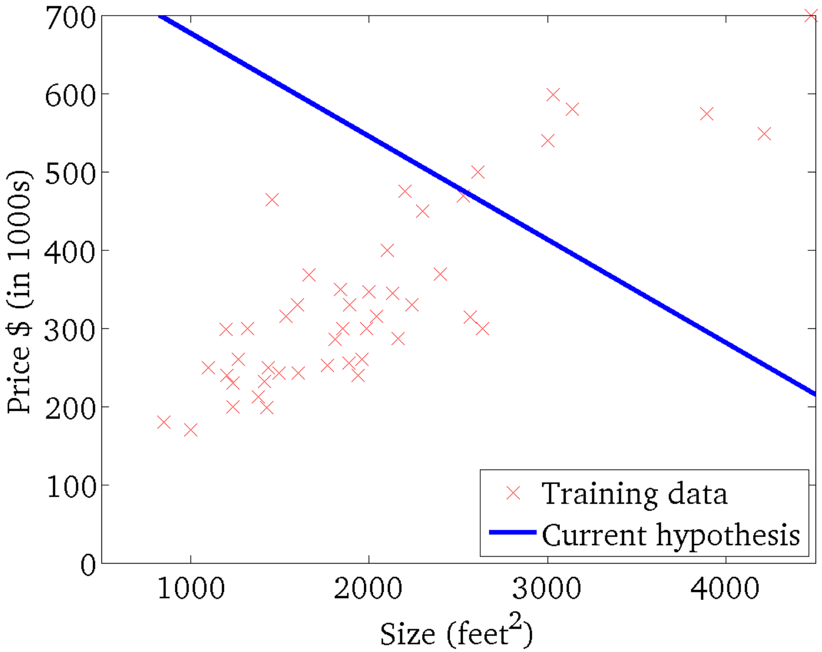
Inicialização:

$$\theta_0 = 900$$

$$\theta_1 = -0.1$$

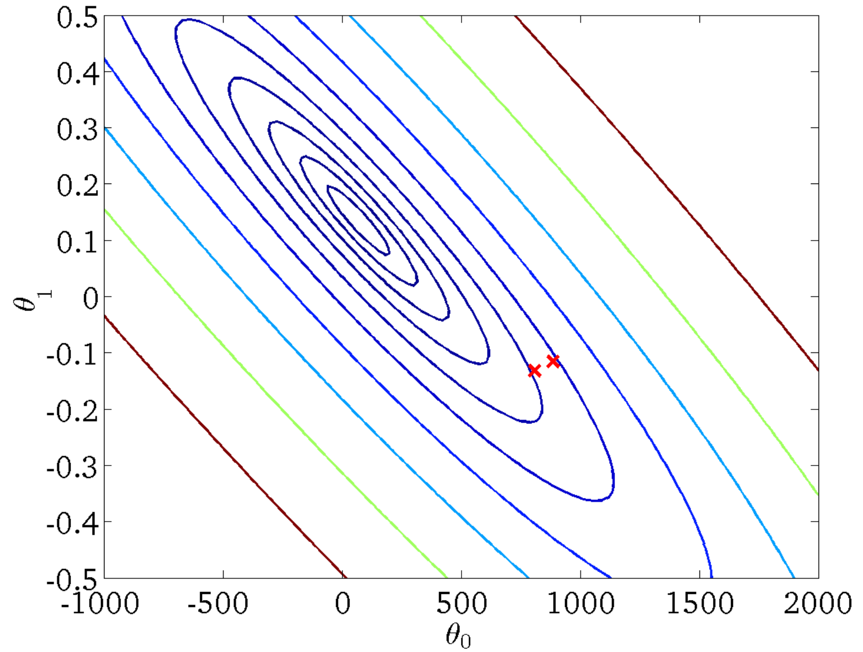
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de x)



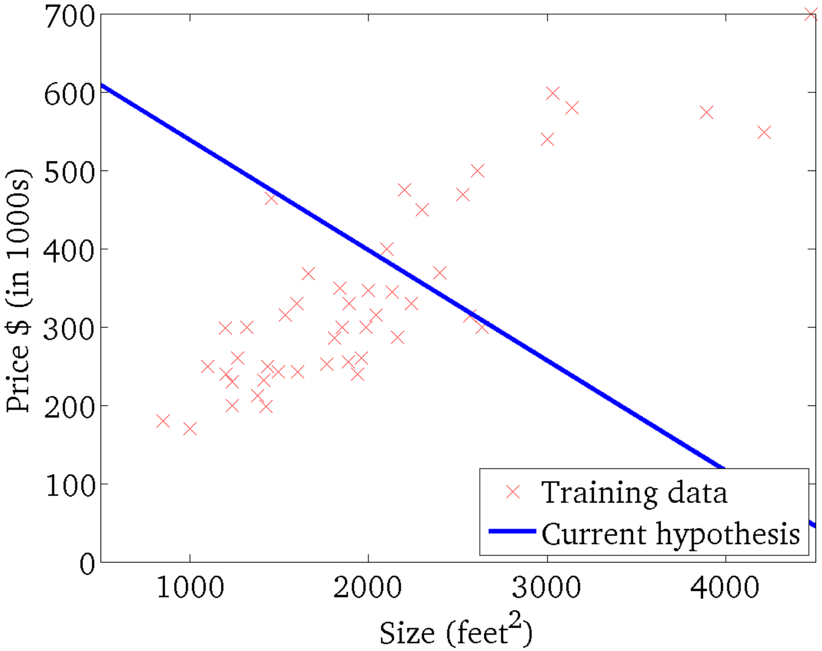
$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)



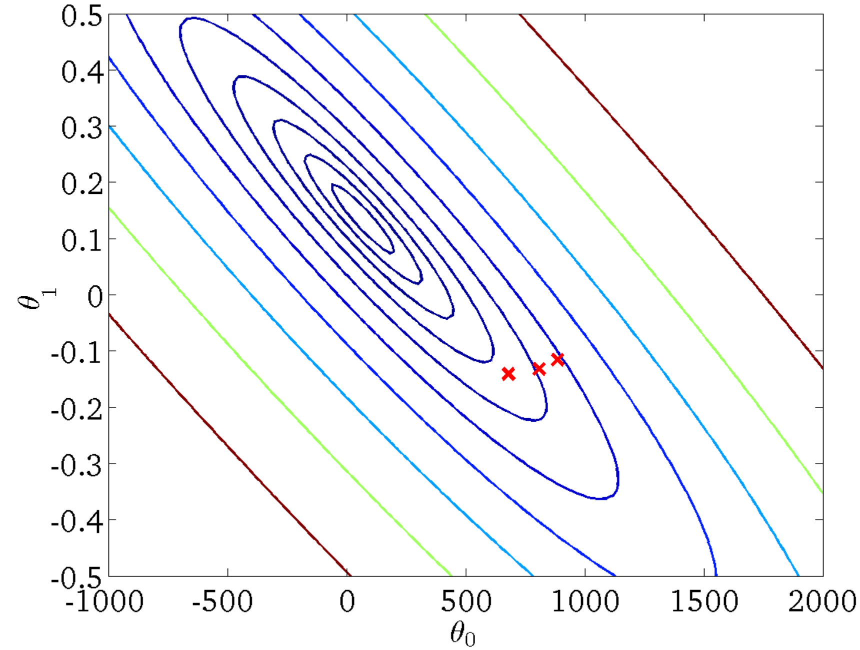
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de *x*)



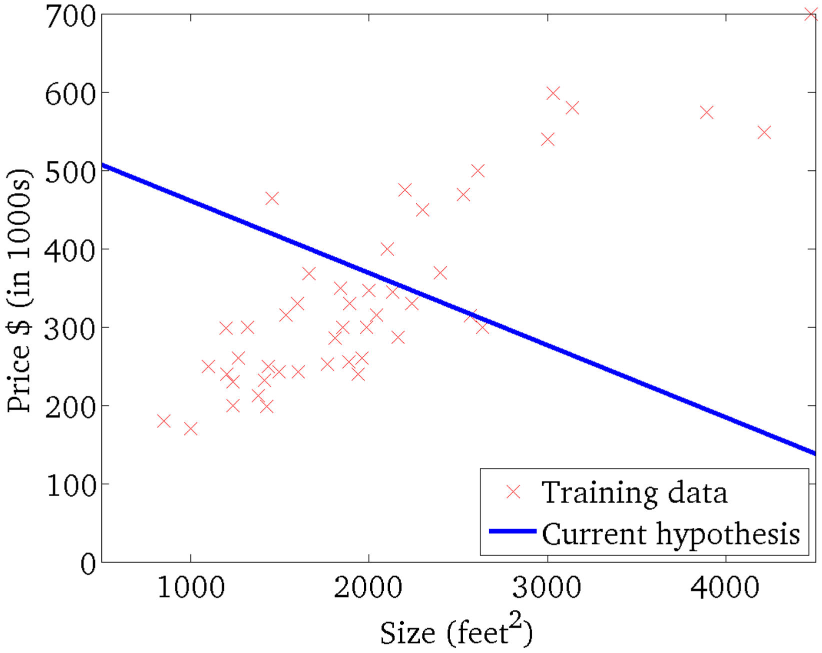
$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)



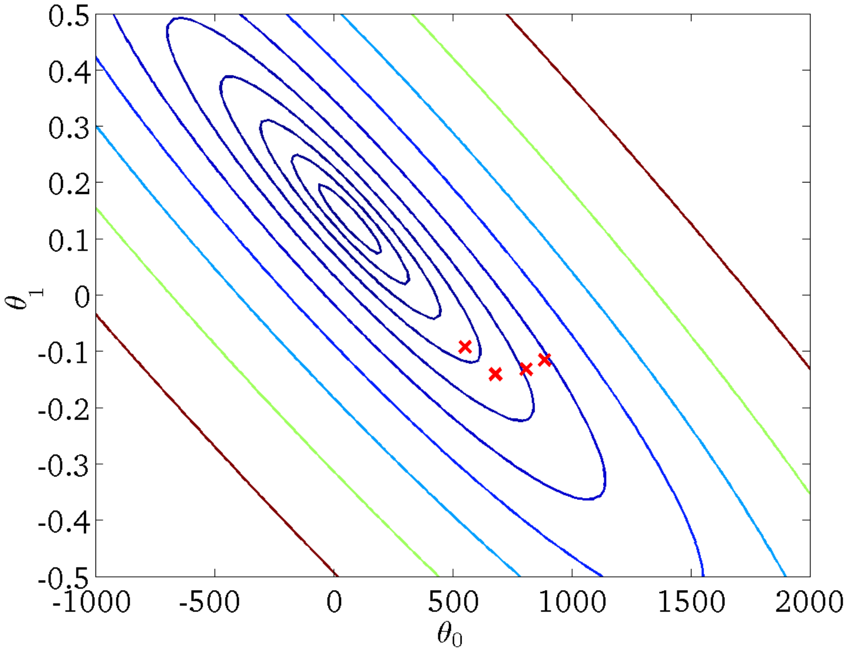
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de *x)*



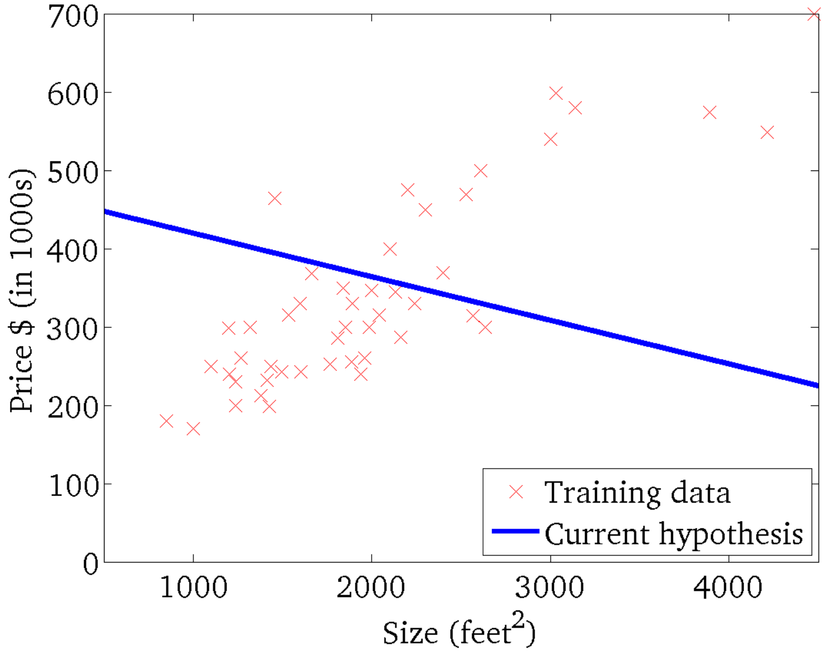
$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)



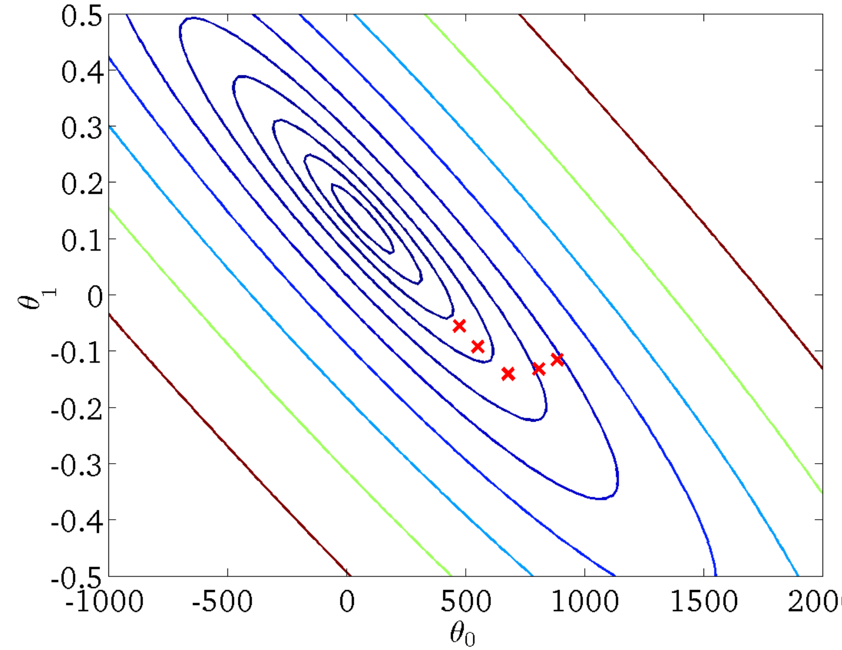
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de *x)*



$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)

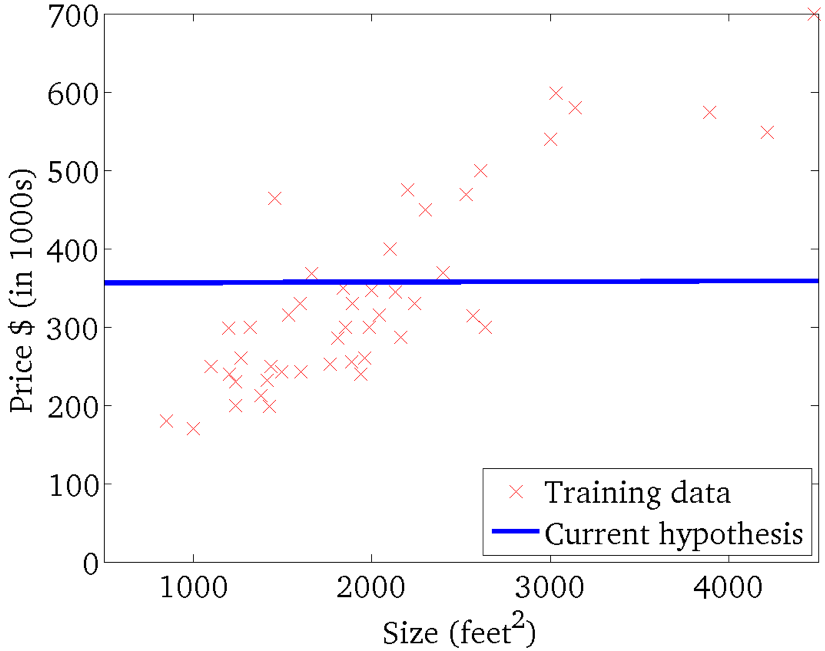


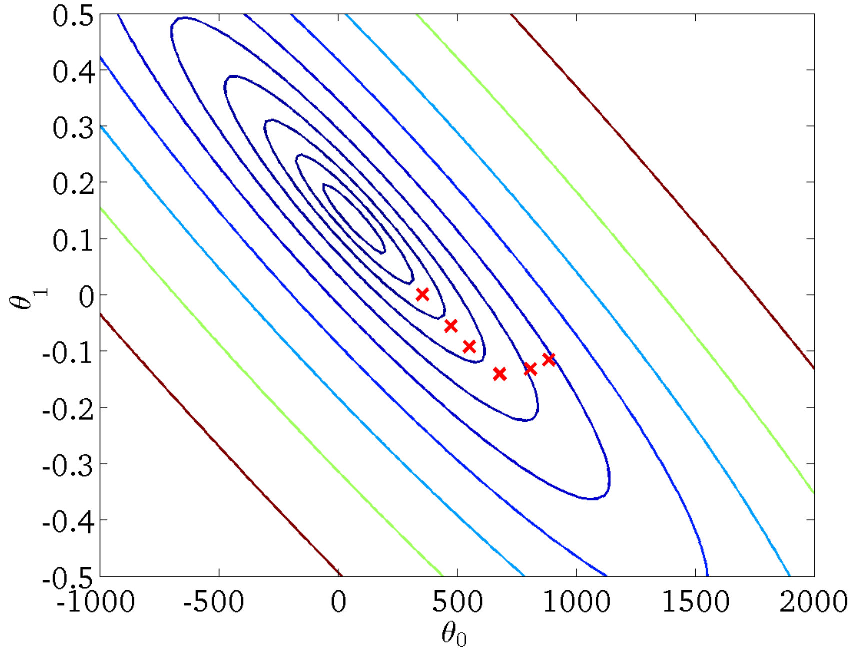
$\hat{f}(x)$

(Para valores fixos de θ_0, θ_1 , é uma função de *x)*

$J(\theta_0, \theta_1)$

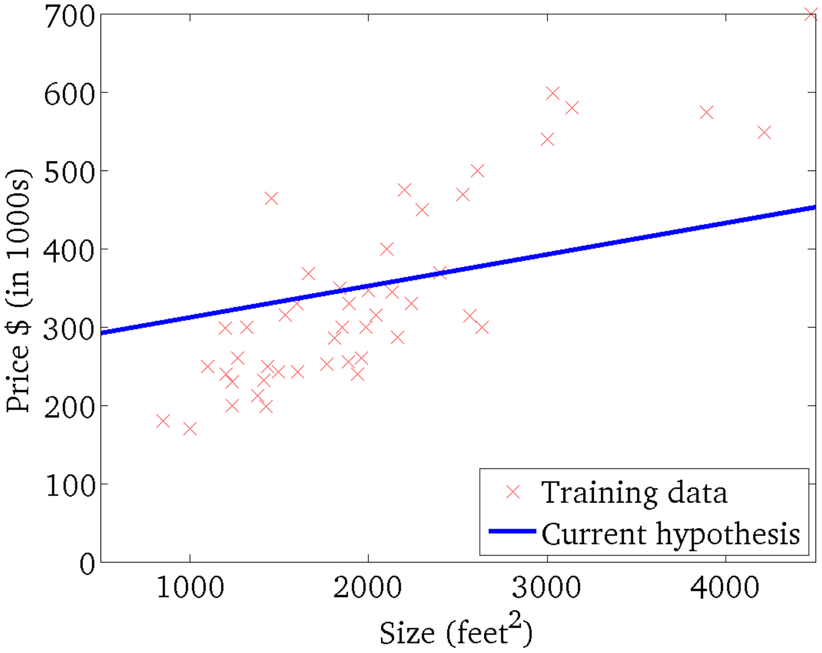
(Função dos parâmetros θ_0, θ_1)





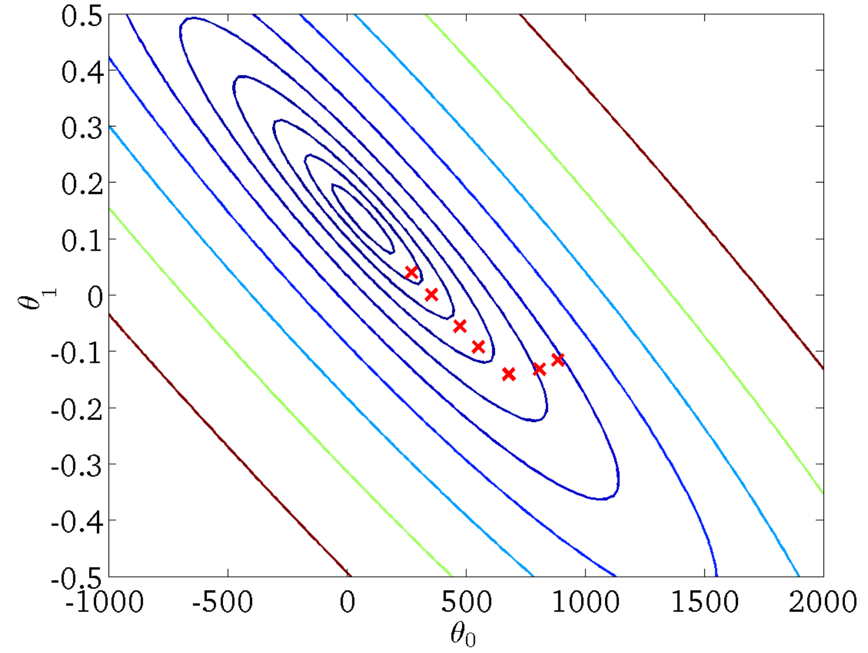
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de *x)*



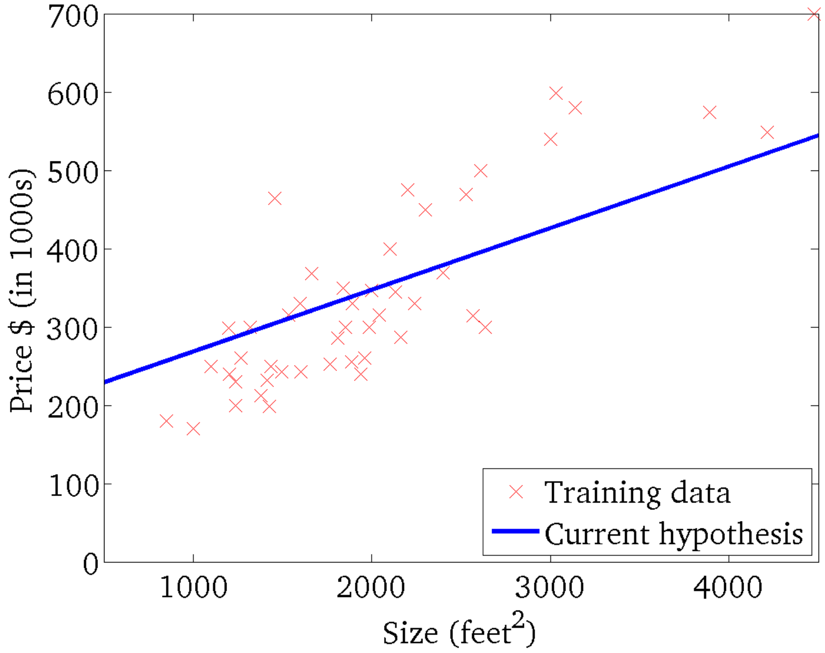
$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)



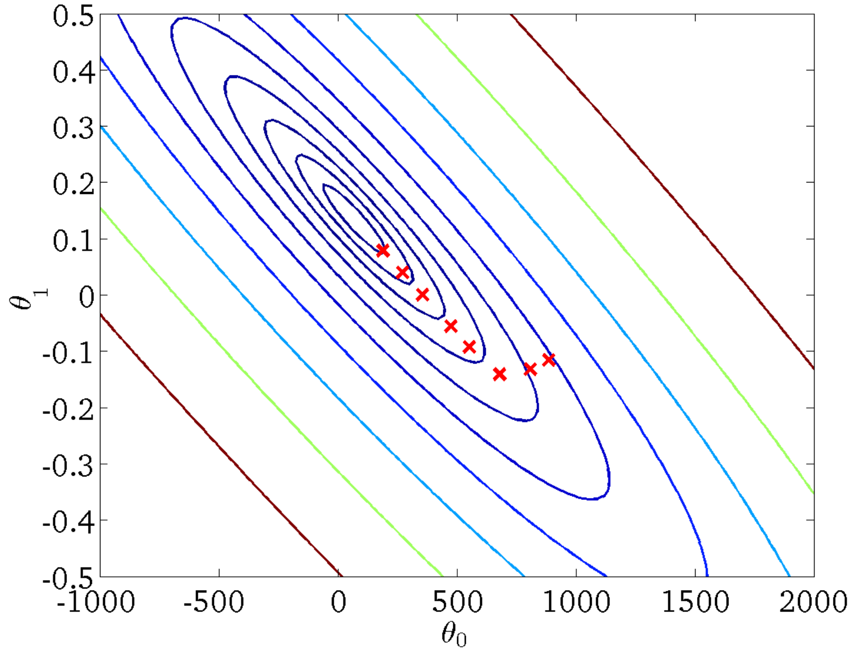
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de x)



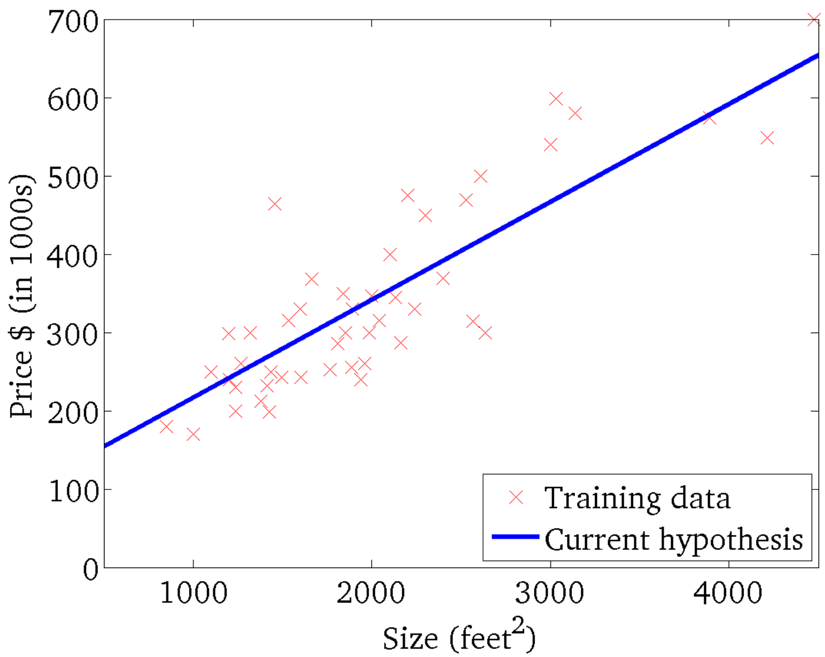
$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)



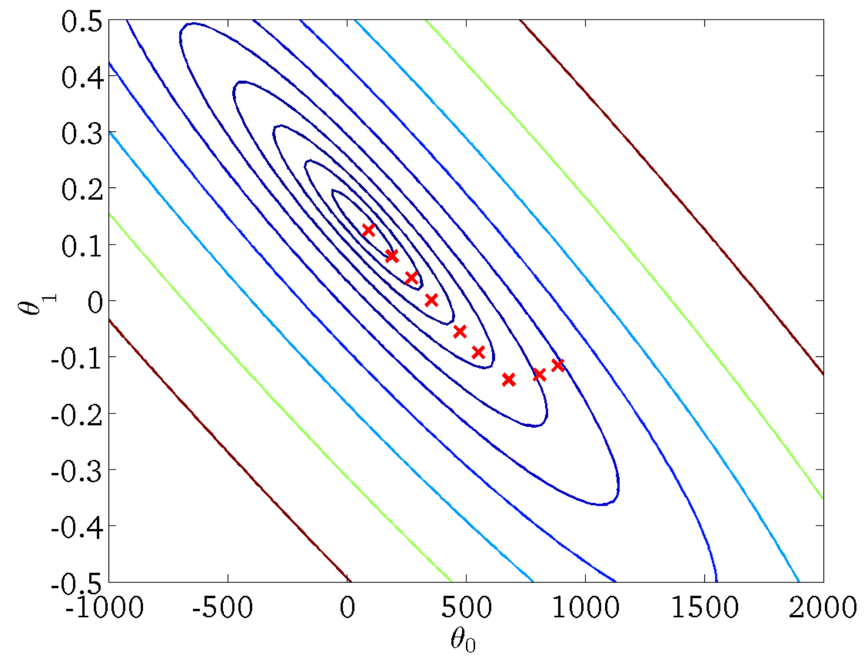
$$\hat{f}(x)$$

(Para valores fixos de θ_0, θ_1 , é uma função de x)



$$J(\theta_0, \theta_1)$$

(Função dos parâmetros θ_0, θ_1)

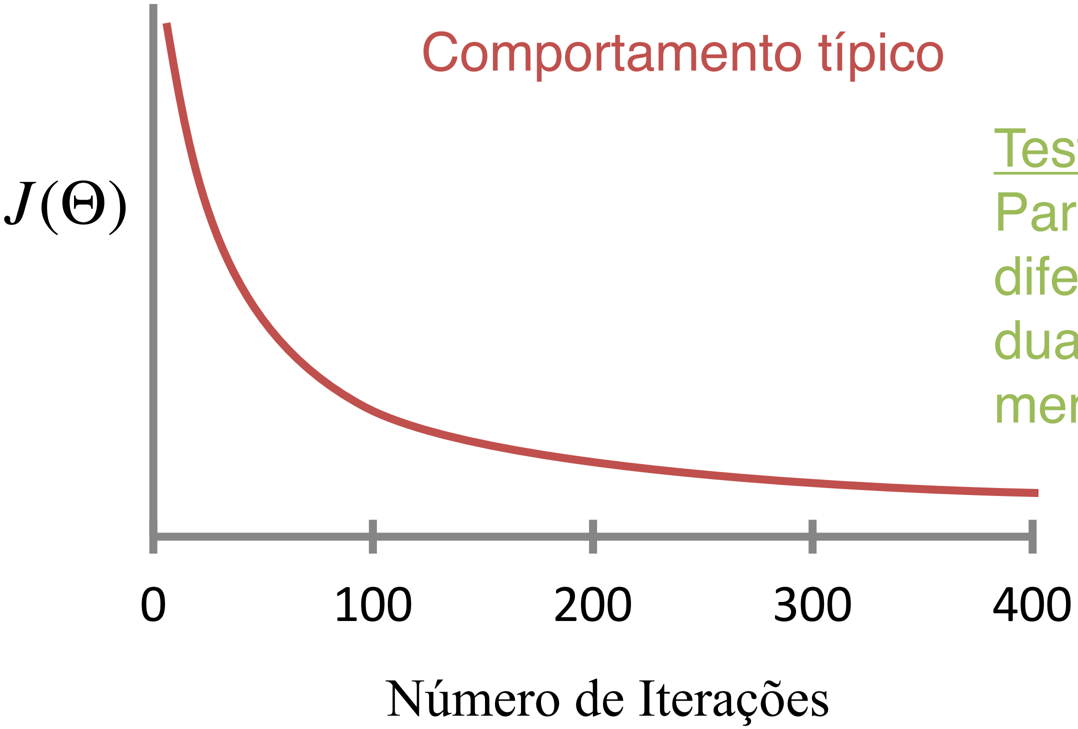


Convergiu!!

Gradiente Descendente para Regressão Multi-variada

- Dicas:
  - Normalizar os atributos para acelerar convergência
    - Utilizar estratégias já vistas em aula como normalização ou transformação para intervalos [0,1] ou [-1,1]
  - Garantir que o gradiente descendente esteja funcionando
    - Plotar função de custo x iteração do gradiente descendente

Debugando o Gradiente Descendente

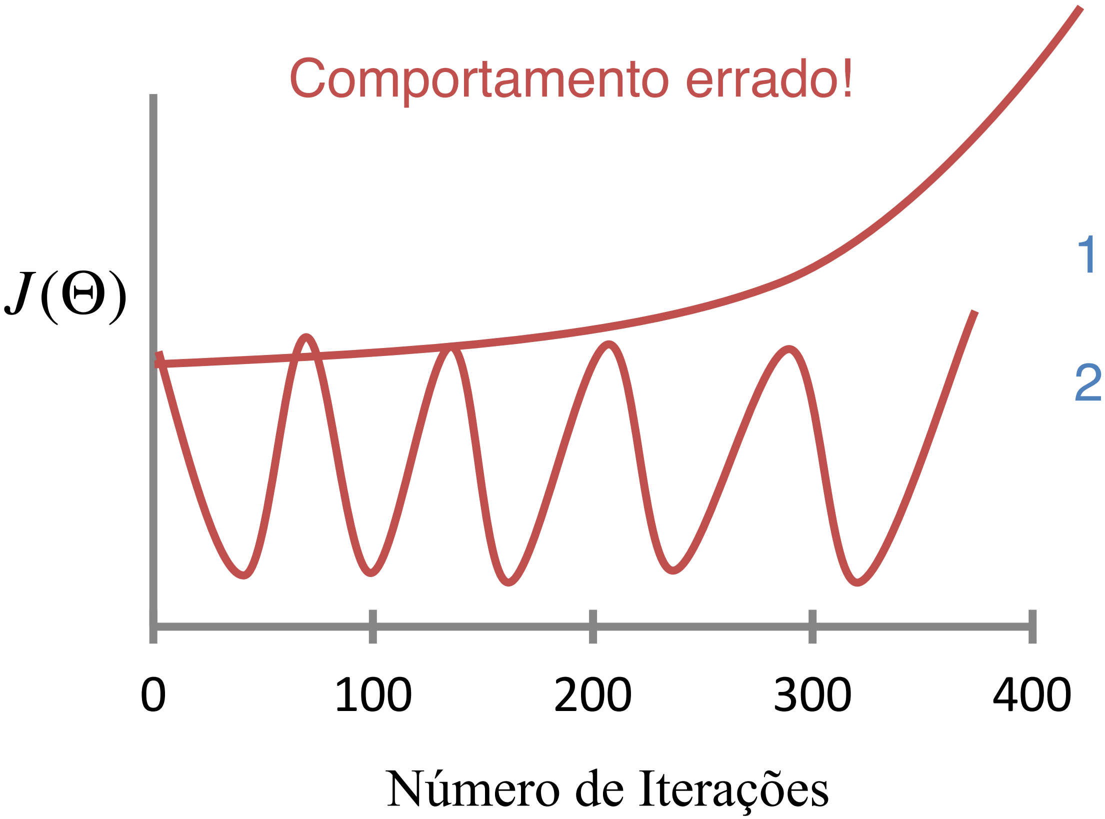


<u>Teste automático de convergência:</u>
Parar de executar o algoritmo se a diferença da função de custo para duas iterações consecutivas for menor que limiar ε

Ex: $\varepsilon = 10^{-3}$

Gradiente Descendente necessariamente deve reduzir a função de custo a cada iteração

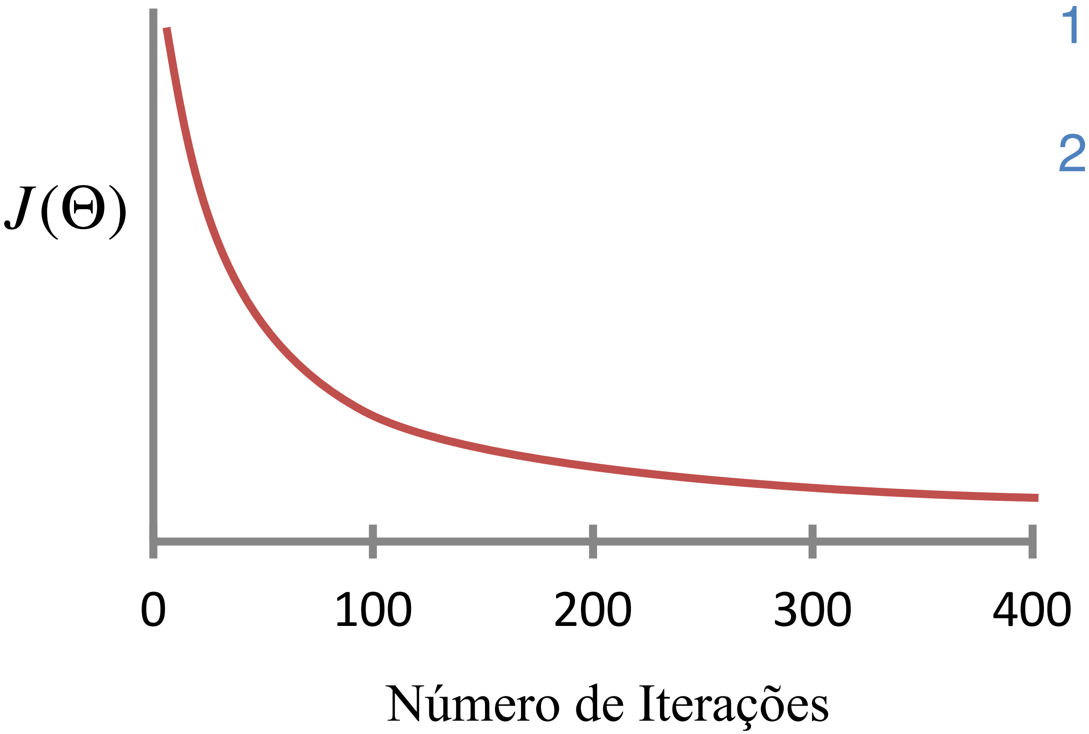
Debugando o Gradiente Descendente



1) Implementação pode estar errada!
2) Se estiver correta, então provavelmente significa que α está muito alta!

Para valores suficientemente baixos de α, gradiente descendente decresce a cada iteração!
Mas atenção! Valores muito baixos podem levar à convergência MUITO lenta!

Debugando o Gradiente Descendente



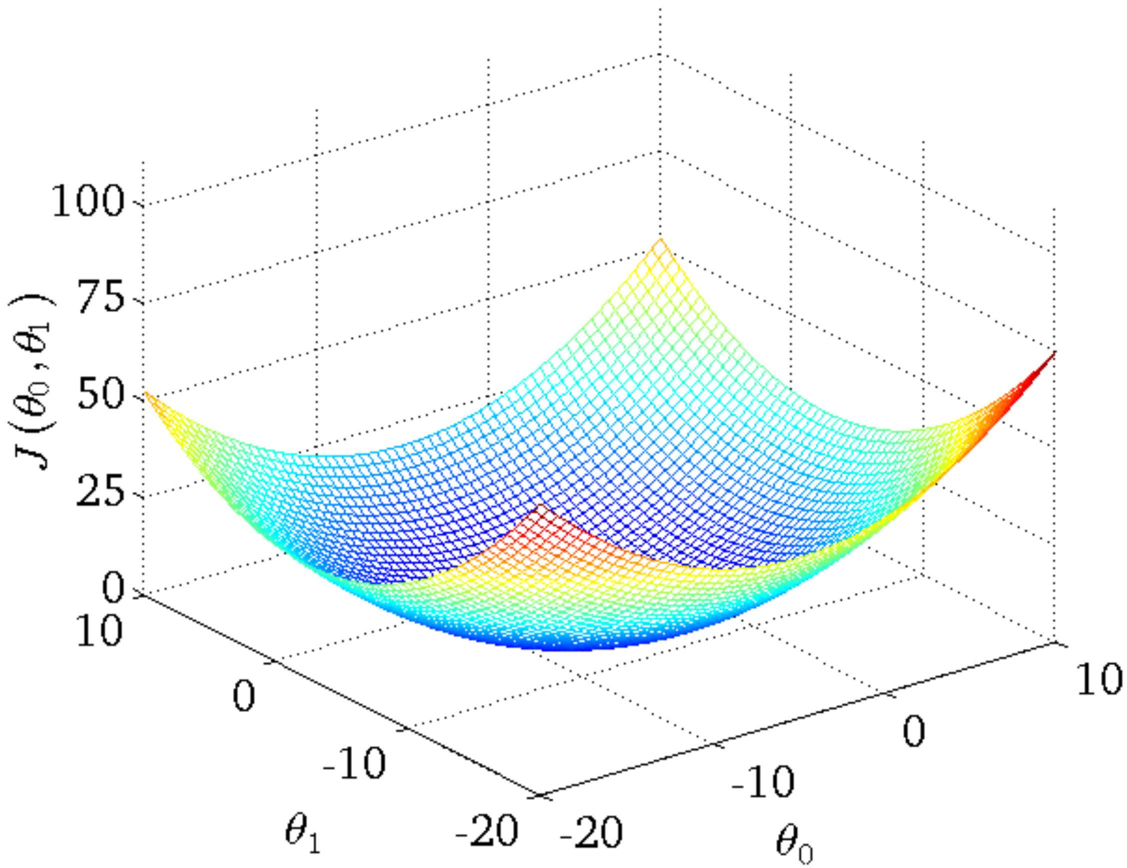
Heurística para escolha de α:

1. Começar com valores pequenos (ex: 0.001)
2. Incrementar o valor por algum fator (ex: 3, 10, etc.) para agilizar convergência, mas sempre conferindo se os valores estão decrescendo após cada iteração

Solução Analítica

Função de Custo do Erro Quadrático é Convexa!

Logo, função tem único mínimo
(Formato de tigela – “bowl-shaped”)

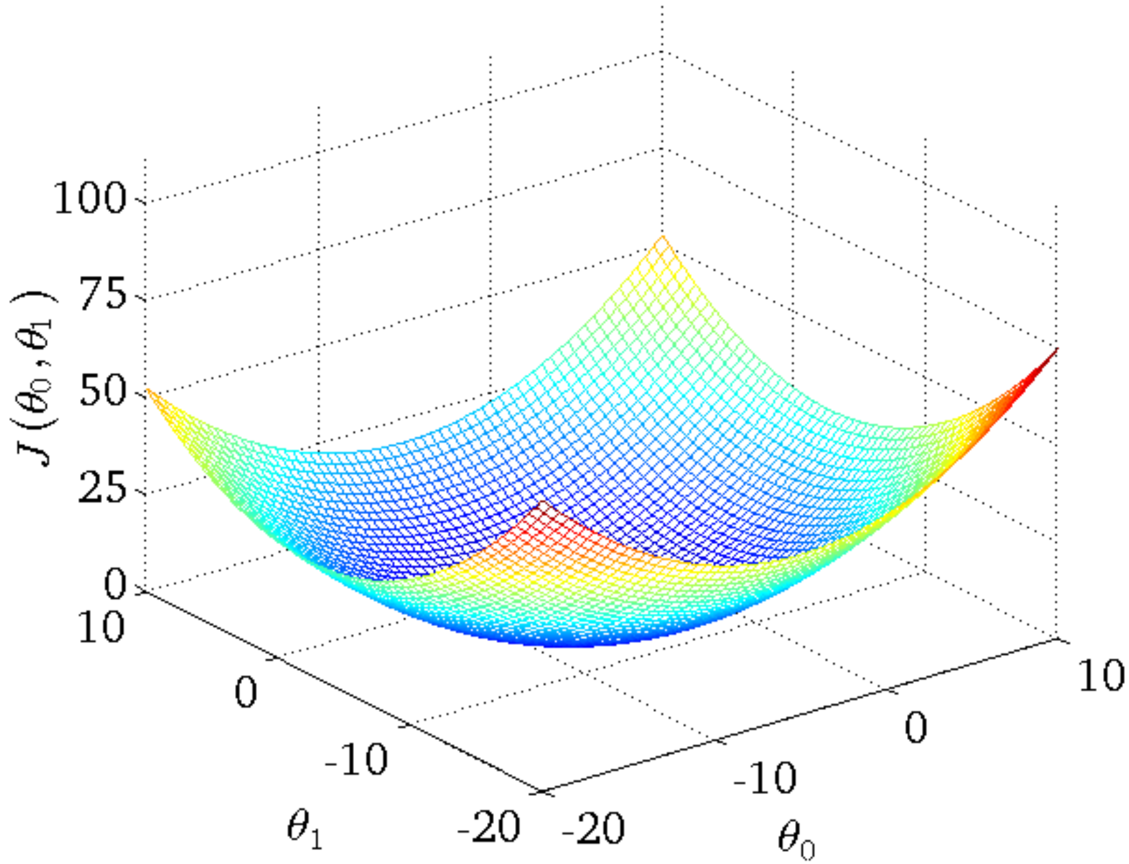


Solução Analítica

- Até então, estamos utilizando o algoritmo do gradiente descendente para minimizar a função de custo *J*, chegando nos valores de parâmetros do hiperplano de regressão que minimizam *J*
- Porém, existe uma *solução analítica* para a descoberta dos parâmetros!

Solução Analítica

- Essa solução existe apenas pela função ser convexa!



Regressão Linear Multivariada

- Como chegamos lá:
  - Igualando a derivada a zero e isolando o vetor de pesos

Regressão Linear Multivariada

- Como chegamos lá:
  - Igualando a derivada a zero e isolando o vetor de pesos

1. $X^T(Xw - y)$
2. $X^T(Xw - y) = 0$
3. $X^T Xw - X^T y \ = 0$
4. $X^T Xw = X^T y$
5. $(X^T X)^{-1} X^T Xw = (X^T X)^{-1} X^T y$
6. $w = (X^T X)^{-1} X^T y$

$$(X^T X)^{-1} X^T y$$

Gradiente Descendente
X
Equação Normal

Gradiente Descendente

- Precisa definir α
- Precisa de muitas iterações
- Funciona bem mesmo para altas dimensionalidades

Equação Normal

- Não precisa definir α
- Não precisa iterar
- Lento se dimensionalidade do problema é muito alta
  - Custo de inverter matriz: $O(m^3)$

Equação Normal

- E se $\left(X^T X\right)$ for uma matriz não-inversível?
  - Utilizar pseudo-inversa (aproximações numéricas)
  - Fazer seleção de atributos para eliminar atributos correlacionados (linearmente dependentes)
  - Pode ser pelo fato do problema ter mais atributos do que instâncias
    - Remover atributos
    - Usar regularização (veremos mais adiante)

Dica Final

- O gradiente descendente que vimos é o tradicional (*batch gradient descent*), que leva em conta todos os dados de treinamento para atualizar os pesos
- Para problemas com muitos dados, fazer atualização com todos os dados é inviável
  - Solução: **gradiente descendente estocástico**
    - Utiliza **mini-batches** para fazer as atualizações
    - Aproxima o gradiente global
    - Mais robusto para funções não-convexas (aleatoriedade evita cair em mínimos)

Aula de Hoje

- Regressão Linear
  - Gradiente Descendente
  - Solução Analítica
- Regressão Logística

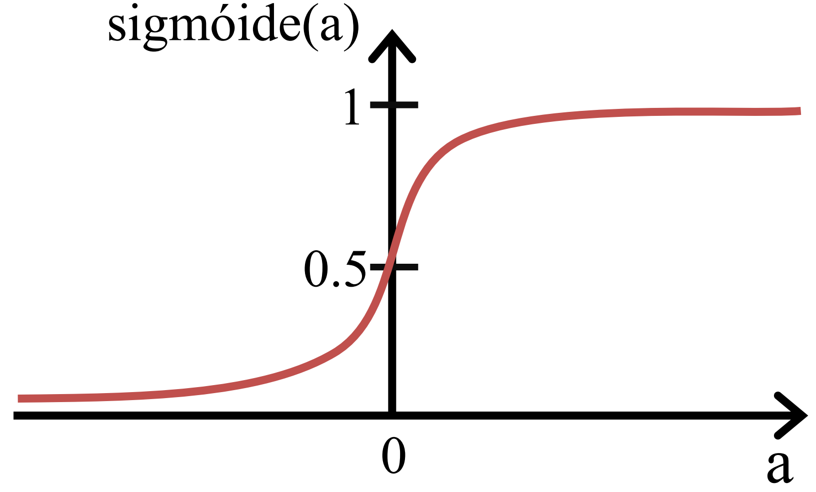
Regressão Logística

- Apesar do nome, é um algoritmo de classificação!
- Utilizado para discriminar entre duas classes
  - $f(\mathbf{x}) = \{0,1\}$
  - Geralmente, 0 indica ausência (classe negativa) e 1 indica presença (classe positiva) do que se deseja classificar
    - Ex: diagnóstico de HIV
      - 0 = sem HIV (classe negativa)
      - 1 = com HIV (classe positva)

Regressão Logística

- Gera função $0 \leq \hat{f}(\mathbf{x}) \leq 1$
- Em regressão linear, $\hat{f}\,(X) = Xw$
- Qual o modelo para regressão logística?
  - $\hat{f}\,(X) = sigmoide(Xw)$

$$sigmoide(a) = \frac{1}{1+e^{-a}}$$



$$\boxed{\hat{f}\,(X) = \frac{1}{1+e^{-Xw}}}$$

Regressão Logística

1. $\log\left(\frac{p(Y = c|X)}{1 - p(Y = c|X)}\right) = Xw$
2. $\frac{p(Y = c|X)}{1 - p(Y = c|X)} = e^{Xw}$
3. $p(Y = c|X) = (1 - p(Y = c|X))e^{Xw}$
4. $p(Y = c|X) + p(Y = c|X)e^{Xw} = e^{Xw}$
5. $p(Y = c|X) = \frac{e^{Xw}}{1+e^{Xw}}$
6. $p(Y = c|X) = \frac{1}{1+e^{-Xw}}$

Probability	Odds	Log Odds
0.100	0.111	-2.197
0.200	0.250	-1.386
0.300	0.428	-0.847
0.400	0.667	-0.405
0.500	1.000	0.000
0.600	1.500	0.405
0.700	2.333	0.847
0.800	4.000	1.386
0.900	9.000	2.197

Regressão Logística

- Interpretação probabilística
  - Probabilidade estimada da classe positiva
    - Ex: classificação de tumor: $f(\mathbf{x})$ = {benigno, maligno}
    - Para paciente com $\hat{f}(\mathbf{x}) = 0.7$
      - Paciente tem 70% de chance de ter tumor maligno

Lembre-se que:

$$P(f(X) = 0|X; w) = 1 - P(f(X) = 1|X; w)$$

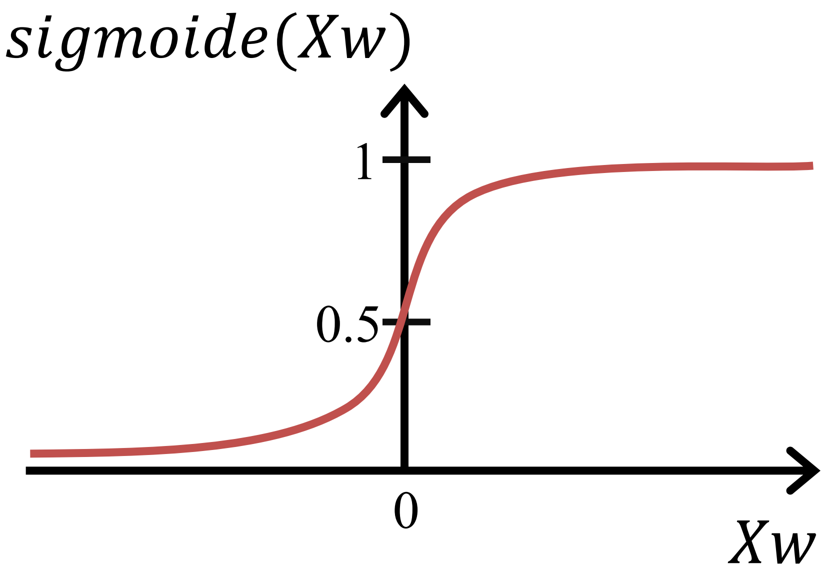
Regressão Logística

- Dada a interpretação probabilística da função logística, o que realmente estamos fazendo é:

$$se\ sigmoide(Xw) \geq 0.5$$
então x é da classe positiva
senão x é da classe negativa

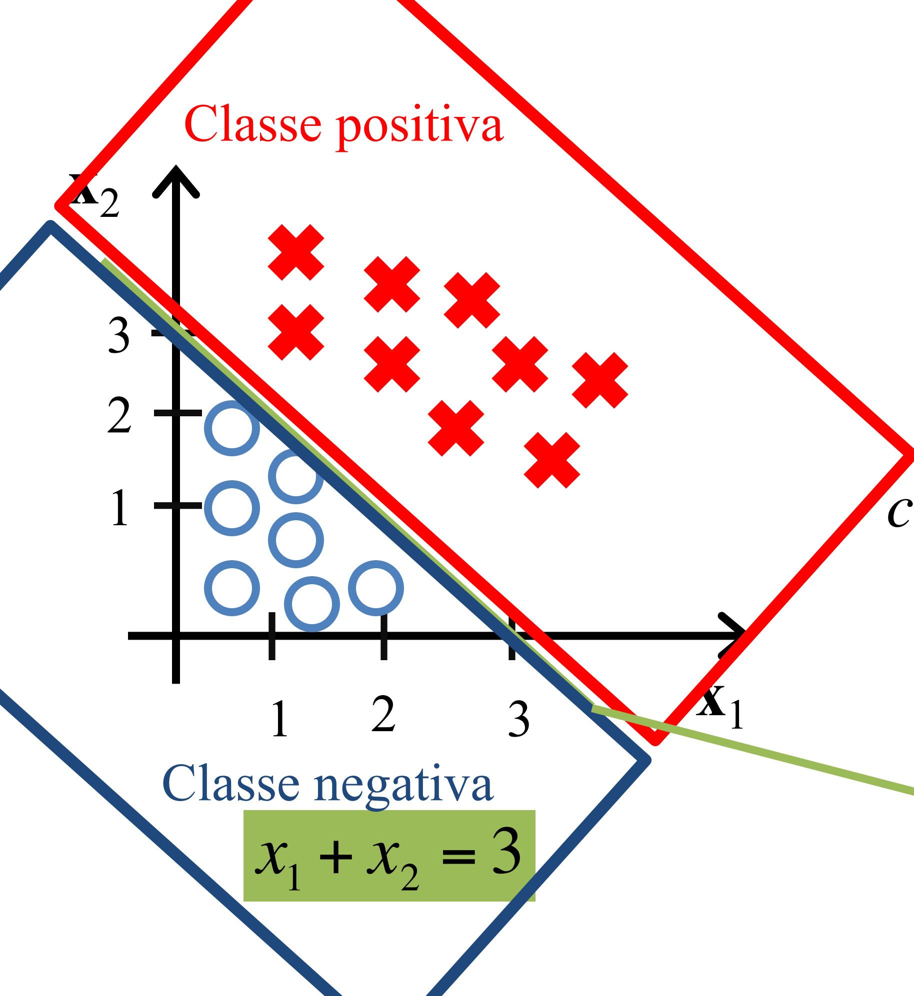
- O que isso nos diz a respeito do vetor de parâmetros w?

$$se\ Xw \geq 0$$
então x é da classe positiva
senão x é da classe negativa



Regressão Logística

- Ex: problema com dois atributos



$$\hat{f}\,(X) = sigmoide(Xw)$$

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

$$classe\ positiva: \ -3 + 1x_1 + 1x_2 \geq 0$$

$$x_1 + x_2 \geq 3$$

Instâncias sobre a reta resultarão em:

$$\hat{f}(\mathbf{x}) = 0.5$$

Regressão Logística

- O problema de otimização novamente se resume a minimizar uma função de custo
- Função de custo quadrática da regressão linear pode ser usada para regressão logística?

$$J(w) = \sum_{i=1}^{N} (\hat{f}\,(x) - f(x))^2$$

Regressão Logística

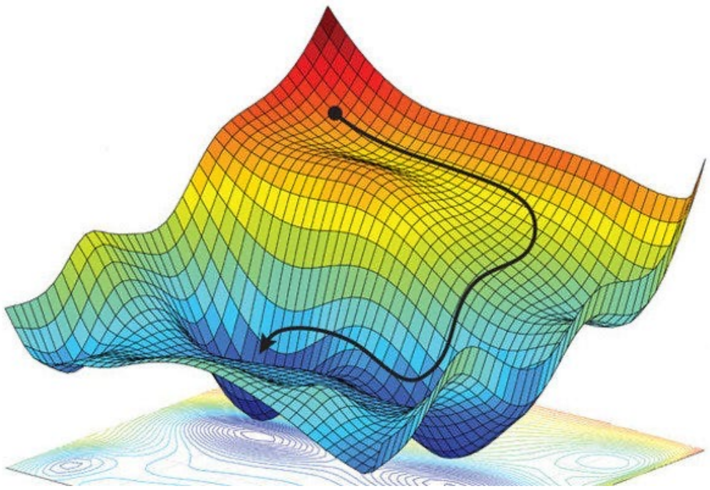
- O problema de otimização novamente se resume a minimizar uma função de custo
- Função de custo quadrática da regressão linear pode ser usada para regressão logística?

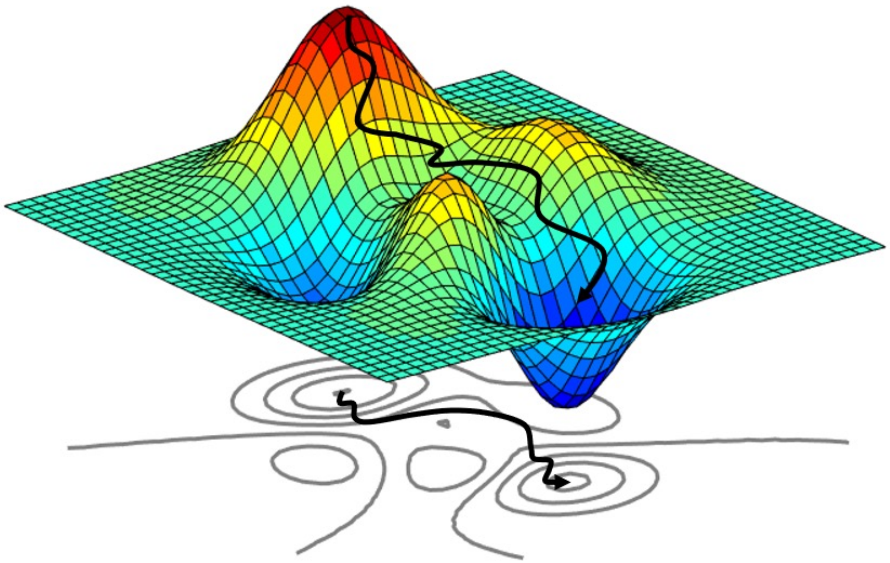
$$J(w) = \sum_{i=1}^{N} (\hat{f}\,(x) - f\,(x))^2$$

NÃO!! POIS *J*(w) VIRA UMA FUNÇÃO NÃO-CONVEXA!!
Logo, possui vários ótimos locais!

Regressão Logística

A cara da função de custo não-convexa



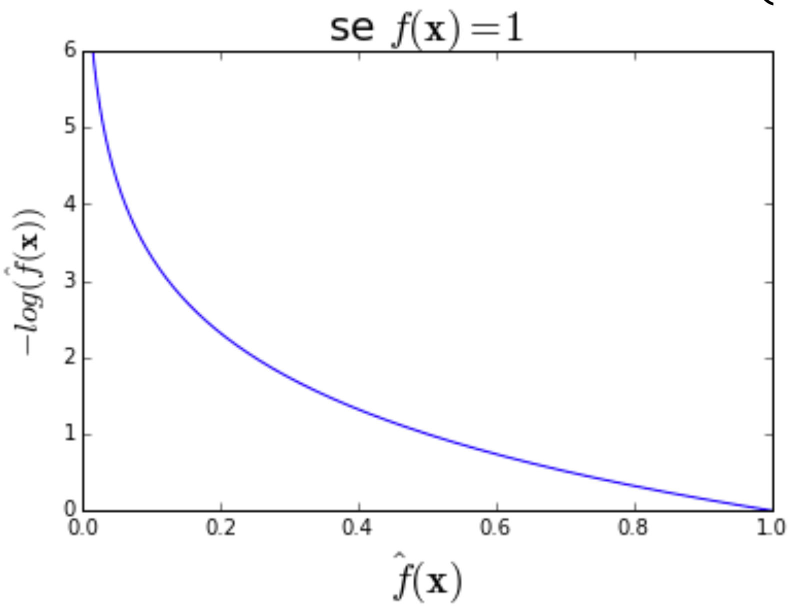


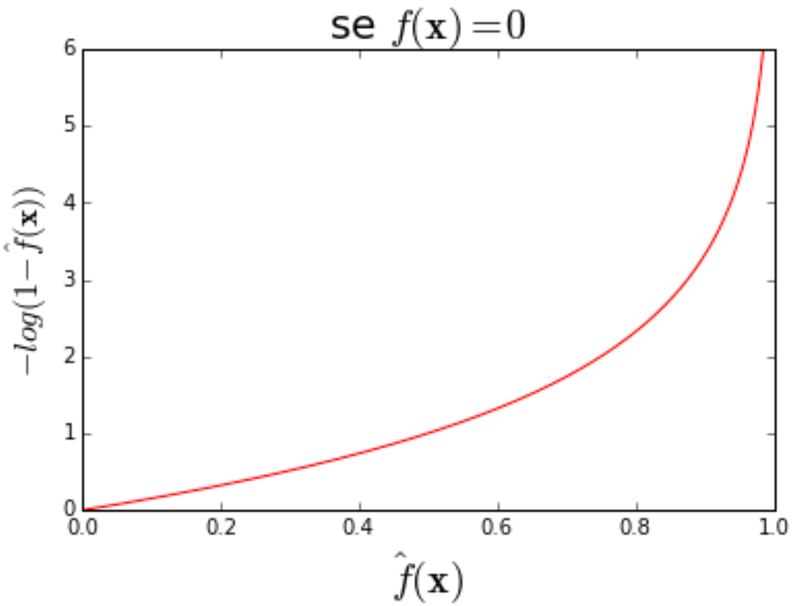
Regressão Logística

- Função de custo a ser minimizada:

$$J(w) = \sum_{i=1}^{N} custo\left(\hat{f}\,(x), f(x)\right)$$

$$\text{custo}(\hat{f}\,(x), f(x)) = \begin{cases} -\log\left(\hat{f}\,(x)\right) & se\ f(x) = 1 \\ -\log\left(1 - \hat{f}\,(x)\right) & se\ f(x) = 0 \end{cases}$$





Regressão Logística

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N} f\left(x^{(i)}\right)\log\left(\hat{f}\left(x^{(i)}\right)\right) + \left(1 - f\left(x^{(i)}\right)\right)\log(1 - \hat{f}\left(x^{(i)}\right))$$

- Sabendo que a nova função de custo é convexa, como minimizá-la?
  - Gradiente descendente!

$$w_{(t+1)} = w_{(t)} - a\frac{1}{N}\sum_{i=1}^{N}\left[{x^{(i)}}^{T}\left(\hat{y}^{(i)} - y^{(i)}\right)\right]$$

Regressão Logística

$$J(w) = -\frac{1}{N}\sum_{i=1}^{N} f\left(x^{(i)}\right)\log\left(\hat{f}\left(x^{(i)}\right)\right) + \left(1 - f\left(x^{(i)}\right)\right)\log(1 - \hat{f}\left(x^{(i)}\right))$$

- Sabendo que a nova função de custo é convexa, como minimizá-la?
  - Gradiente descendente!

$$w_{(t+1)} = w_{(t)} - a\frac{1}{N}\sum_{i=1}^{N}\left[{x^{(i)}}^{T}\left(\hat{y}^{(i)} - y^{(i)}\right)\right]$$

$$w_{(t+1)} = w_{(t)} - a\frac{1}{N}X^{T}(Xw_t - y)$$

Regressão Logística

Pseudo code

We discussed all necessary components to perform Logistic Regression. Let's bring them together in form of pseudo code:

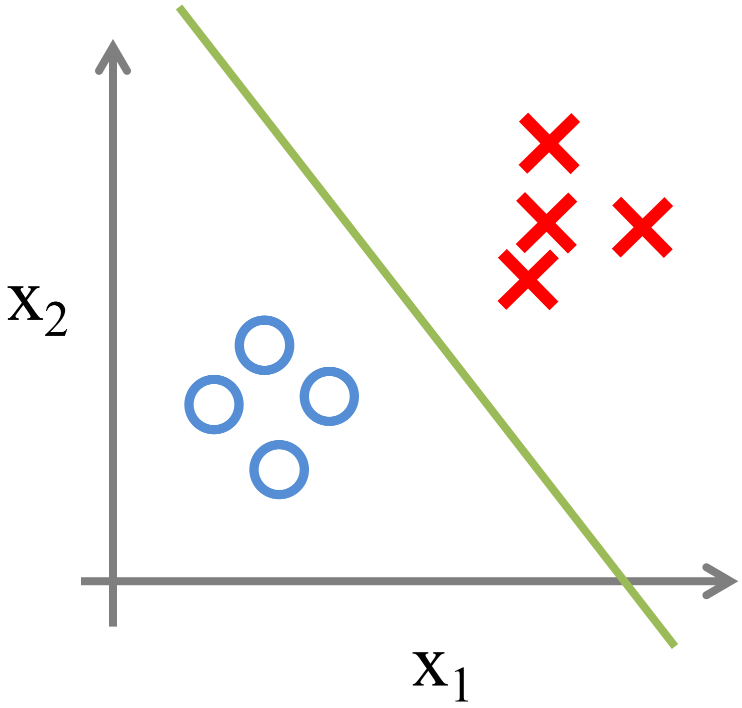
Step 1: Initialize all parameters (B_0, B_1, etc.)
Step 2: Calculate (predict) dependent variable ($h_\theta(x)$)
Step 3: Calculate cost function ($Cost(h_\theta(x), y)$)
Step 4: Calculate gradient for cost function
Step 5: Update all parameters
Step 6: Repeat steps 2 to 5

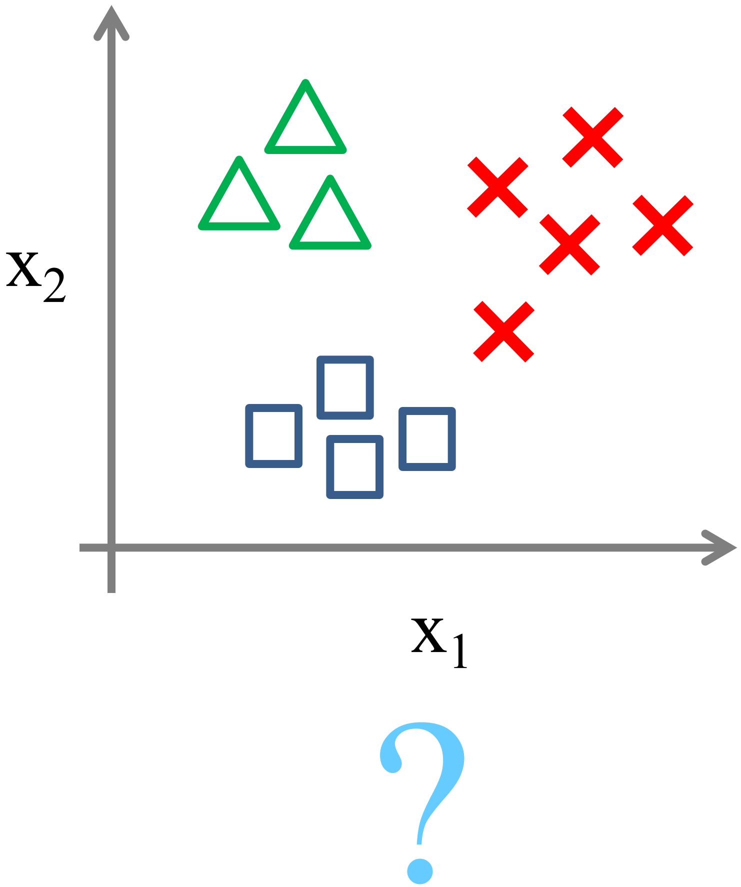
Regressão Logística

- Existem algoritmos de otimização numérica mais efetivos que o gradiente descendente (porém menos eficientes)
  - Gradiente conjugado de Newton
  - BFGS (algoritmo de Broyden-Fletcher-Goldfarb-Shanno)
  - L-BFGS (Limited-memory BFGS)

- Algoritmos complexos e de implementação não-trivial
  - Utilizar bibliotecas renomadas
    - Python (SciPy: scipy.optimize)
    - Matlab (Optimization toolbox: fminunc)
    - Octave (fminunc)

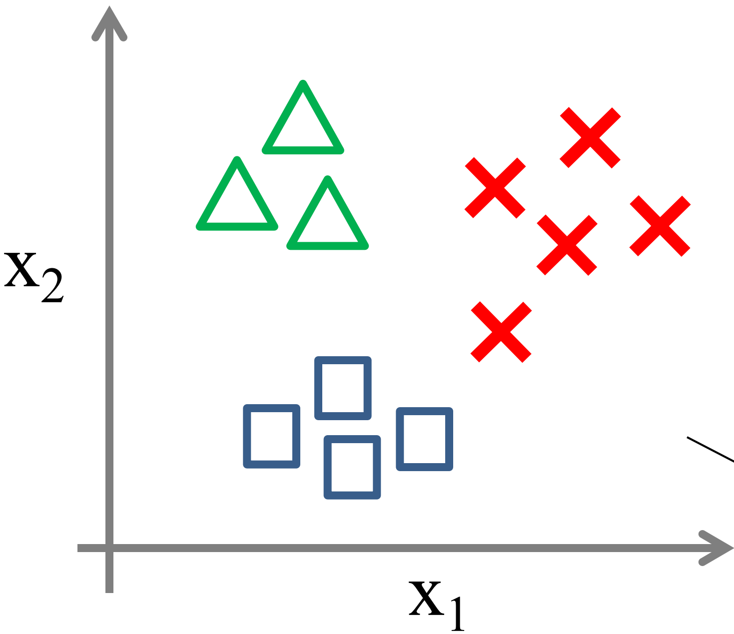
Regressão Logística





Regressão Logística

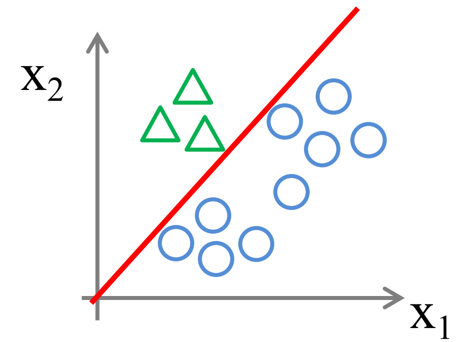
Abordagem one-vs-all



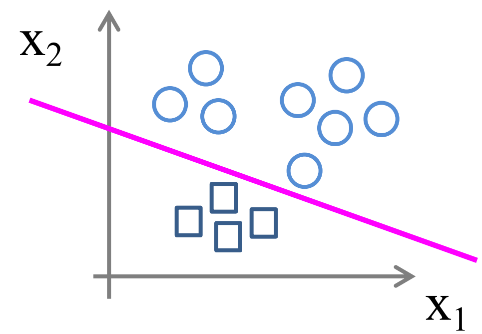
Classe 1: △
Classe 2: □
Classe 3: ✕

$$\hat{f}\,(x)^i = P\big(f\,(x)^i = 1|X;w\big)$$

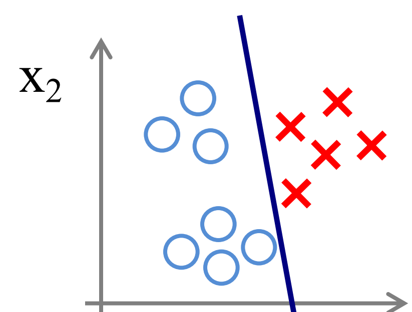
$$\max_i \hat{f}(\mathbf{x})^i$$



$\hat{f}\,(x)^1 = P(f\,(x)^1 = 1|X;w)$



$\hat{f}\,(x)^2 = P(f\,(x)^2 = 1|X;w)$



$\hat{f}\,(x)^3 = P(f\,(x)^3 = 1|X;w)$

Sugestão de Leitura

- Seções 3.2 e 3.3 do livro:
  – Abu-Mostafa et al. **Learning from Data** – A Short Course. 2012.

- Apêndice D (Tan et al. 2006)

- Assistir MOOC Coursera “Machine Learning” – 2 primeiros algoritmos ensinados

Créditos

Slides adaptados dos originais do prof. Andrew Ng

MOOC – Machine Learning (Coursera)