Machine Learning and Data Analytics
ME 5013- Fall 2019

UTSA®
The University of Texas at San Antonio™

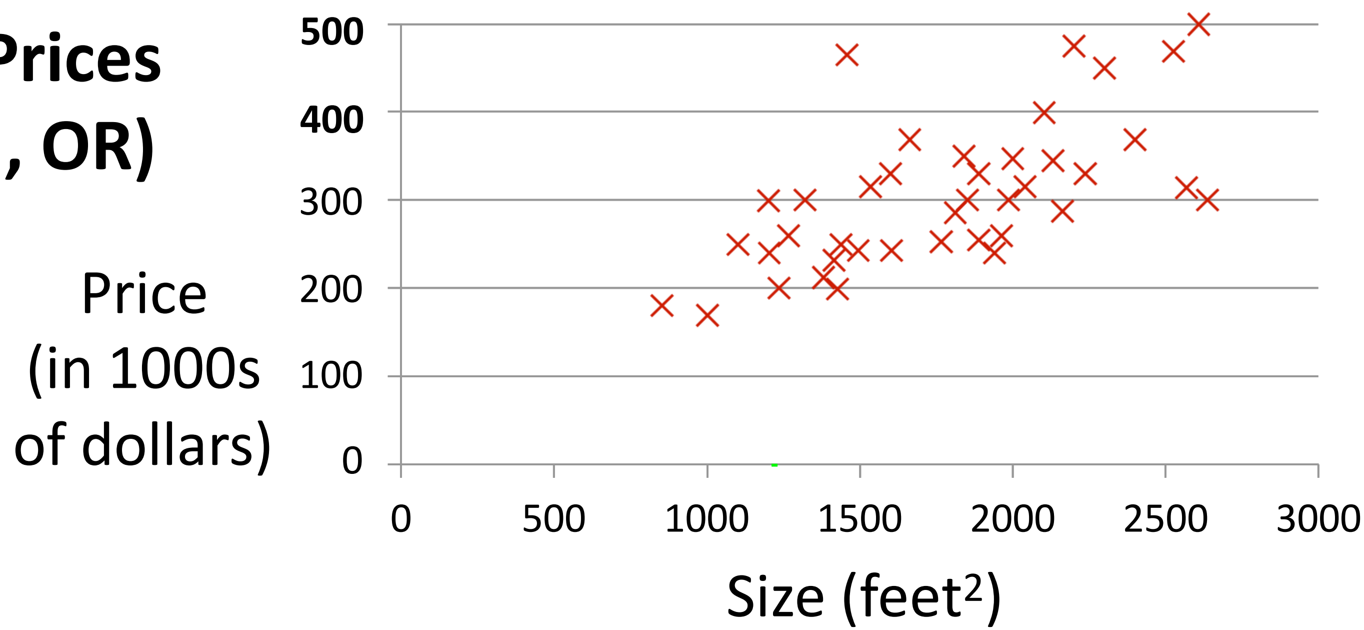
Lecture 05

- Cost Function
- Gradient Descent

Adel Alaeddini, PhD
Associate Professor of Mechanical Engineering
Advanced Data Engineering Lab
adel.alaeddini@utsa.edu

Disclosure: some of the slides are adopted from Dr. Andrew Ng lecture notes on "Machine Learning" at Stanford university

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-‐valued output

Training Set

$$Y = \beta_0 + \beta_1 X + \epsilon$$

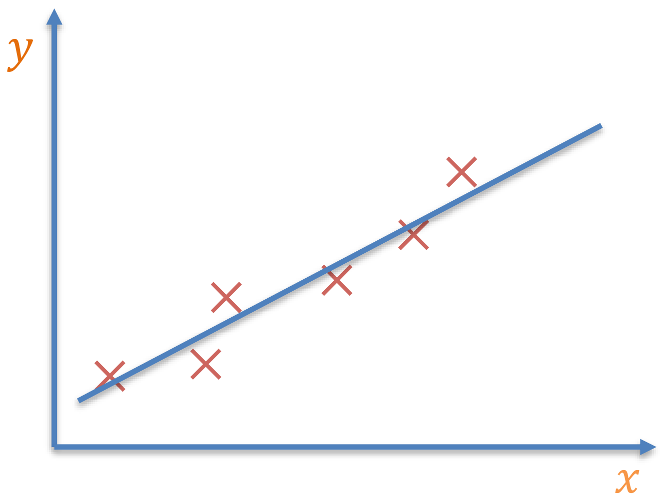
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

Hypothesis: $h_\beta(x) = \beta_0 + \beta_1 x$

β_i's: Parameters

How to choose β_i's ?

Simple linear regression



Idea: Choose β_0, β_1 so that $h_\beta(x)$ is close to y for our (x, y) training examples

Cost function I: $RSS = e_1^2 + e_2^2 + \cdots + e_n^2$

$$RSS = (y_1 - \widehat{\beta}_0 - \widehat{\beta}_1 x_1)^2 + \cdots + (y_n - \widehat{\beta}_0 - \widehat{\beta}_1 x_n)^2$$

$$RSS = \sum_{i=1}^{n} (y_i - \widehat{\beta}_0 - \widehat{\beta}_1 x_i)^2$$

Closed formed formula:

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\widehat{\beta}_0 = \bar{y} - \widehat{\beta}_1 \bar{x}$$

Cost function II:

Hypothesis: $h_\beta(x) = \beta_0 + \beta_1 x$

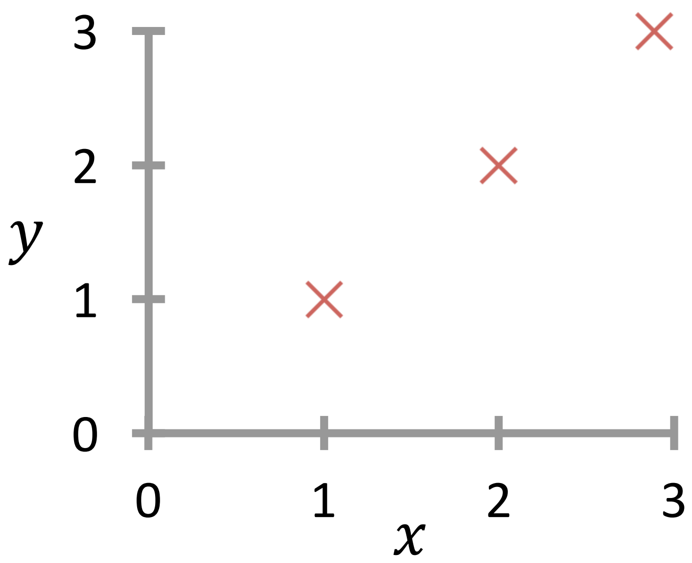
$$\min_{\beta_0, \beta_1} J(\beta_0, \beta_1) = \frac{1}{2n} \sum_{i=1}^{n} (y_n - \beta_0 - \beta_1 x_i)^2$$

Intuition

- Let $\beta_0 = 0$

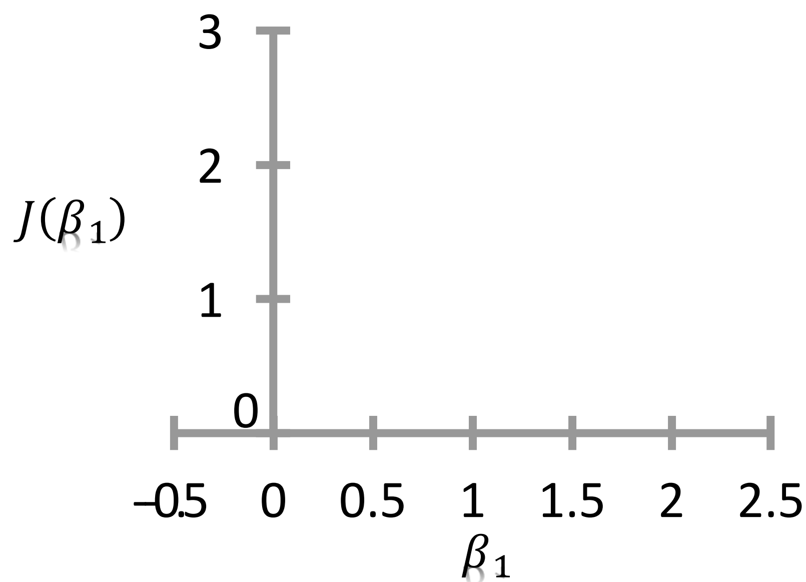
$h_{\beta_1}(x)$

For fixed β_1 this is a function of x)



$J(\beta_1)$

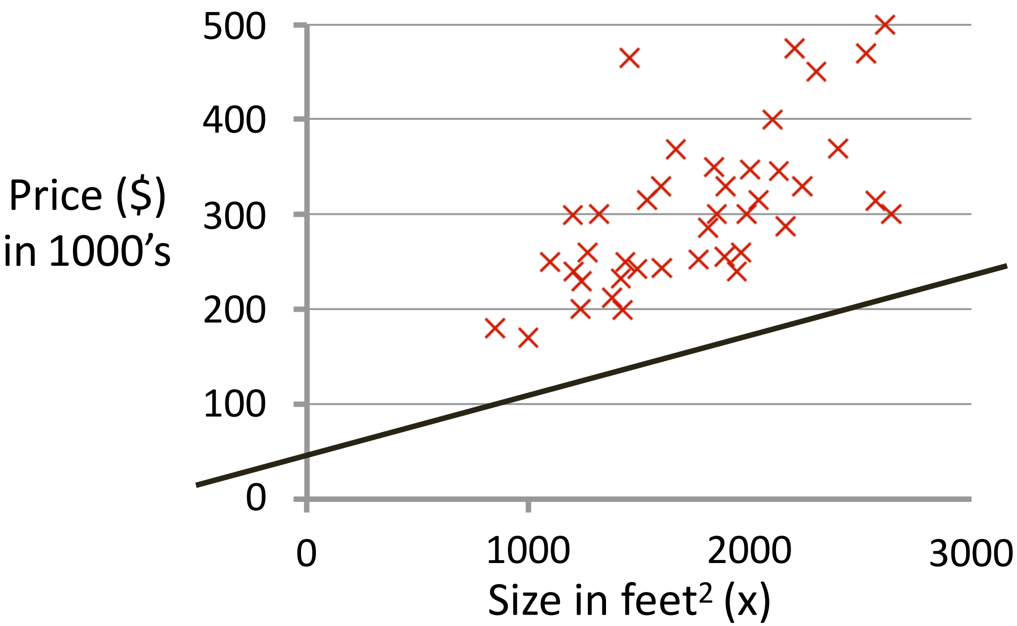
(function of the parameter β_1)



Intuition

$h_{\beta_0,\beta_1}(x)$

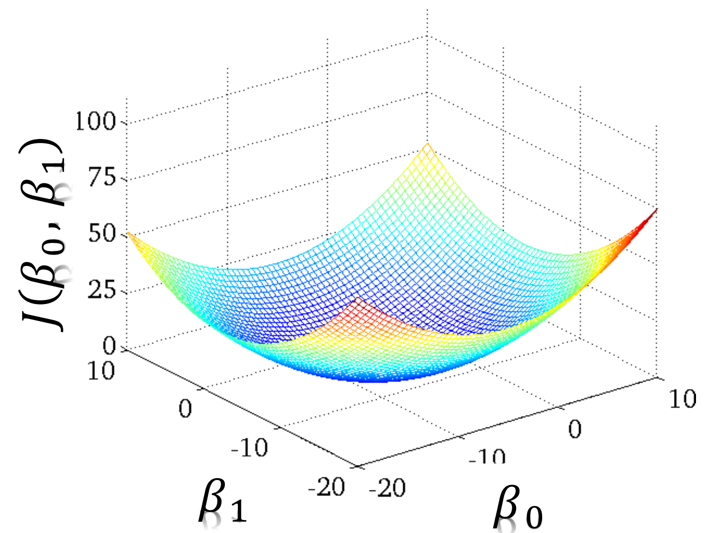
For fixed β_0, β_1 this is a function of x)



$$h_{\beta_0,\beta_1}(x) = 50 + 0.06x$$

$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



Gradient Descent

Have some function $J(\beta_0, \beta_1)$

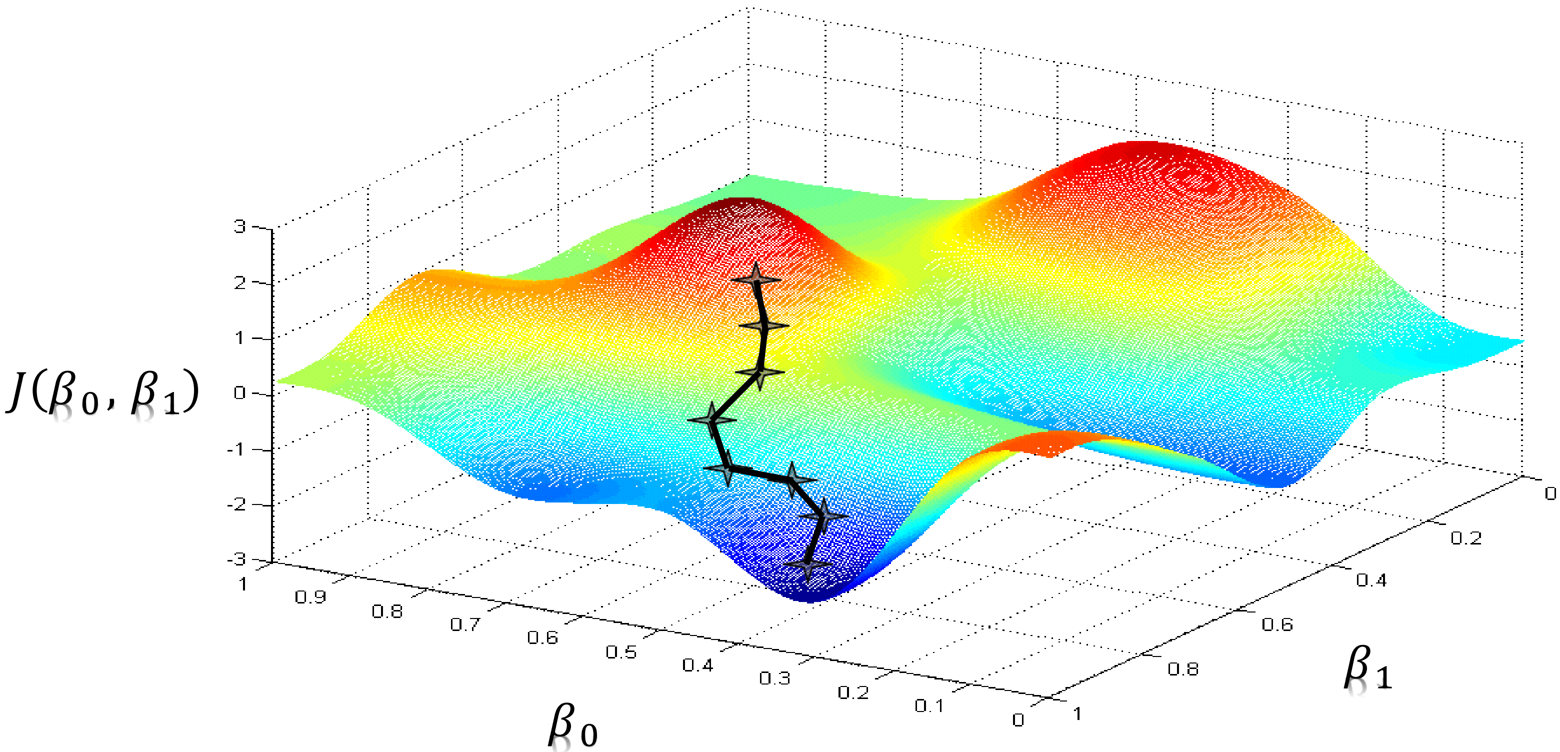
Want $\min_{\beta_0, \beta_1} J(\beta_0, \beta_1)$

Outline:

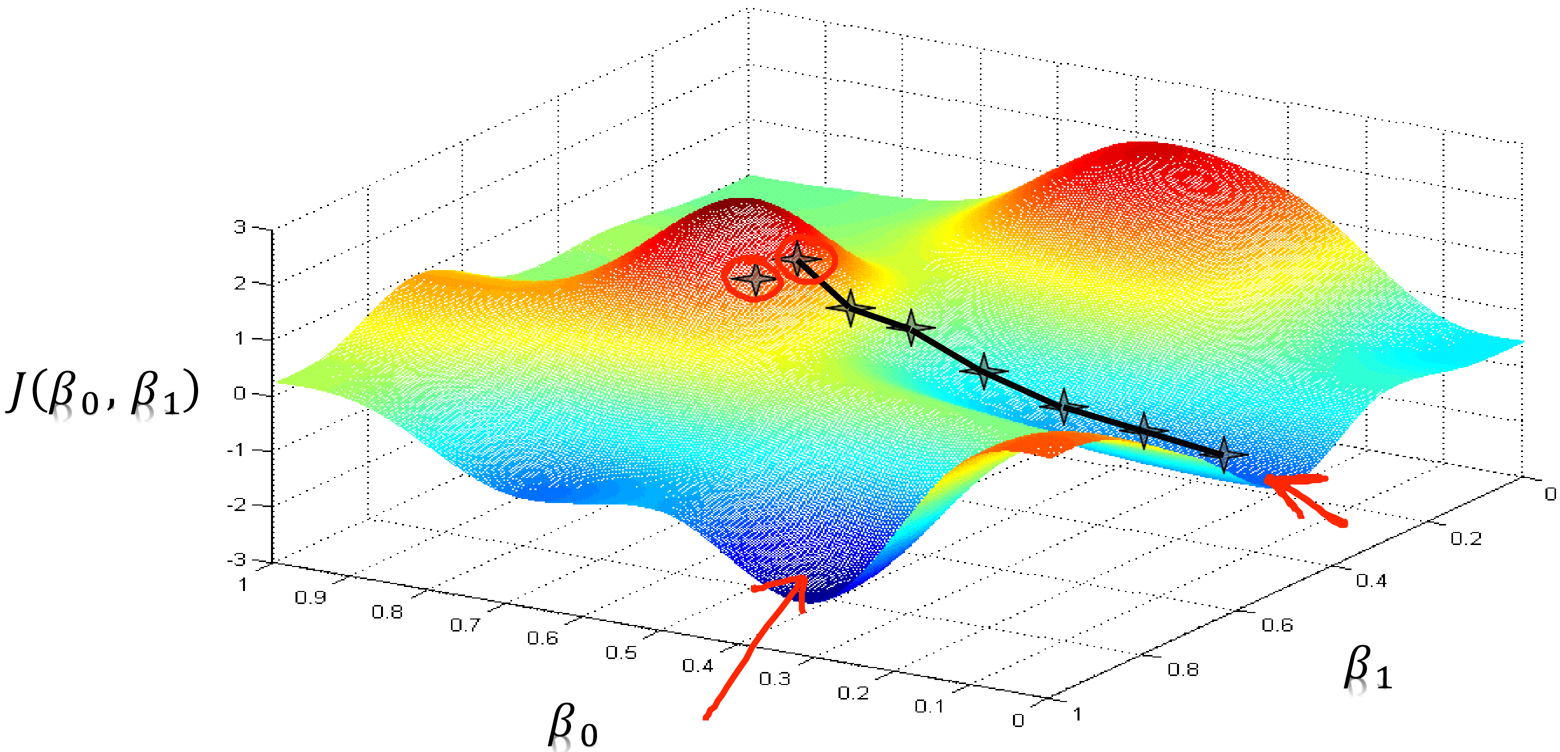
- Start with some β_0, β_1
- Keep changing β_0, β_1 to reduce $J(\beta_0, \beta_1)$

  Until we hopefully end up at a minimum

Gradient Descent



Gradient Descent



Gradient descent algorithm

repeat until convergence {

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta_0, \beta_1) \quad \text{(for } j = 0 \text{ and } j = 1\text{)}$$

}

Correct: Simultaneous update

$$\text{temp0} := \beta_0 - \alpha \frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1)$$

$$\text{temp1} := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1)$$

$$\beta_0 := \text{temp0}$$

$$\beta_1 := \text{temp1}$$

Incorrect:

$$\text{temp0} := \beta_0 - \alpha \frac{\partial}{\partial \beta_0} J(\beta_0, \beta_1)$$

$$\beta_0 := \text{temp0}$$

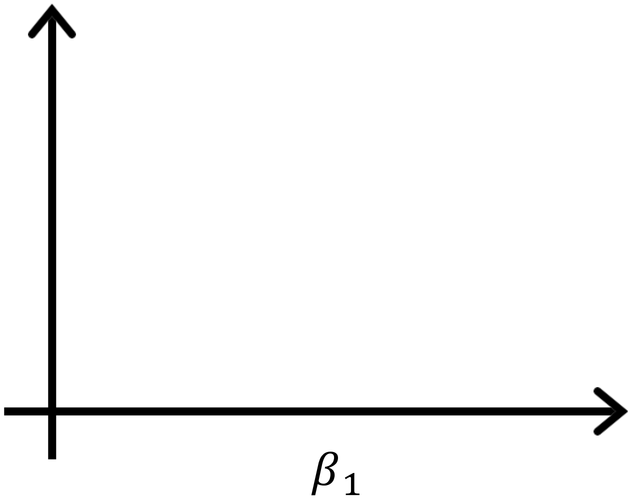
$$\text{temp1} := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1)$$

$$\beta_1 := \text{temp1}$$

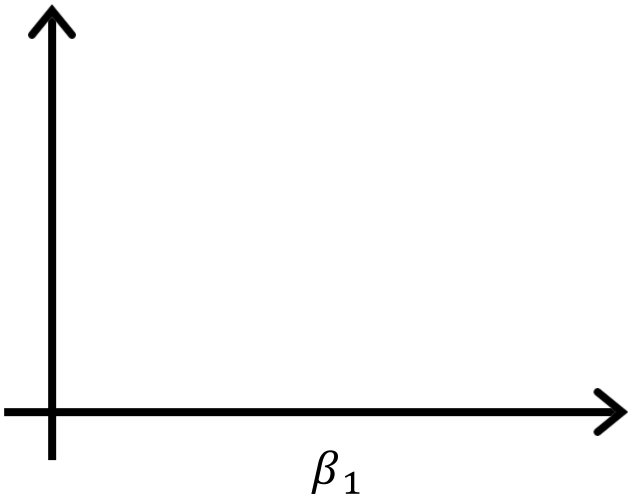
Gradient Descent

$$\beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1)$$

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



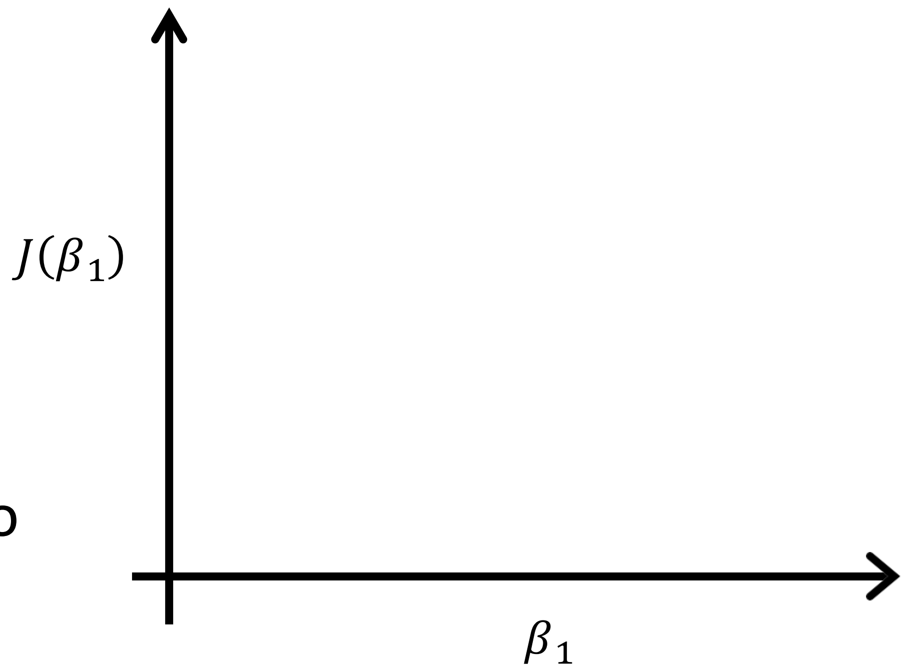
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



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

$$\beta_1 := \beta_1 - \alpha \frac{\partial}{\partial \beta_1} J(\beta_0, \beta_1)$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over 6me.



Gradient descent for linear regression

Gradient descent algorithm

repeat until convergence {

$$\beta_j := \beta_j - \alpha \frac{\partial}{\partial \beta_j} J(\beta_0, \beta_1)$$

(for $j = 0$ and $j = 1$)

}

Hypothesis: $h_\beta(x) = \beta_0 + \beta_1 x$

$$\min_{\beta_0, \beta_1} J(\beta_0, \beta_1)$$

$$= \frac{1}{2n} \sum_{i=1}^{n} (y_n - \beta_0 - \beta_1 x_i)^2$$

Gradient descent algorithm for regression

repeat until convergence $\{$

$$\beta_0 := \beta_0 + \alpha \frac{1}{n}\sum_{i=1}^{n}(y_n - \beta_0 - \beta_1 x_i)$$

$$\beta_1 := \beta_1 + \alpha \frac{1}{n}\sum_{i=1}^{n}(y_n - \beta_0 - \beta_1 x_i)\, x_i$$

$\}$

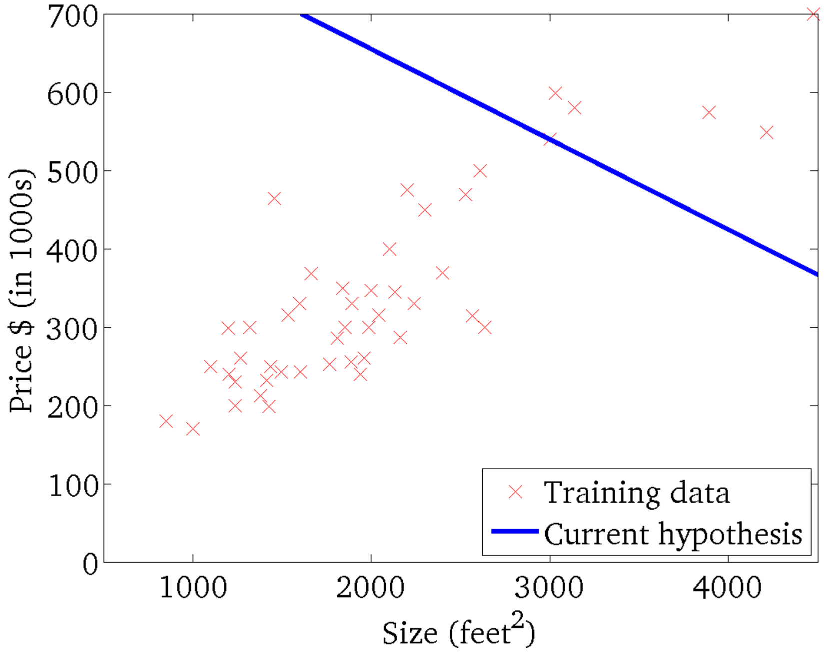
update β_0 and β_1 simultaneously

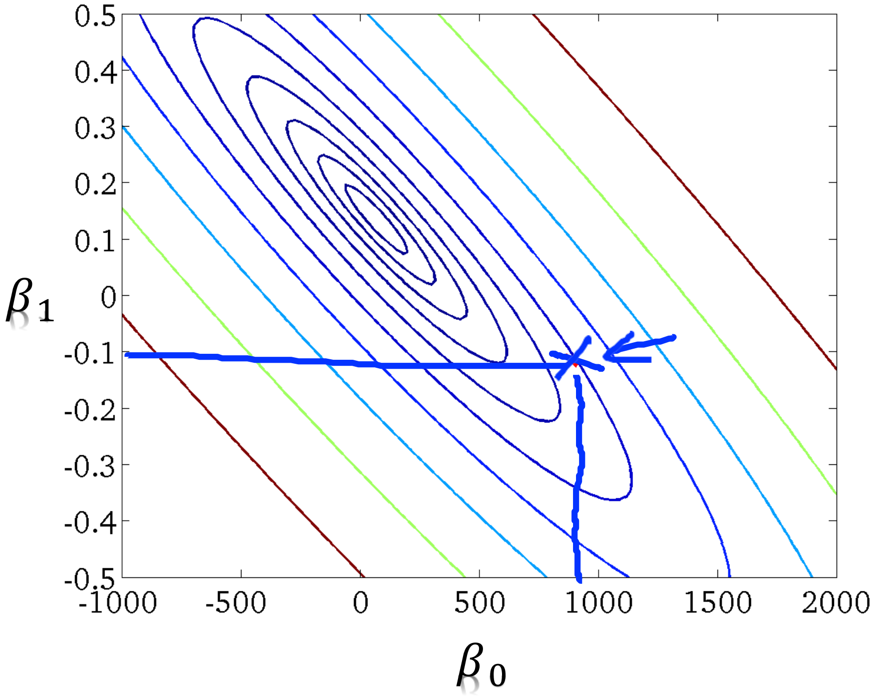
$h_{\beta_0,\beta_1}(x)$

$J(\beta_0, \beta_1)$

For fixed β_0, β_1 this is a function of x)

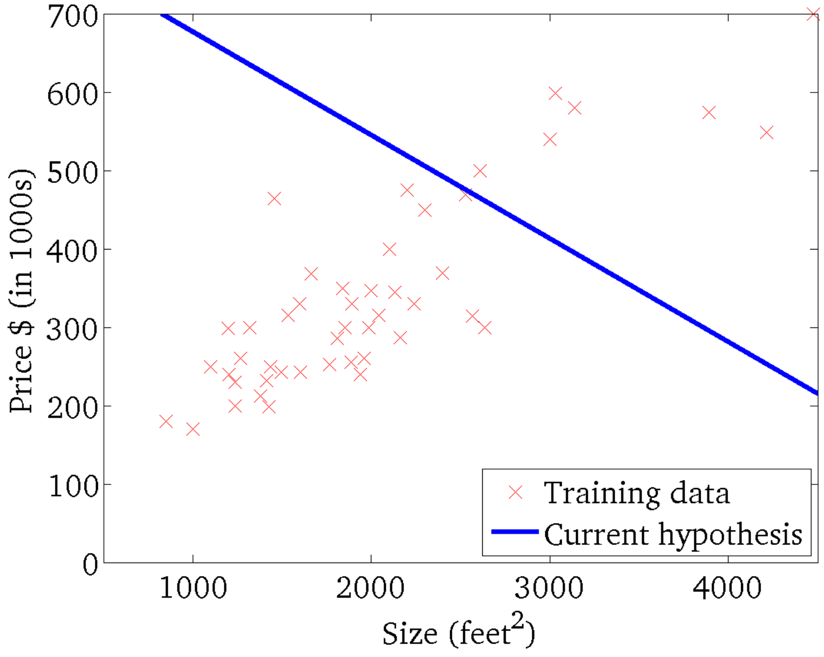
(function of the parameters β_0, β_1)





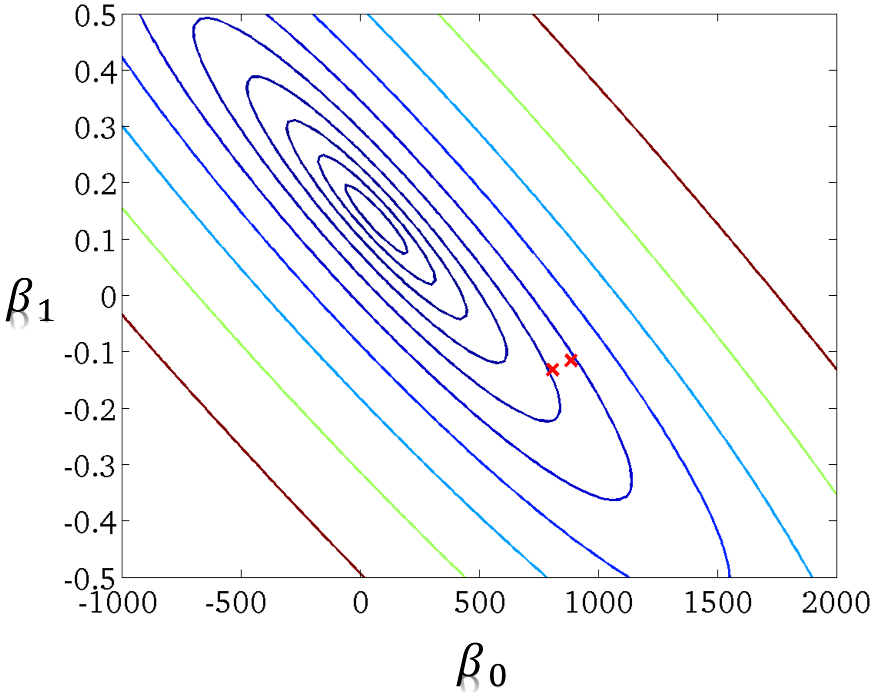
$h_{\beta_0,\beta_1}(x)$

For fixed β_0, β_1 this is a function of x)



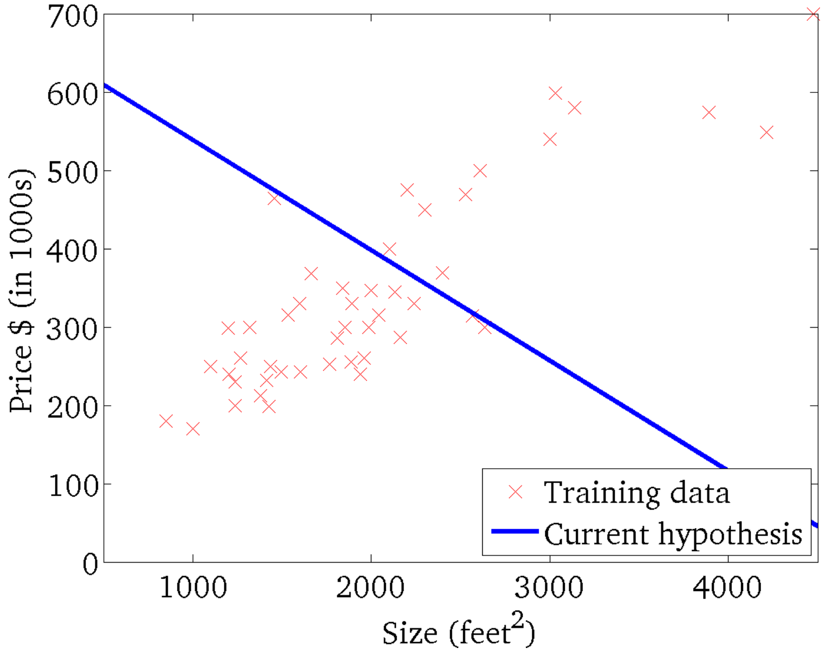
$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



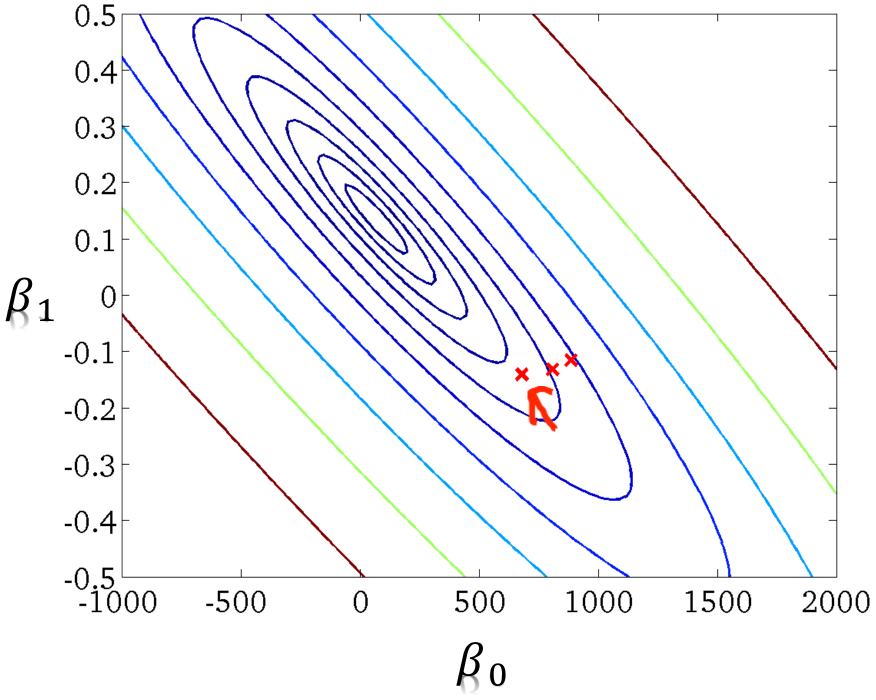
$h_{\beta_0,\beta_1}(x)$

For fixed β_0, β_1 this is a function of x)



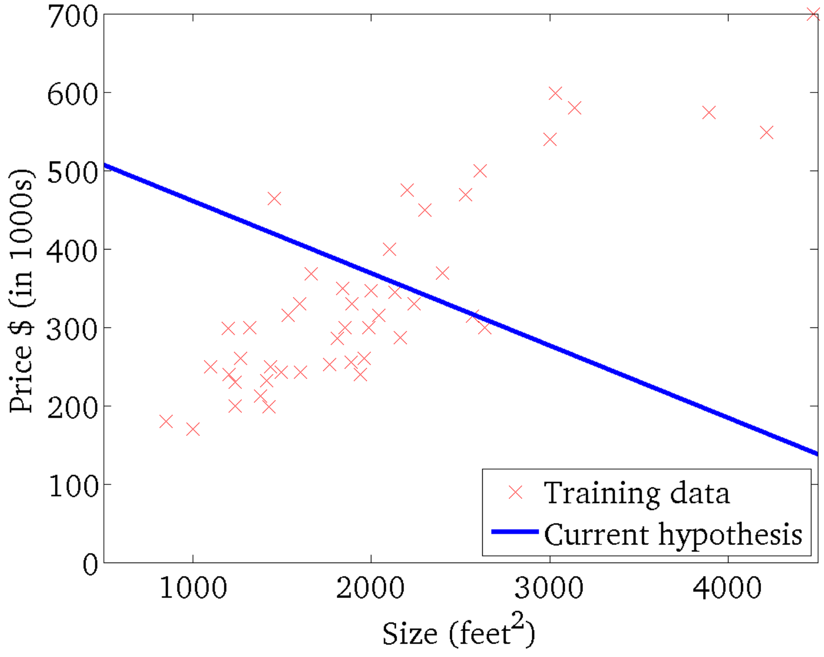
$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



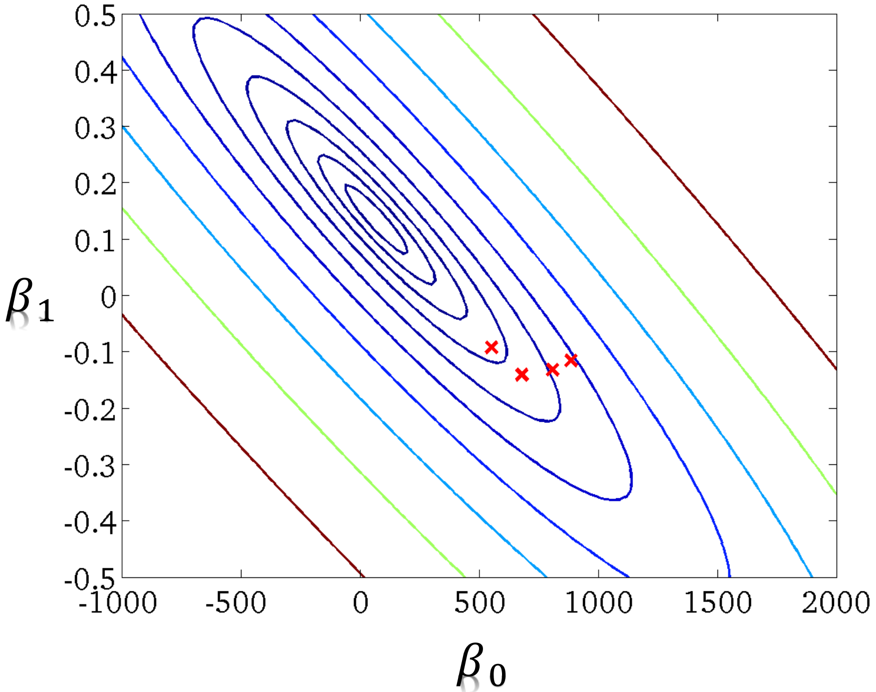
$h_{\beta_0,\beta_1}(x)$

For fixed β_0, β_1 this is a function of x)



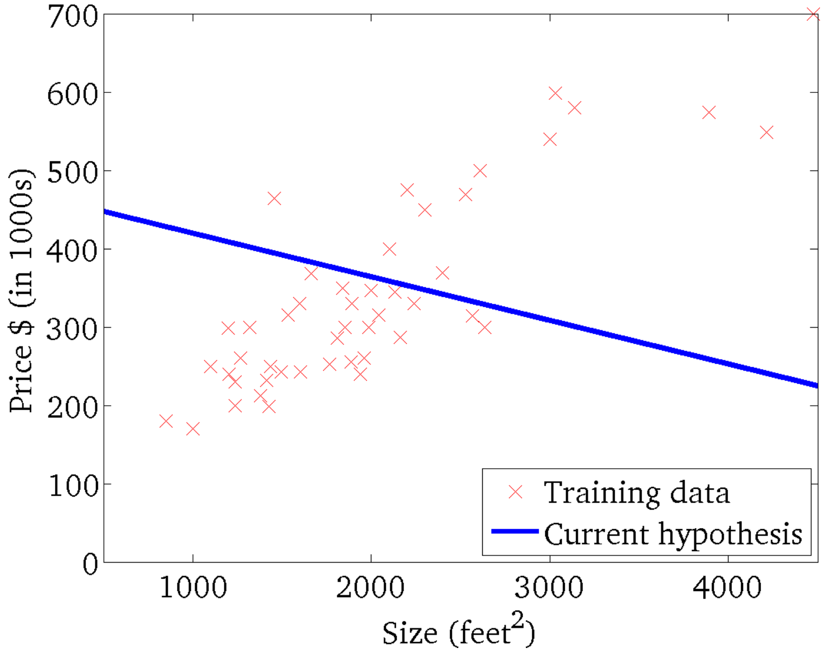
$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



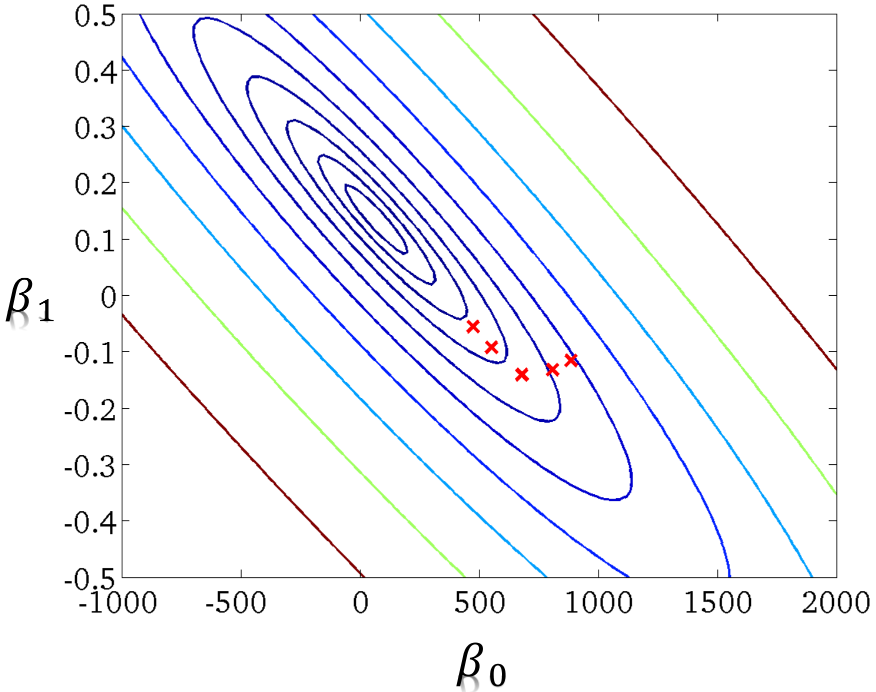
$h_\beta(x)$

For fixed β_0, β_1 this is a function of x)



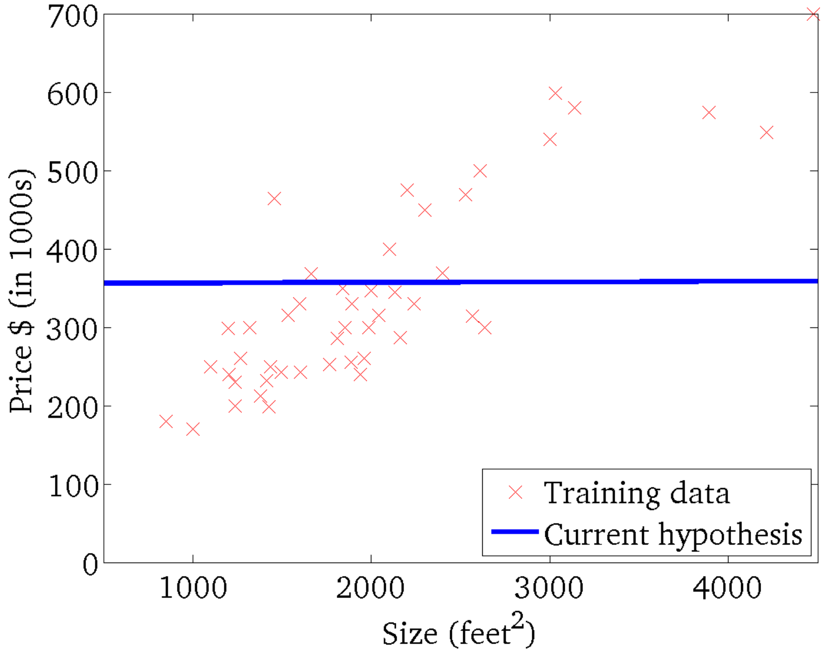
$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



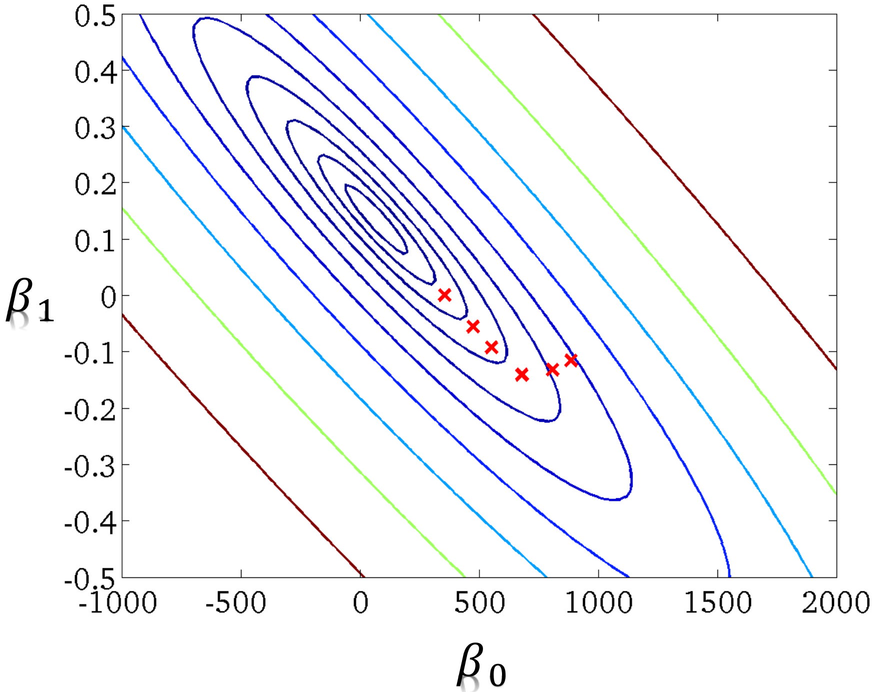
$h_\beta(x)$

For fixed β_0, β_1 this is a function of x)



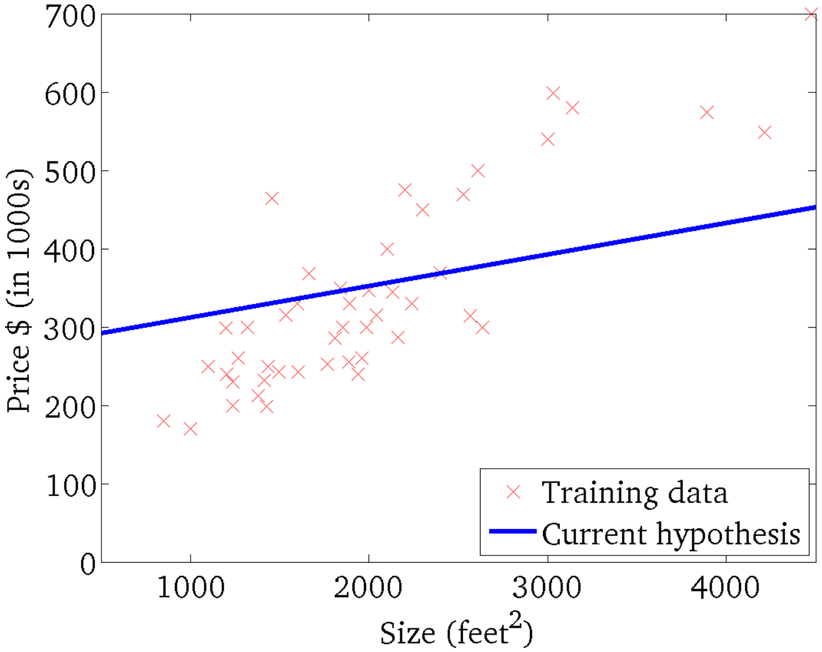
$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



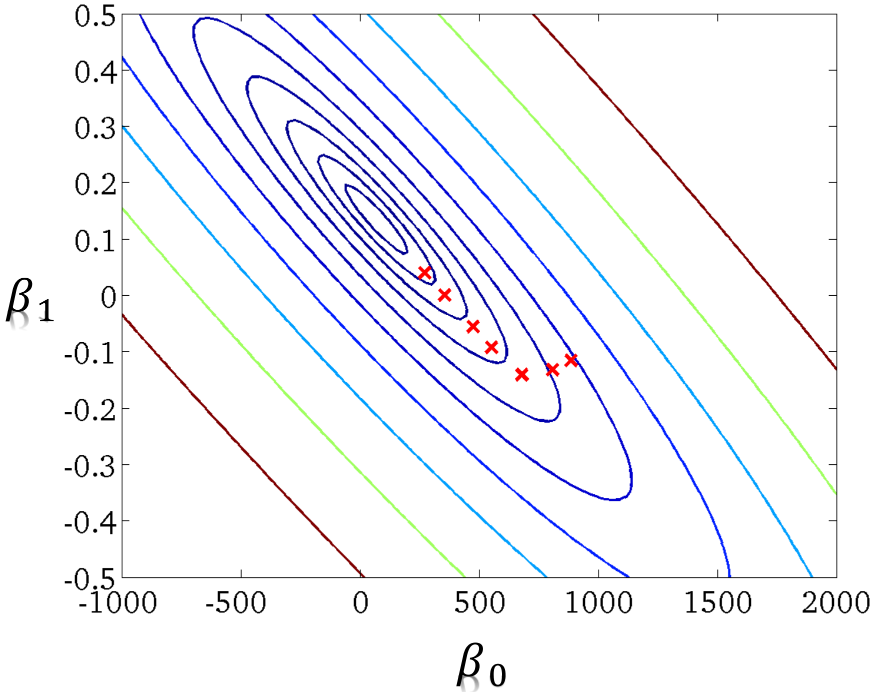
$h_{\beta_0,\beta_1}(x)$

For fixed β_0, β_1this is a function of x)



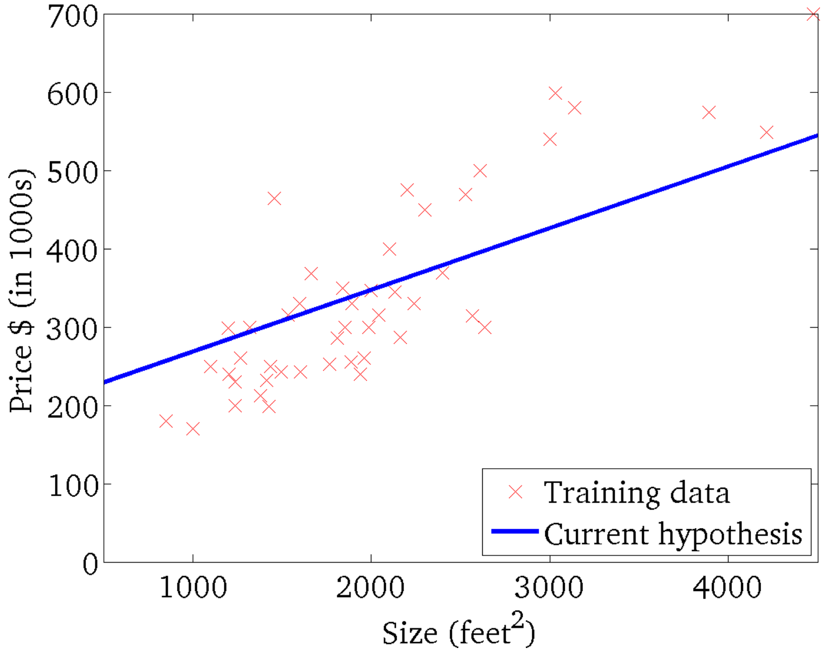
$J(\beta_0, \beta_1)$

(function of the parametersβ_0, β_1)



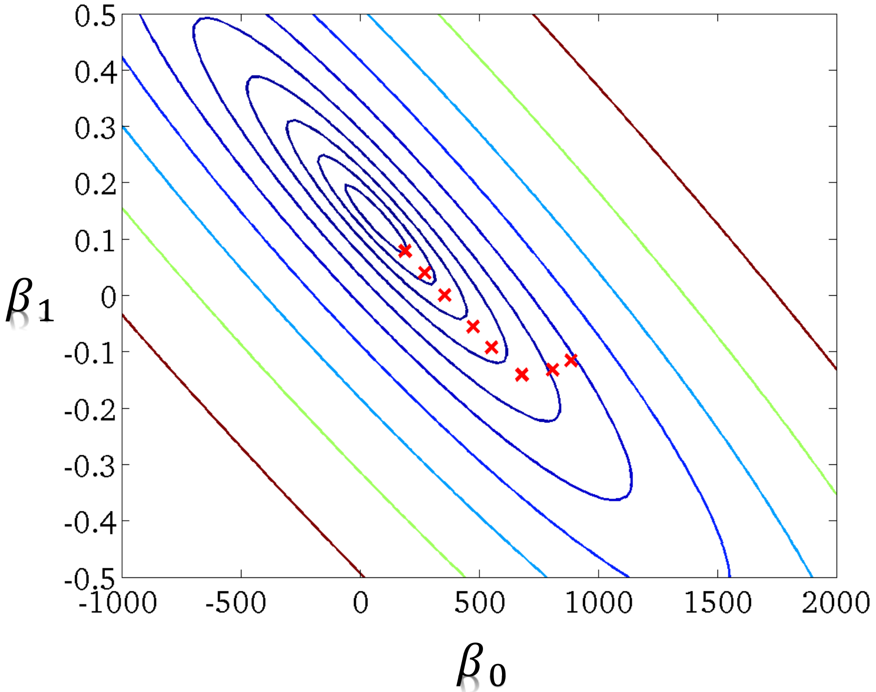
$h_{\beta_0,\beta_1}(x)$

For fixed β_0, β_1 this is a function of x)



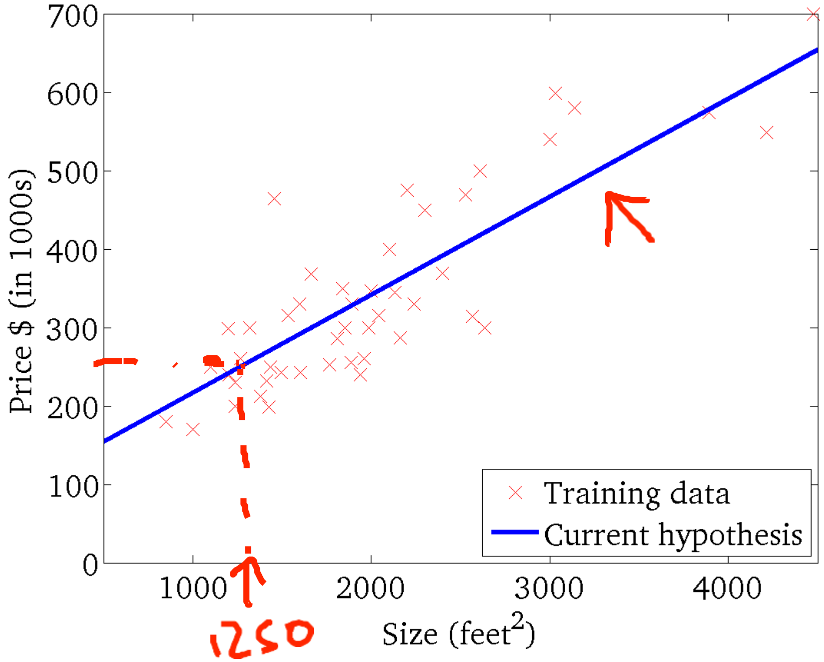
$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



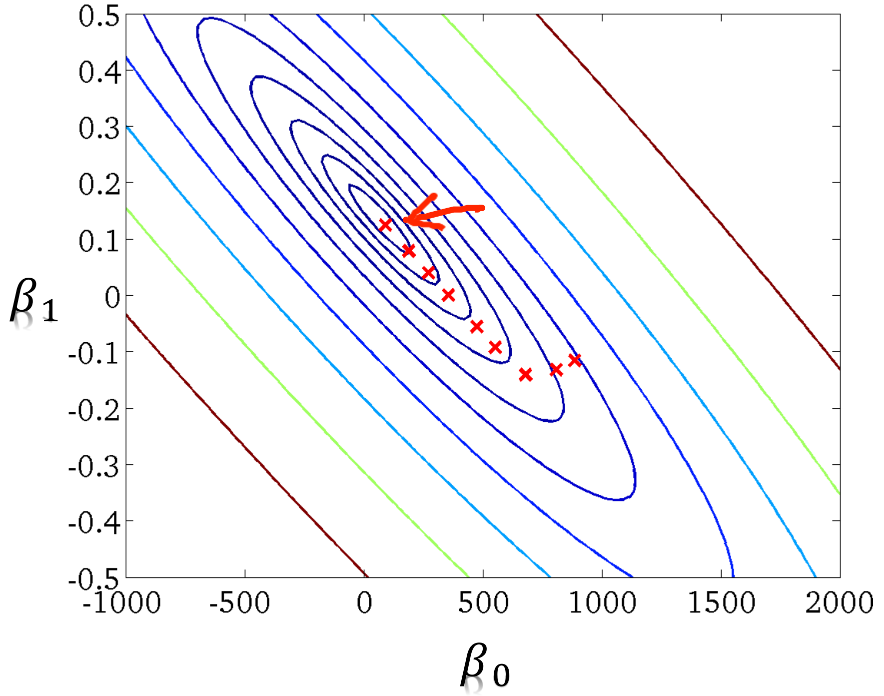
$h_{\beta_0,\beta_1}(x)$

For fixed β_0, β_1 this is a function of x)



$J(\beta_0, \beta_1)$

(function of the parameters β_0, β_1)



“Batch” Gradient Descent

“Batch”: Each step of gradient descent uses all the training examples.