

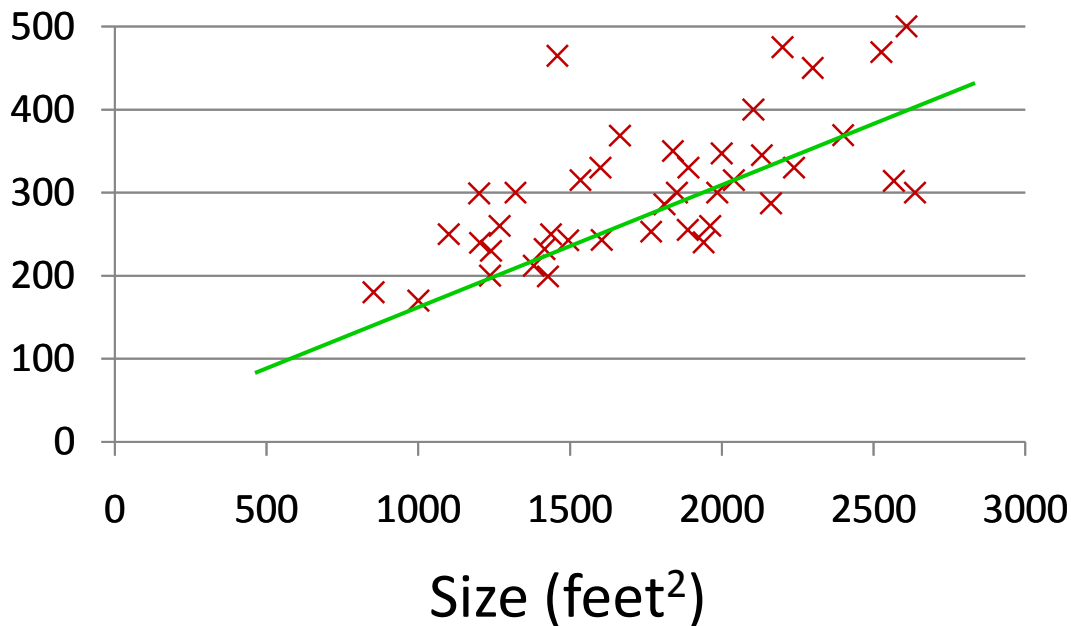
Machine Learning

Linear regression
with one variable

Model
representation

Housing Prices (Portland, OR)

Price
(in 1000s
of dollars)



Supervised Learning

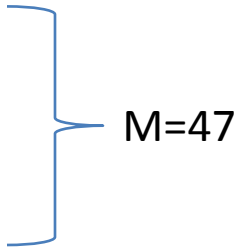
Given the “right answer” for each example in the data.

Regression Problem

Predict real-valued output
How about classification?

Training set of housing prices (Portland, OR)

Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...



M=47

Notation:

m = Number of training examples

x's = "input" variable / features

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

(x,y)-one training example

($x^{(i)}$, $y^{(i)}$)-ith training example

$$x^{(1)}=2104$$

$$x^{(2)}=1416$$

$$y^{(1)}=460$$

$$(x^{(1)}, y^{(1)})=(2104,460)$$

Training Set



Learning Algorithm



Size of
house



h



Estimated
price

x

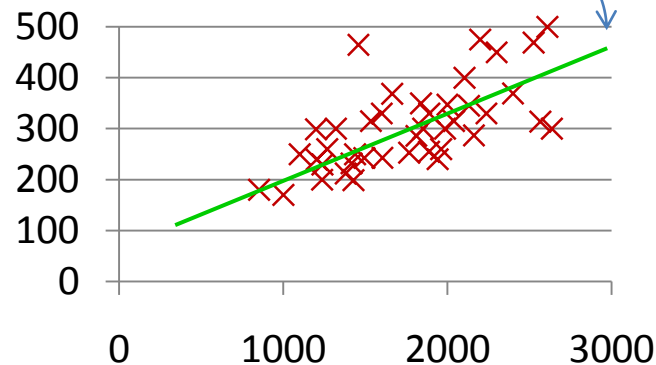
hypothesis

Estimated
value of y

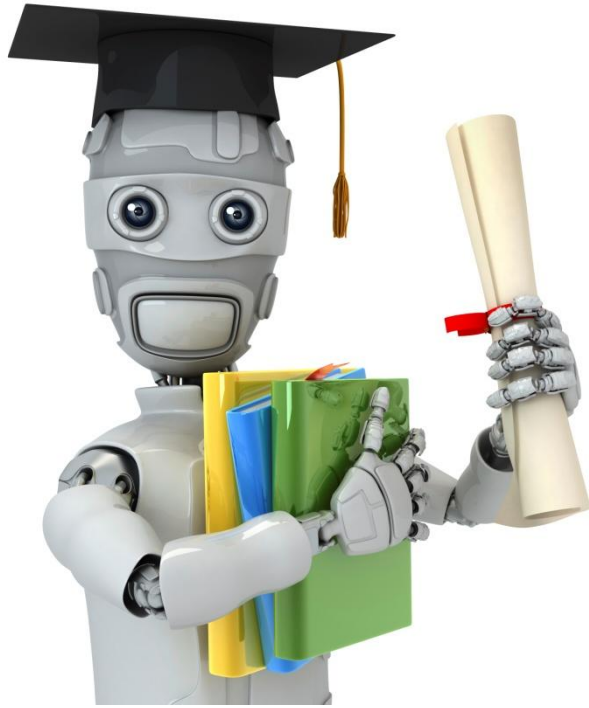
How do we represent h ?

h maps from x 's to y 's

example: $h_{\theta}(x) = \theta_0 + \theta_1 x$



Linear regression with one variable (x).
Univariate linear regression.



Machine Learning

Linear regression
with one variable

Cost function

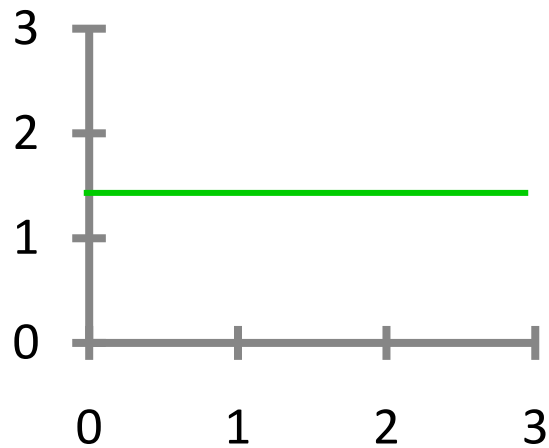
Training Set	Size in feet ² (x)	Price (\$) in 1000's (y)
	2104	460
	1416	232
	1534	315
	852	178

Hypothesis: $h_{\theta}(x) = \theta_0 + \theta_1 x$

θ_i 's: Parameters

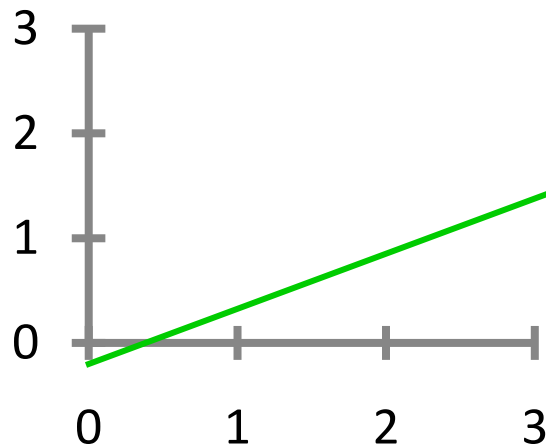
How to choose θ_i 's ?

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



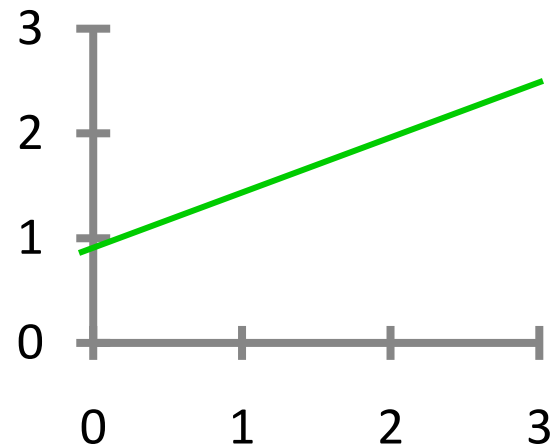
$$\theta_0 = 1.5$$

$$\theta_1 = 0$$



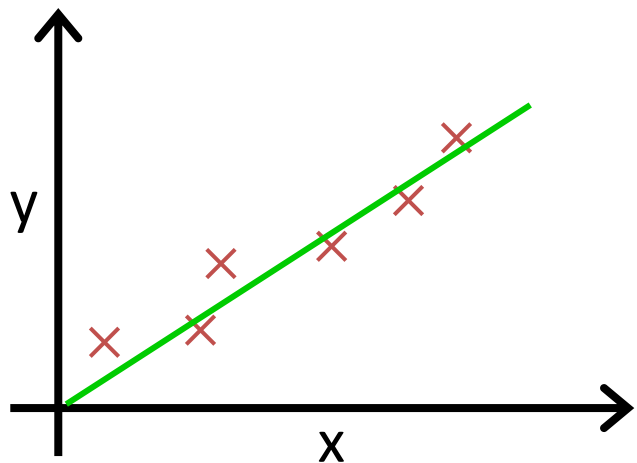
$$\theta_0 = 0$$

$$\theta_1 = 0.5$$



$$\theta_0 = 1$$

$$\theta_1 = 0.5$$



Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

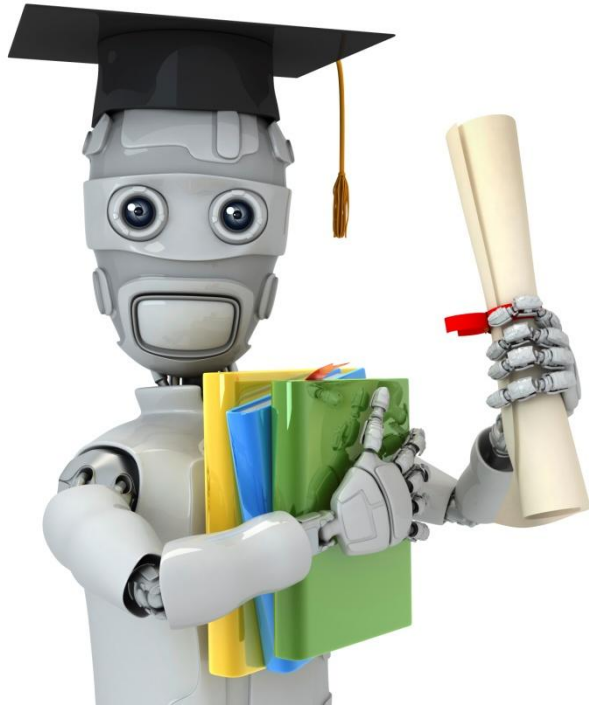
↗ #training example

$$\underset{\theta_0, \theta_1}{\text{minimize}} \quad \frac{1}{2m} \sum_{i=1}^m \underbrace{\left(h_\theta(x^{(i)}) - y^{(i)} \right)^2}_{h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}}$$

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

$$\underset{\theta_0, \theta_1}{\text{minimize}} \quad \underbrace{J(\theta_0, \theta_1)}$$

Cost function



Machine Learning

Linear regression
with one variable

Cost function
intuition I

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$

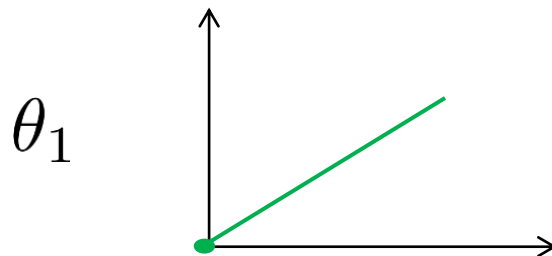
Cost Function:

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

Goal: minimize $J(\theta_0, \theta_1)$
 θ_0, θ_1

Simplified

$$h_{\theta}(x) = \theta_1 x \quad \text{Set } \theta_0 = 0$$

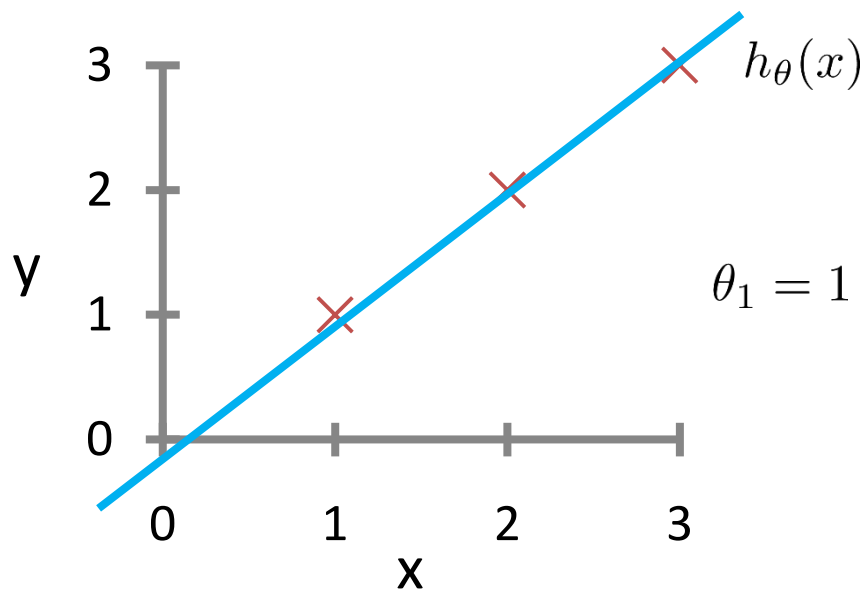


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

minimize $J(\theta_1)$
 θ_1

$$h_{\theta}(x)$$

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

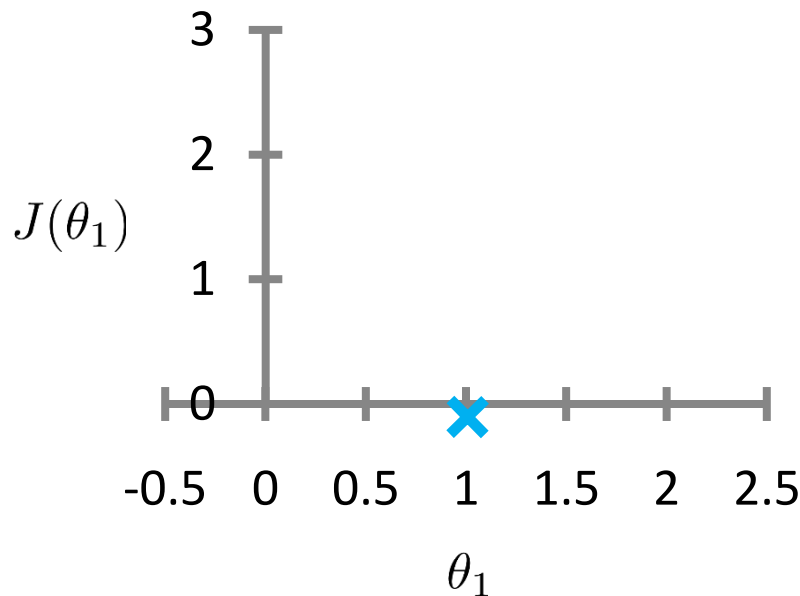


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^m (\theta_1 x^{(i)} - y^{(i)})^2 = \frac{1}{2m} (0^2 + 0^2 + 0^2) = 0$$

$$J(\theta_1)$$

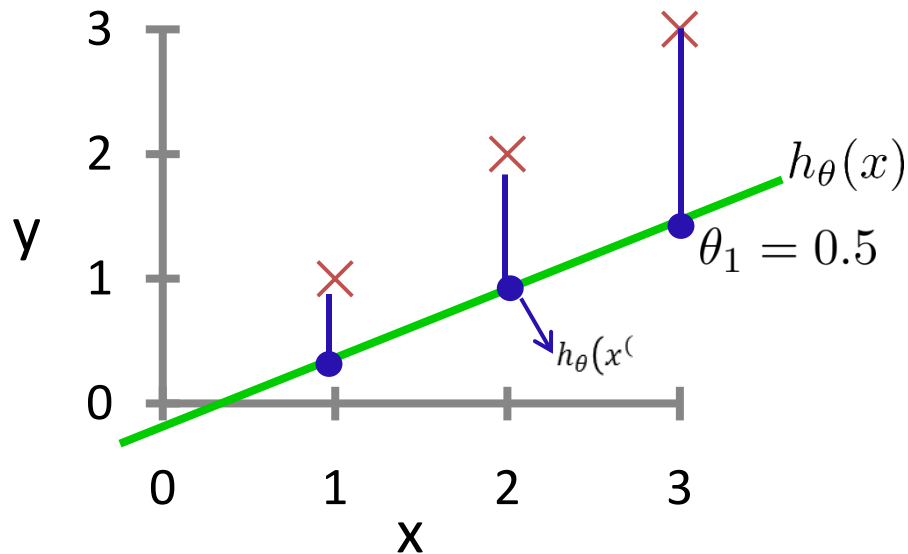
(function of the parameter θ_1)



$$J(\theta_1) = 0$$

$$h_{\theta}(x)$$

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

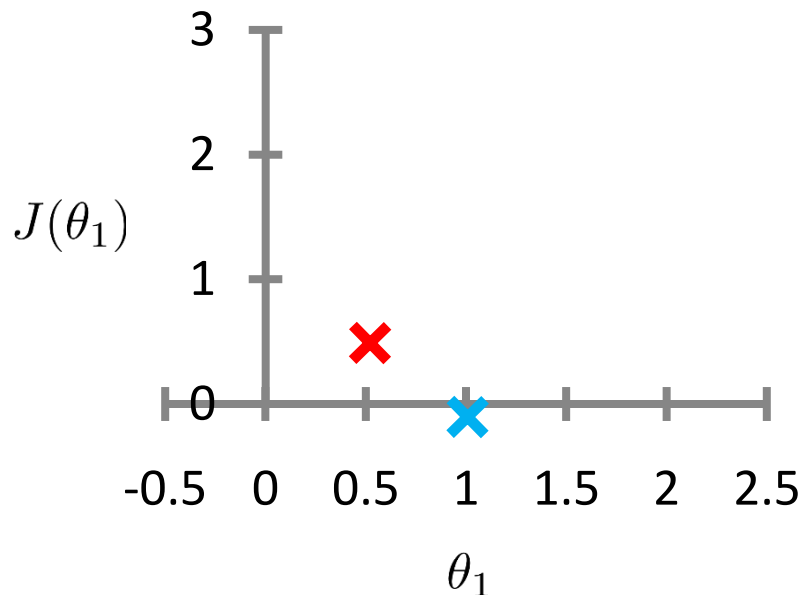


$$J(\theta_1) = \frac{1}{2m} [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2]$$

$$= \frac{1}{2 * 3} (3.5) \approx 0.58$$

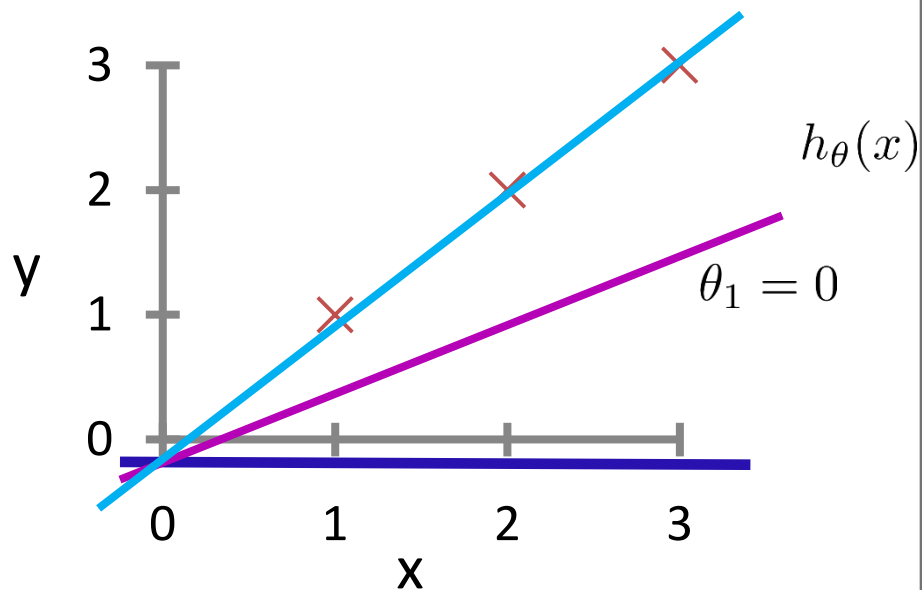
$$J(\theta_1)$$

(function of the parameter θ_1)



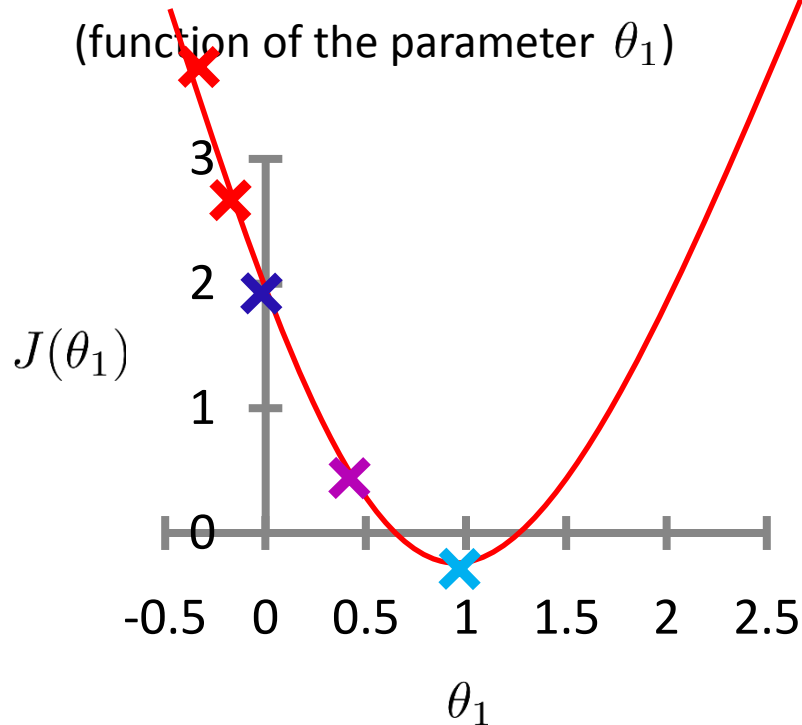
$$h_{\theta}(x)$$

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



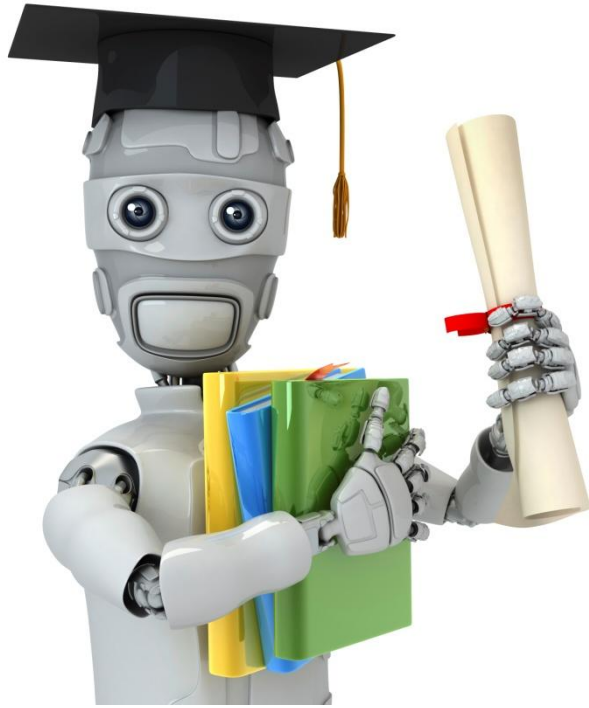
$$J(\theta_1)$$

(function of the parameter θ_1)



minimize $J(\theta_1) = 0$
 θ_1

$$\theta_1 = 1$$



Machine Learning

Linear regression
with one variable

Cost function
intuition II

Hypothesis: $h_{\theta}(x) = \theta_0 + \theta_1 x$

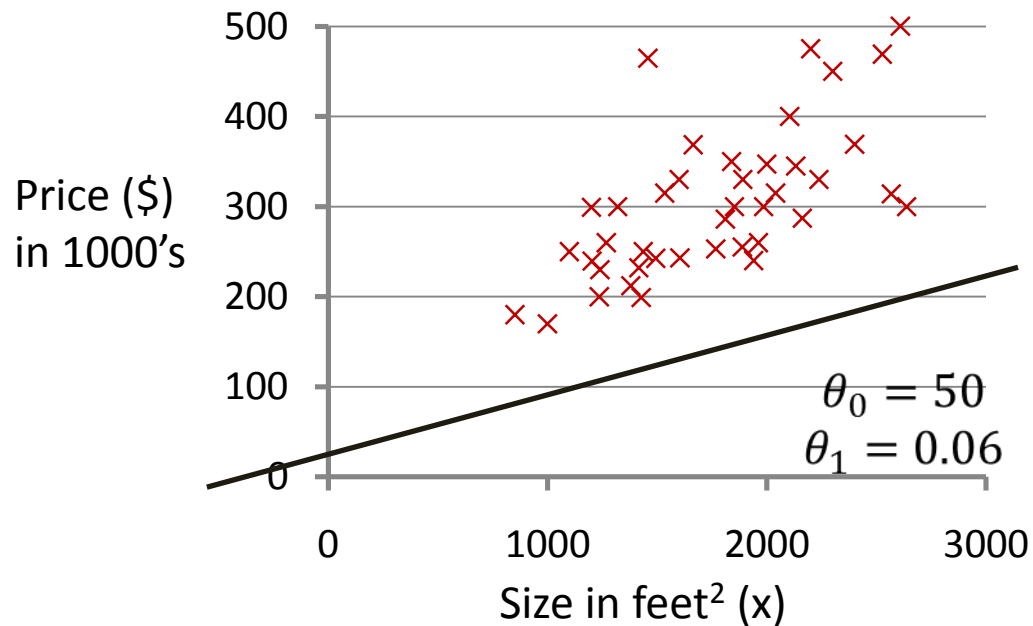
Parameters: θ_0, θ_1

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

Goal: minimize $J(\theta_0, \theta_1)$
 θ_0, θ_1

$$h_{\theta}(x)$$

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

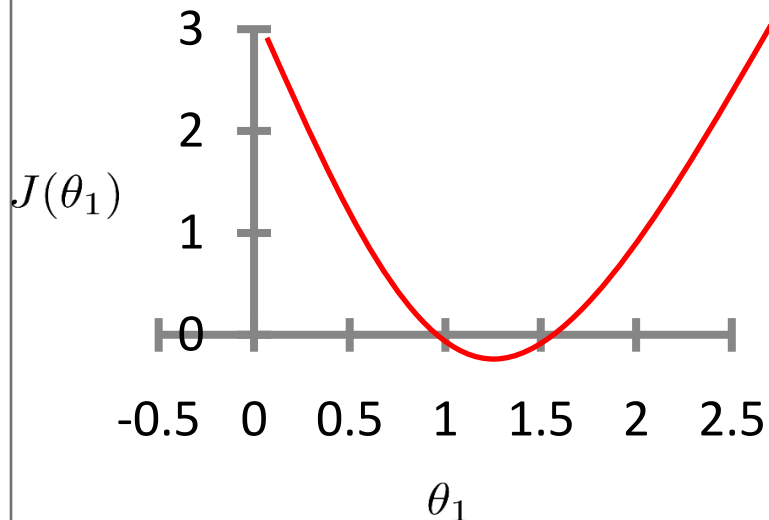


$$h_{\theta}(x) = 50 + 0.06x$$

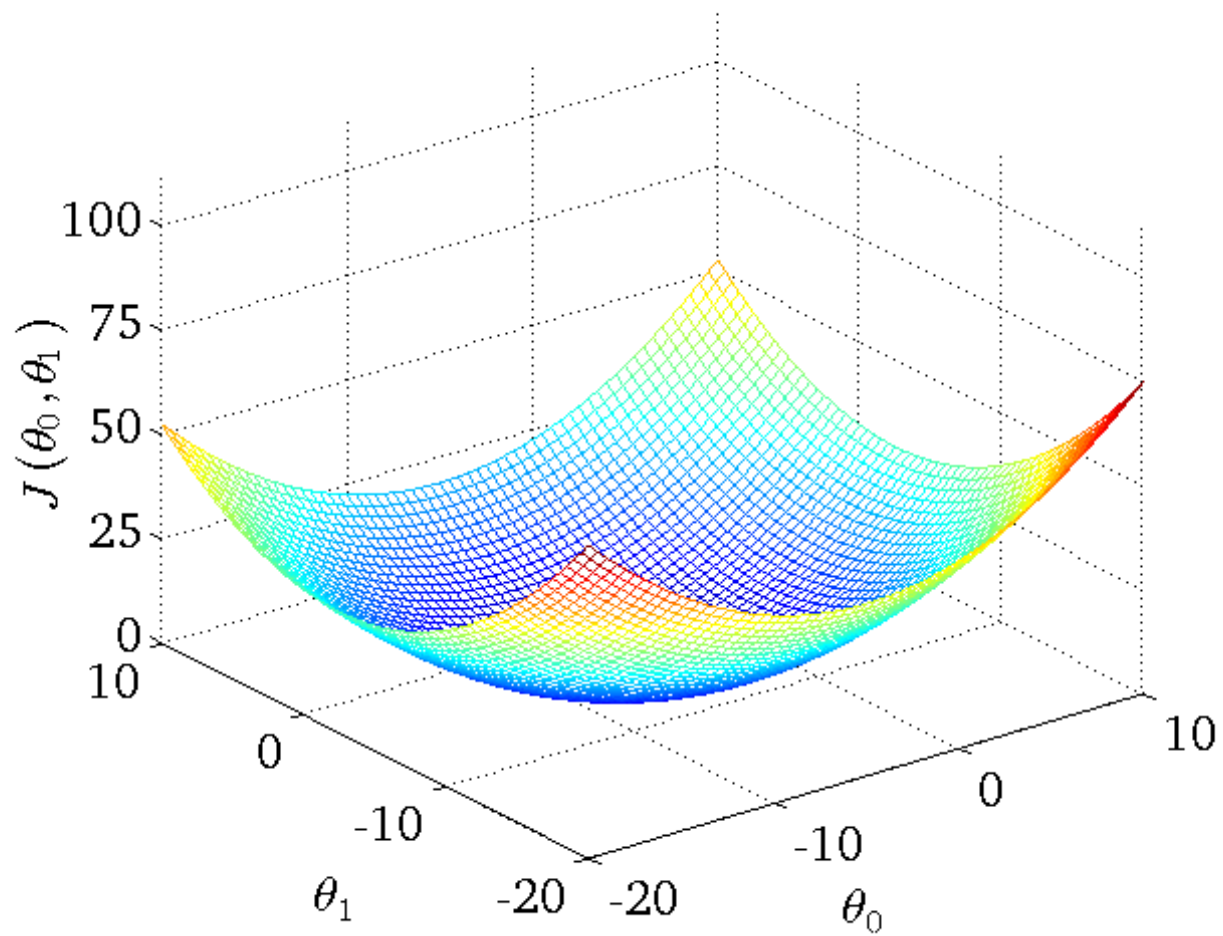
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

Cannot plot like this:

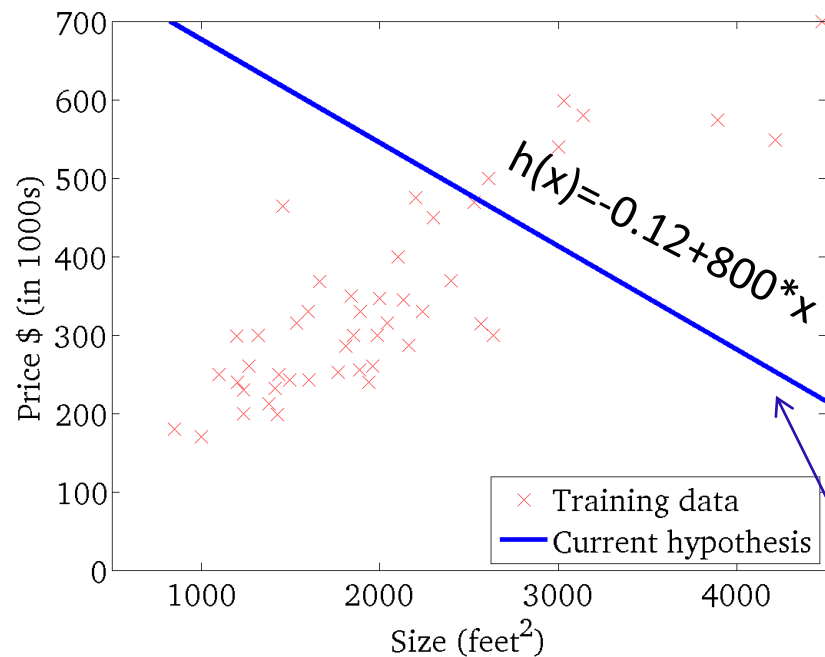


Because we have two parameters.



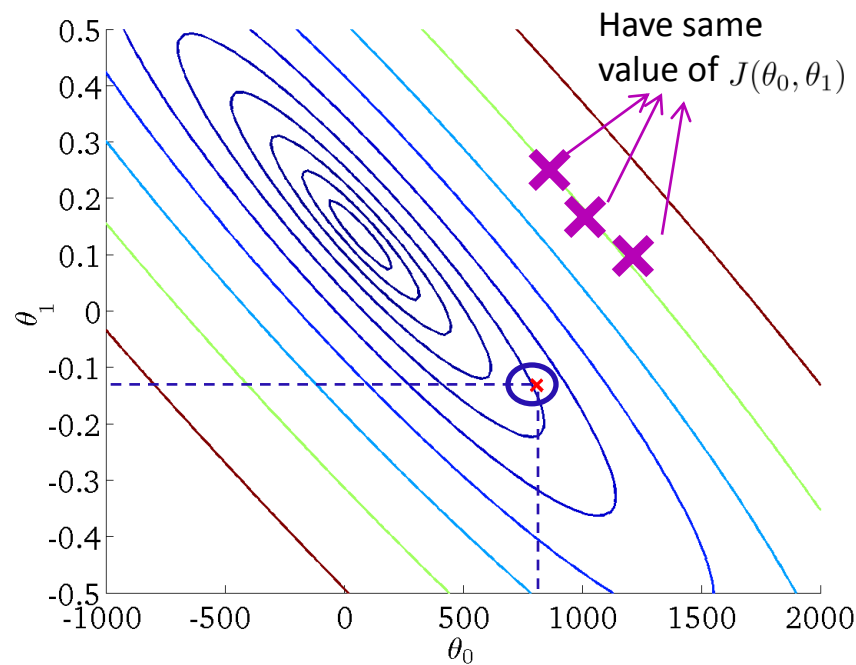
$$h_{\theta}(x)$$

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



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

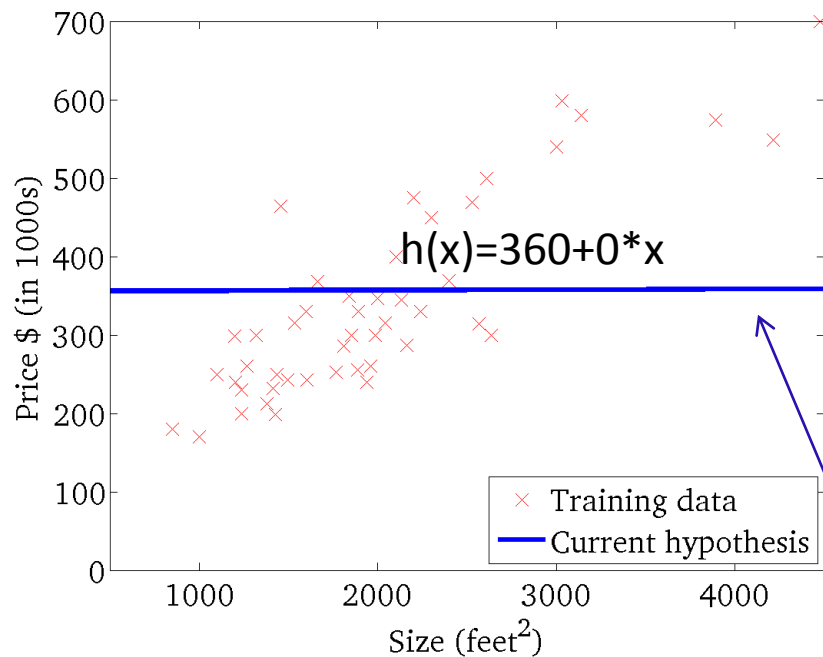
(function of the parameters θ_0, θ_1)



$$\begin{cases} \theta_0 = -0.12 \\ \theta_1 = 800 \end{cases}$$

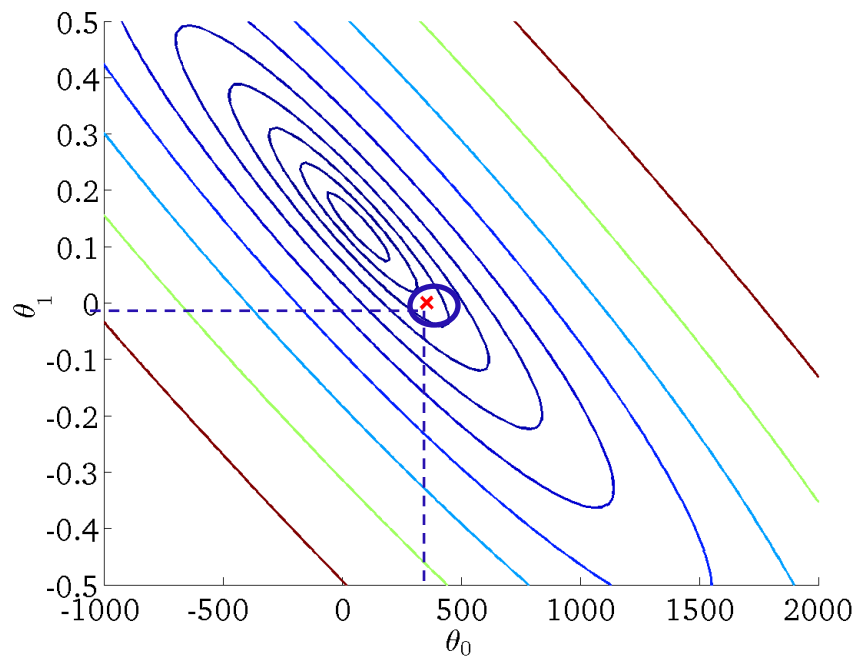
$$h_{\theta}(x)$$

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



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

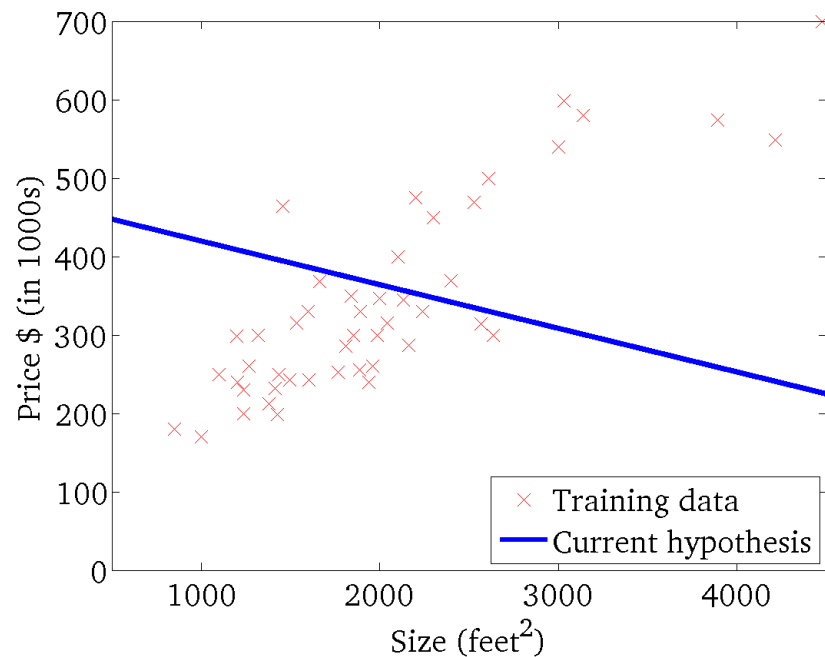
(function of the parameters θ_0, θ_1)



$$\begin{cases} \theta_0 = 360 \\ \theta_1 = 0 \end{cases}$$

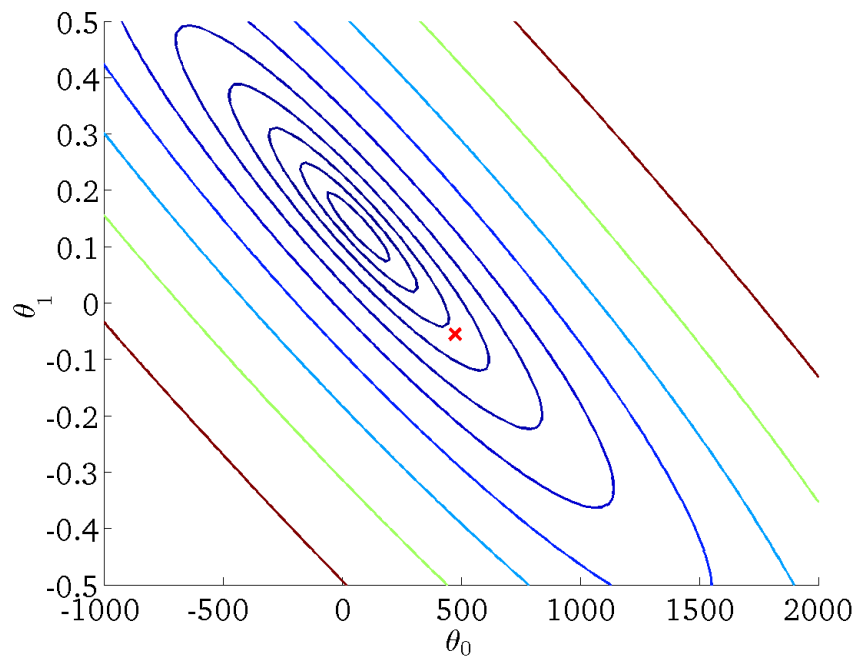
$$h_{\theta}(x)$$

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



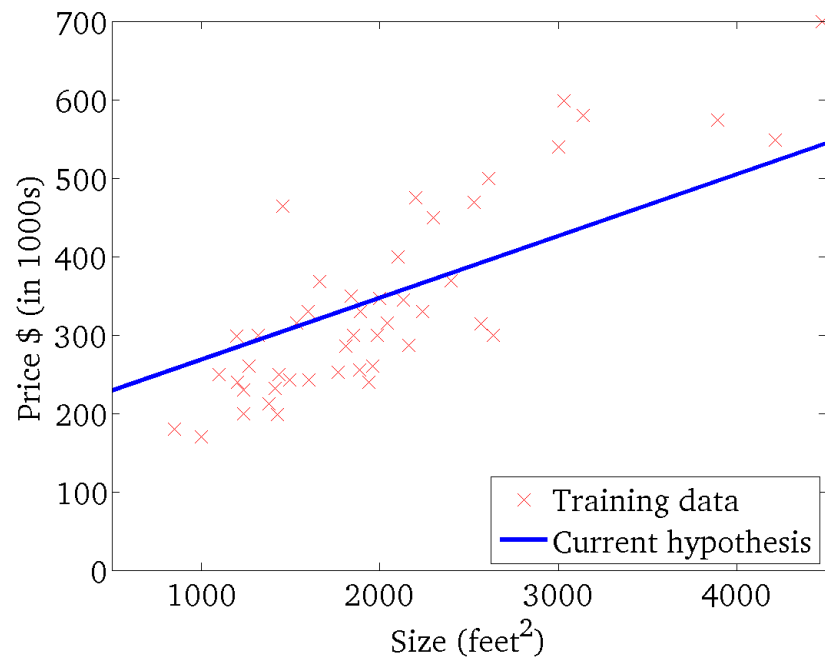
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



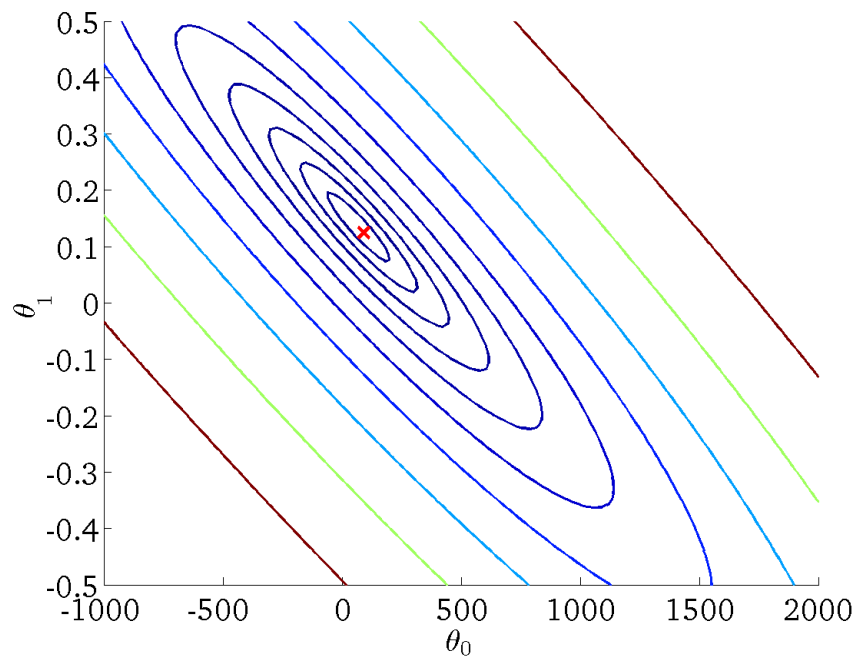
$$h_{\theta}(x)$$

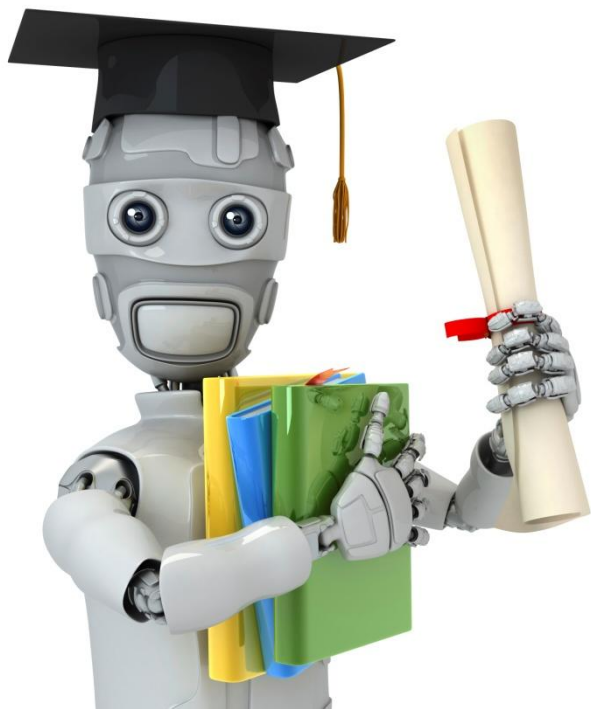
(for fixed θ_0, θ_1 , this is a function of x)



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

(function of the parameters θ_0, θ_1)





Machine Learning

Linear regression
with one variable

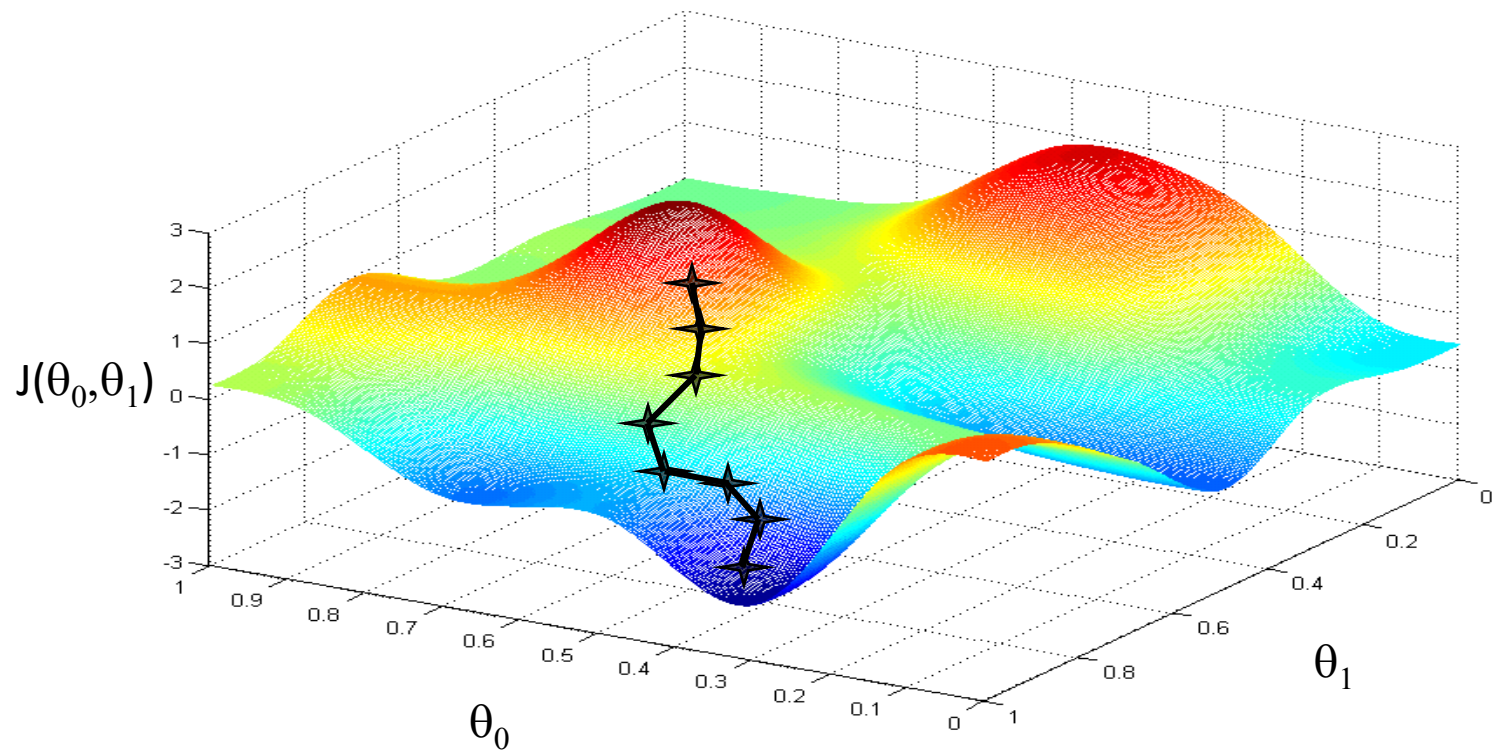
Gradient
descent

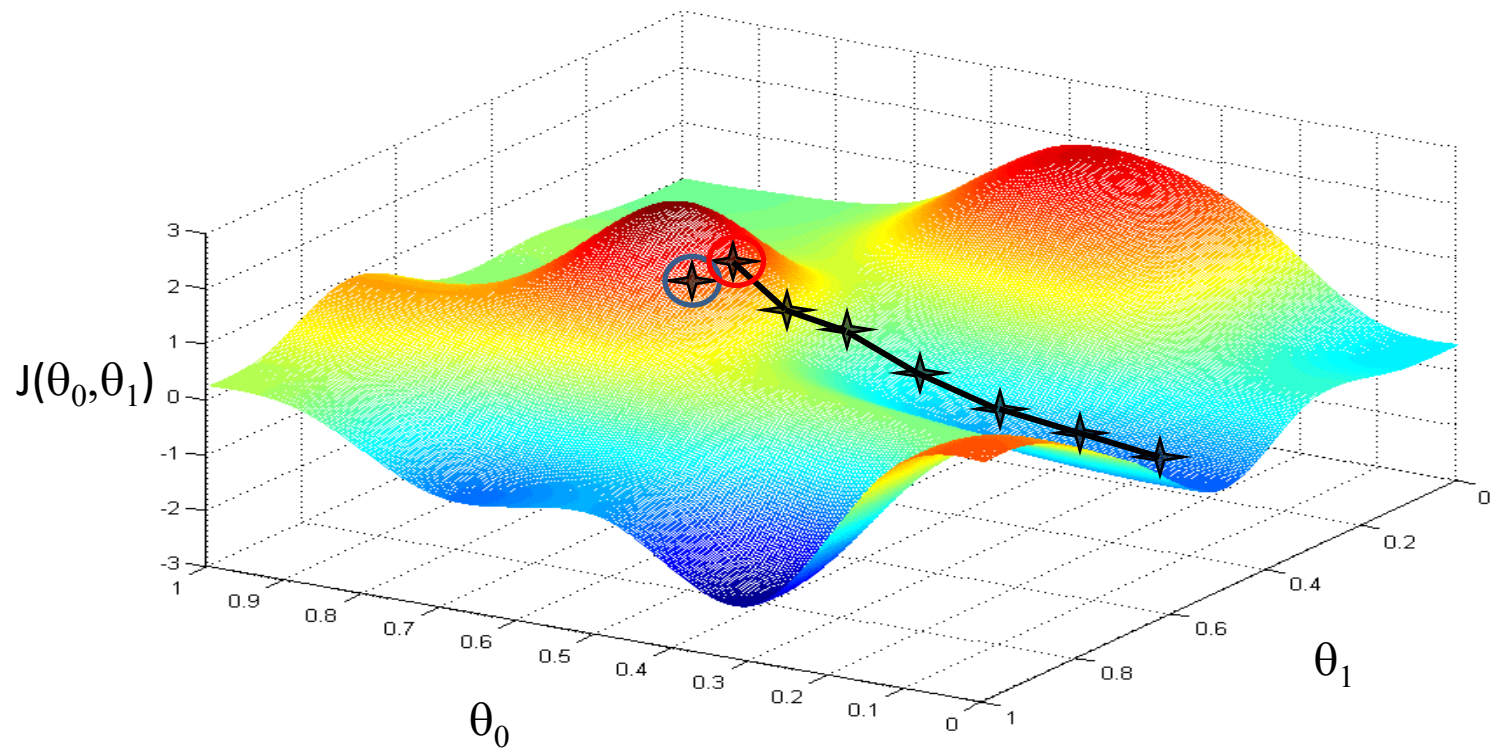
Have some function $J(\theta_0, \theta_1)$ or $J(\theta_0, \theta_2, \theta_2, \dots, \theta_n)$

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

Outline:

- Start with some θ_0, θ_1 e.g. $\theta_0 = 0, \theta_1 = 0$
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$
until we hopefully end up at a minimum





Gradient descent algorithm

repeat until convergence {

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

}

Learning rate

Simultaneously update θ_0 and θ_1

Correct: Simultaneous update

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

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

$$\theta_0 := \text{temp0}$$

$$\theta_1 := \text{temp1}$$

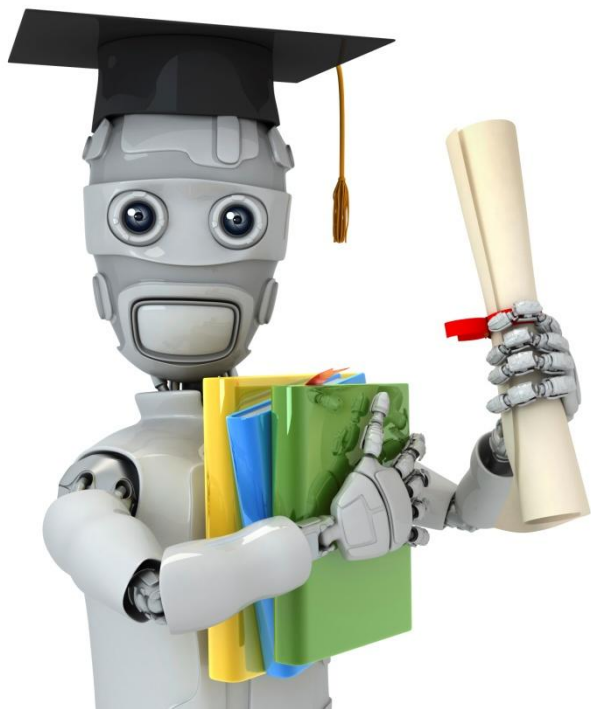
Incorrect:

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

$$\theta_0 := \text{temp0}$$

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

$$\theta_1 := \text{temp1}$$



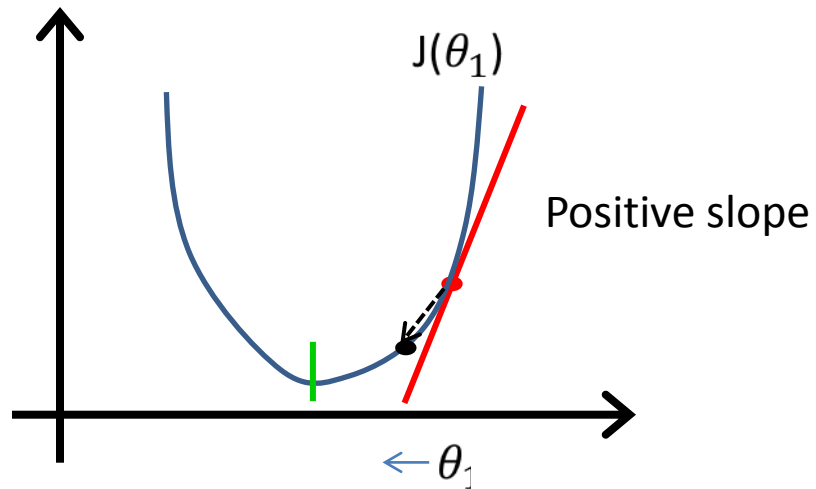
Machine Learning

Linear regression
with one variable

Gradient descent
intuition

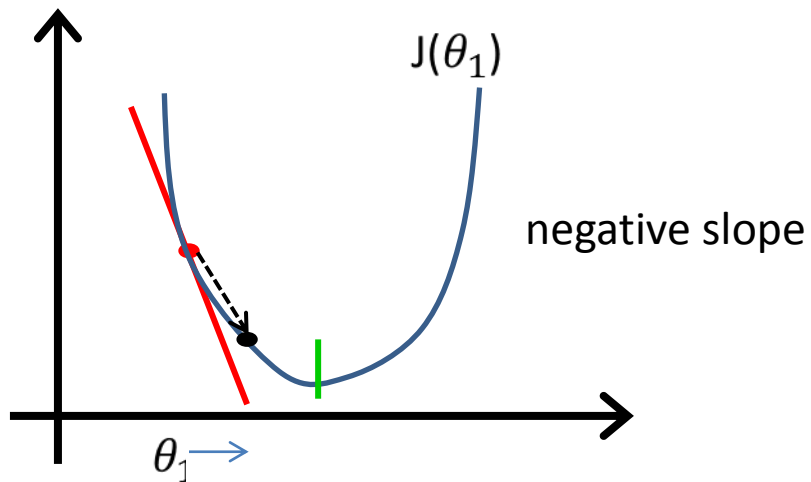
Gradient descent algorithm

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



$$\frac{\partial}{\partial \theta_1} J(\theta_1) > 0$$

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

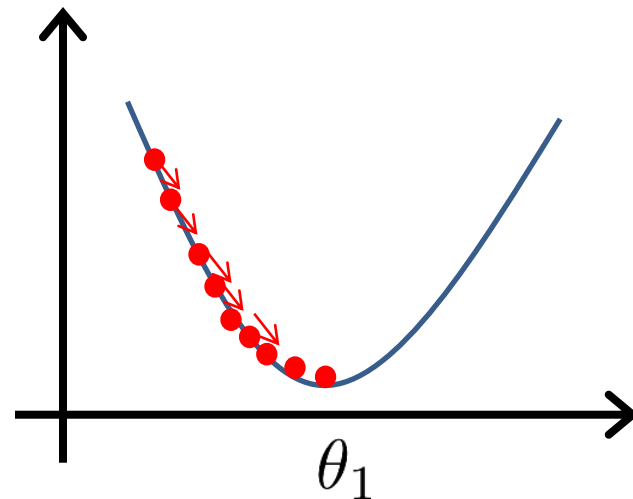


$$\frac{\partial}{\partial \theta_1} J(\theta_1) \leq 0$$

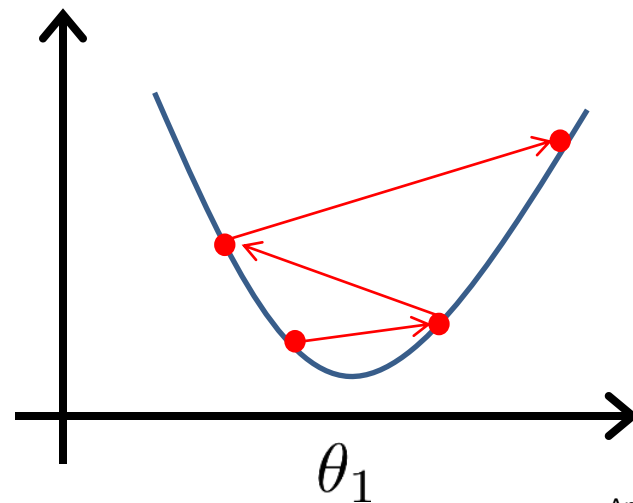
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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

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



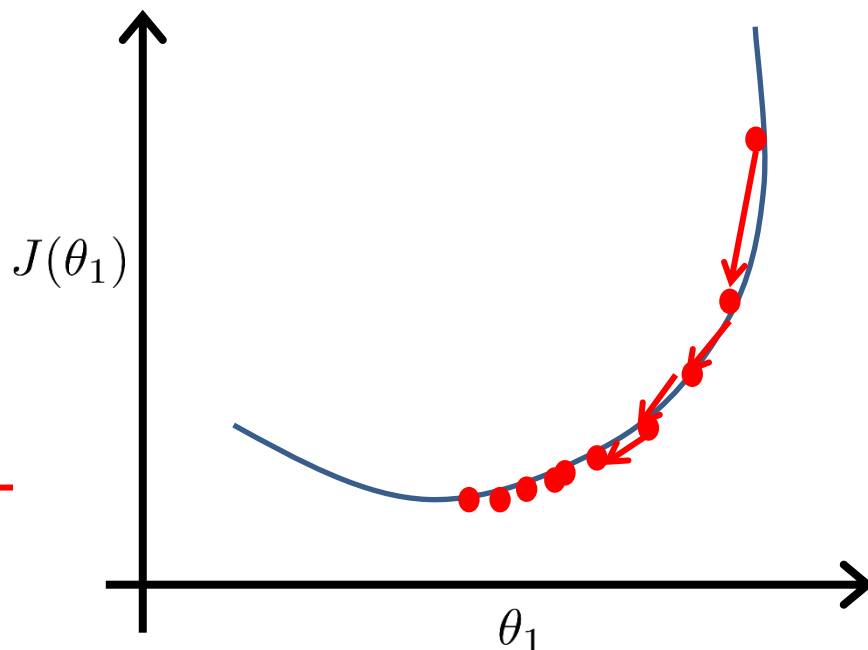
If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.

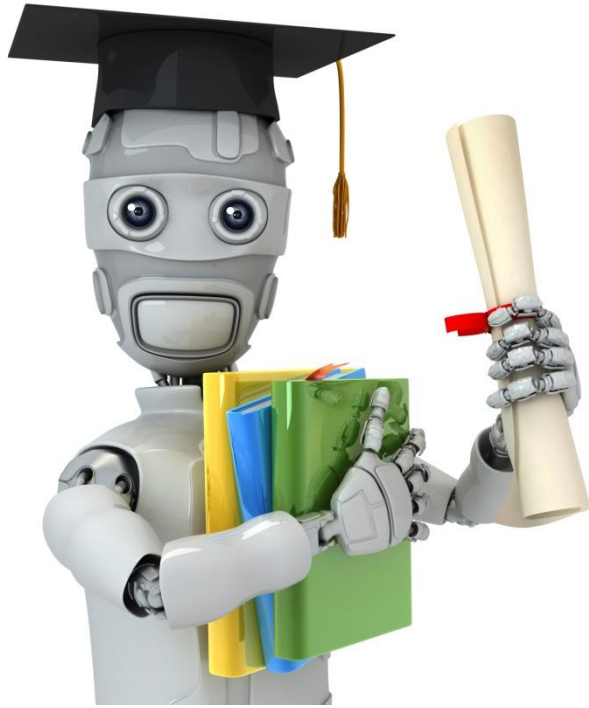


Gradient descent can converge to a local minimum, even with the learning rate α fixed.

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

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





Machine Learning

Linear regression with one variable

Gradient descent for linear regression

Gradient descent algorithm

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

Linear Regression Model

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

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

$$\begin{aligned}\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2m} \sum_{i=1}^m (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2\end{aligned}$$

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

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

Gradient descent algorithm

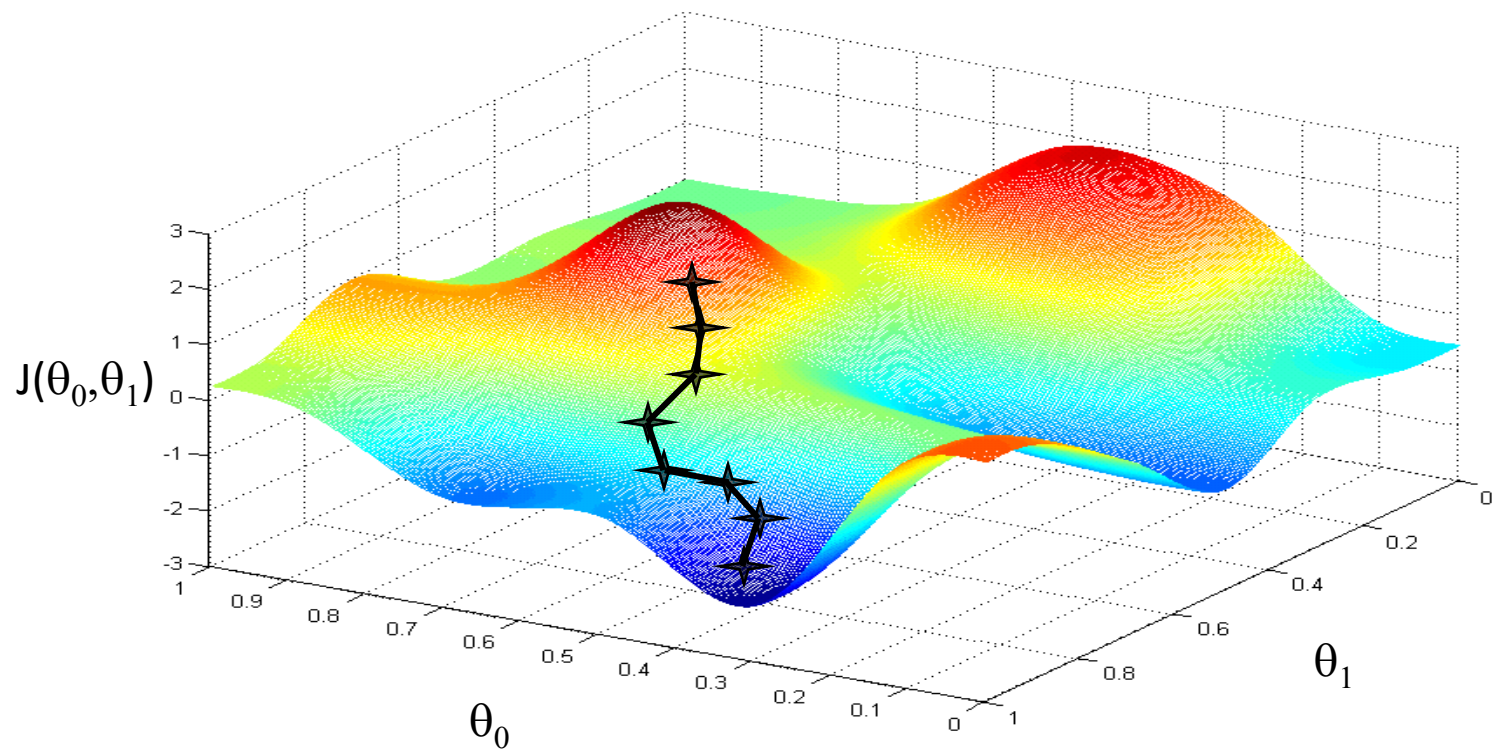
repeat until convergence {

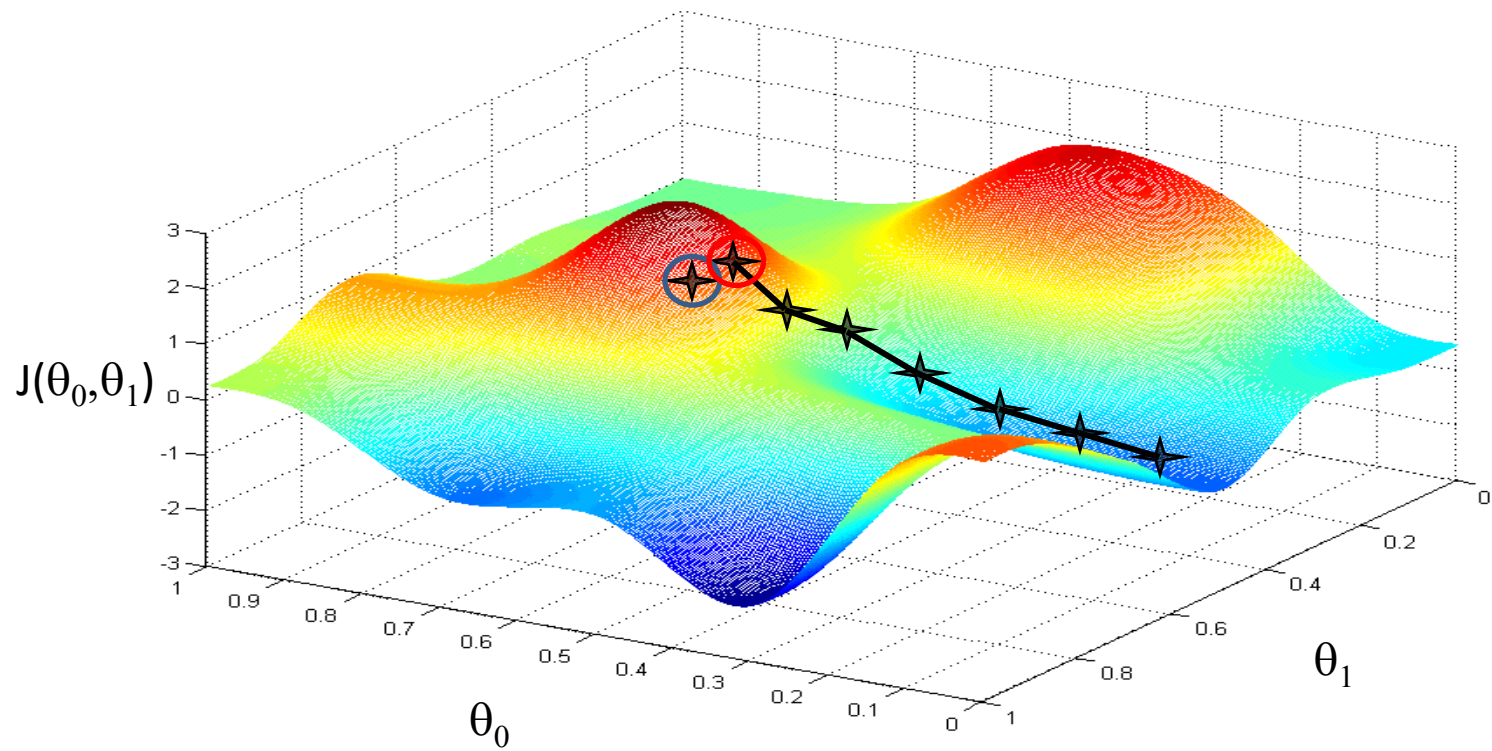
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})$$

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

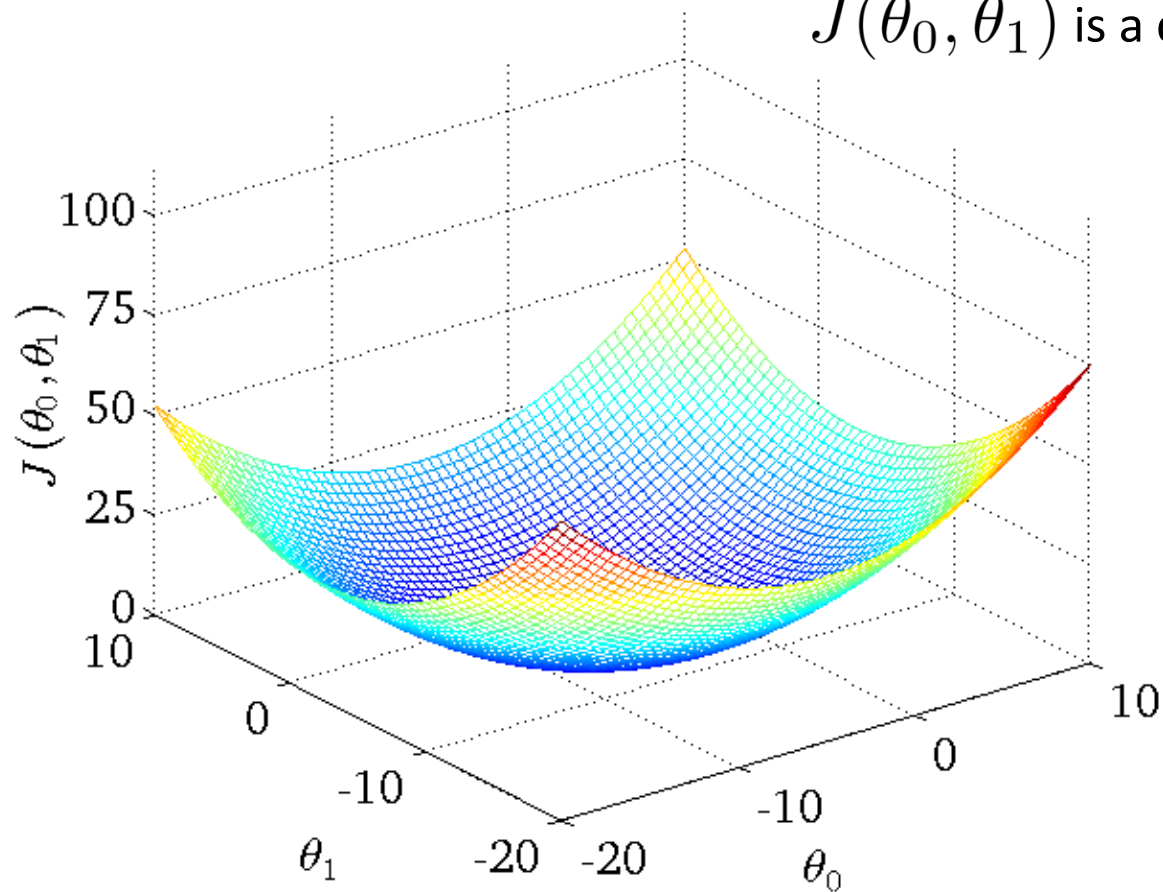
}

update
 θ_0 and θ_1
simultaneously



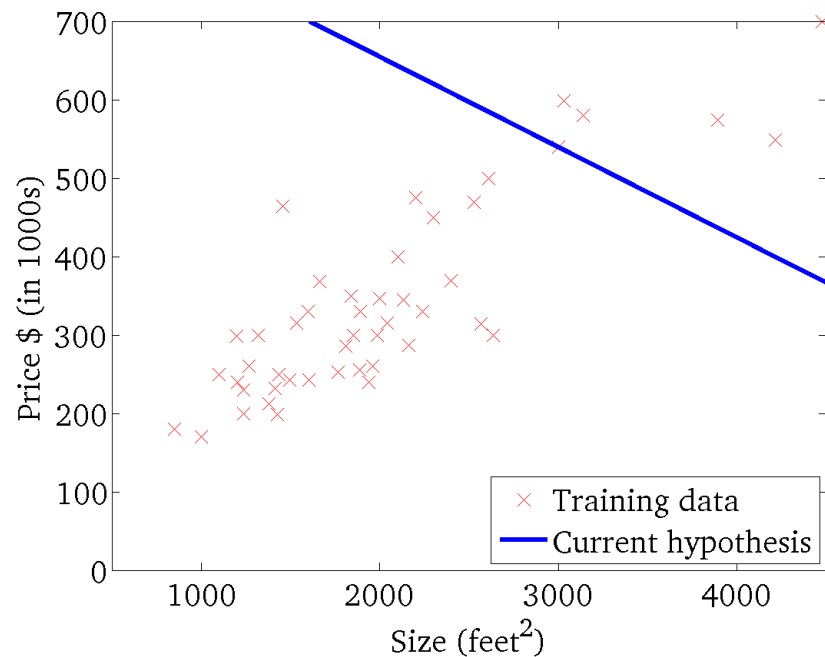


$J(\theta_0, \theta_1)$ is a convex function



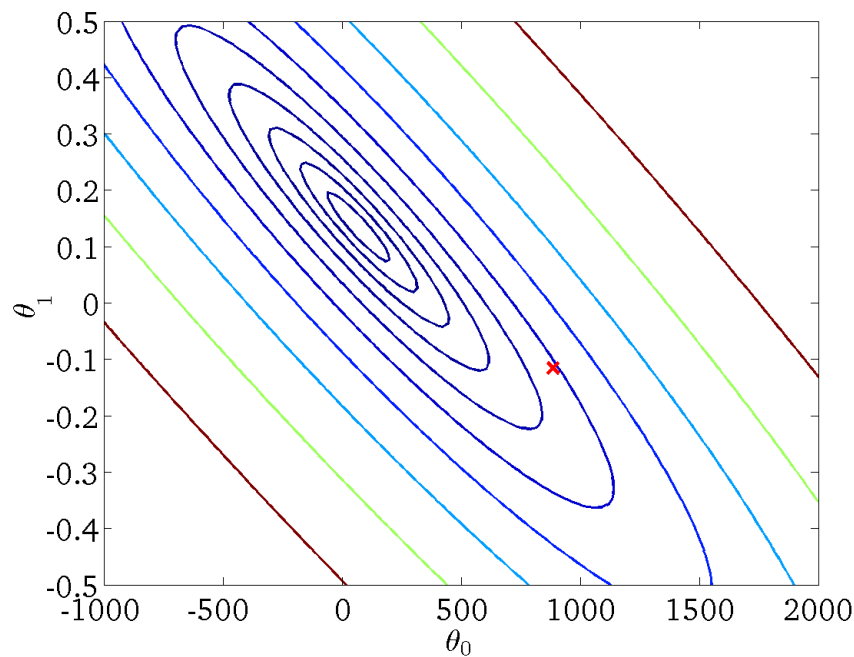
$$h_{\theta}(x)$$

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



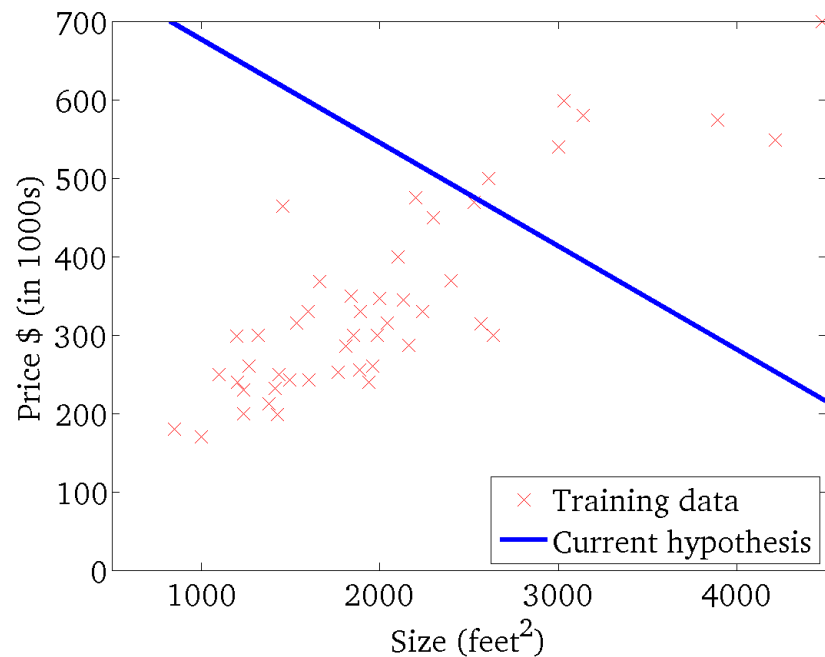
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



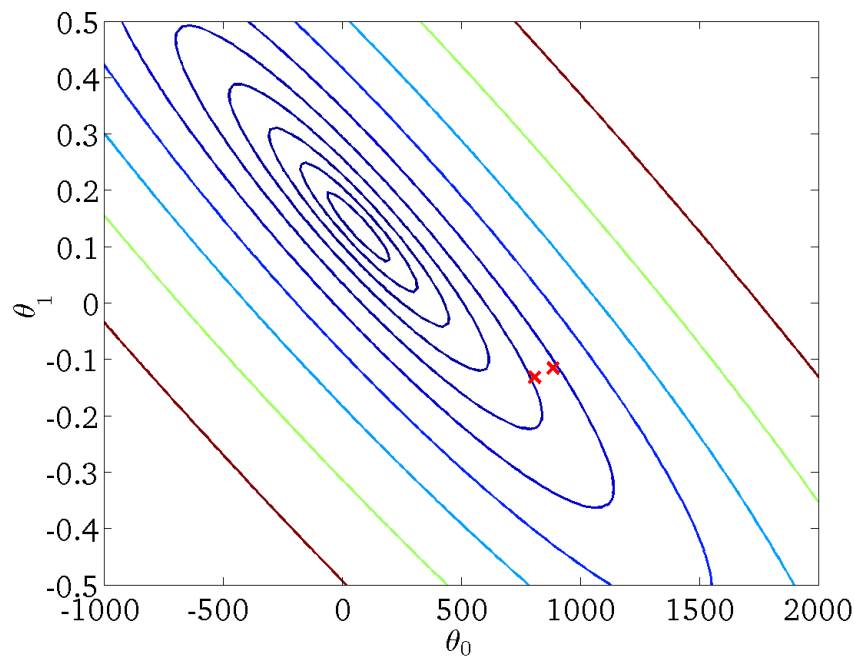
$$h_{\theta}(x)$$

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



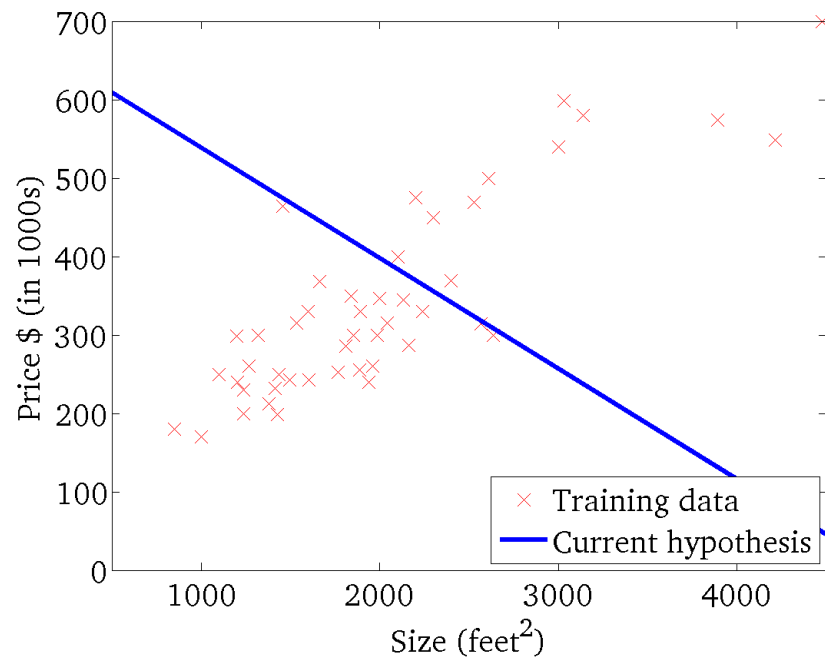
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



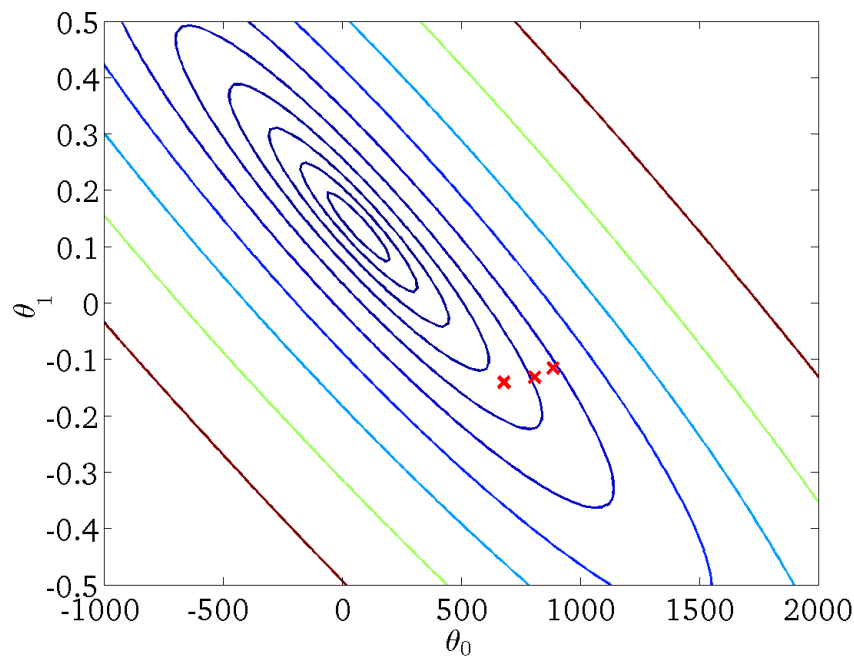
$$h_{\theta}(x)$$

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



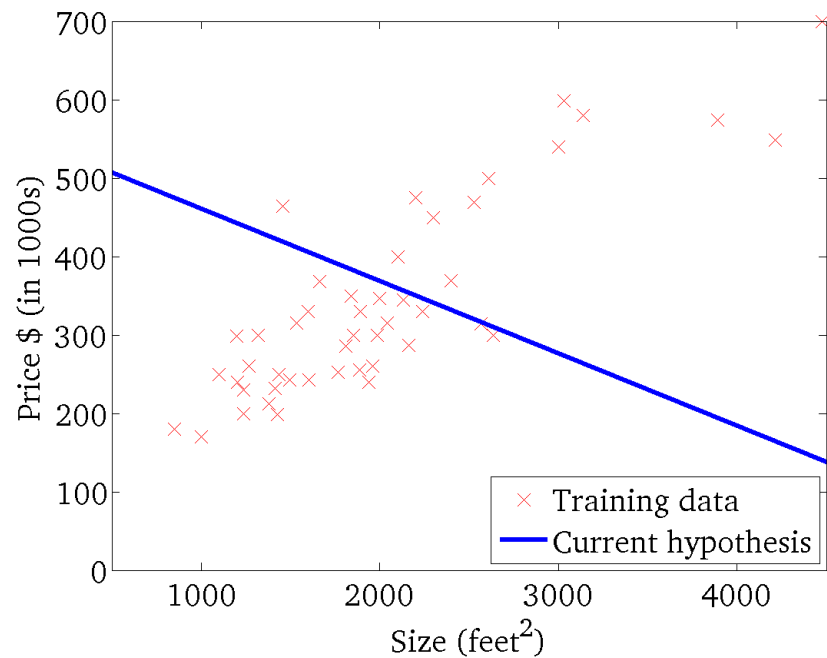
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



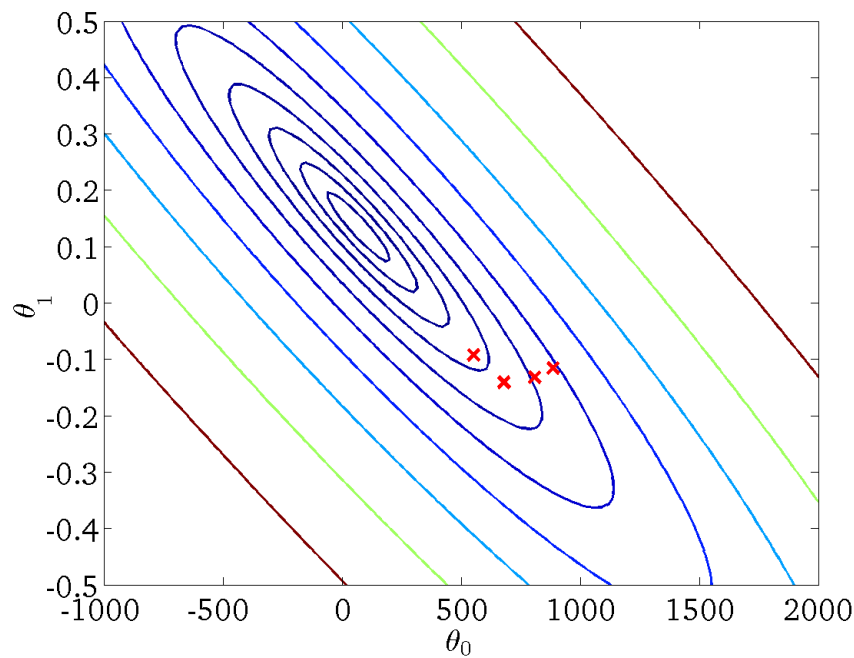
$$h_{\theta}(x)$$

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



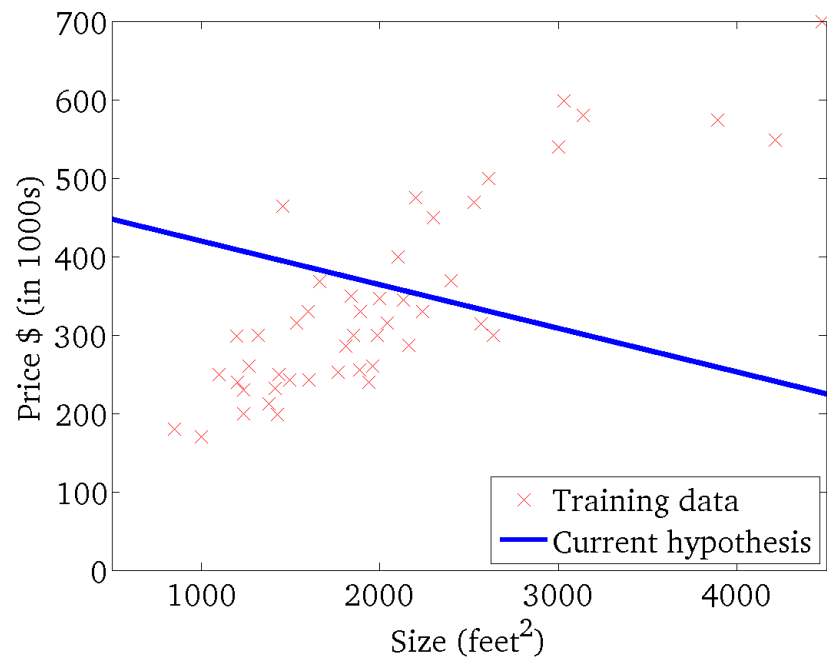
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



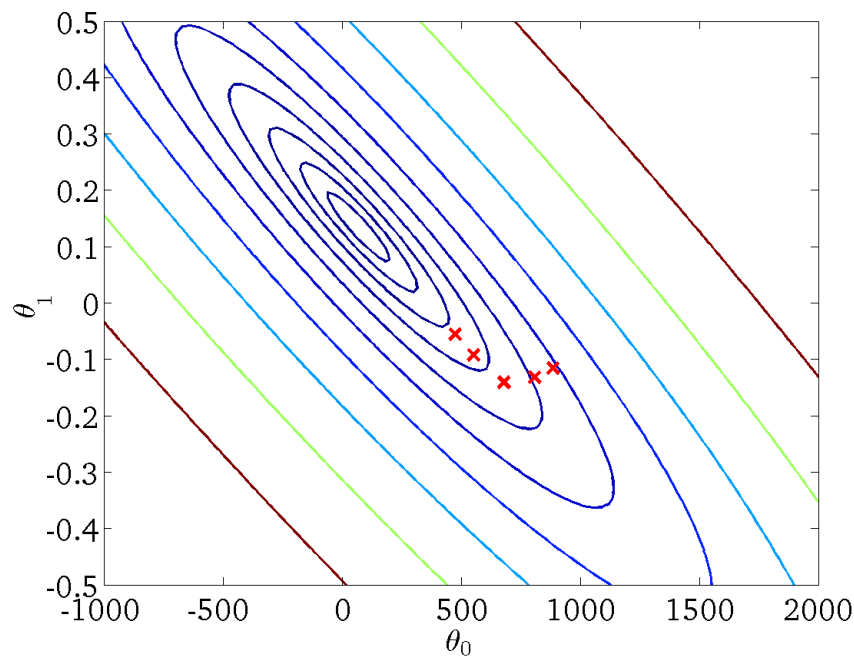
$$h_{\theta}(x)$$

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



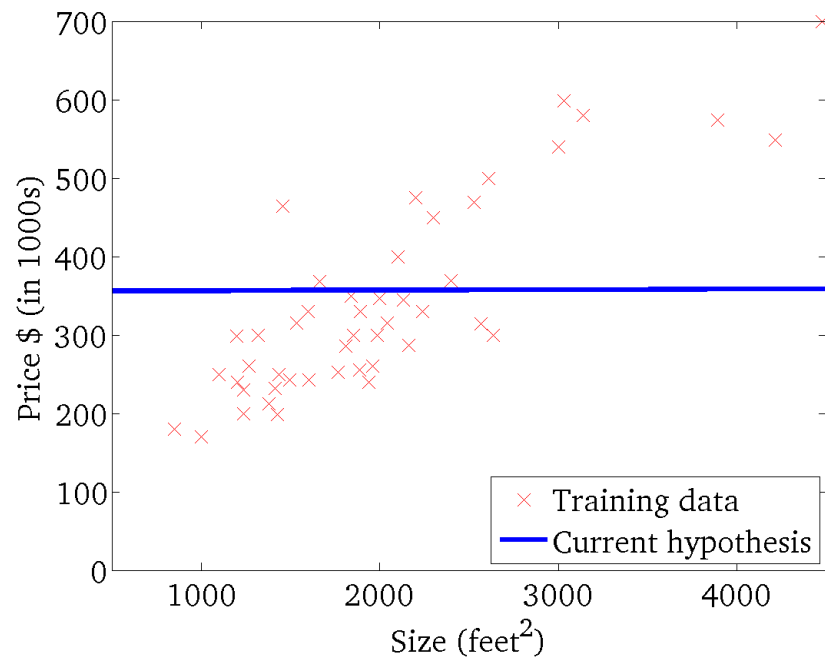
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



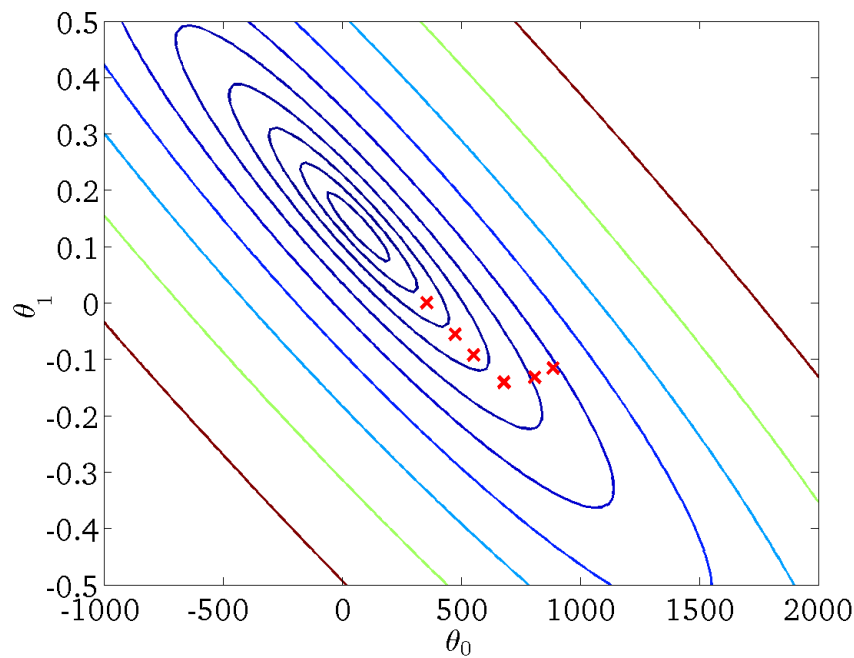
$$h_{\theta}(x)$$

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



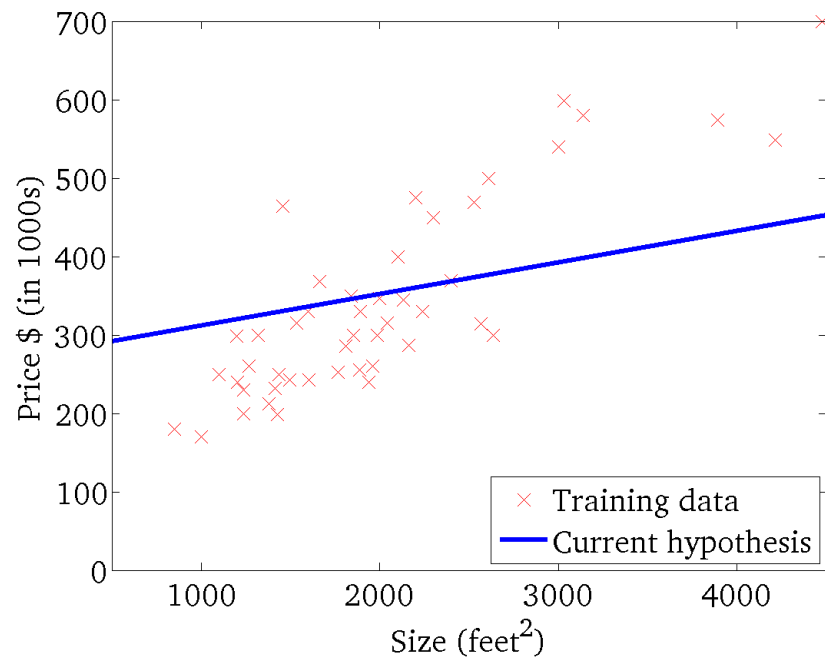
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



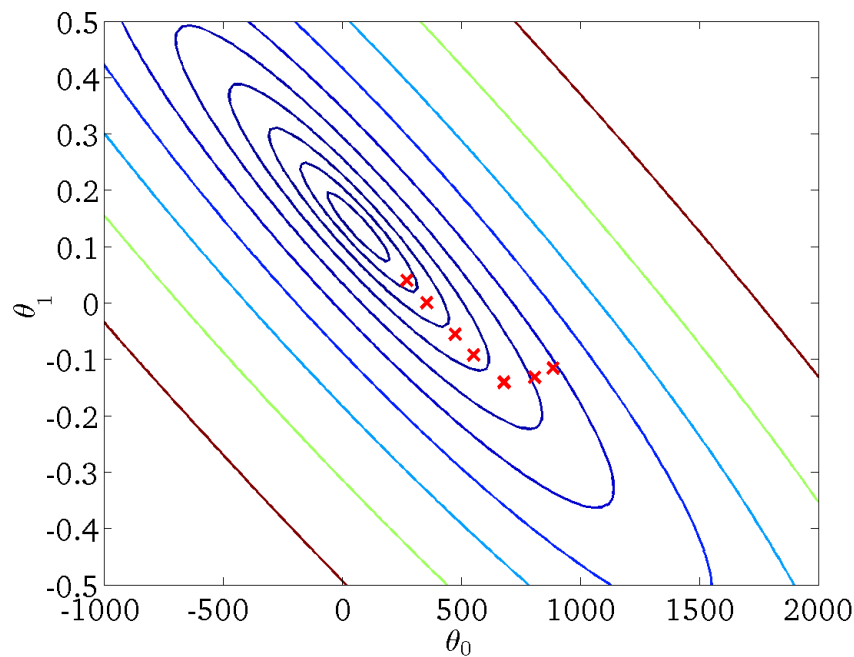
$$h_{\theta}(x)$$

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



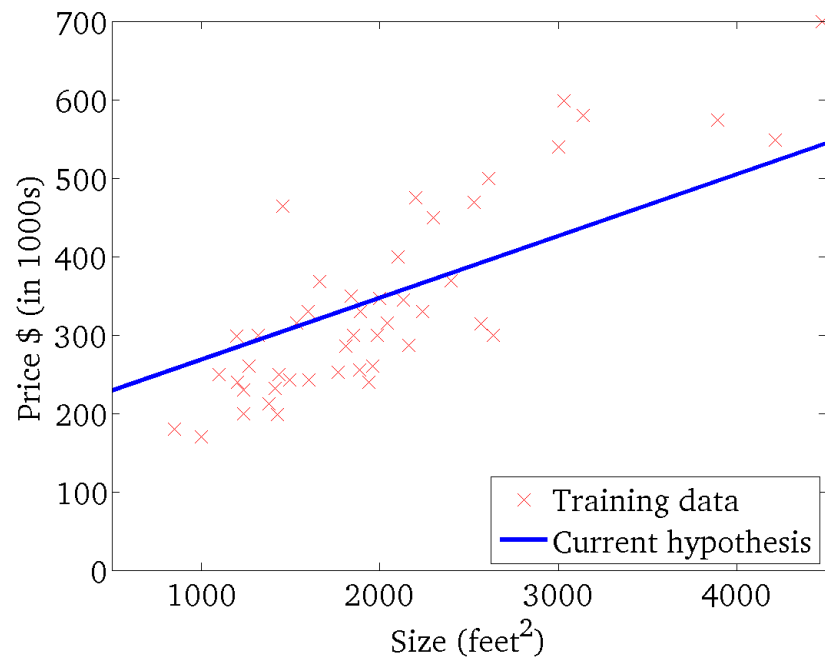
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



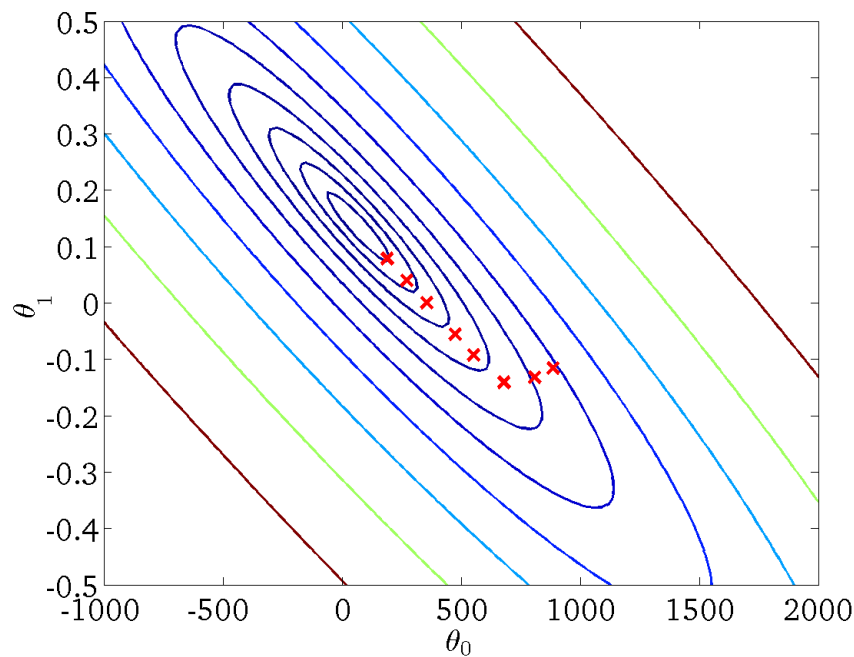
$$h_{\theta}(x)$$

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



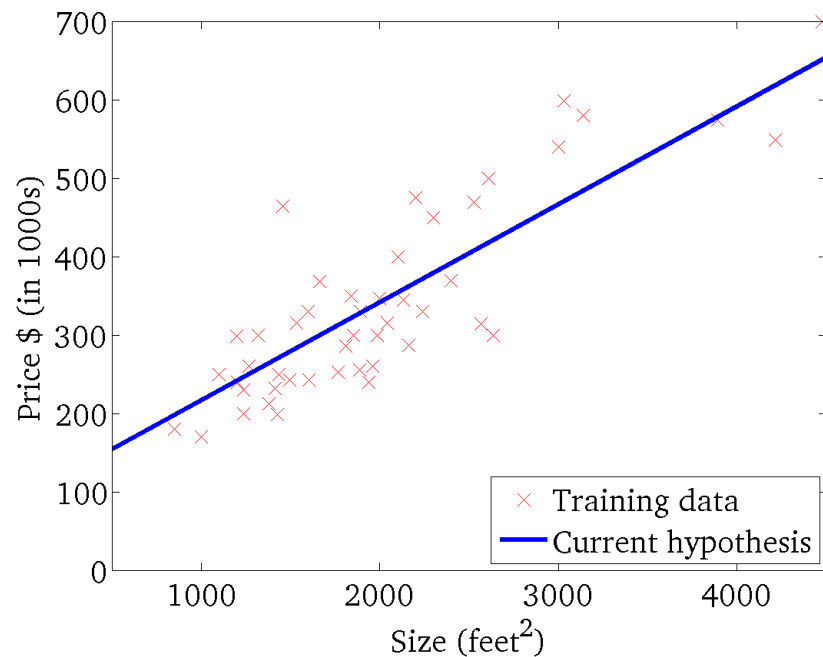
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



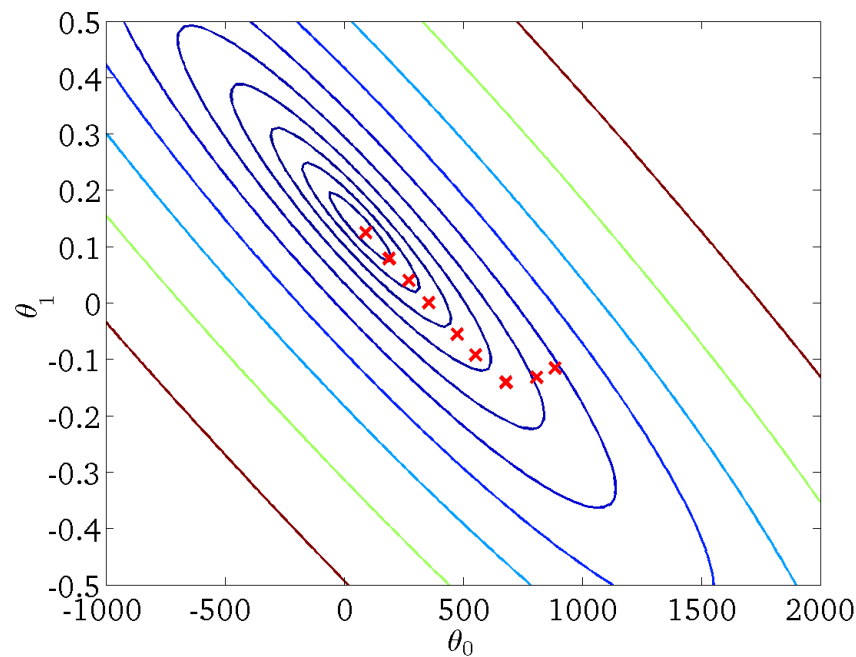
$$h_{\theta}(x)$$

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



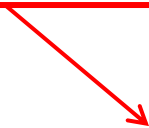
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



“Batch” Gradient Descent

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


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