

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

Classification: Discrete-valued output

# Training set of housing prices (Portland, OR)

Size in feet <sup>2</sup> (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

$(x^{(i)}, y^{(i)})$  - i<sup>th</sup> training example

$$\left\{ \begin{array}{l} x^{(1)} = 2104 \\ x^{(2)} = 1416 \\ y^{(1)} = 460 \end{array} \right.$$

Training Set

Learning Algorithm

Size of house  
x

h

Estimated price  
(estimated value of y)

hypothesis

h maps from x's to y's.

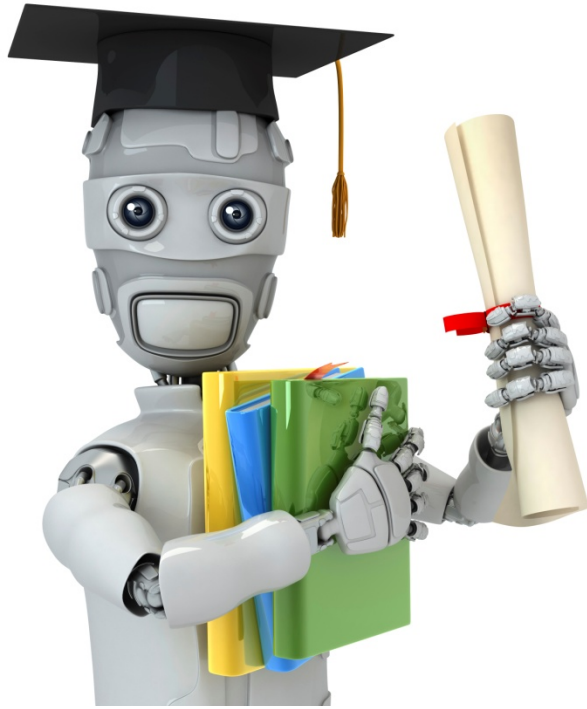
How do we represent  $h$ ?

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

Shorthand:  $h(x)$



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



Machine Learning

# Linear regression with one variable

---

## Cost function

Training Set

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

}  $m = 47$

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

$\theta_i$ 's: Parameters

↑      ↑

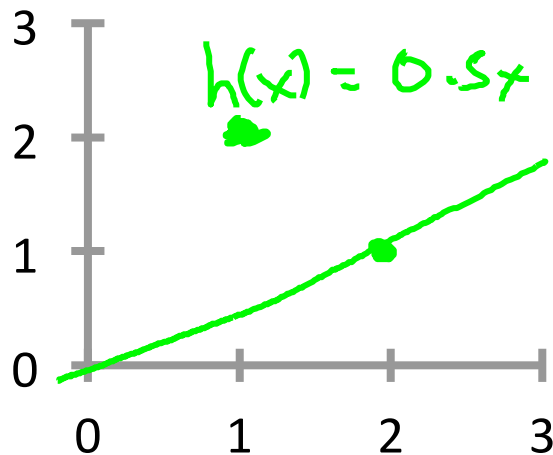
How to choose  $\theta_i$ 's ?

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



$$\rightarrow \theta_0 = 1.5$$

$$\rightarrow \theta_1 = 0$$



$$\rightarrow \theta_0 = 0$$

$$\rightarrow \theta_1 = 0.5$$



$$\rightarrow \theta_0 = 1$$

$$\rightarrow \theta_1 = 0.5$$



Idea: Choose  $\underline{\theta_0}, \underline{\theta_1}$  so that  $\underline{h_\theta(x)}$  is close to  $\underline{y}$  for our training examples  $\underline{(x, y)}$

$$\begin{array}{l} \boxed{\text{minimize } \underline{\theta_0, \theta_1}} \quad \frac{1}{2m} \sum_{i=1}^m \underbrace{\left( \underbrace{h_\theta(x^{(i)})}_{h_\theta(x^{(i)}) = \underline{\theta_0} + \underline{\theta_1} x^{(i)}} - \underline{y^{(i)}} \right)^2}_{\text{\#training examples}} \end{array}$$

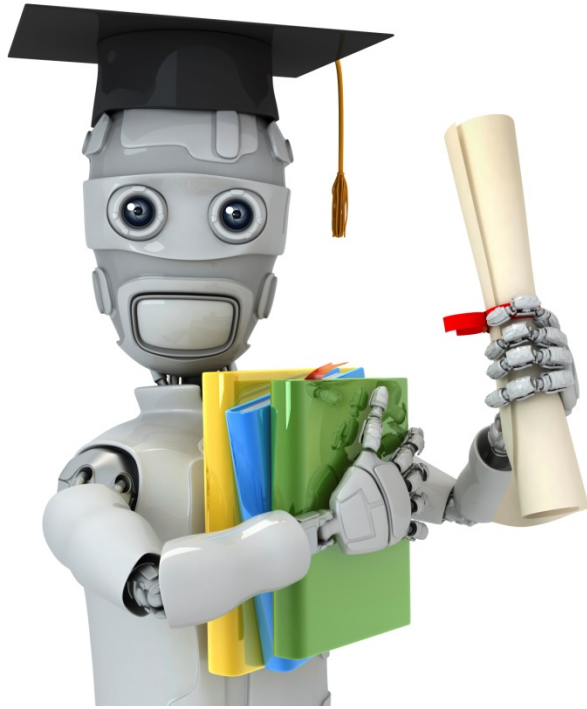
$$J(\underline{\theta_0}, \underline{\theta_1}) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\begin{array}{l} \text{minimize } J(\underline{\theta_0}, \underline{\theta_1}) \\ \underline{\theta_0, \theta_1} \end{array}$$

Cost function

Squared error function





Machine Learning

Linear regression  
with one variable

---

Cost function  
intuition I

Hypothesis:

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

Parameters:

$$\underline{\theta_0, \theta_1}$$



Cost Function:

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

Goal: minimize  $J(\theta_0, \theta_1)$   
 $\nearrow \theta_0, \theta_1$

Simplified

$$h_{\theta}(x) = \underline{\theta_1 x}$$

$$\theta_0 = 0$$

$$\underline{\theta_1}$$



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

minimize  $J(\theta_1)$   
 $\theta_1$

$$\theta, x^{(i)}$$

→  $h_{\theta}(x)$

(for fixed  $\theta_1$ , this is a function of  $x$ )



$J(\theta_1)$

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

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

→  $J(\theta_1)$

(function of the parameter  $\theta_1$ )

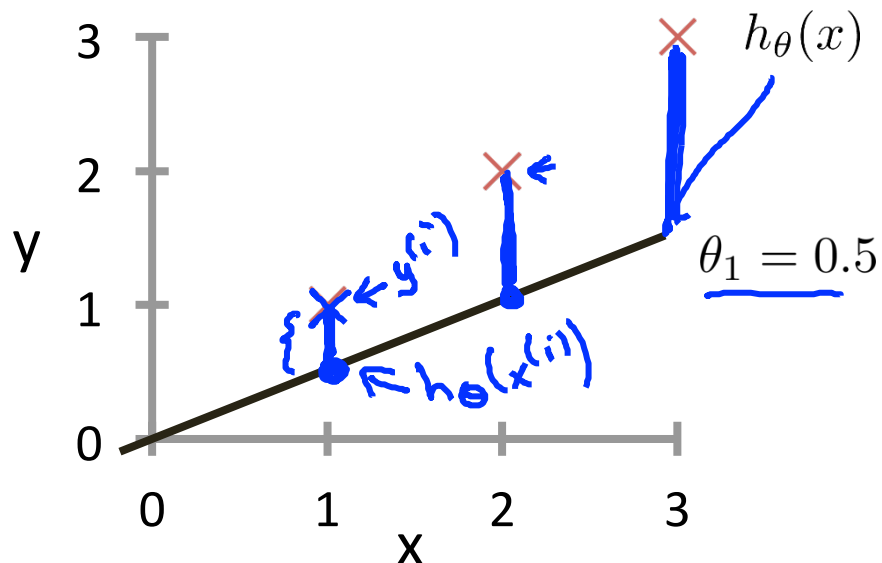


$\theta_1 = 0.5?$

$J(1) = 0$

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

(for fixed  $\theta_1$ , this is a function of  $x$ )



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

$$= \frac{1}{2 \times 3} (3.5) = \frac{3.5}{6} \approx \underline{0.58}$$

$$J(\theta_1)$$

(function of the parameter  $\theta_1$ )

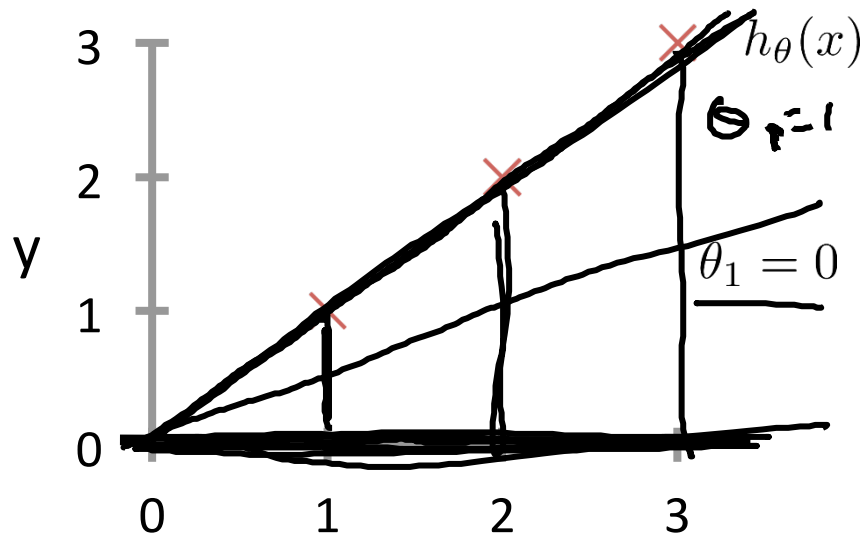


$$\theta_1 = 0?$$

$$J(0) = ?$$

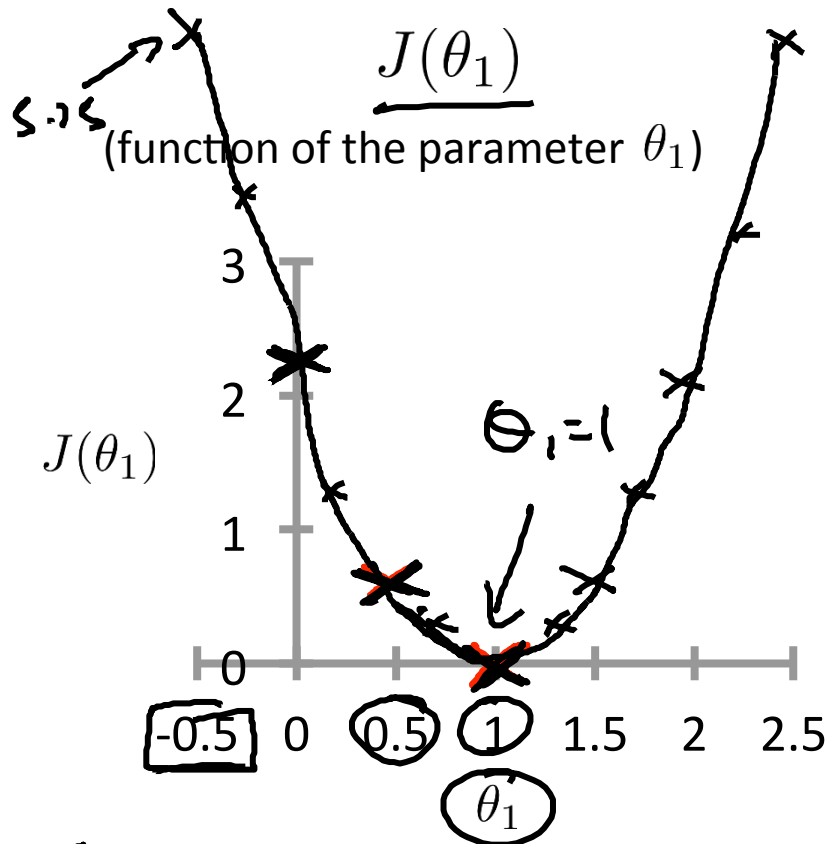
$$h_{\theta}(x)$$

(for fixed  $\theta_1$ , this is a function of  $x$ )



$$J(0) = \frac{1}{2m} (1^2 + 2^2 + 3^2)$$

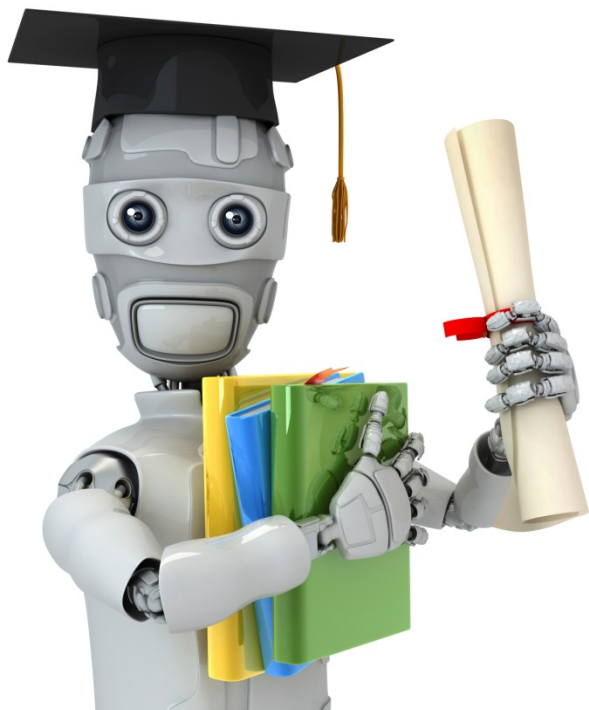
$$= \frac{1}{6} \cdot 14 \approx 2.3$$



$$h(x) = -0.5x$$

minimize  $J(\theta_1)$

$h(x) \theta_1$



Machine Learning

Linear regression  
with one variable

---

Cost function  
intuition II

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

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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

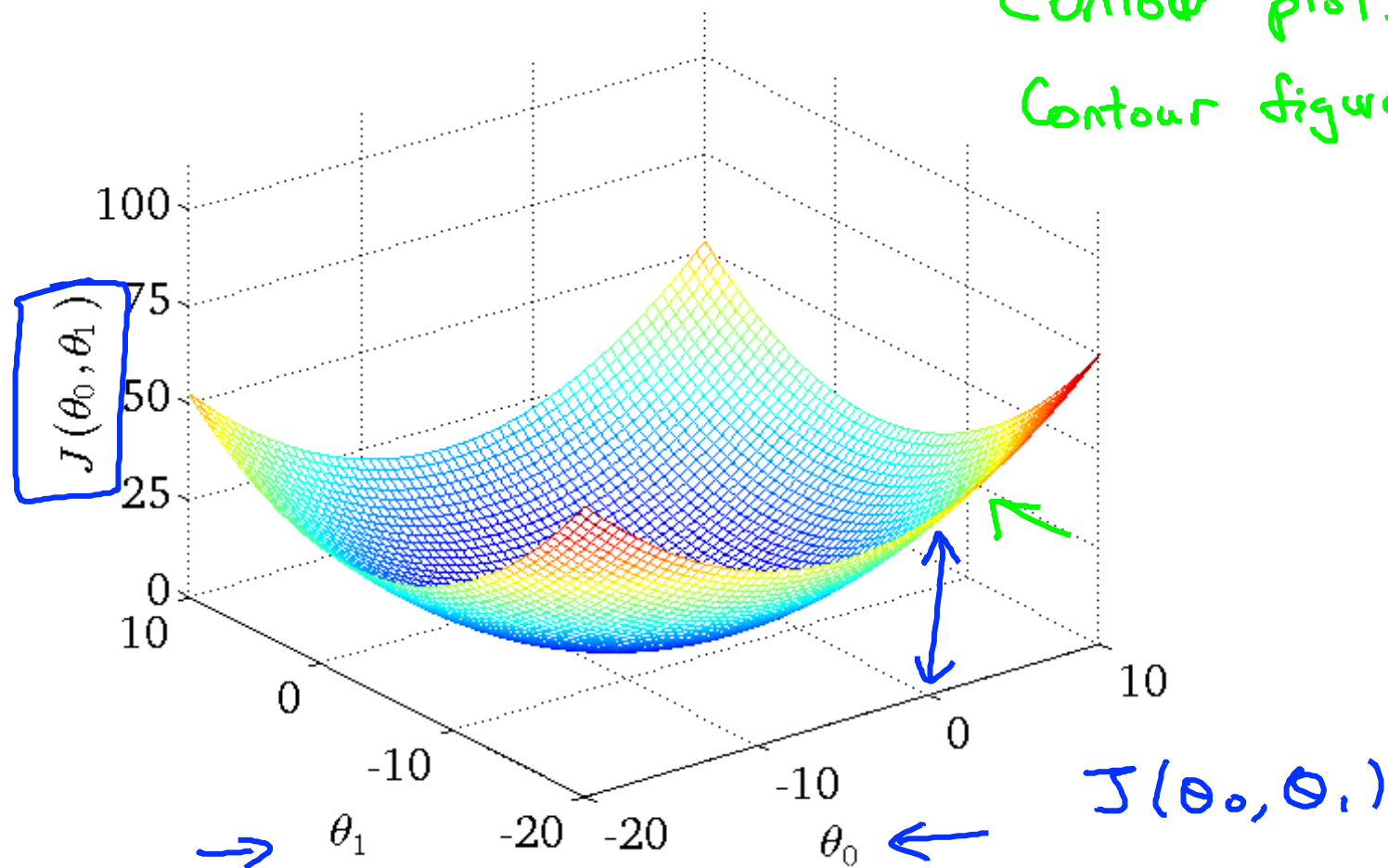
$$\underline{\underline{J(\theta_0, \theta_1)}}$$

(function of the parameters  $\theta_0, \theta_1$ )





Contour plots  
Contour figures -



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

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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



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

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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



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

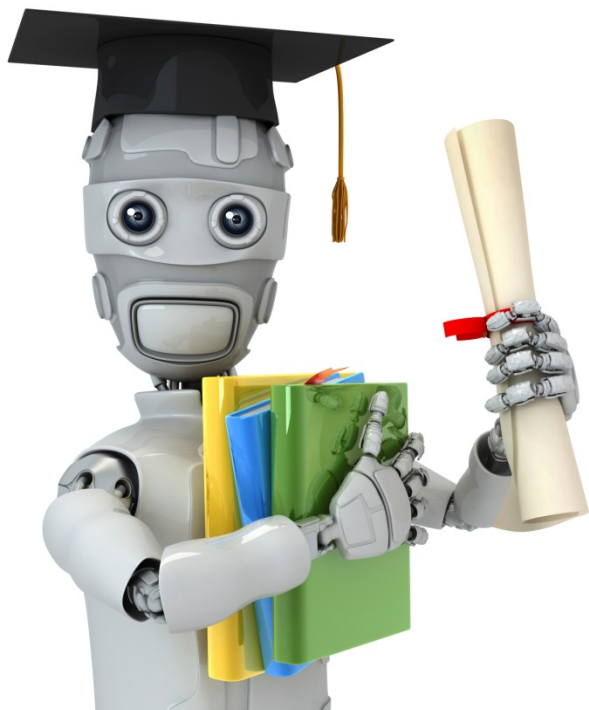
(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )





Machine Learning

Linear regression  
with one variable

---

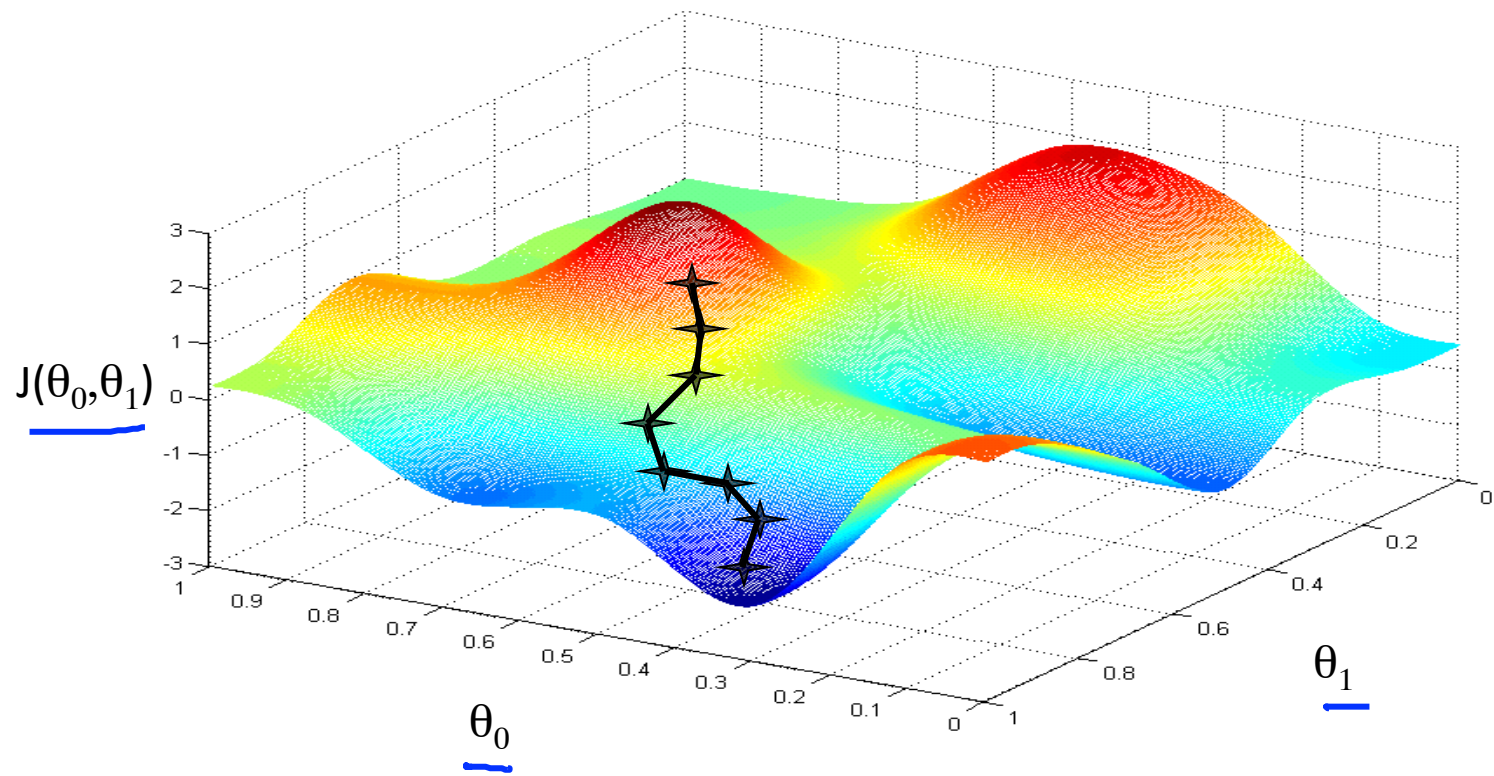
Gradient  
descent

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

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

## Outline:

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







# Gradient descent algorithm

$\theta_0, \theta_1$

repeat until convergence {

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

learning rate

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

Simultaneously update  
 $\theta_0$  and  $\theta_1$

Assignment

$$\begin{aligned} & \rightarrow a := b \\ & \quad \uparrow \\ & \quad a := a + 1 \end{aligned}$$

Truth assertion

$$a = b \leftarrow$$

$$a = a + 1 \times$$

Correct: Simultaneous update

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

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

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

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

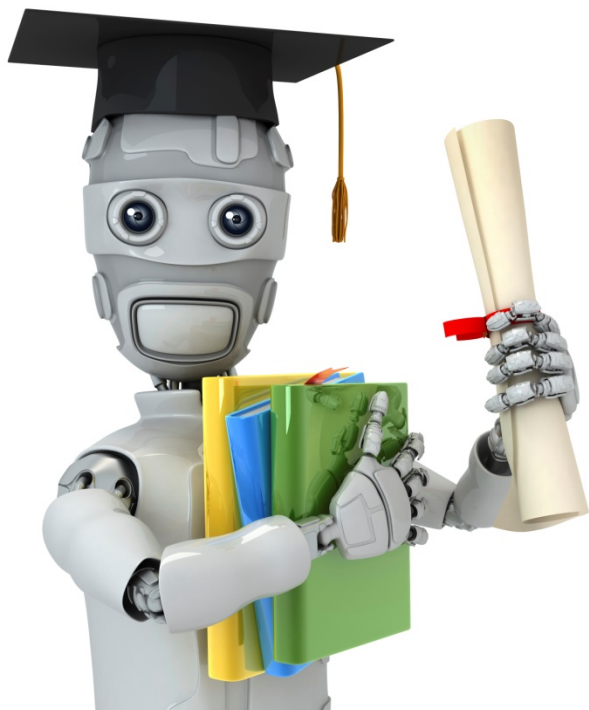
Incorrect:

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

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

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

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



Machine Learning

Linear regression  
with one variable

---

Gradient descent  
intuition

# Gradient descent algorithm

repeat until convergence {

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

}

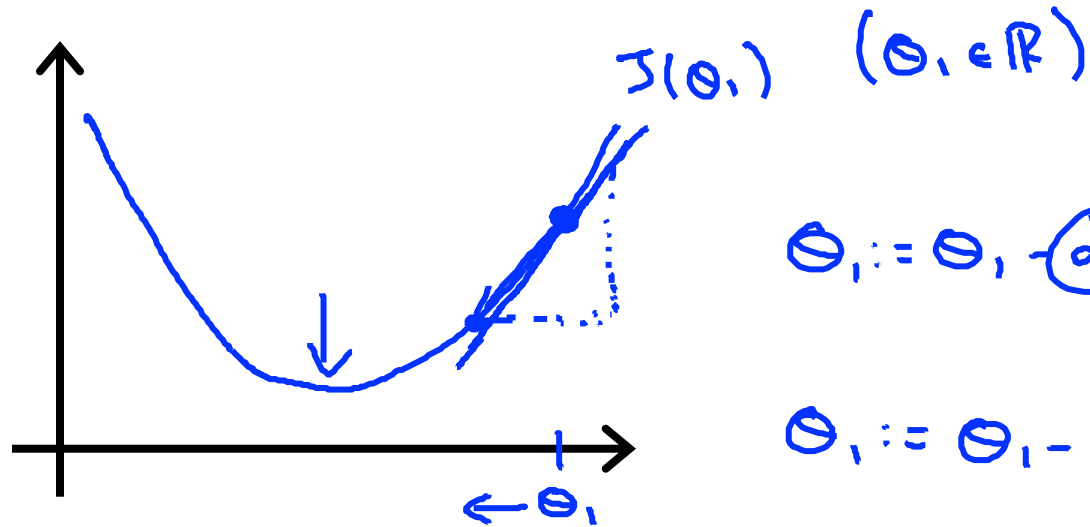
learning  
rate

derivative

(simultaneously update  
 $j = 0$  and  $j = 1$ )

$$\min_{\theta_1} J(\theta_1)$$

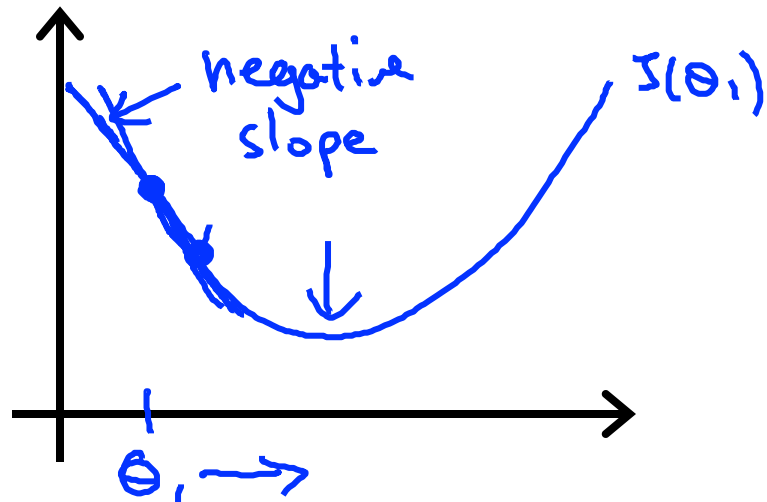
$$\theta_1 \in \mathbb{R}.$$



$$\theta_1 := \theta_1 - \alpha \left( \frac{\partial}{\partial \theta_1} J(\theta_1) \right) \geq 0$$

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

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



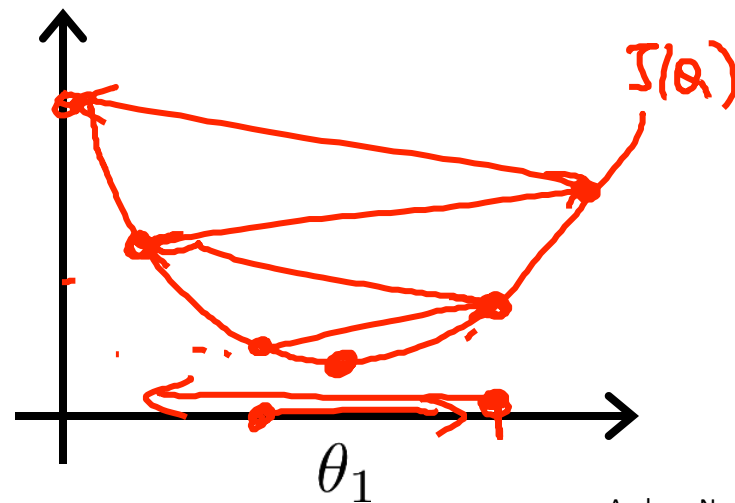
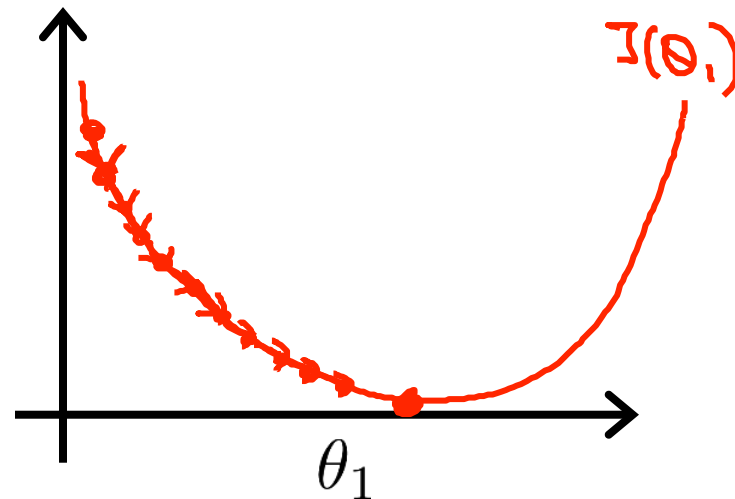
$$\frac{\partial}{\partial \theta_1} J(\theta_1) \leq 0$$

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

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

If  $\alpha$  is too small, gradient descent can be slow.

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





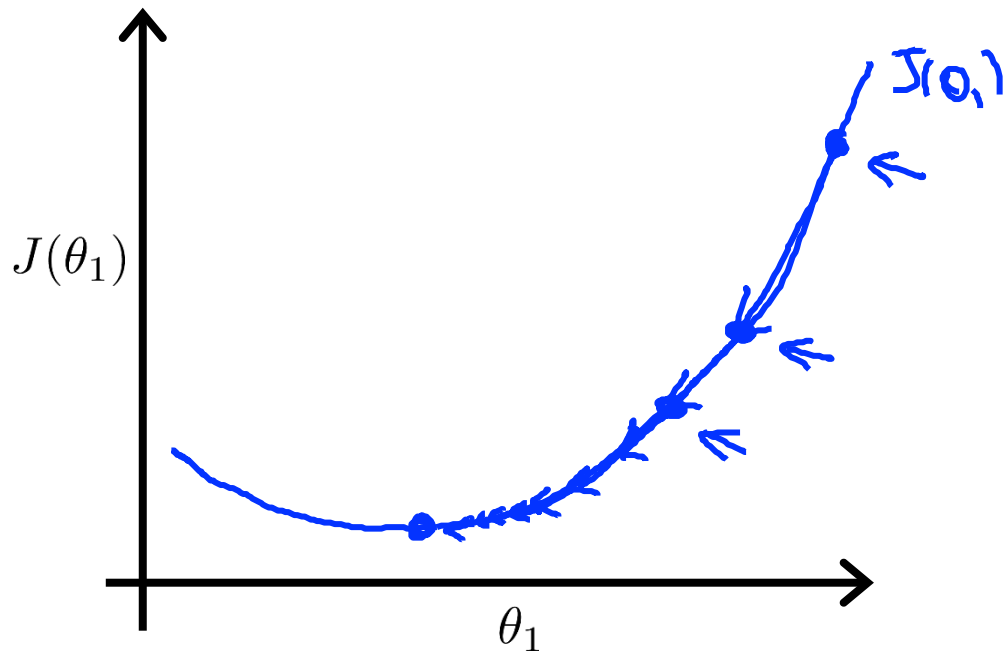
Current value of  $\theta_1$

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

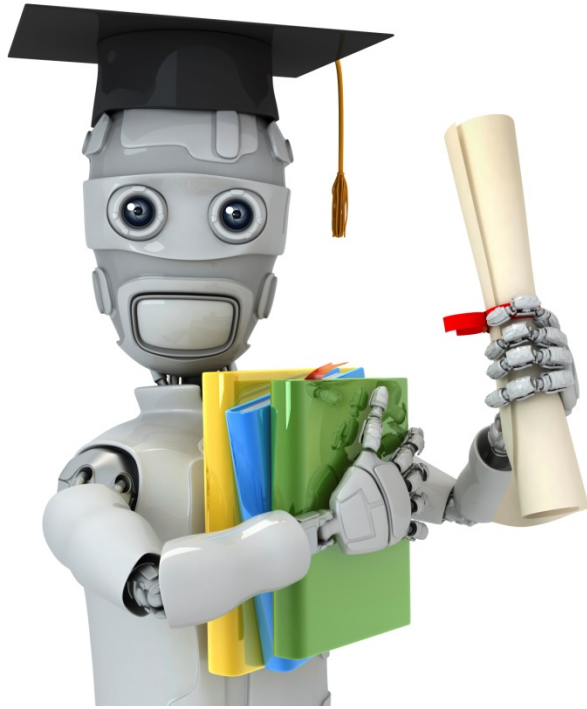
Gradient descent can converge to a local minimum, even with the learning rate  $\alpha$  fixed.

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

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease  $\alpha$  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

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

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

$$\frac{\partial}{\partial \theta_j} \underline{J(\theta_0, \theta_1)} = \frac{2}{2\theta_j} \frac{1}{2m} \sum_{i=1}^m \underline{(h_0(x^{(i)}) - y^{(i)})^2}$$

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

$$j = 0 : \underline{\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)} = \frac{1}{m} \sum_{i=1}^m (h_0(x^{(i)}) - y^{(i)})$$

$$j = 1 : \underline{\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)} = \frac{1}{m} \sum_{i=1}^m (h_0(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

# Gradient descent algorithm

repeat until convergence {

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

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

}

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

update  
 $\theta_0$  and  $\theta_1$   
simultaneously

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





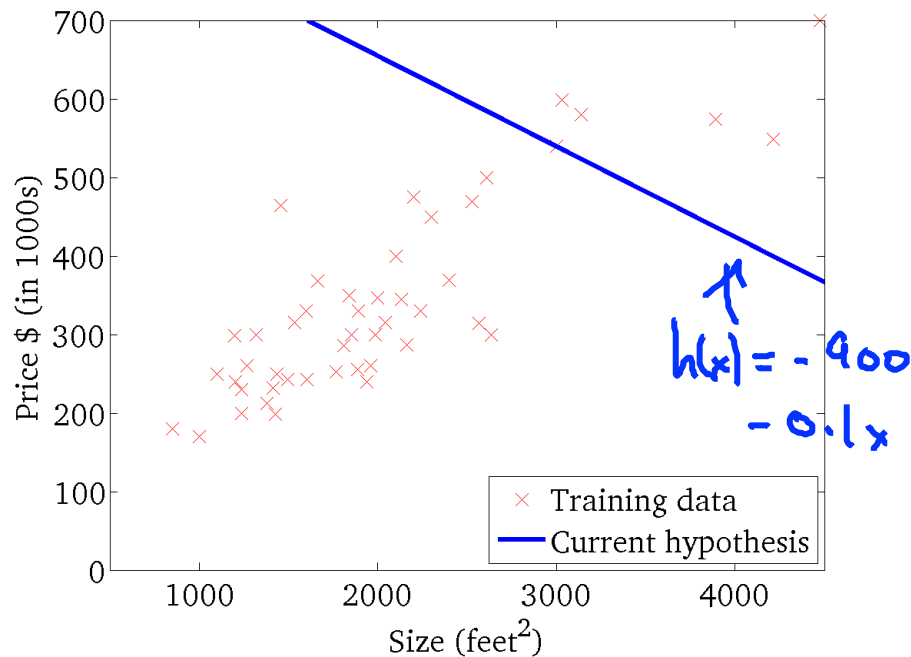


"Convex function"

Bowl-shaped

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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



$$\underline{J(\theta_0, \theta_1)}$$

(function of the parameters  $\theta_0, \theta_1$ )





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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



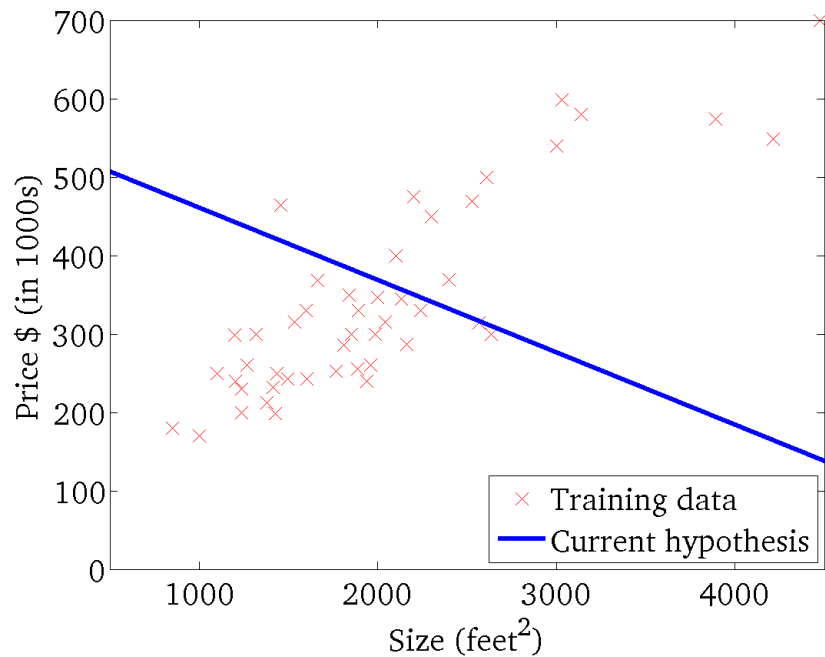
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



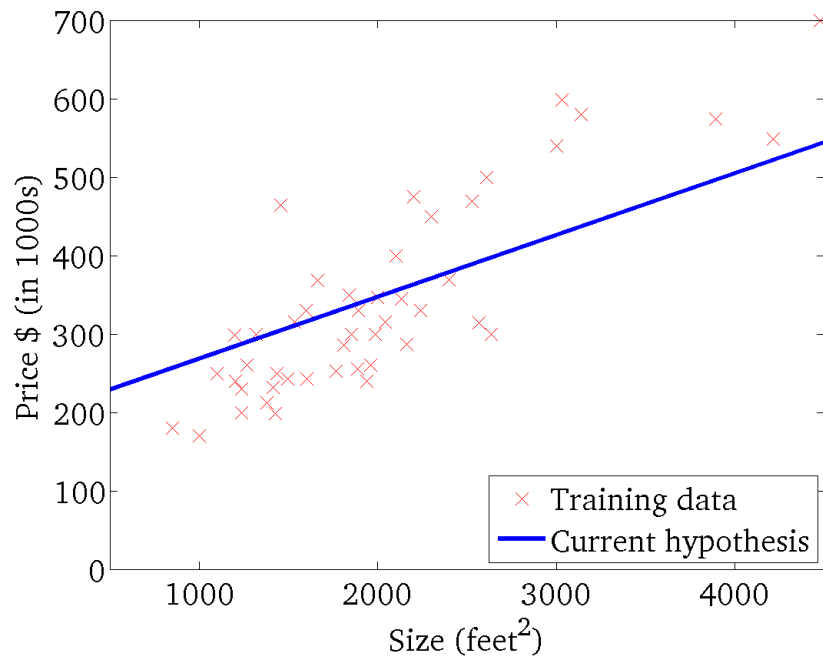
$$J(\theta_0, \theta_1)$$

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )



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

(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

(function of the parameters  $\theta_0, \theta_1$ )





## “Batch” Gradient Descent

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

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