

Machine Learning

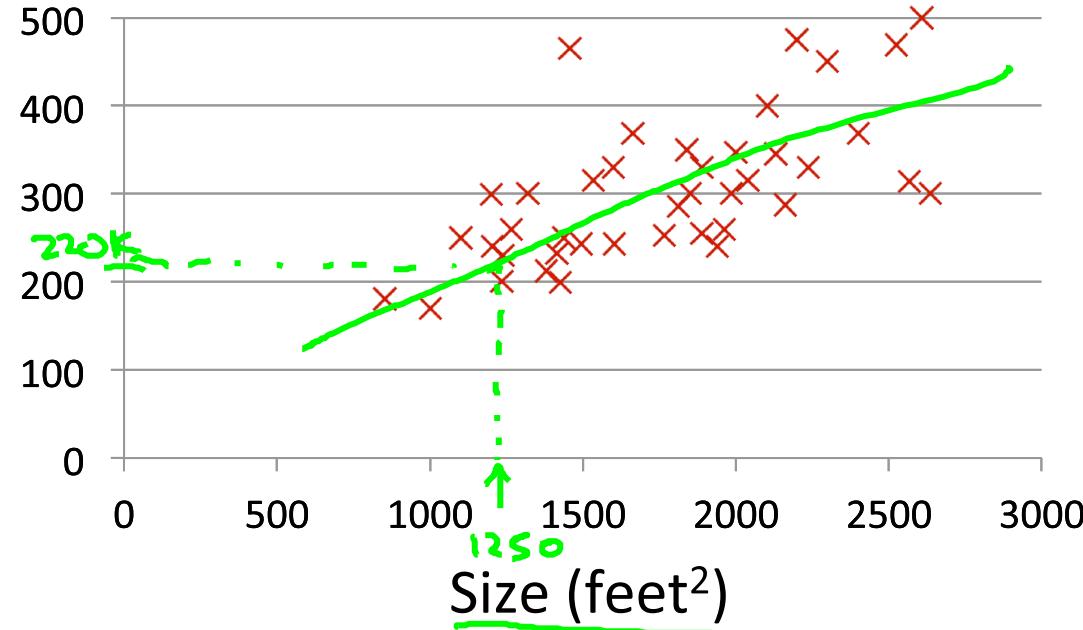
## Linear regression with one variable

---

## 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
...	...
i	i

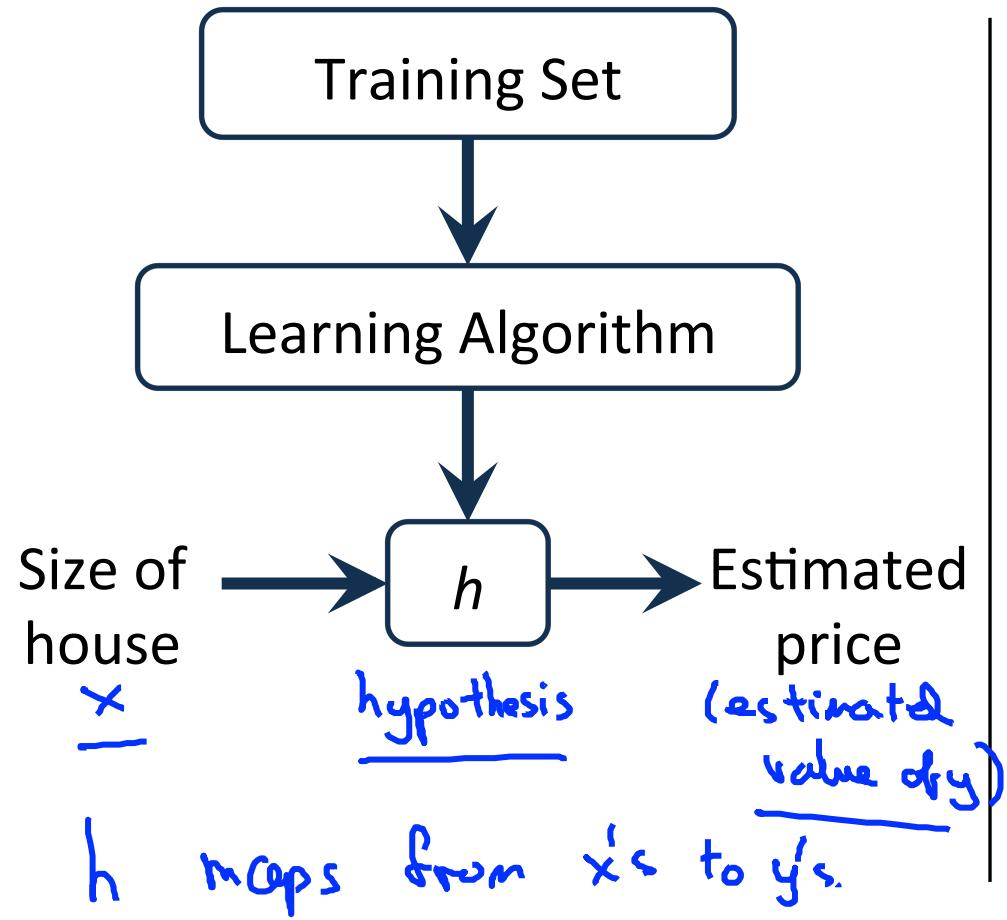
Notation:

- $m$  = 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

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

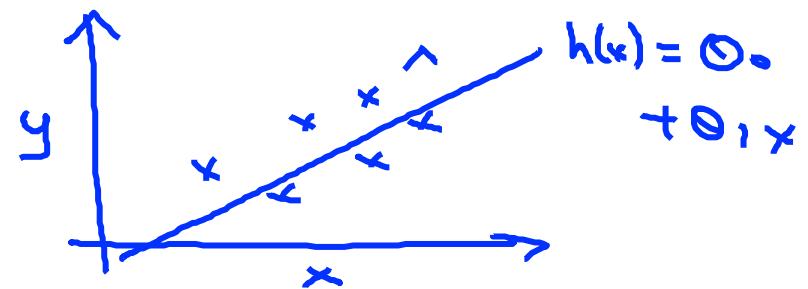
Andrew Ng



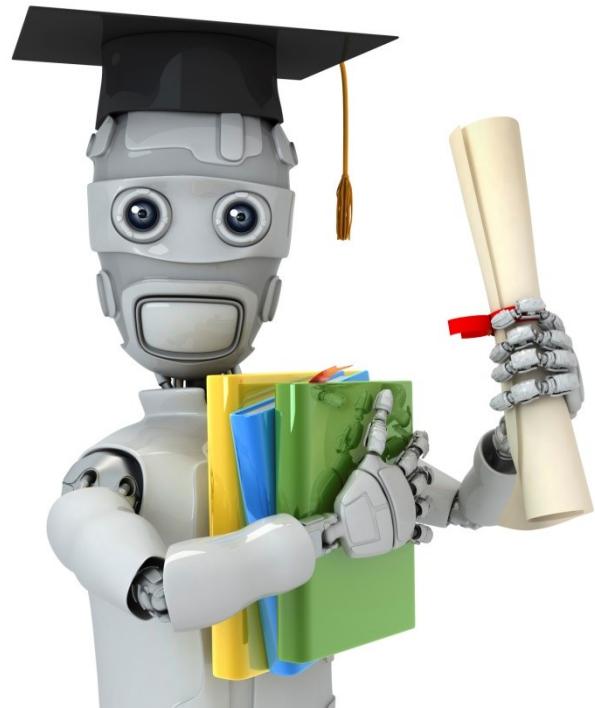
How do we represent  $h$  ?

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

Shorthand:  $\hat{h}(x)$



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
	...	...

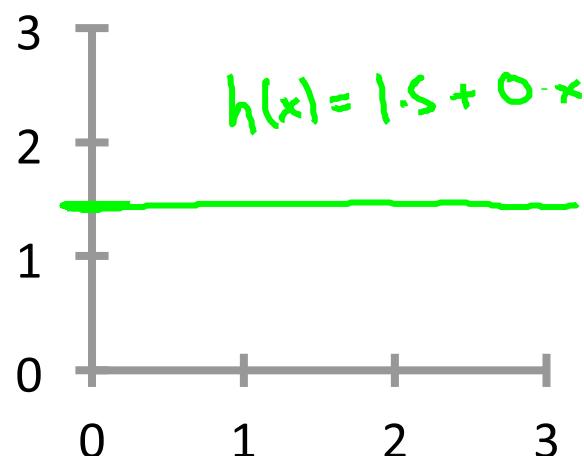
$m = 47$

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

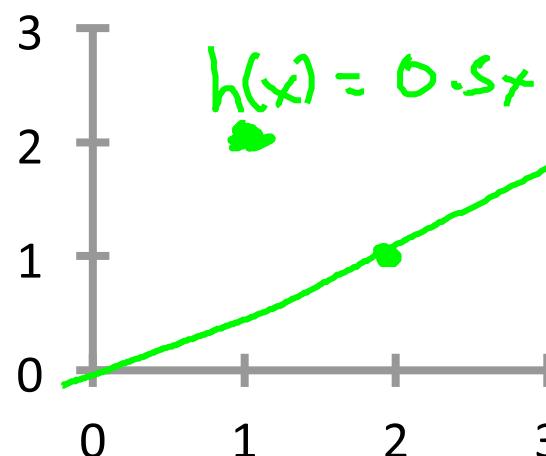
$\theta_i$ 's: Parameters

How to choose  $\theta_i$ 's ?

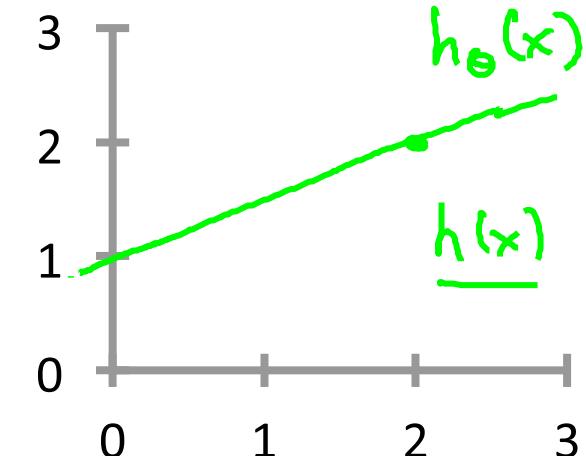
$$h_{\theta}(x) = \underline{\theta_0 + \theta_1 x}$$



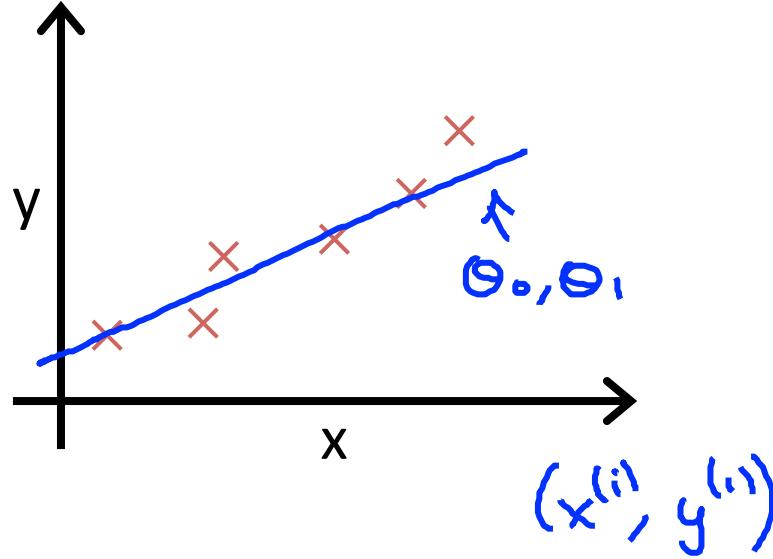
$$\begin{aligned} \rightarrow \theta_0 &= 1.5 \\ \rightarrow \theta_1 &= 0 \end{aligned}$$



$$\begin{aligned} \rightarrow \theta_0 &= 0 \\ \rightarrow \theta_1 &= 0.5 \end{aligned}$$



$$\begin{aligned} \rightarrow \theta_0 &= 1 \\ \rightarrow \theta_1 &= 0.5 \end{aligned}$$



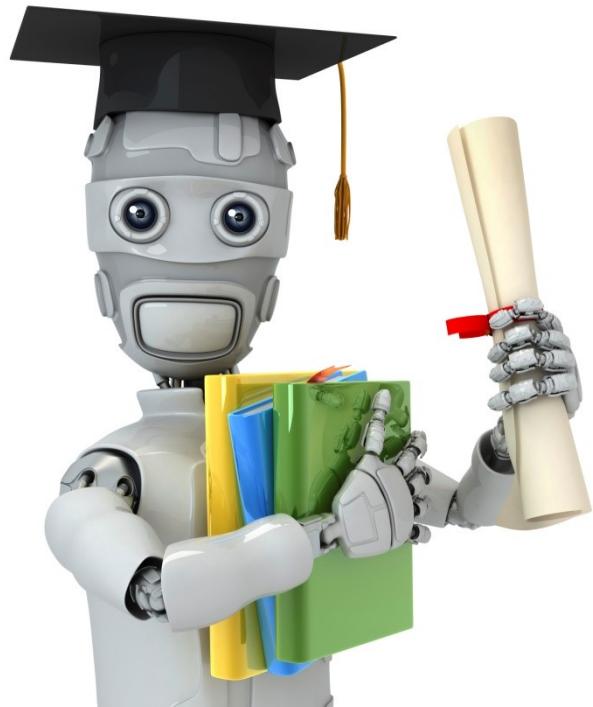
Idea: Choose  $\underline{\theta_0}, \underline{\theta_1}$  so that  $\underline{h_\theta(x)}$  is close to  $\underline{y}$  for our training examples  $\underline{(x, y)}$

$x, y$

$$\begin{array}{l} \text{minimize}_{\theta_0, \theta_1} \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \\ h_\theta(x^{(i)}) = \underline{\theta_0} + \underline{\theta_1} x^{(i)} \\ J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2 \end{array}$$

minimize  $\underline{\theta_0, \theta_1}$   $\underbrace{J(\theta_0, \theta_1)}$   
Cost function  
Squared error function

Andrew Ng



Machine Learning

## Linear regression with one variable

---

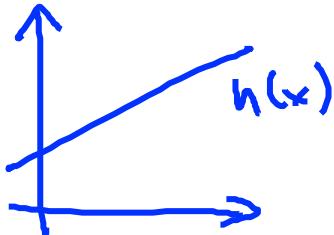
## Cost function intuition I

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  $J(\theta_0, \theta_1)$

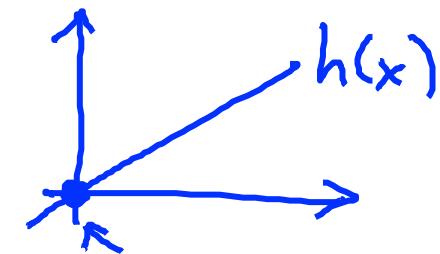
$$\overbrace{\theta_0, \theta_1}$$

Simplified

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

$$\underline{\theta_0 = 0}$$

$$\underline{\theta_1}$$

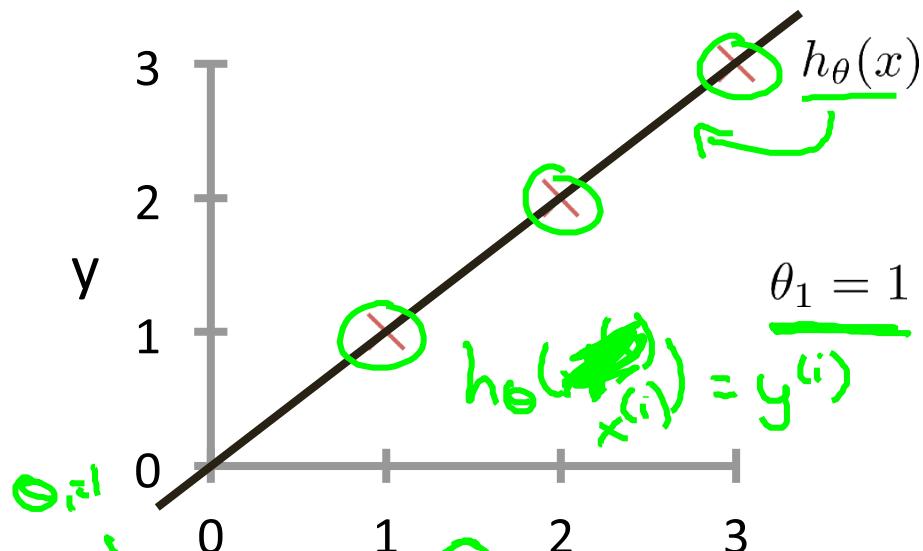


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (\underline{h_{\theta}(x^{(i)})} - y^{(i)})^2$$

$$\text{minimize } \underline{J(\theta_1)}$$

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

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

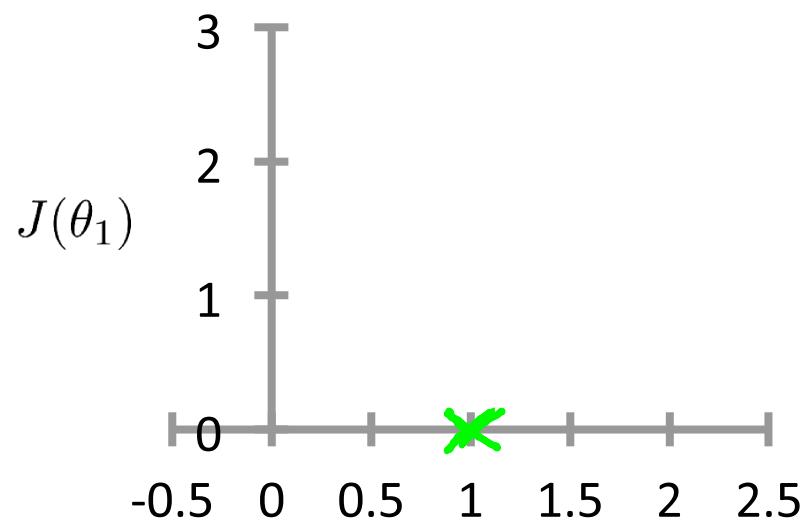


$$\underline{J(\theta_1)} = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^m (\underline{\Theta_1 x^{(i)}} - y^{(i)})^2 \approx \frac{1}{2m} (0^2 + 0^2 + 0^2) = 0^2$$

$\rightarrow \underline{J(\theta_1)}$

(function of the parameter  $\theta_1$ )

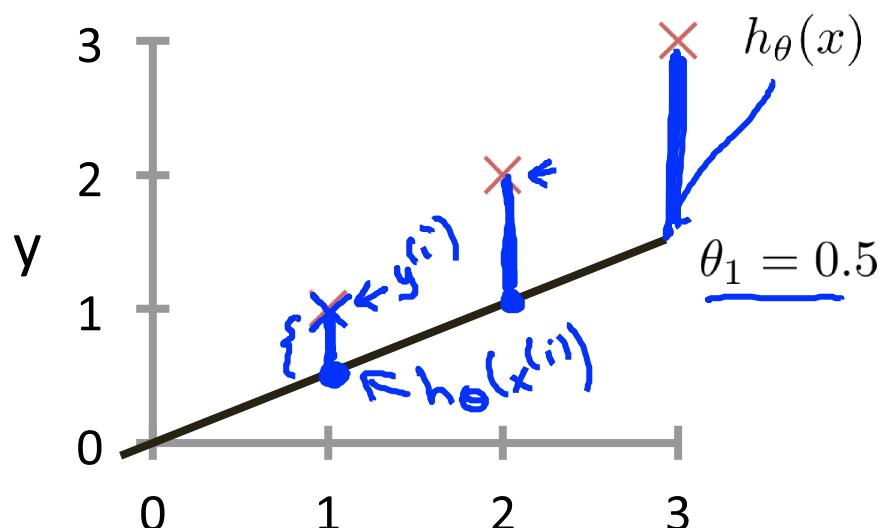


$$\Theta_1 = 0.5 ?$$

$$\underline{J(1)} = 0$$

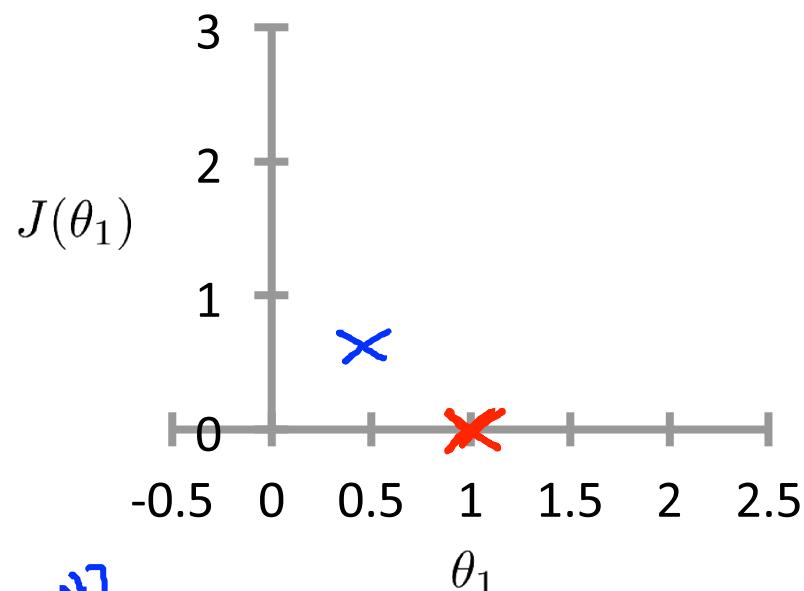
Andrew Ng

$h_{\theta}(x)$   
(for fixed  $\theta_1$ , this is a function of  $x$ )

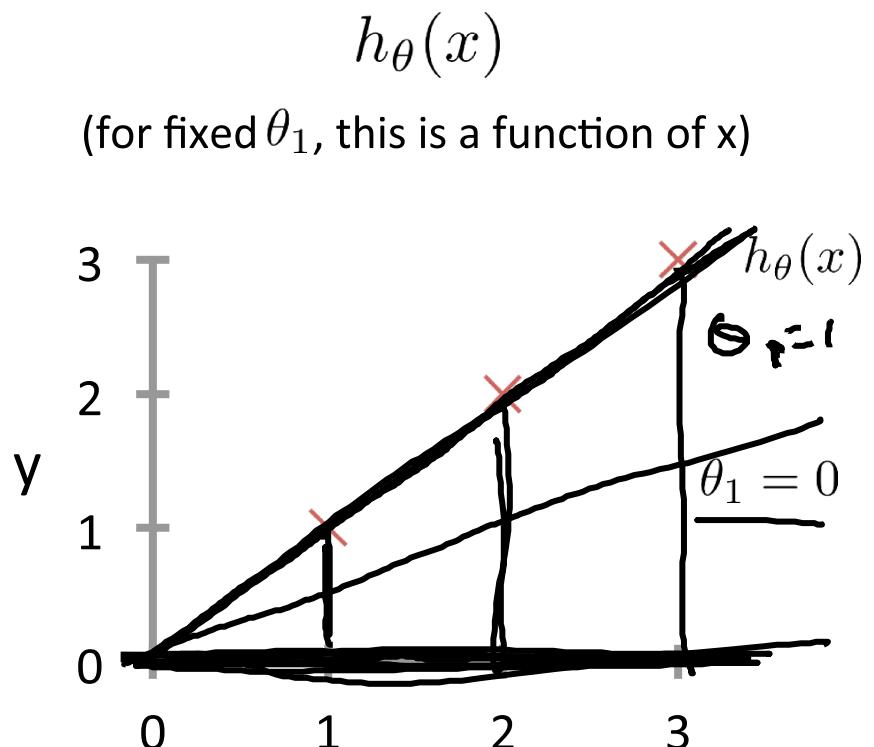


$$\begin{aligned} J(0.5) &= \frac{1}{2m} \sum_{i=1}^m [(0.5 - 1)^2 + (1 - 2)^2 + (1.5 - 3)^2] \\ &= \frac{1}{2 \times 3} (3.5) = \frac{3.5}{6} \approx 0.58 \end{aligned}$$

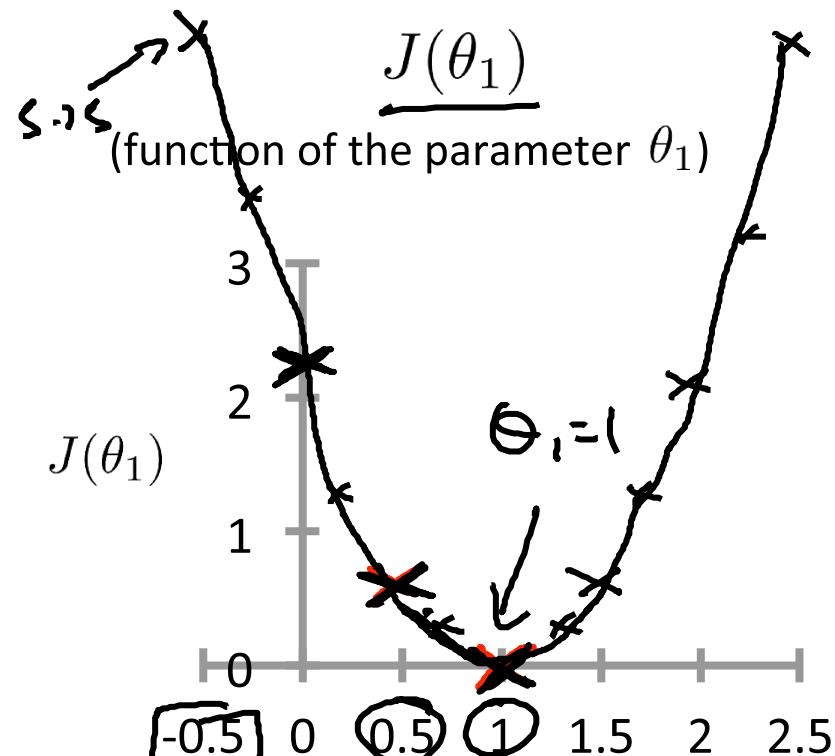
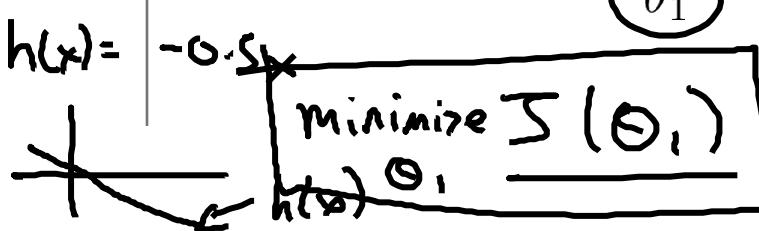
$J(\theta_1)$   
(function of the parameter  $\theta_1$ )

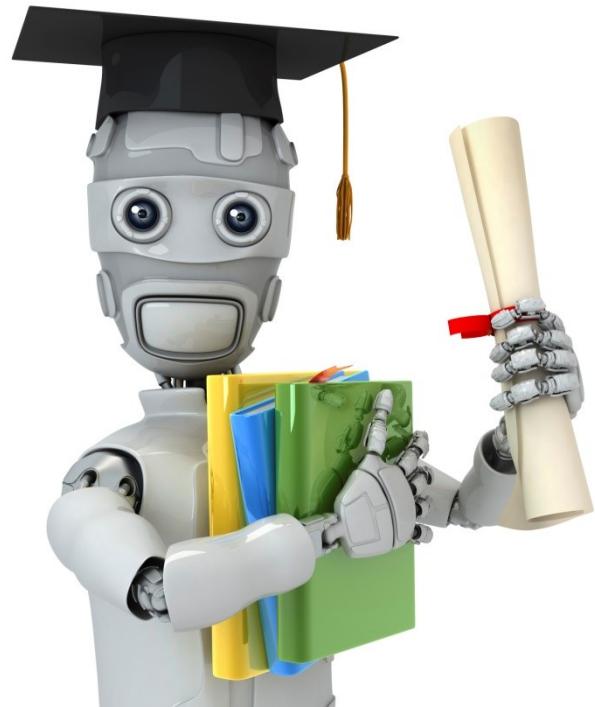


$$\begin{aligned} \theta_1 &\approx 0? \\ J(0) &=? \end{aligned}$$



$$\begin{aligned}
 J(0) &= \frac{1}{2m} \left( 1^2 + 2^2 + 3^2 \right) \\
 &= \frac{1}{6} \cdot 14 \approx 2.3
 \end{aligned}$$





Machine Learning

## Linear regression with one variable

---

## Cost function intuition II

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

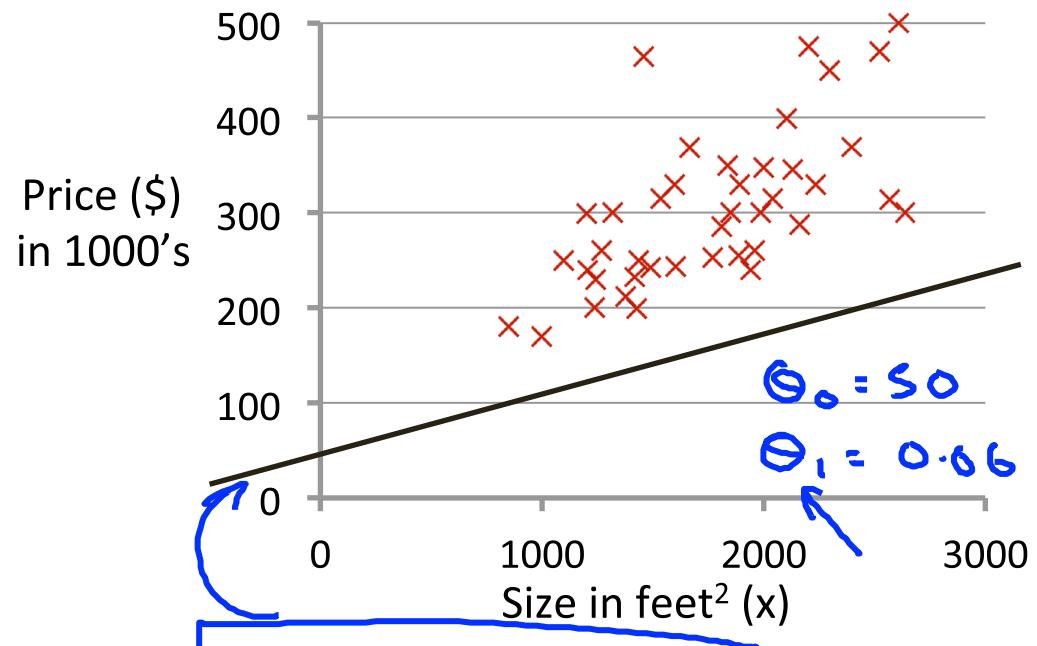
Parameters:  $\underline{\theta_0, \theta_1}$

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

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

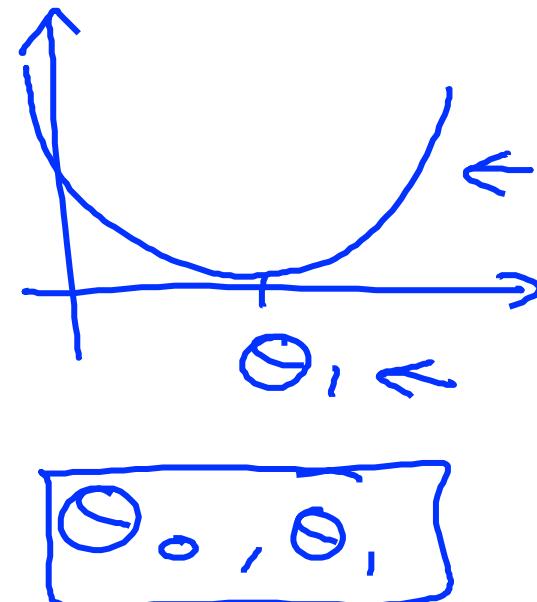
$$\underline{h_{\theta}(x)}$$

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

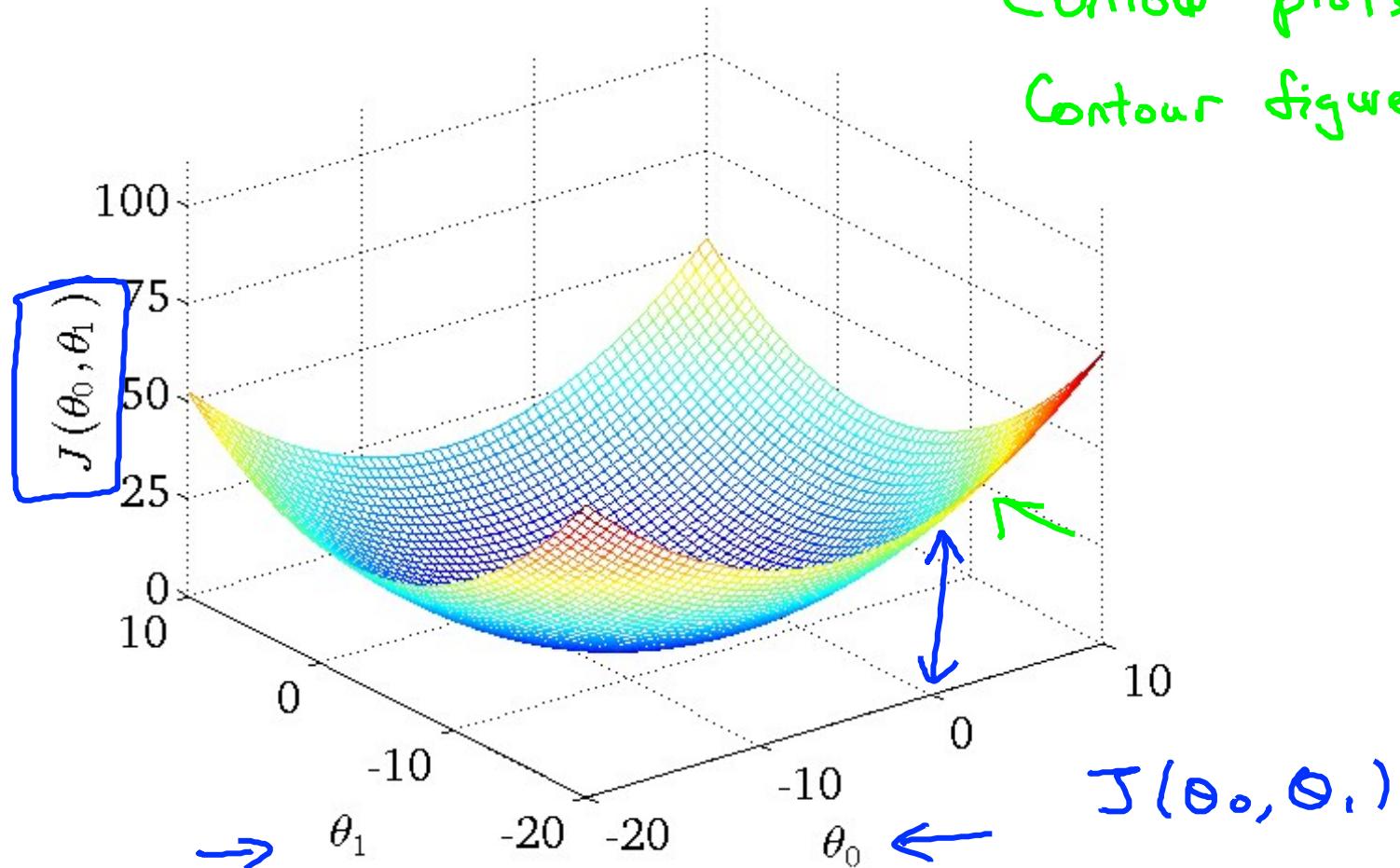


$$\underline{J(\theta_0, \theta_1)}$$

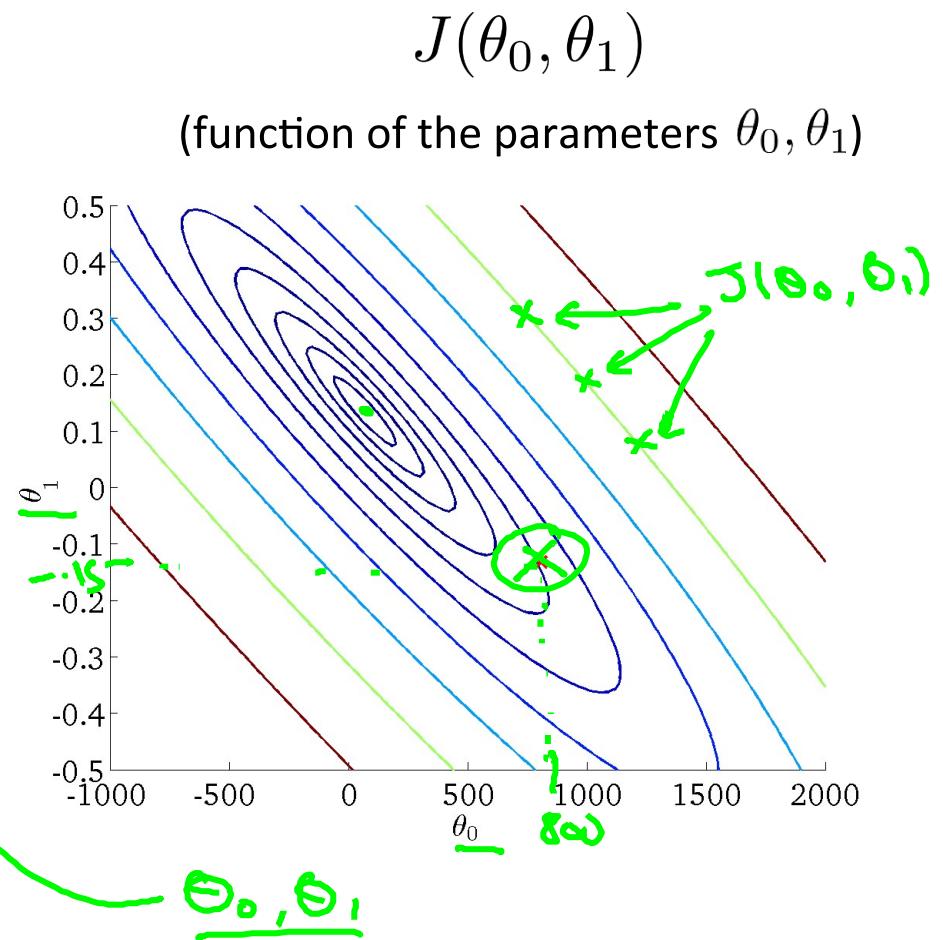
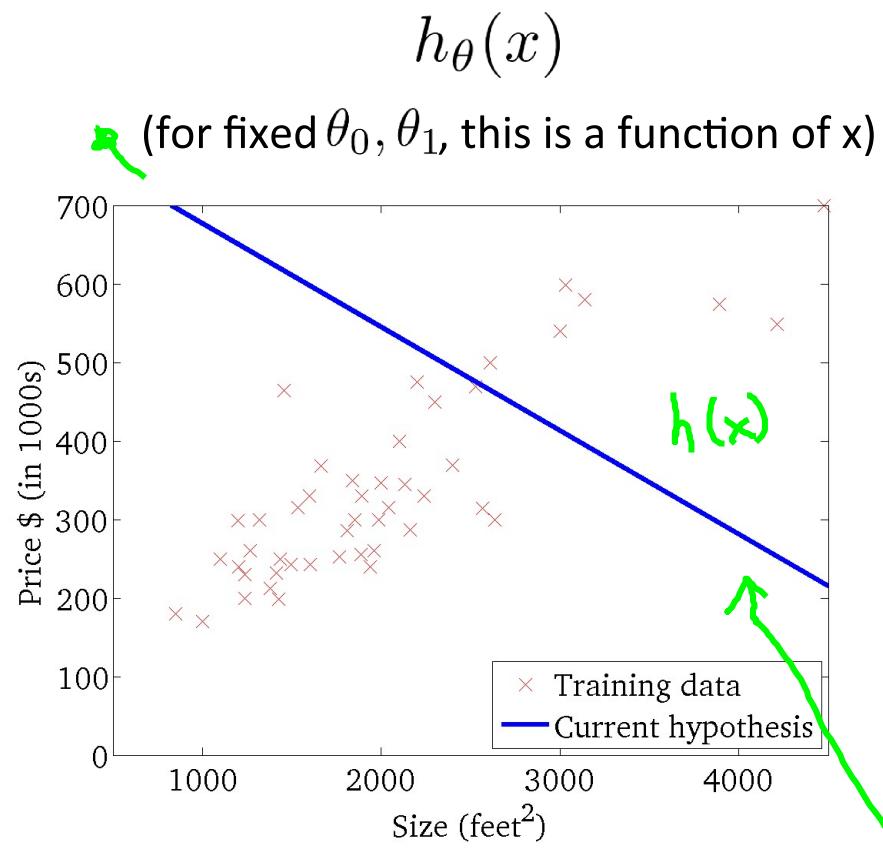
(function of the parameters  $\theta_0, \theta_1$ )



Contour plots  
Contour figures -

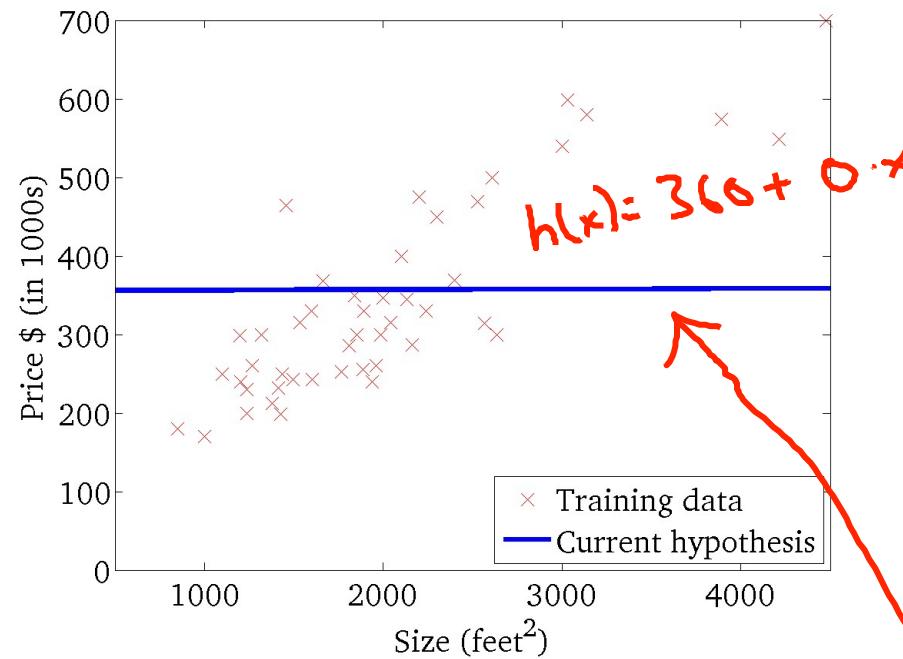


Andrew Ng



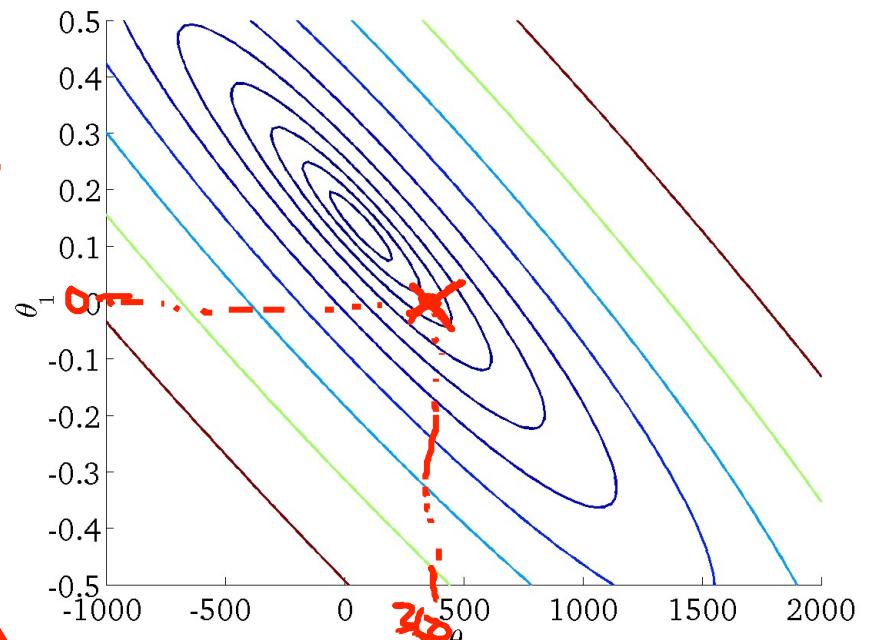
$$h_{\theta}(x)$$

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



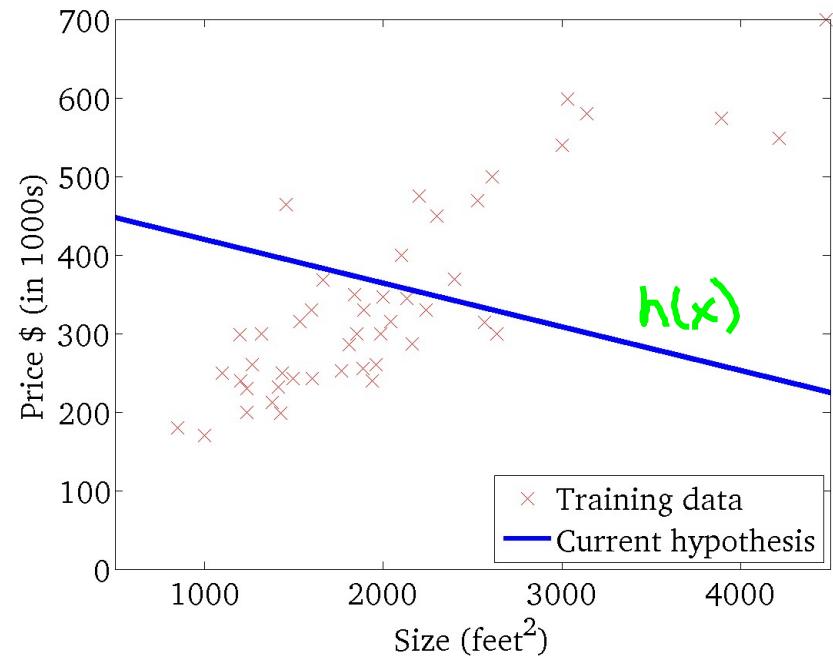
$$J(\theta_0, \theta_1)$$

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



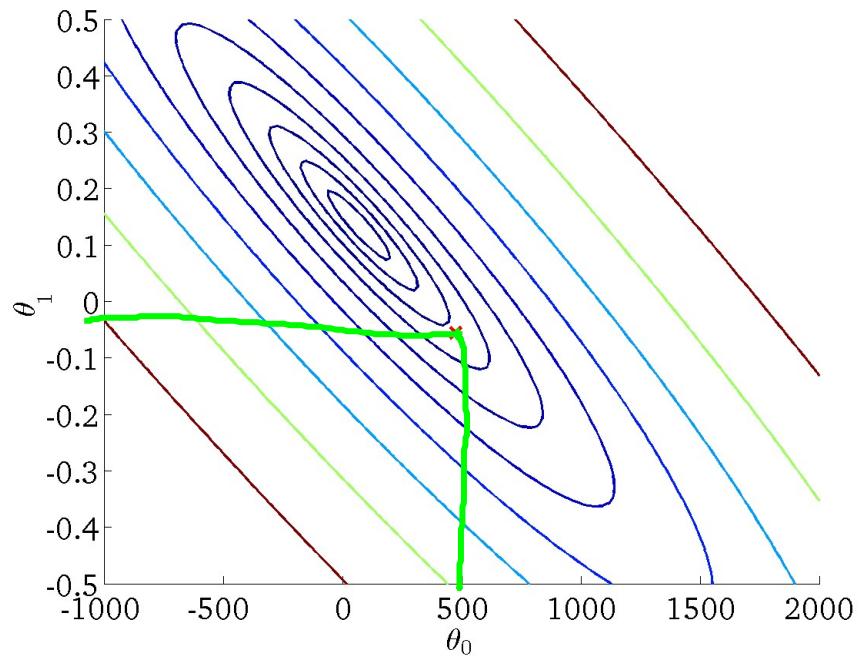
$$h_{\theta}(x)$$

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



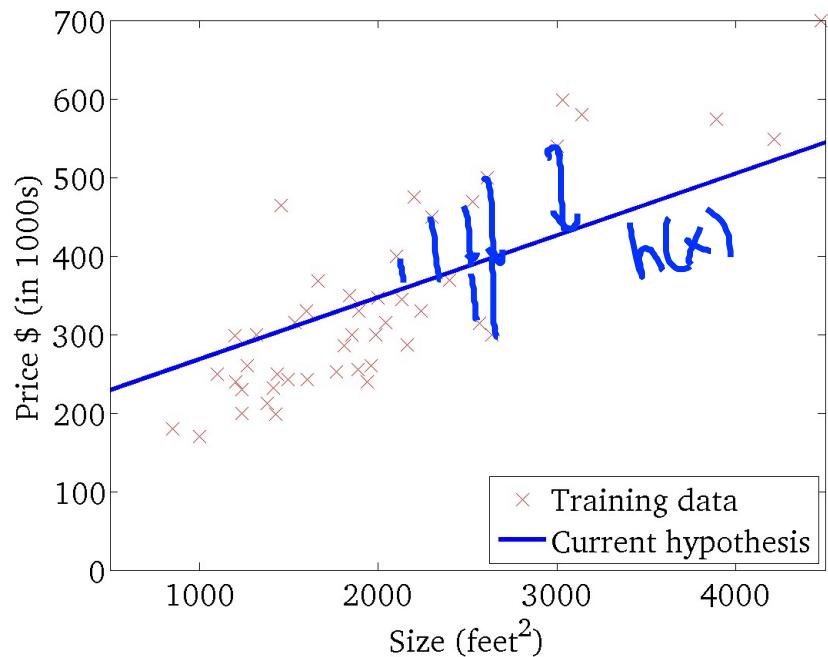
$$J(\theta_0, \theta_1)$$

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



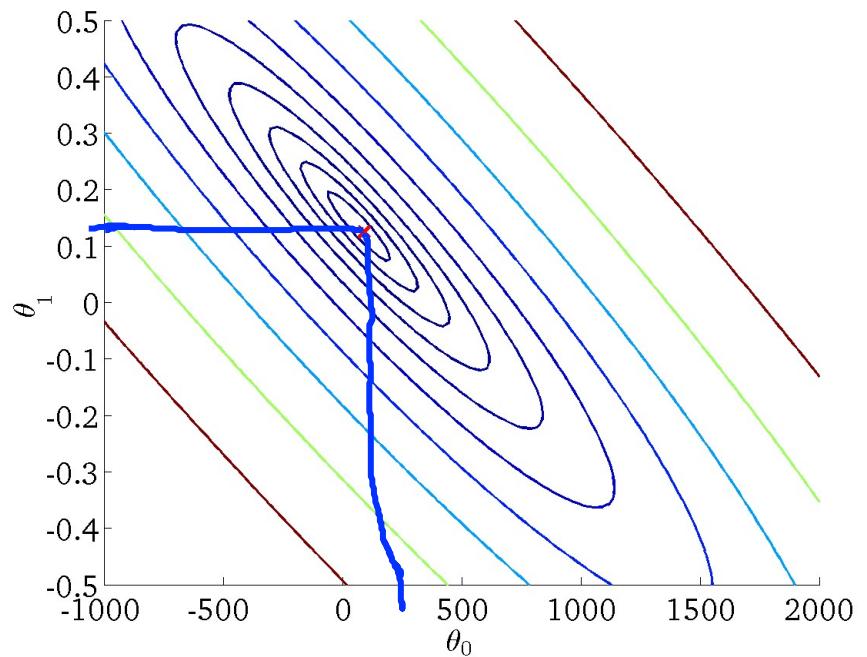
$$h_{\theta}(x)$$

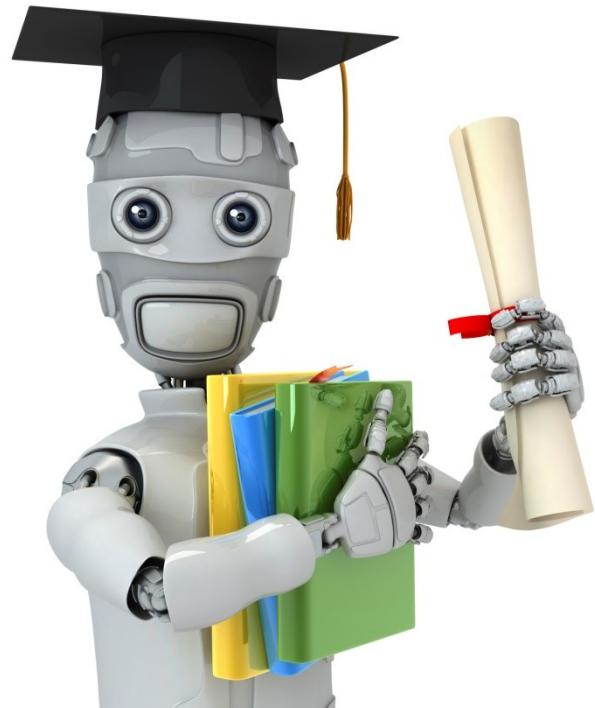
(for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )



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

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





Machine Learning

Linear regression  
with one variable

---

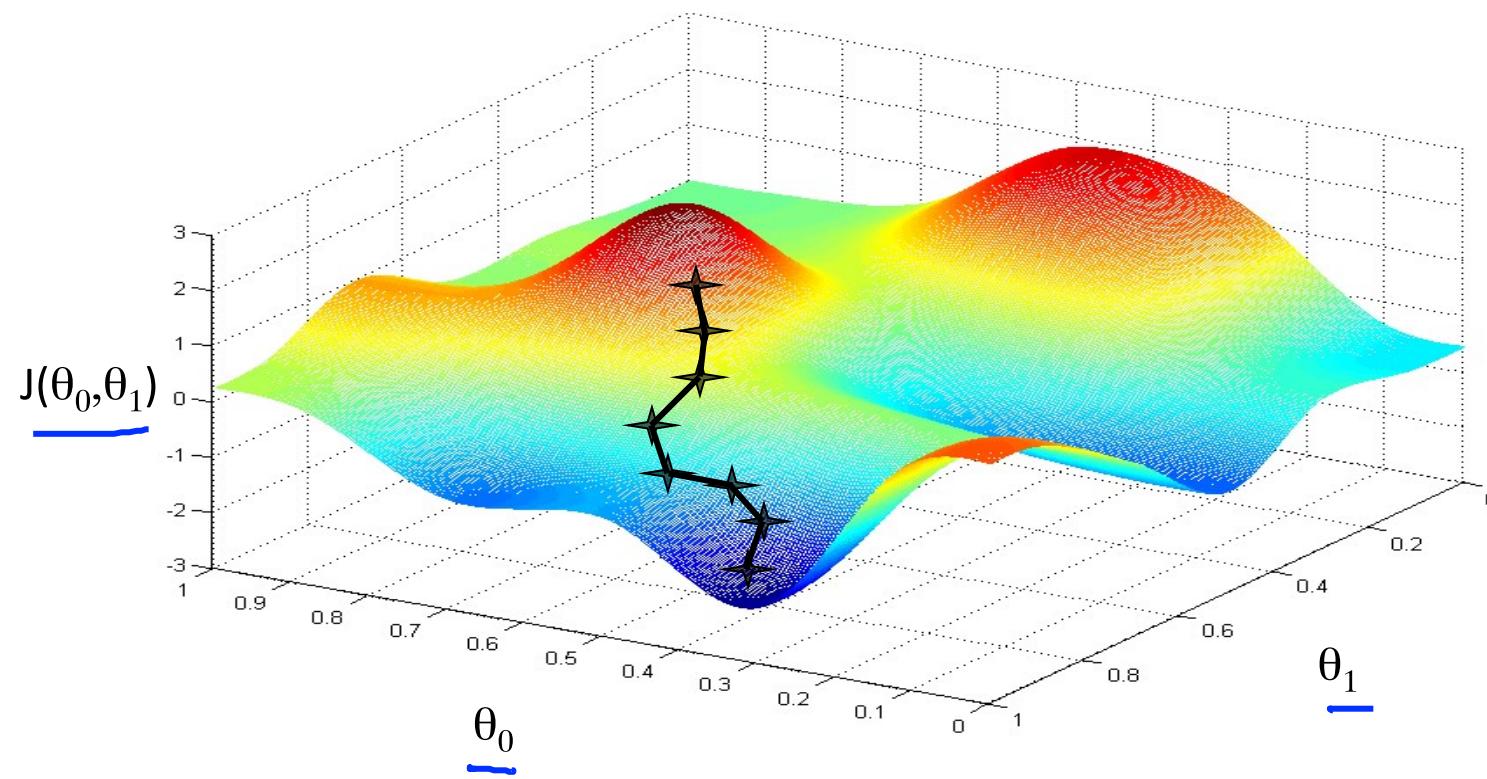
Gradient  
descent

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

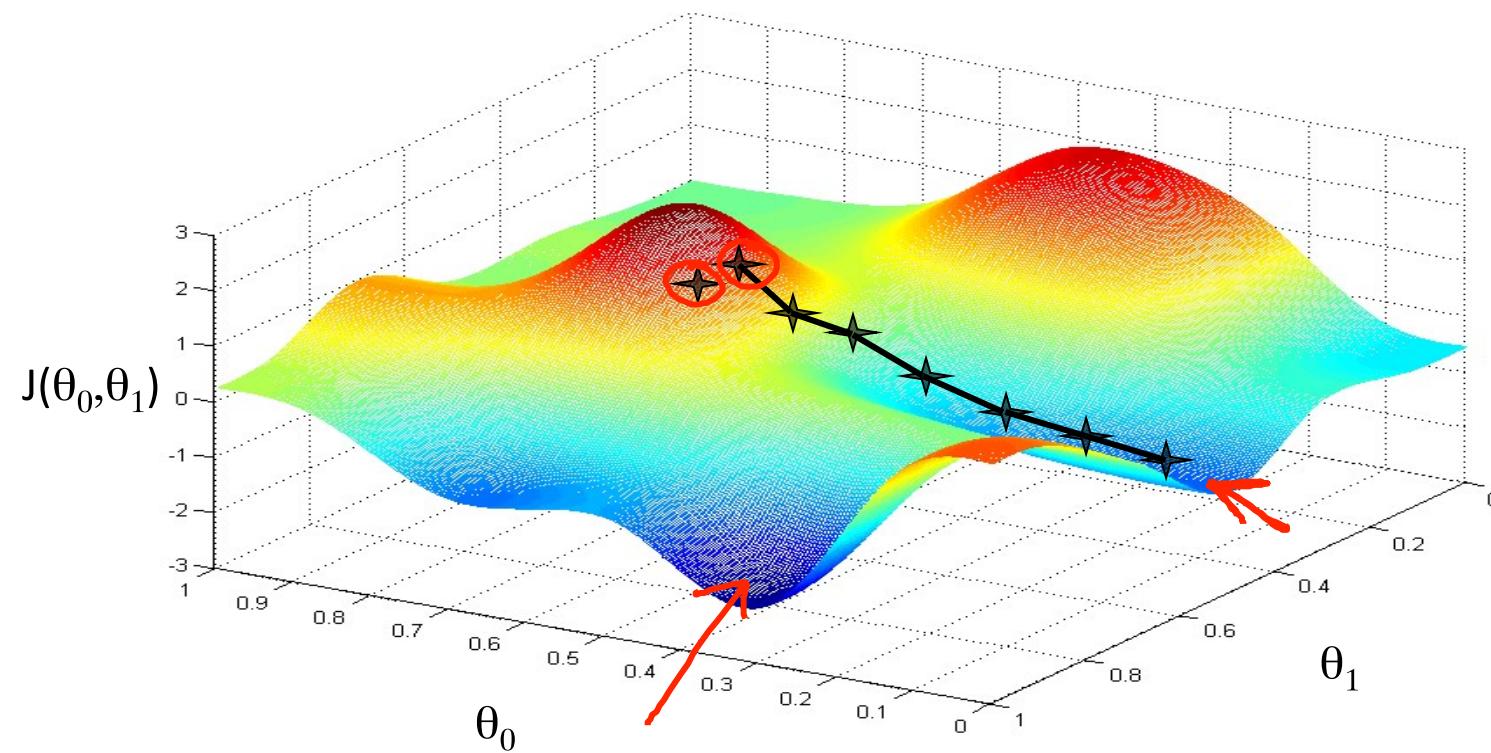
Want  $\min_{\theta_0, \theta_1} \underline{J(\theta_0, \theta_1)}$   $\underline{\min_{\theta_0, \dots, \theta_n} J(\theta_0, \dots, \theta_n)}$

## Outline:

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



Andrew Ng



Andrew Ng

## Gradient descent algorithm

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  $\theta_1$

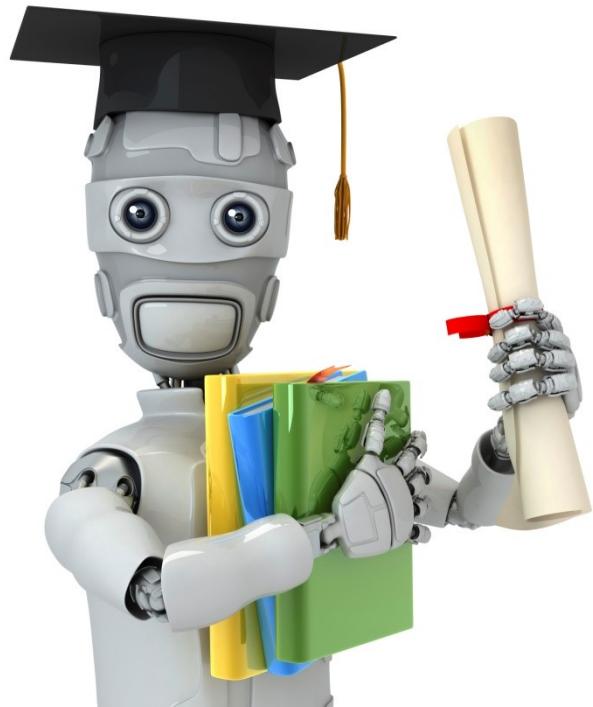
Assignment  
 $a := b$   
 $\underline{a := a + 1}$   
 Truth assertion  
 $a = b$  ↗  
 $a = a + 1$  ✗

### 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

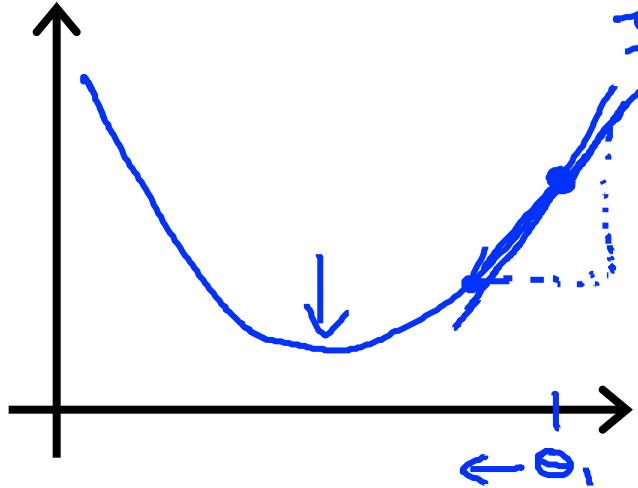
# Gradient descent algorithm

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

learning rate      derivative

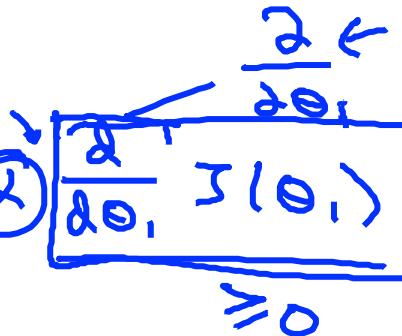
(simultaneously update  
 $j = 0$  and  $j = 1$ )

$$\min_{\theta_1} J(\theta_1) \quad \theta_1 \in \mathbb{R}.$$

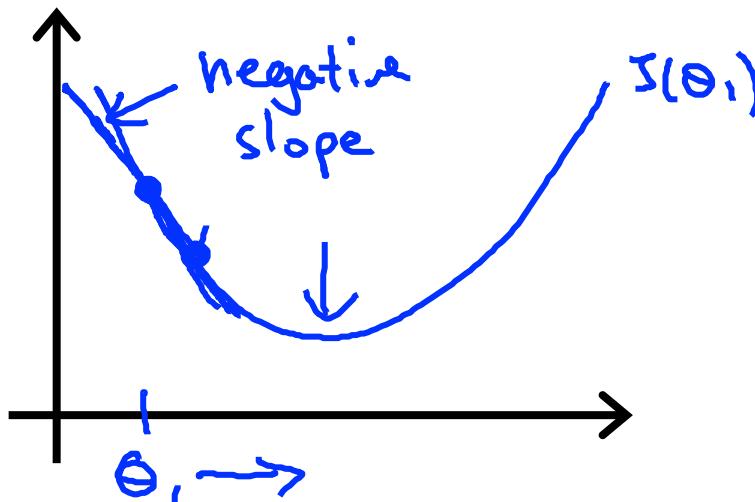


$J(\theta_0)$  ( $\theta_0 \in \mathbb{R}$ )

$$\theta_1 := \theta_0 - \alpha$$



$$\theta_1 := \theta_1 - \frac{\alpha}{\text{positive number}} \cdot (dJ/d\theta_1)$$



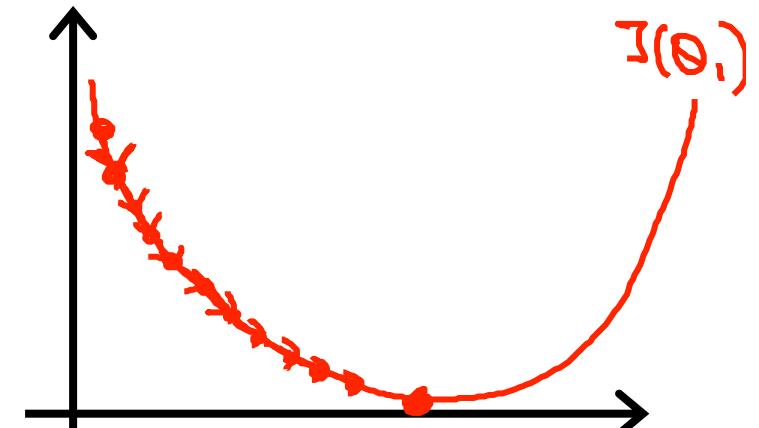
negative slope

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

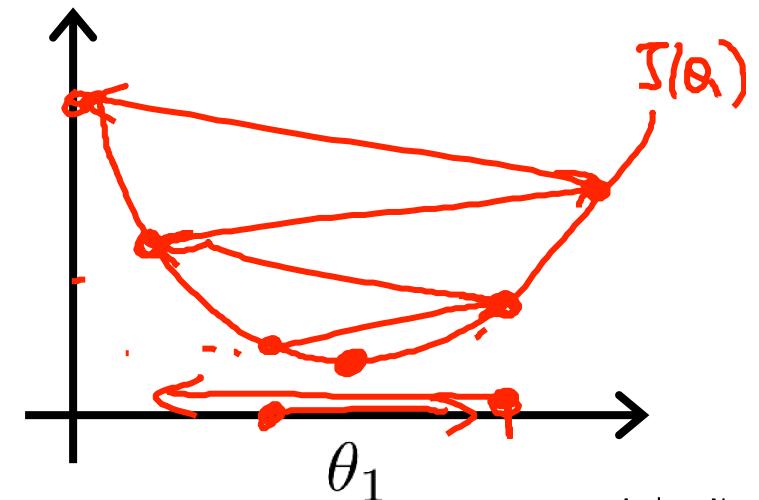
$$\theta_1 := \theta_1 - \frac{\alpha}{\text{negative number}} \cdot (dJ/d\theta_1)$$

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

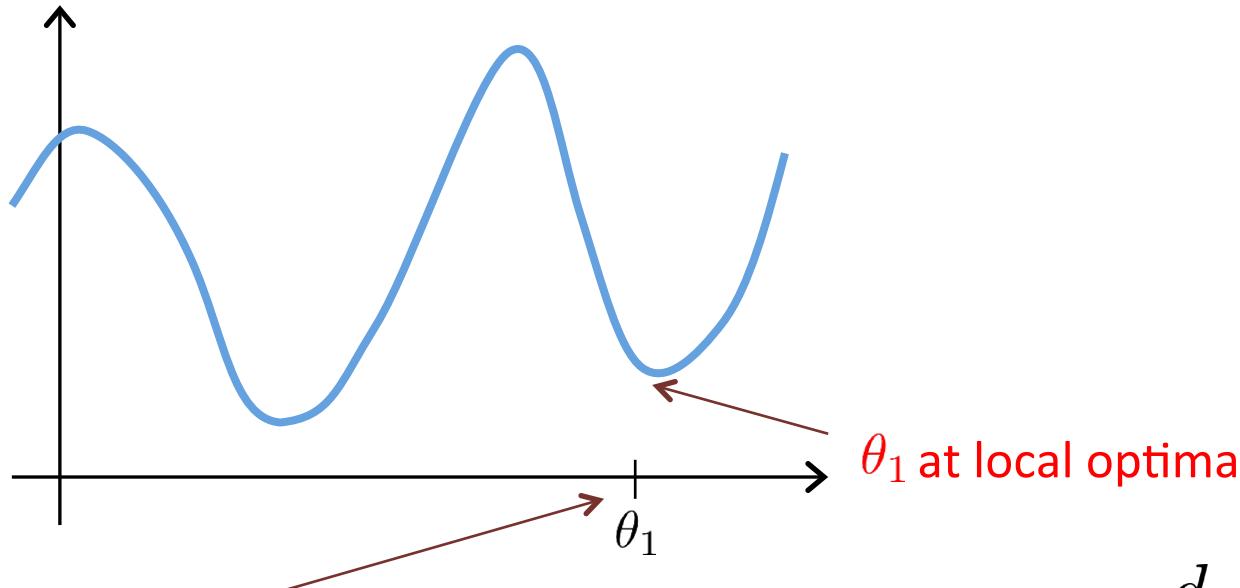
If  $\alpha$  is too small, gradient descent can be slow.



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



Andrew Ng

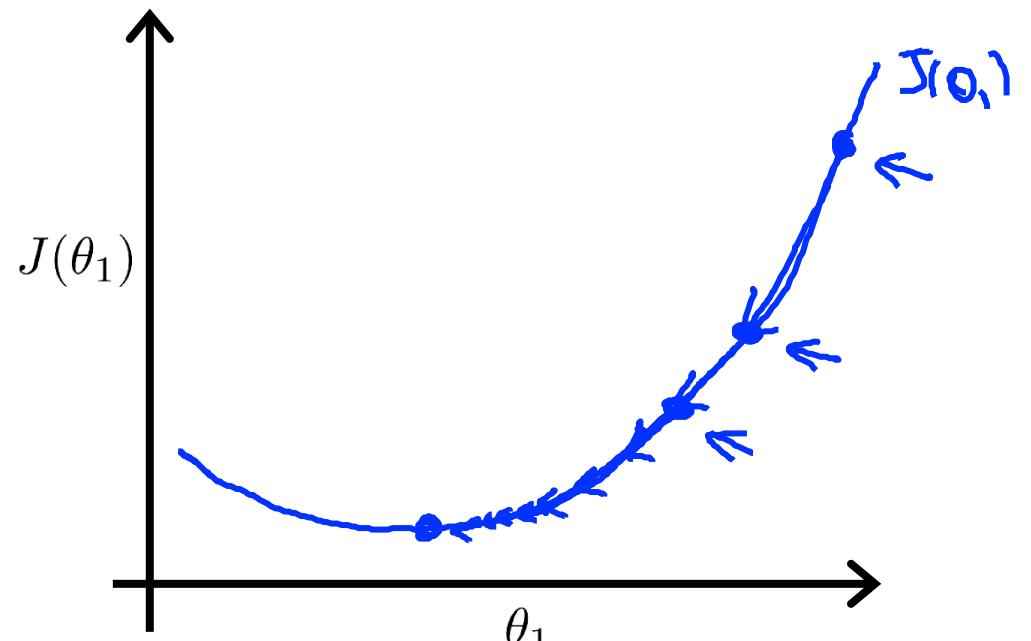


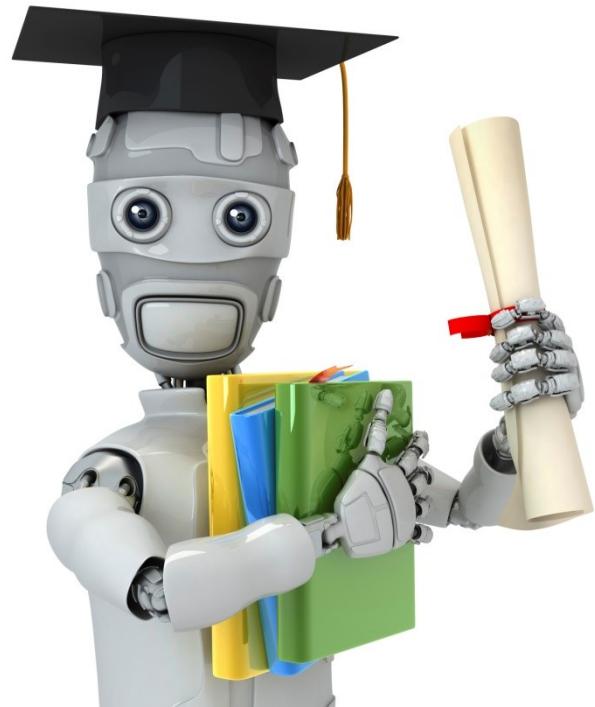
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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

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

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

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

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

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

## Gradient descent algorithm

repeat until convergence {

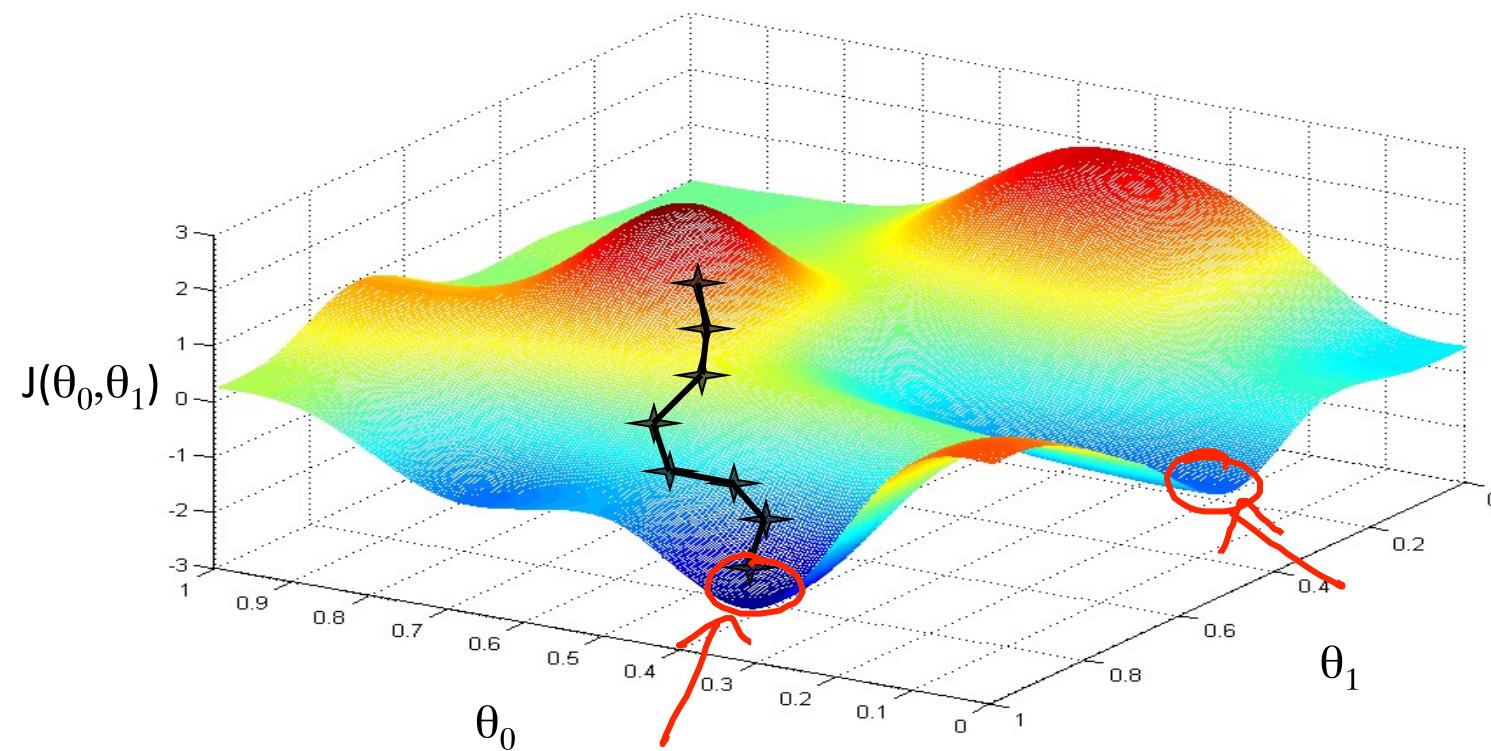
$$\theta_0 := \theta_0 - \alpha \left[ \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \right]$$
$$\theta_1 := \theta_1 - \alpha \left[ \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)} \right]$$

}

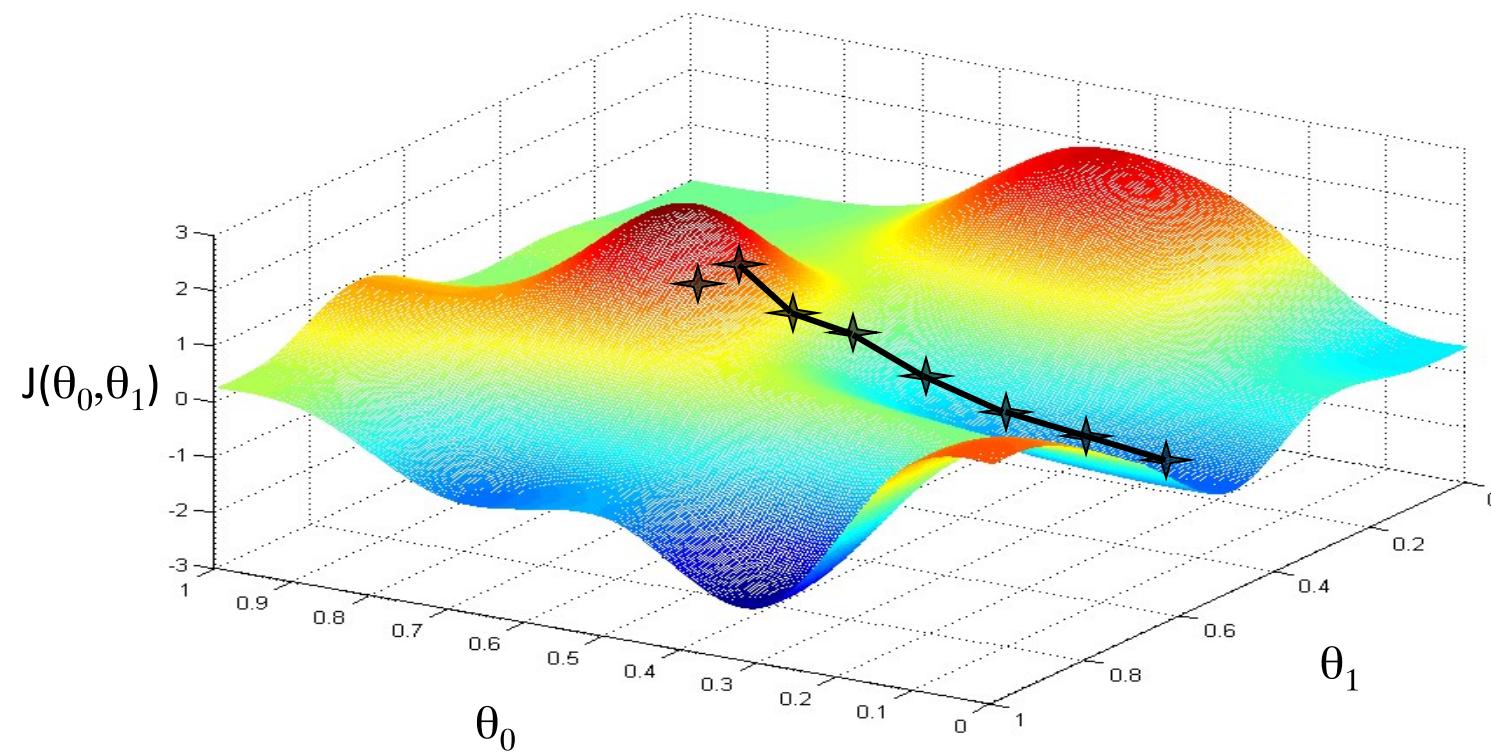
$$\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

update  
 $\theta_0$  and  $\theta_1$   
simultaneously

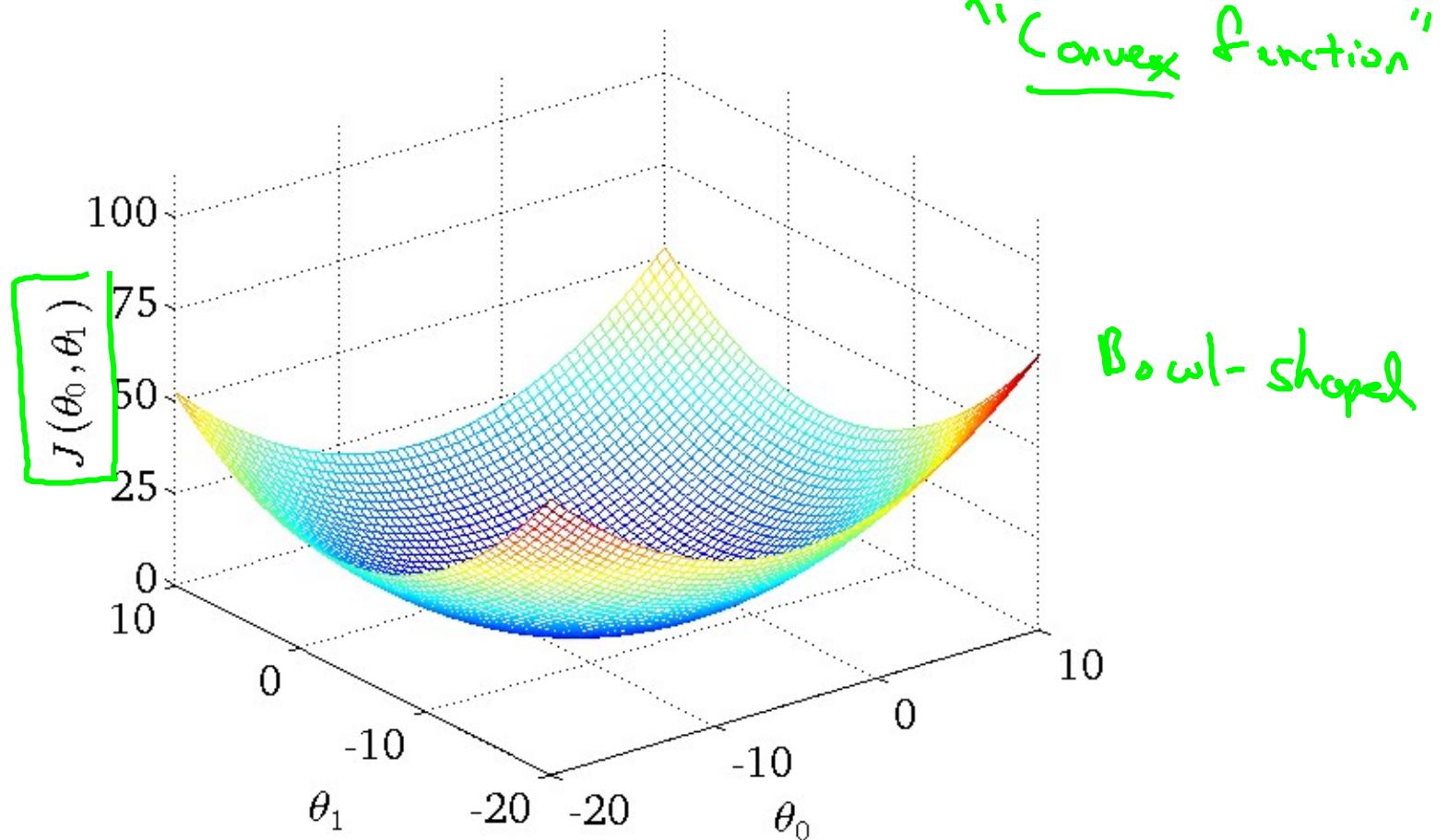
$$\frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$



Andrew Ng

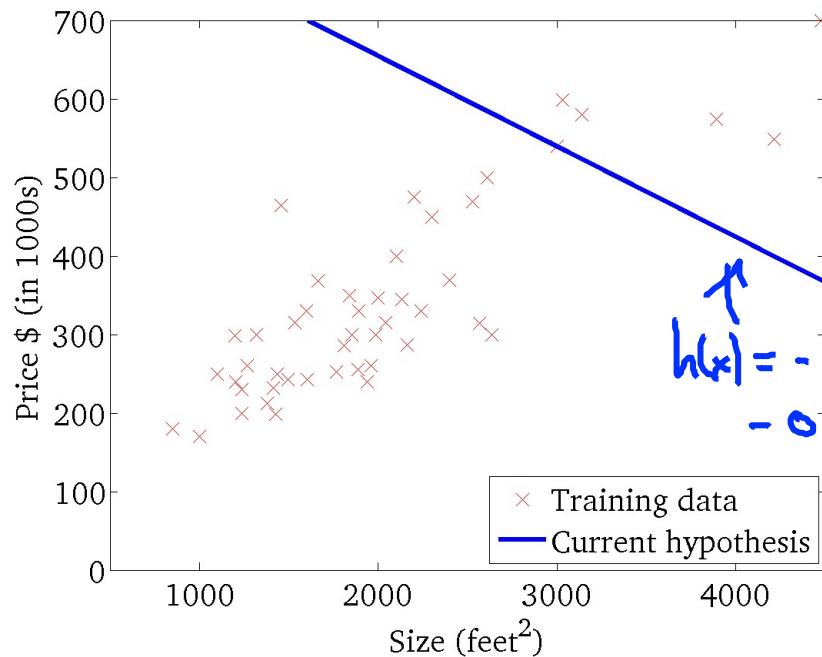


Andrew Ng

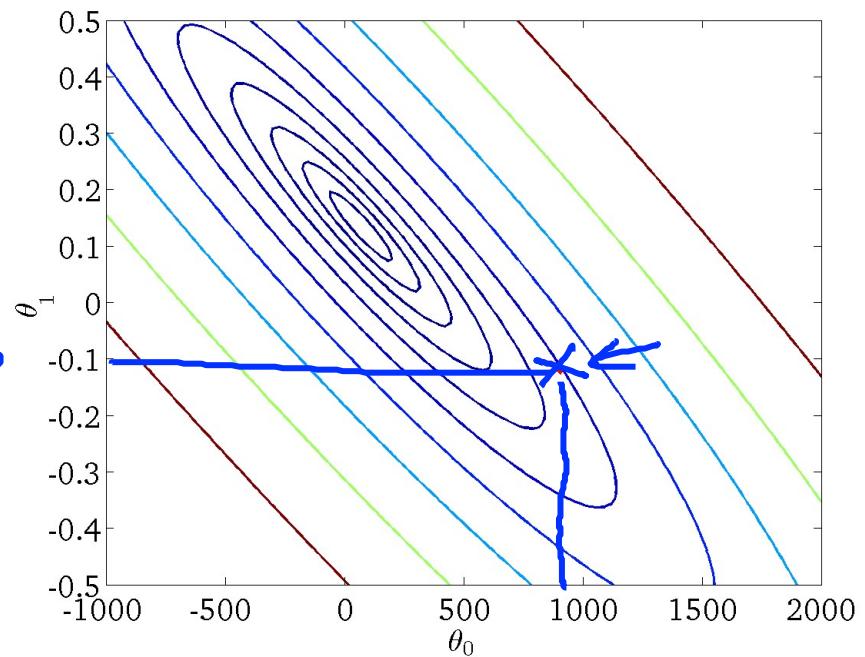


Andrew Ng

$\underline{h_{\theta}(x)}$   
 (for fixed  $\theta_0, \theta_1$ , this is a function of  $x$ )

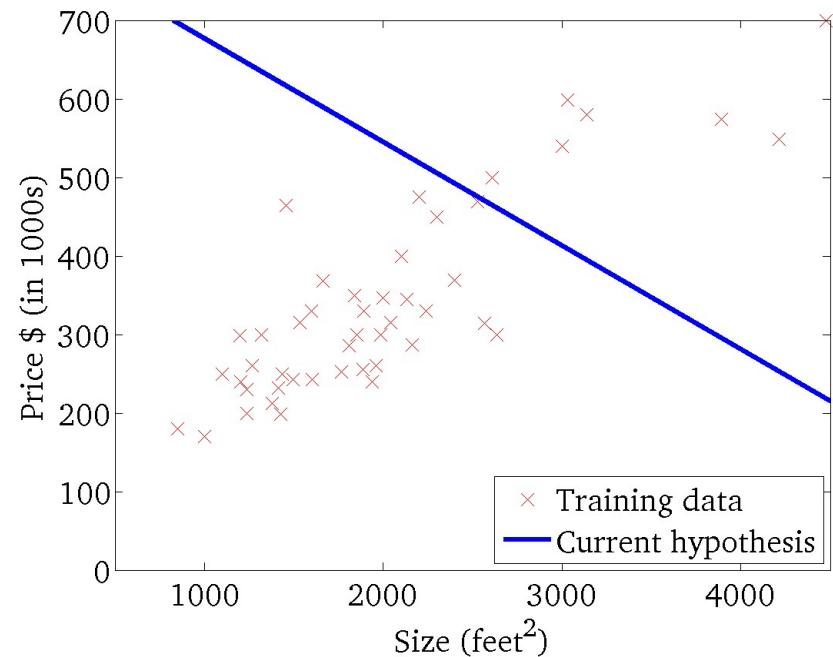


$\underline{J(\theta_0, \theta_1)}$   
 (function of the parameters  $\theta_0, \theta_1$ )



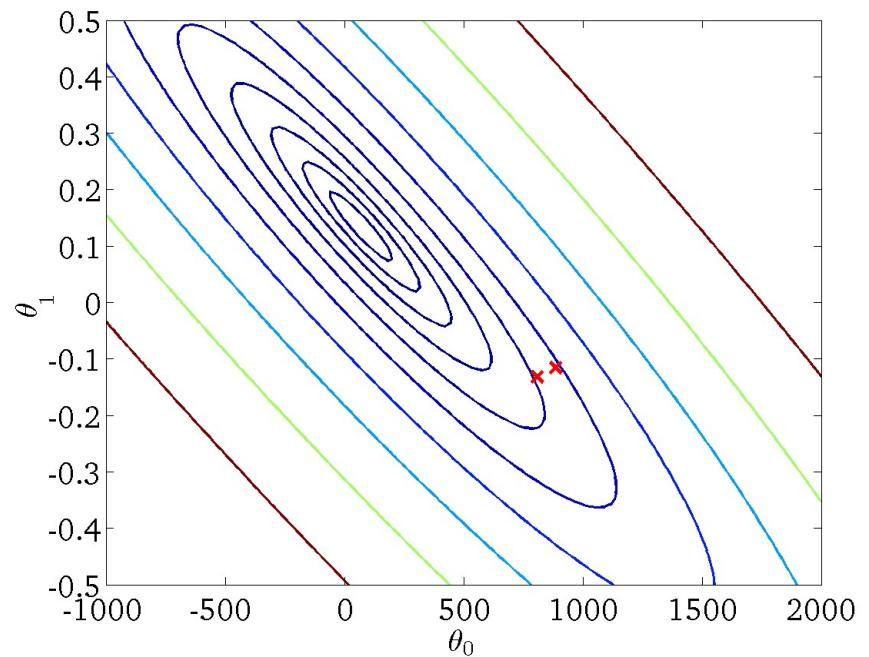
$$h_{\theta}(x)$$

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



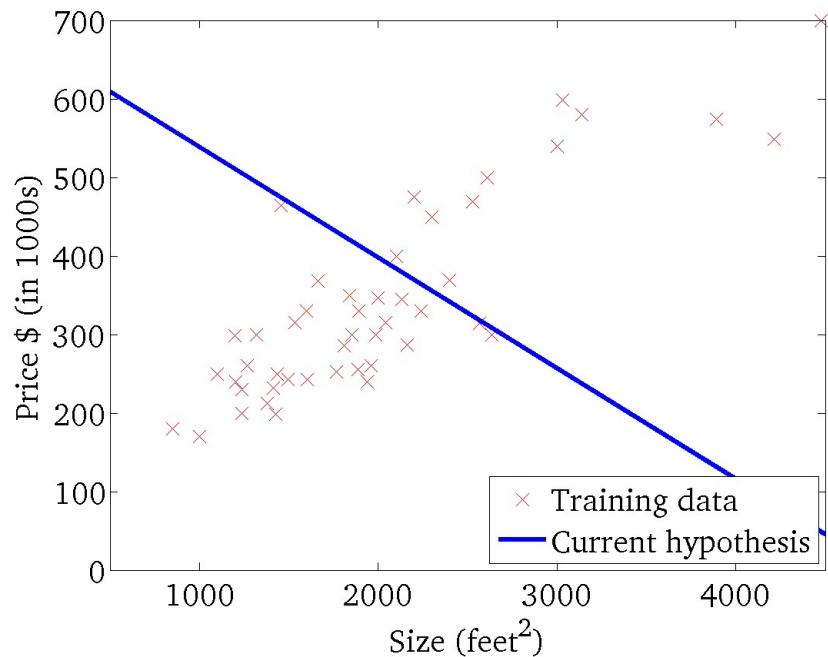
$$J(\theta_0, \theta_1)$$

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



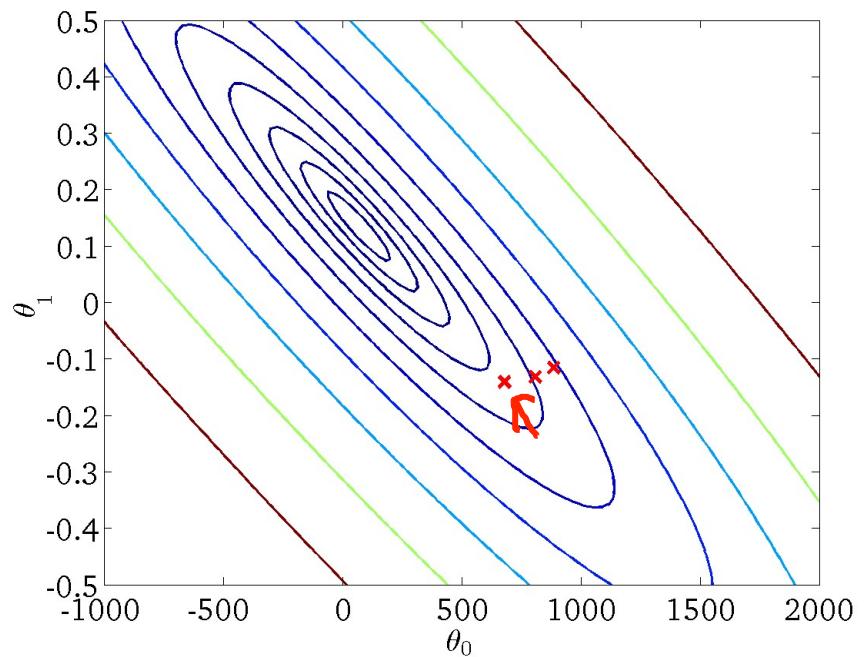
$$h_{\theta}(x)$$

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



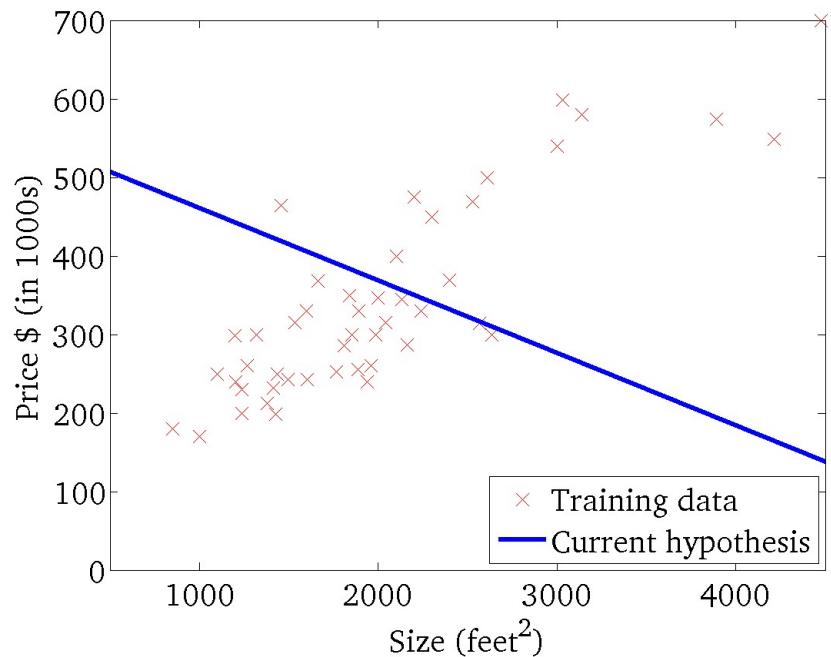
$$J(\theta_0, \theta_1)$$

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



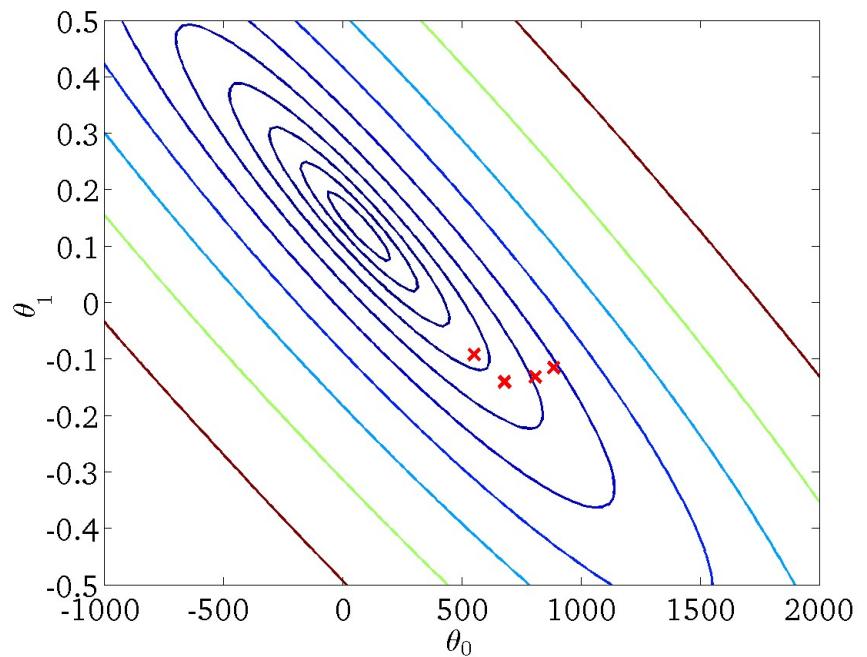
$$h_{\theta}(x)$$

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



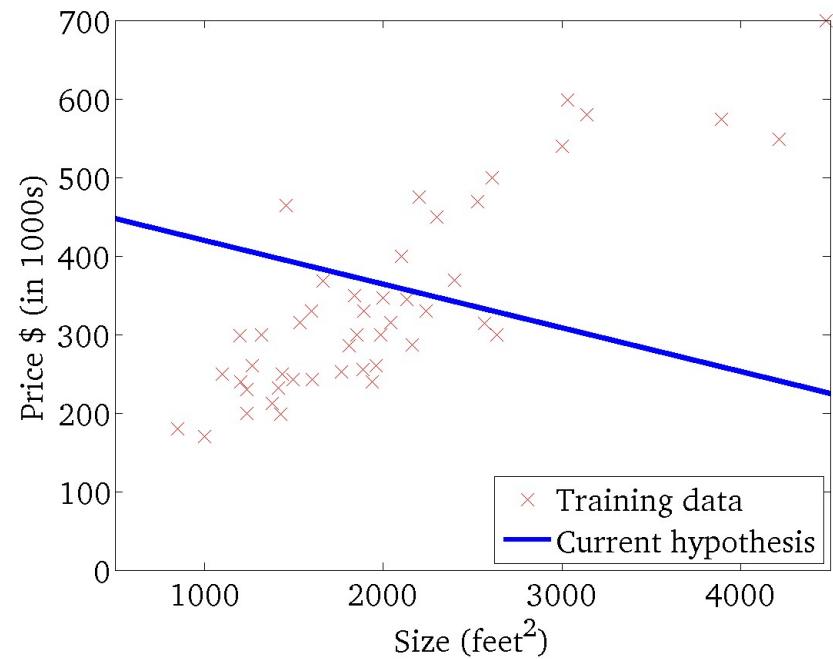
$$J(\theta_0, \theta_1)$$

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



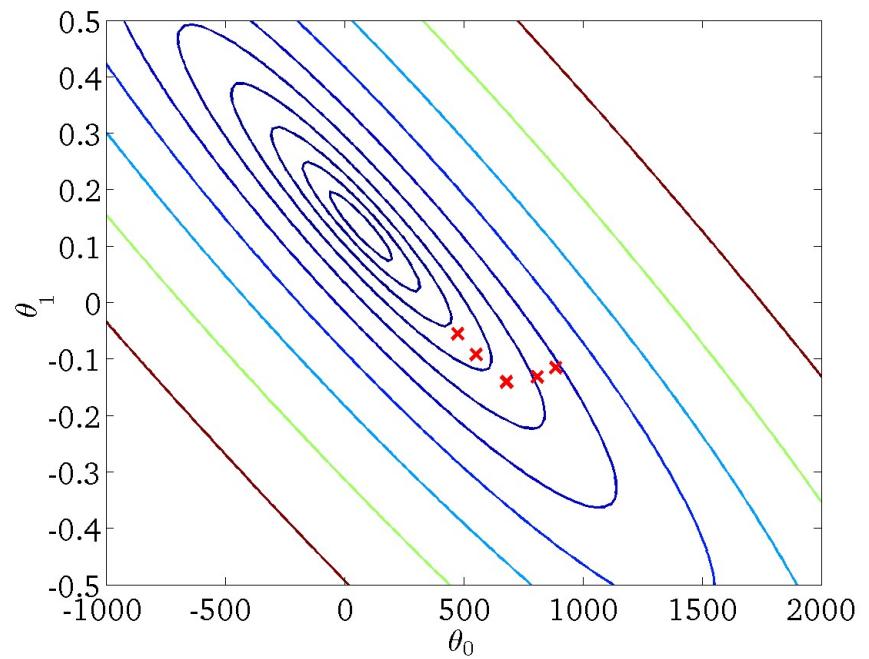
$$h_{\theta}(x)$$

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



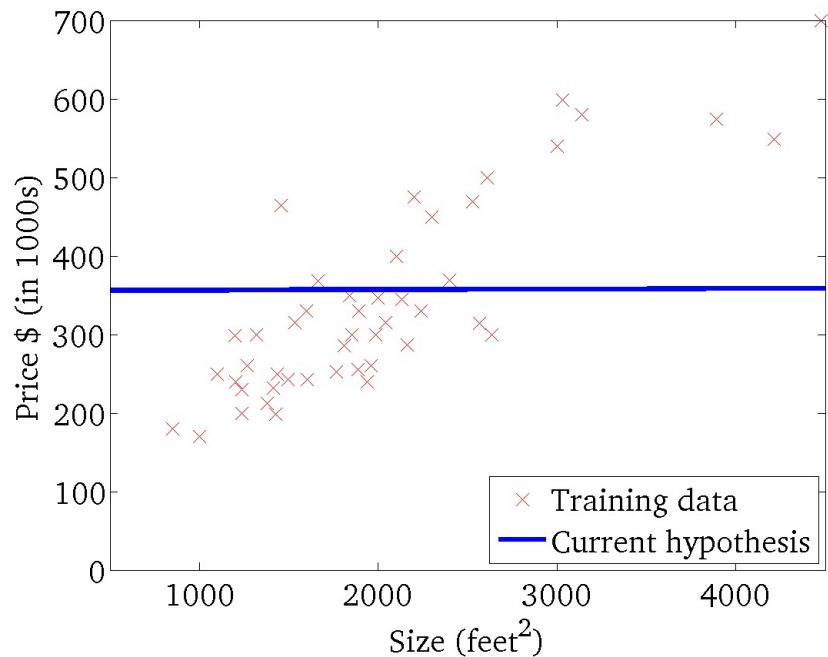
$$J(\theta_0, \theta_1)$$

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



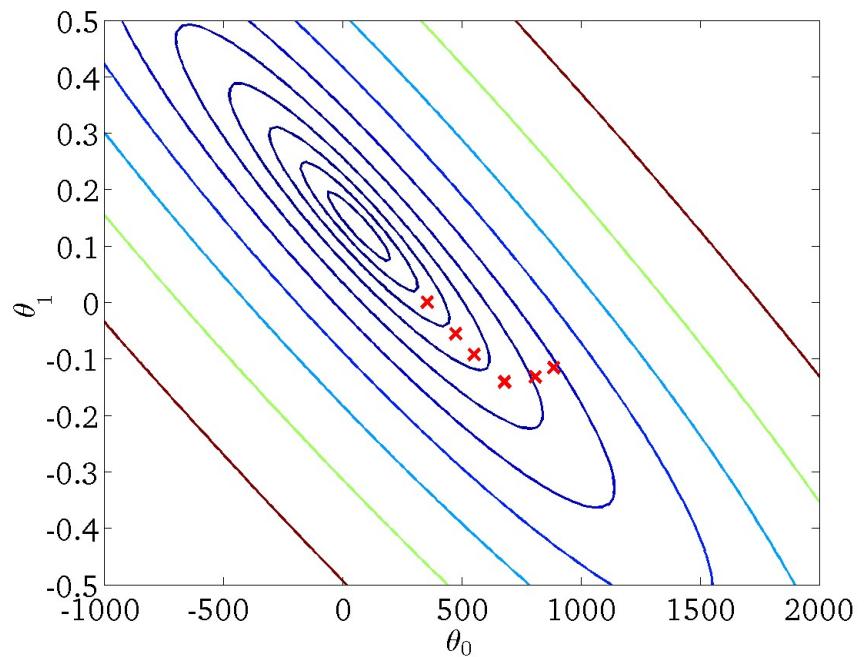
$$h_{\theta}(x)$$

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



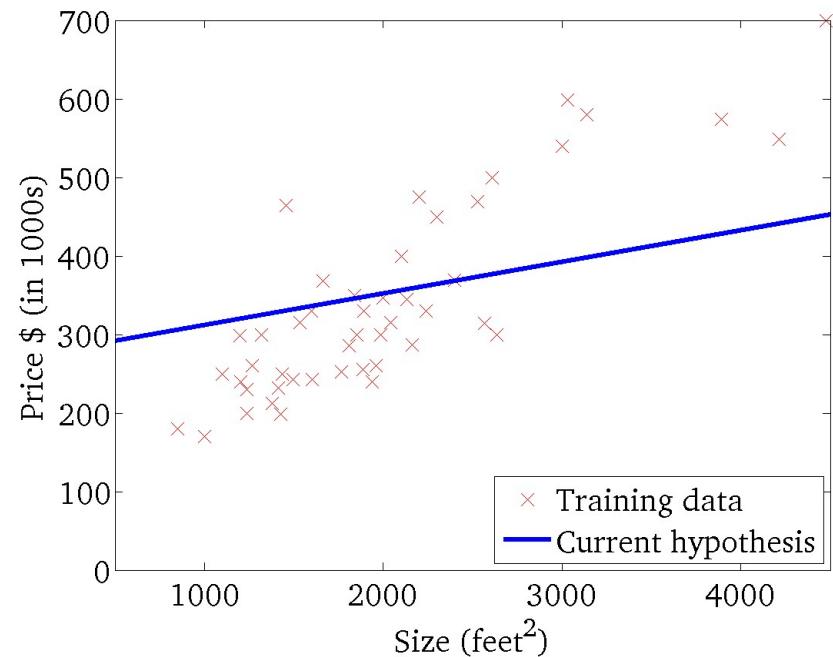
$$J(\theta_0, \theta_1)$$

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



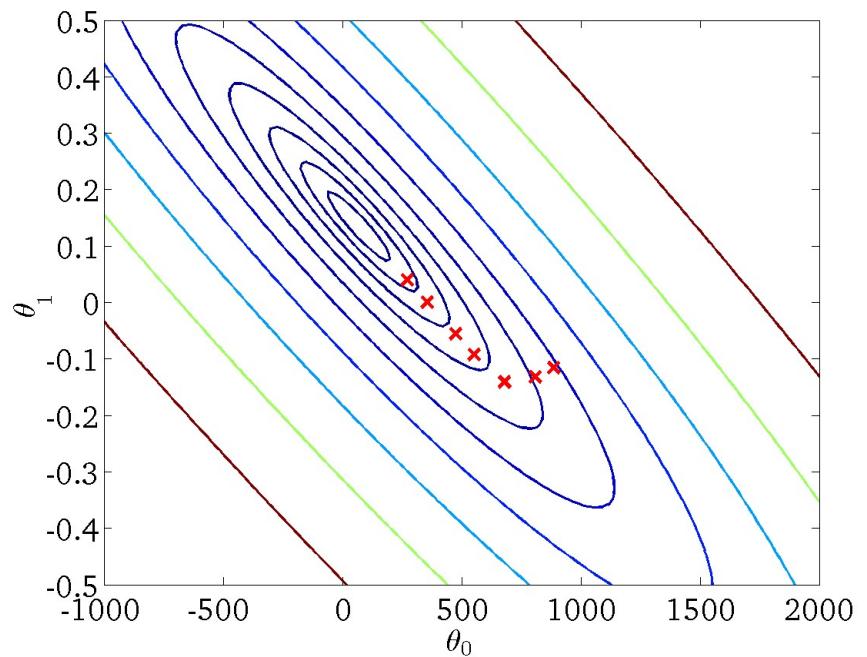
$$h_{\theta}(x)$$

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



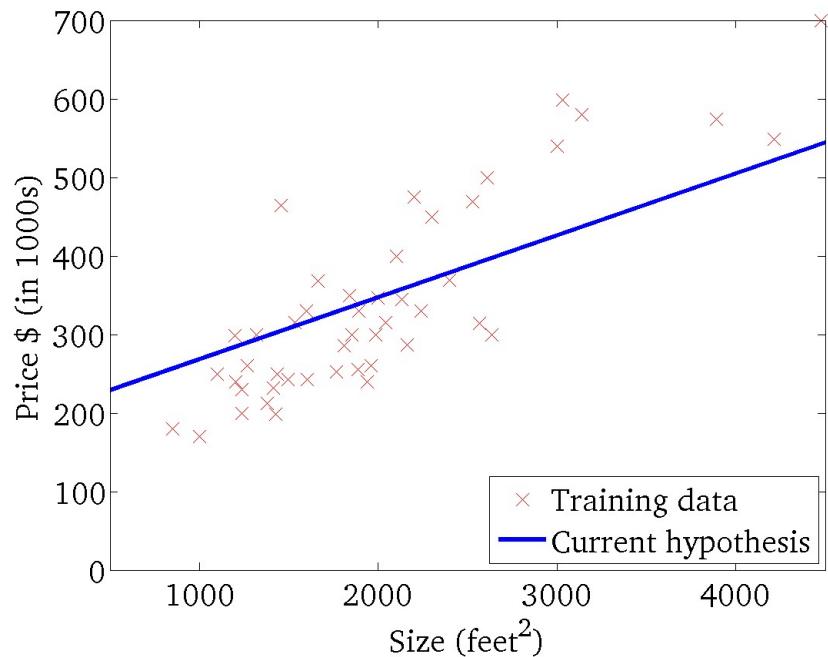
$$J(\theta_0, \theta_1)$$

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



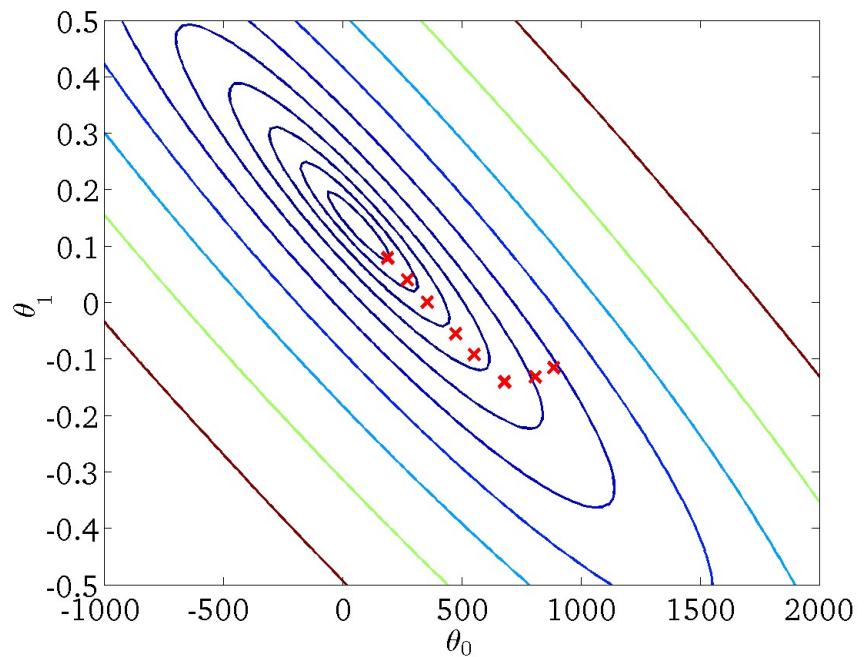
$$h_{\theta}(x)$$

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



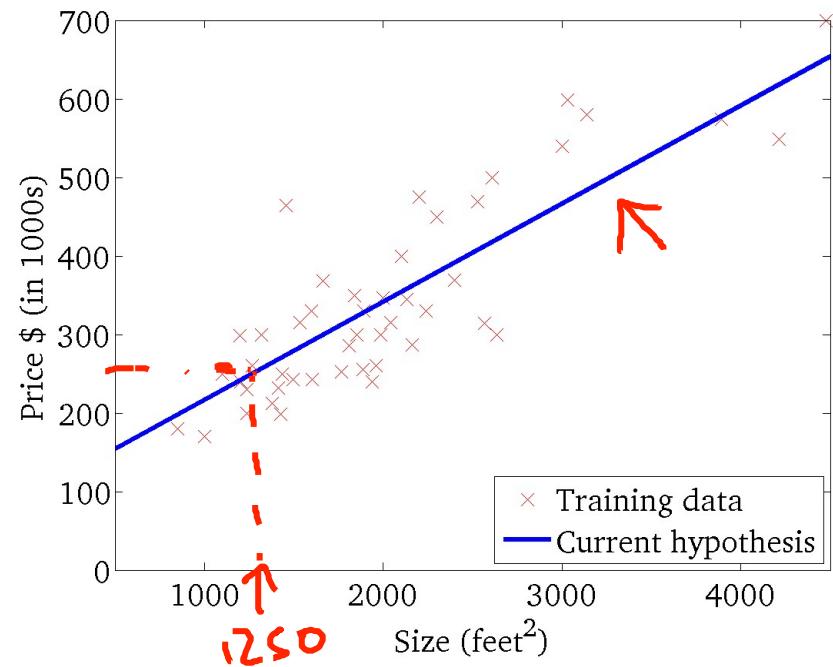
$$J(\theta_0, \theta_1)$$

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



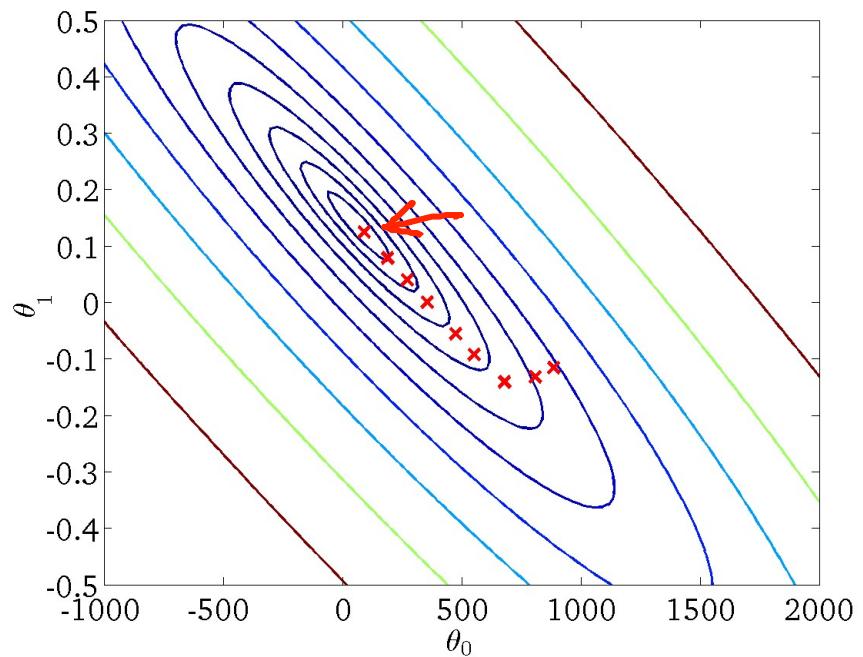
$$h_{\theta}(x)$$

(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.

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