

Regression

A Learning Problem

- A *computer program* is said to *learn* from experience T with respect to some task F and some performance measure P, if its *performance on F, as measured by P, improves with experience T.*

Regression

- Regression is the determination of a function $y=h_{\theta}(x)$ fitting a set of points $T=\{(x^{(i)}, y^{(i)}), i=1, \dots, m, x^{(i)} \in R^n, y^{(i)} \in R\}$.
- The function depends on a number of *unknown* parameters $\theta=[\theta_1, \theta_2, \dots, \theta_n]^T$

Regression framework

Learning phase

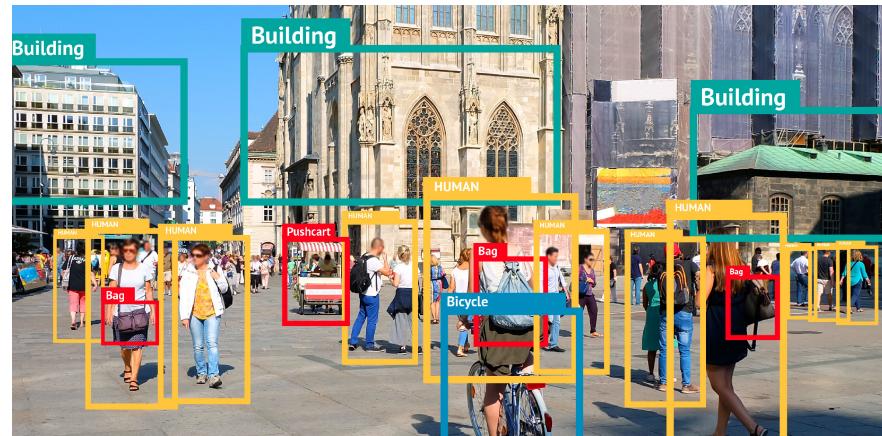
- Given a training set $T=\{(x^{(i)}, y^{(i)}), i=1, \dots, m, y^{(i)} \in R\}$
- Given some hypotheses regarding $h_{\theta}(x)$
- Estimate parameters $\theta=[\theta_1, \theta_2, \dots, \theta_n]^T$ of $h_{\theta}(x)$ by using T

Estimation phase

- Use the estimated parameters $\theta=[\theta_1, \theta_2, \dots, \theta_n]^T$ to determine $h_{\theta}(x)$ at new x

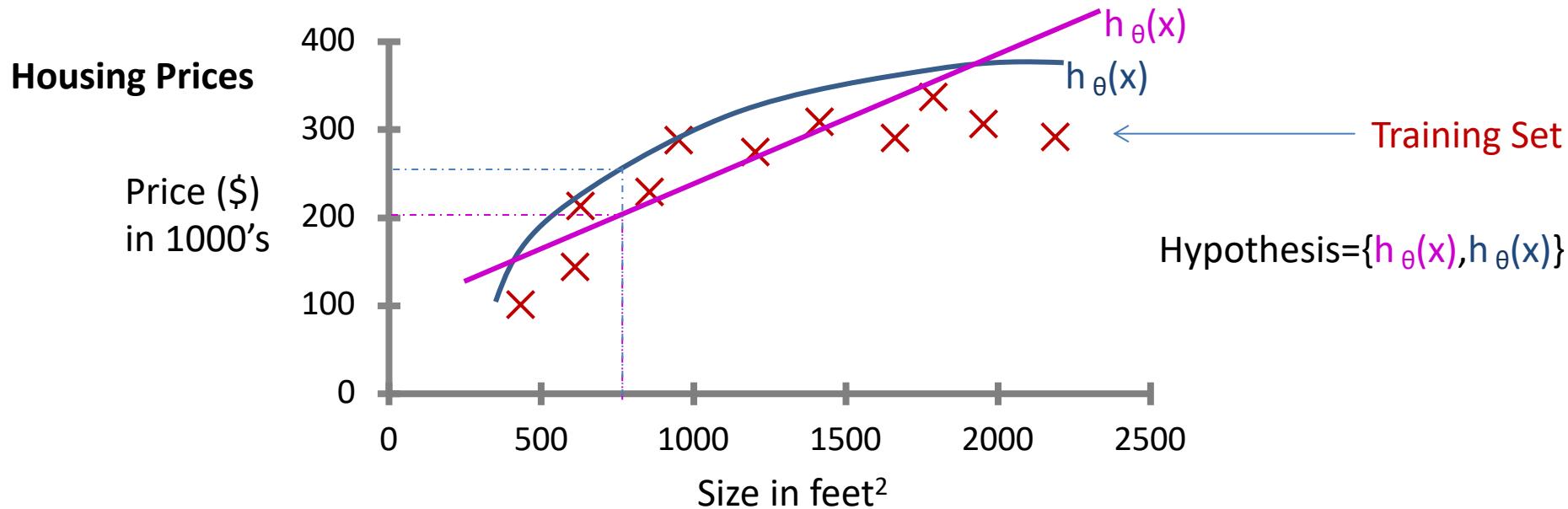
Examples

- Image Based Localization
- Object Detection
- Pose Estimation
- :
- :



Linear Regression with one Variable

Supervised Learning Problem: Regression



Supervised Learning

"right answers" are given for each example in the data

Regression: Predict continuous valued output (price)

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
...	...

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^{th} training example



What is $y^{(3)}$?

Training Set



Learning Algorithm

Size of house

x



hypothesis

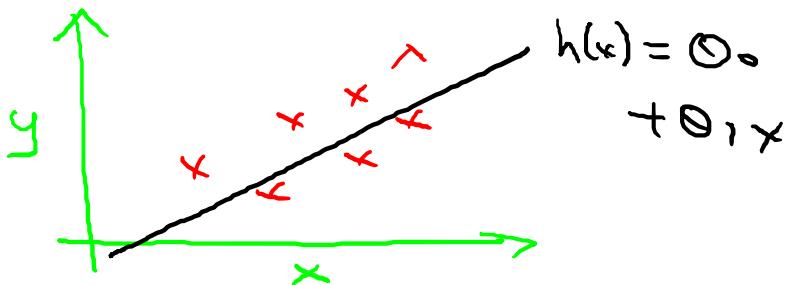
Estimated price
(estimated value of y)

h maps from x's to y's.

How do we represent h in Regression?

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

Shorthand: $h(x)$



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

One variable

Training Set

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

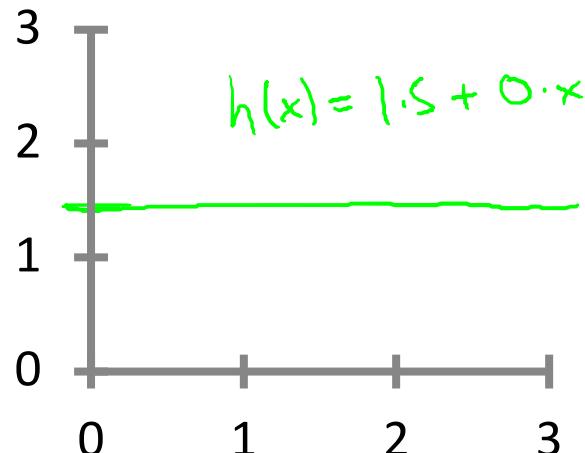
Hypothesis - Model: $h_{\theta}(x) = \theta_0 + \theta_1 x$

θ_i 's: Parameters

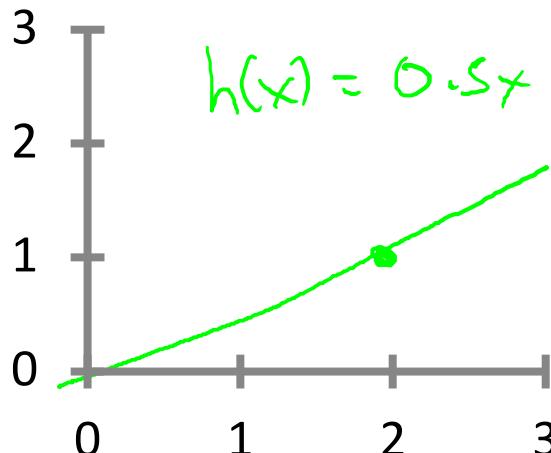
How to choose θ_i 's ?

By choosing different parameters we have different hypothesis

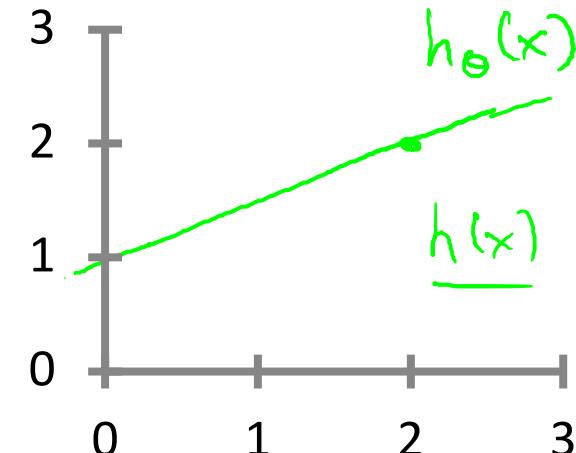
$$\underline{h_{\theta}(x) = \theta_0 + \theta_1 x}$$



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



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

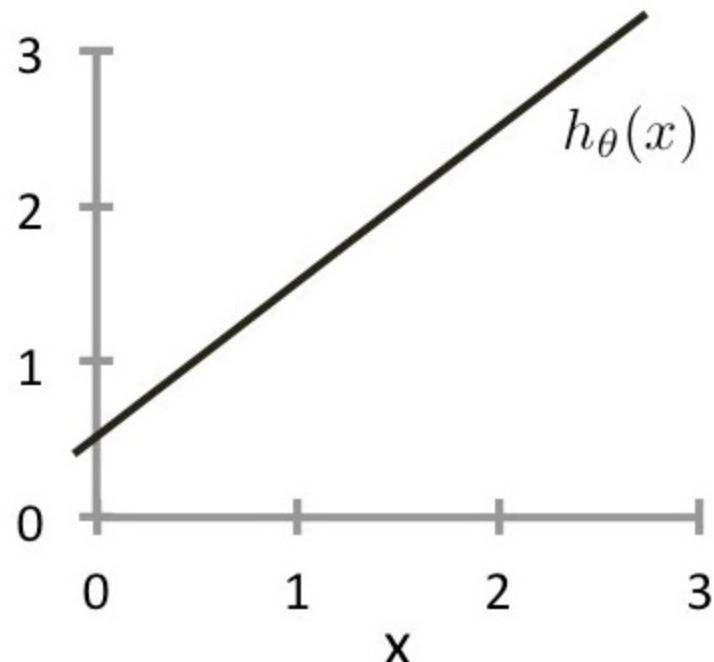


$$\begin{aligned}\rightarrow \theta_0 &= 1 \\ \rightarrow \theta_1 &= 0.5\end{aligned}$$

Consider the plot below of $h_{\theta}(x) = \theta_0 + \theta_1 x$. What are θ_0 and θ_1 ?

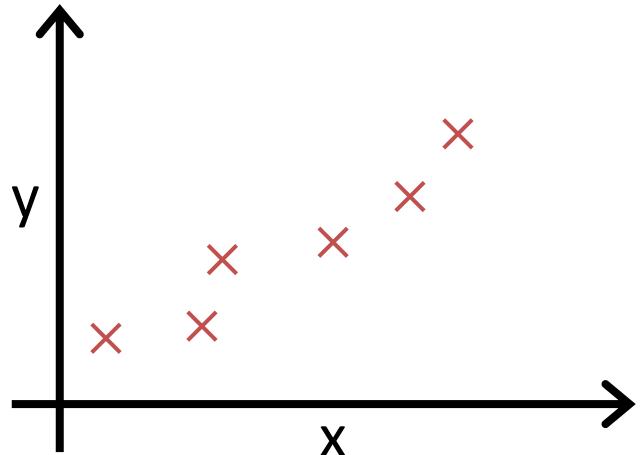


- $\theta_0 = 0, \theta_1 = 1$
- $\theta_0 = 0.5, \theta_1 = 1$
- $\theta_0 = 1, \theta_1 = 0.5$
- $\theta_0 = 1, \theta_1 = 1$



The Cost Function

The Cost Function J



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

#training examples

\uparrow

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

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

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

x, y

Minimize $J(\theta_0, \theta_1)$
 θ_0, θ_1

Squared error function

What cost function is doing and why we want to use it?

Linear Regression Problem (one variable)

Hypothesis - Model:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

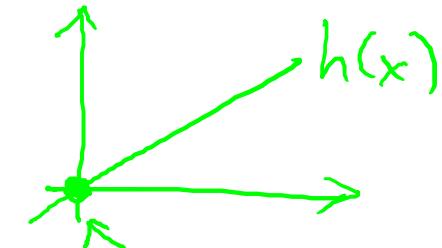
Goal: minimize $J(\theta_0, \theta_1)$

Simplified

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

$$\Theta_0 = 0$$

$$\theta_1$$

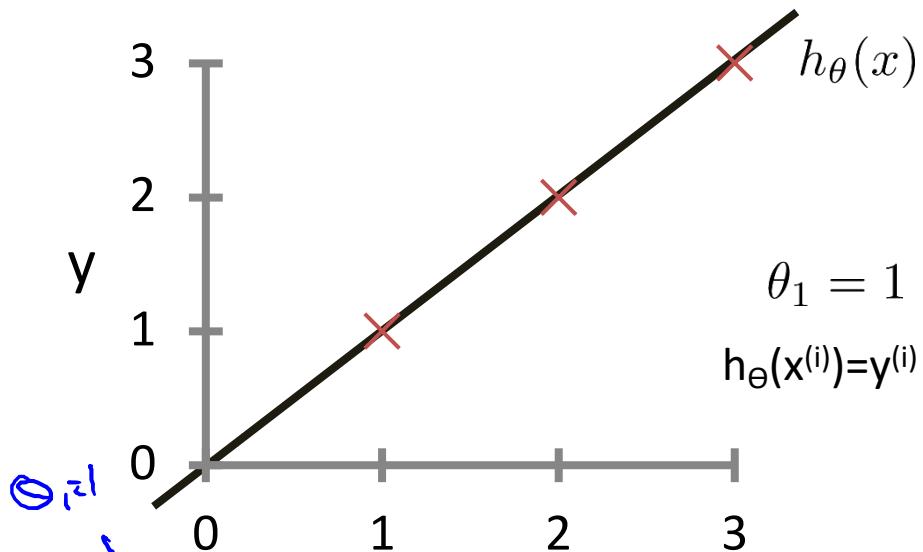


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

minimize $\underset{\theta_1}{J(\theta_1)}$

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

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

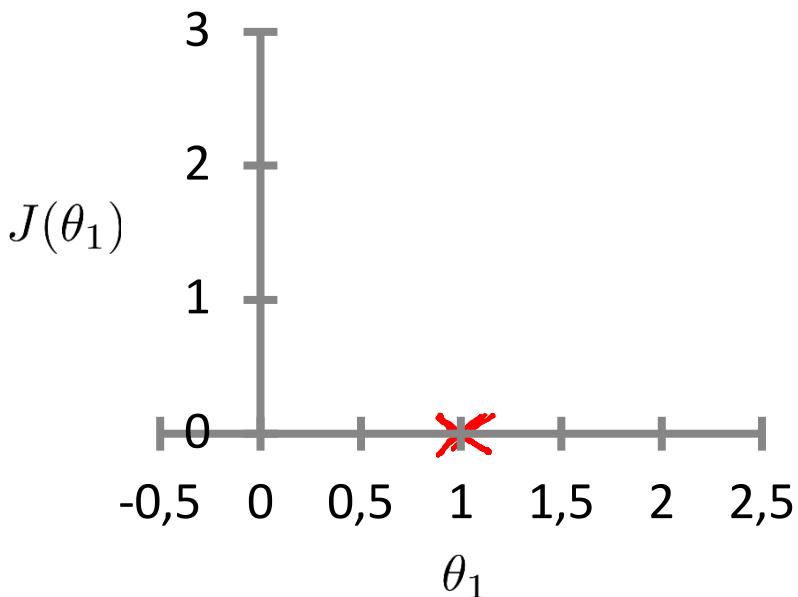


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{2m} \sum_{i=1}^m (\theta_1 x^{(i)} - y^{(i)})^2 = \frac{1}{2m} (6^2 + 0^2 + 0^2) = 0$$

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

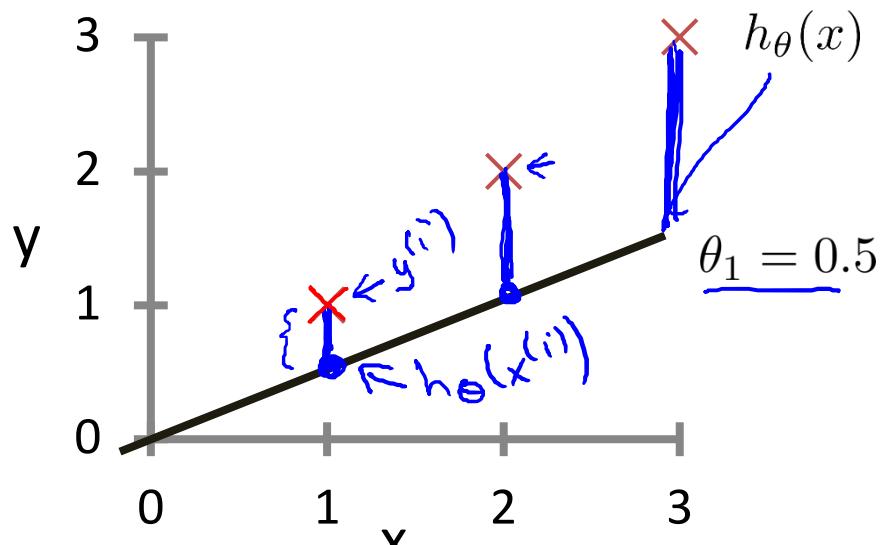
(function of the parameter θ_1)



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

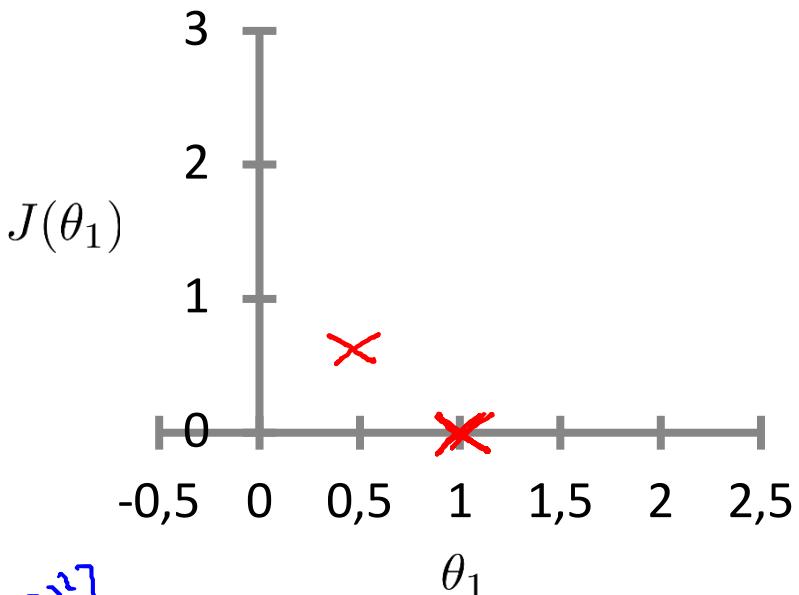
$$h_{\theta}(x) = \theta_1 x$$

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



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

(function of the parameter θ_1)

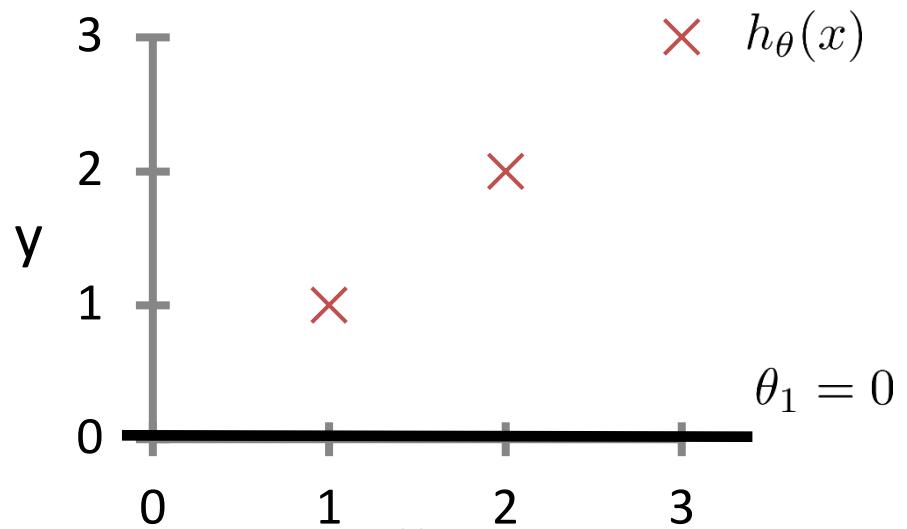


$$J(0.5) = \frac{1}{2m} [(0.5-1)^2 + (1-2)^2 + (1.5-3)^2]$$

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

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

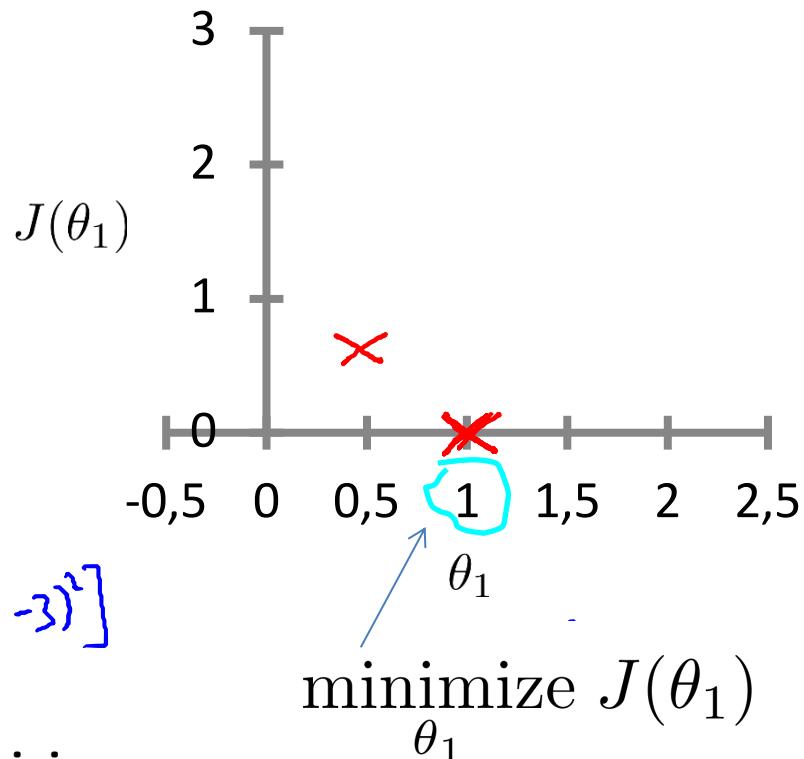
(for fixed θ_1 , this is a function of x)



$$J(0) = \frac{1}{2m} \left[(-1)^2 + (-2)^2 + (-3)^2 \right]$$

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

(function of the parameter θ_1)



We need a procedure to automatically find the minimum

Original Problem

Hypothesis:

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

Parameters:

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

Cost Function:

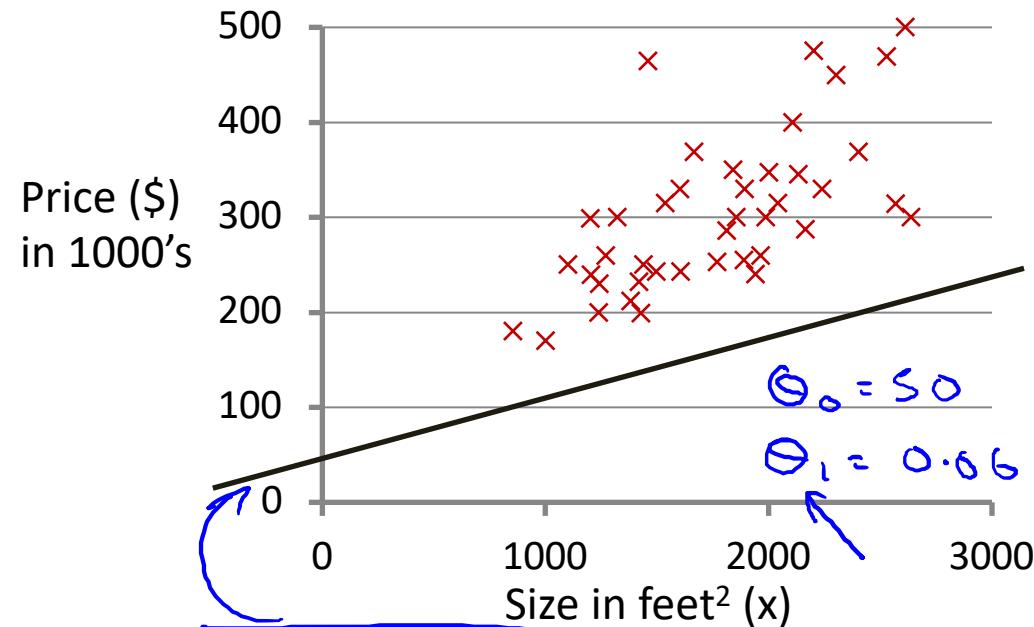
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Goal:

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

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

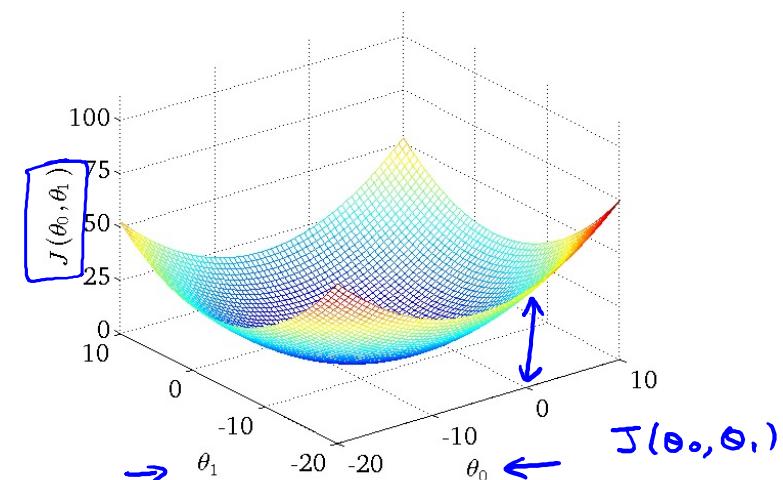
(for fixed θ_0, θ_1 , this is a function of x)



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

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

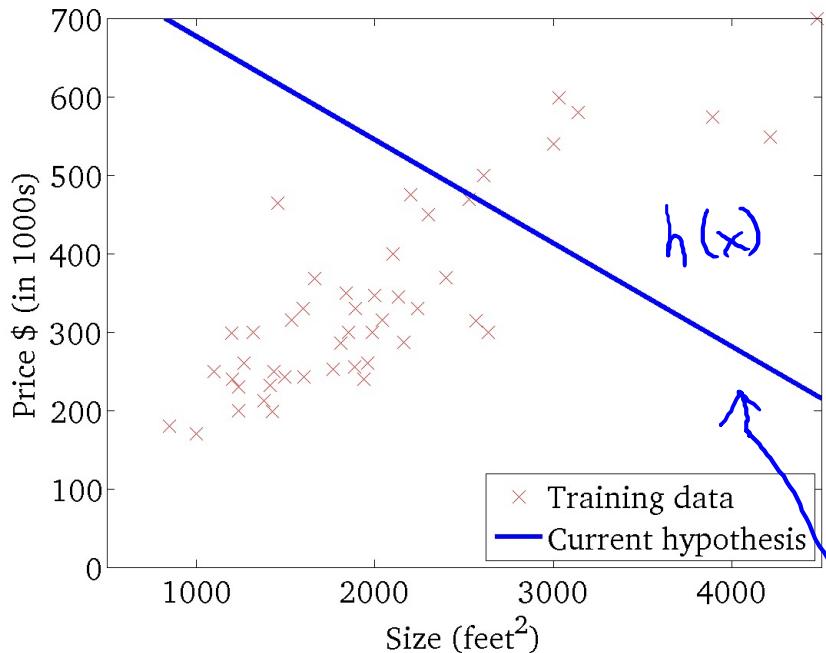
(function of the parameters θ_0, θ_1)



Note that we have just one global minimum.
Contour Plots....

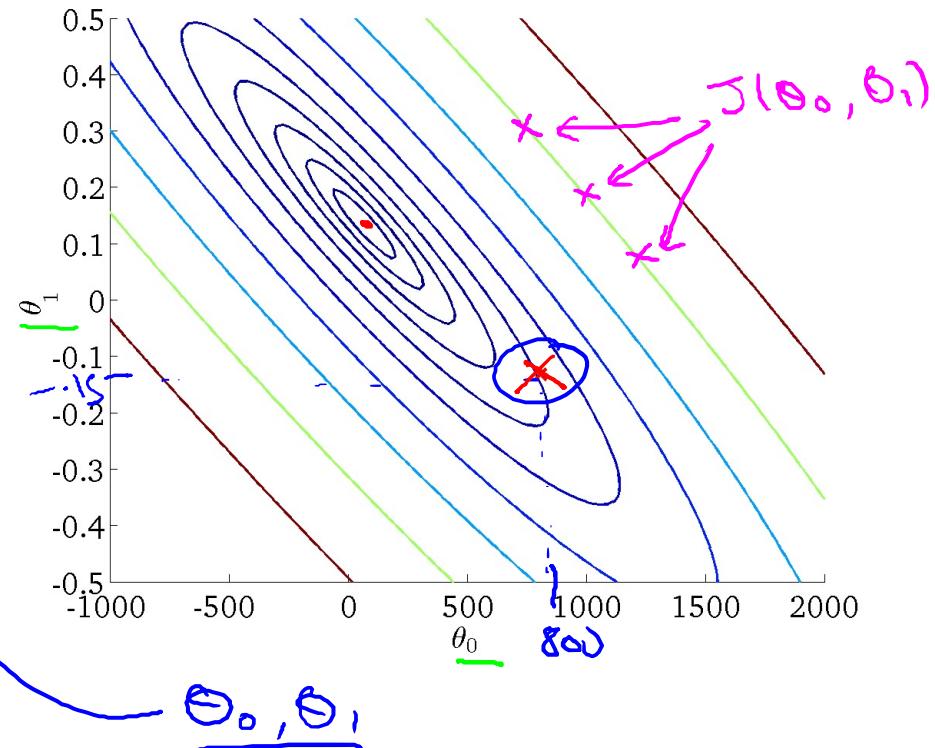
$$h_{\theta}(x)$$

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



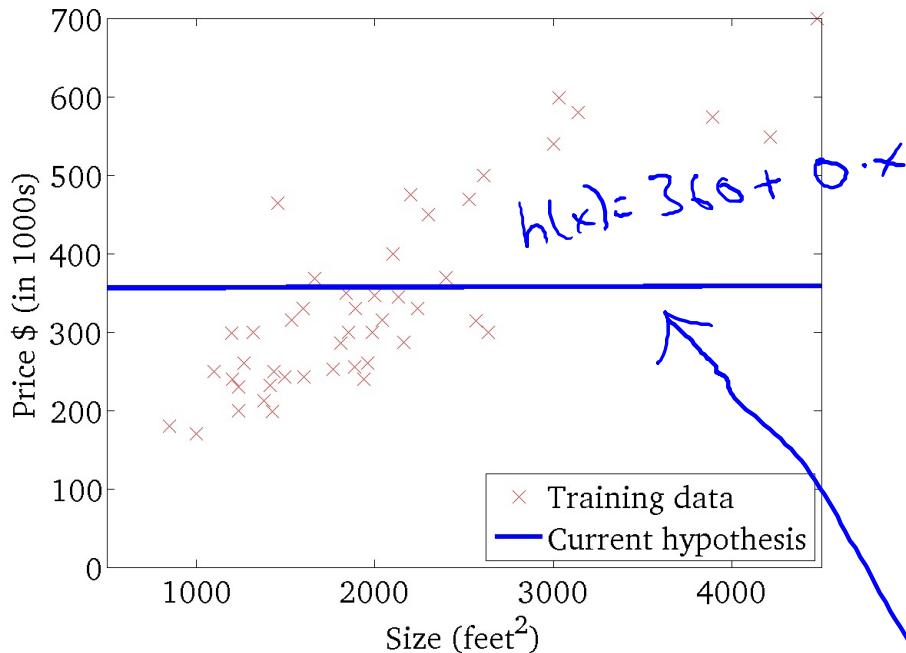
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



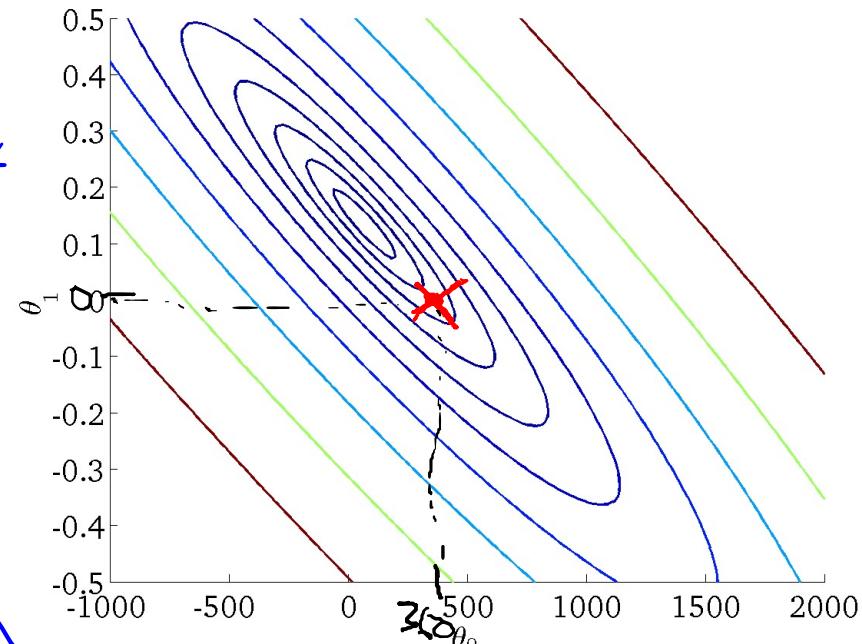
$$h_{\theta}(x)$$

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



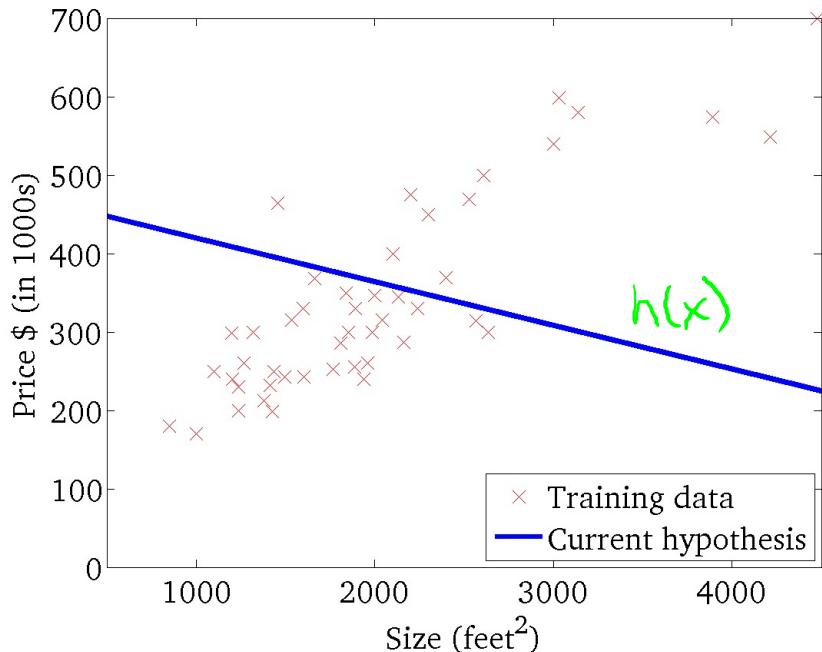
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



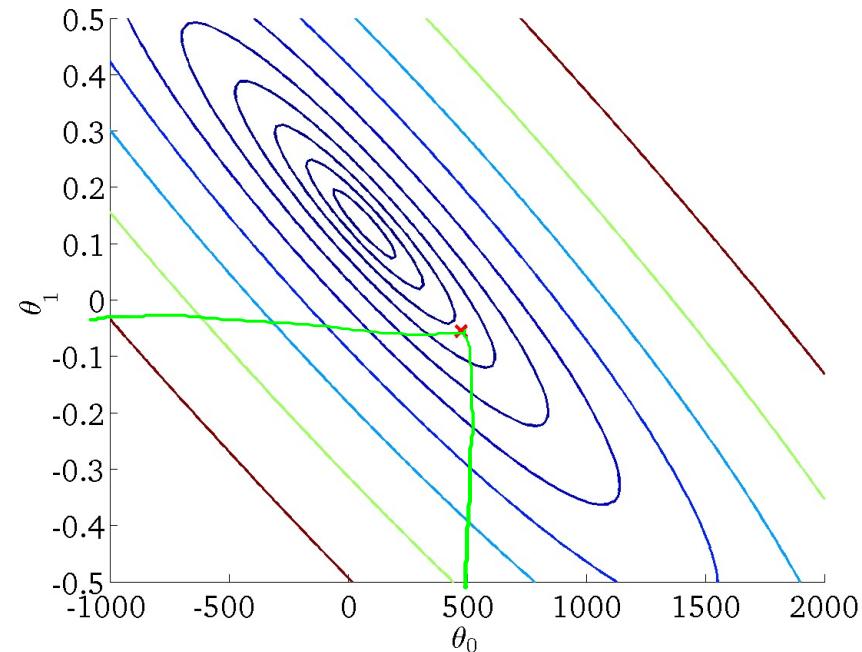
$$h_{\theta}(x)$$

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



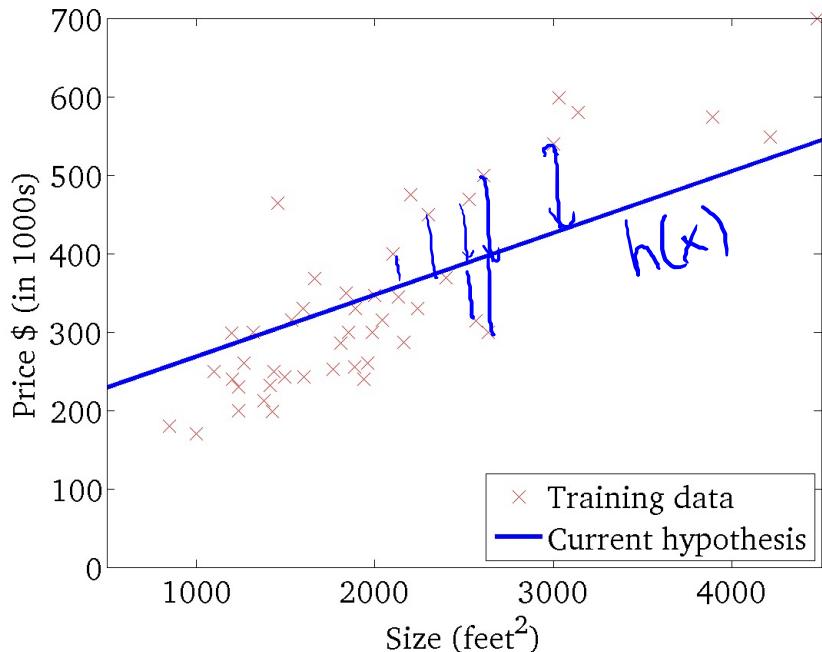
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



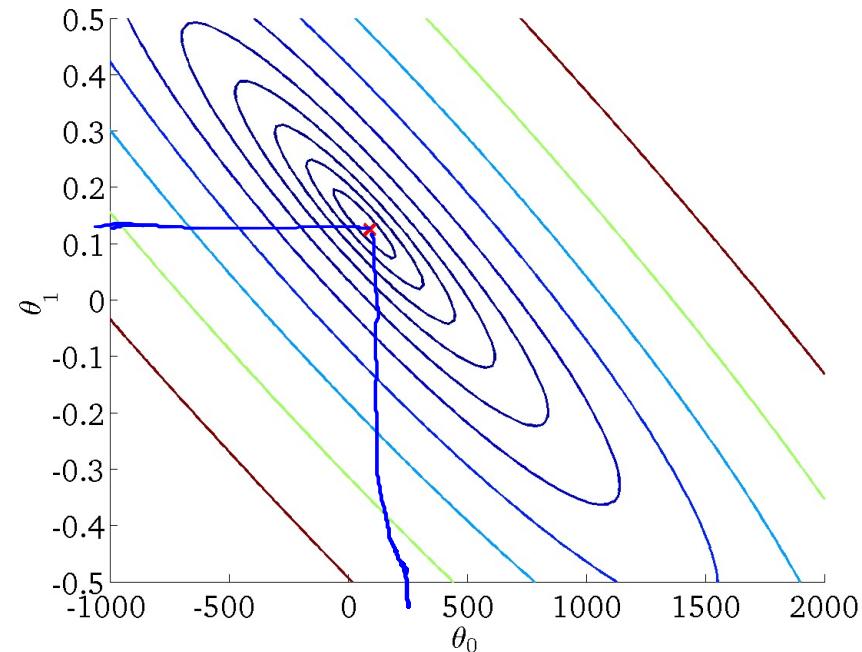
$$h_{\theta}(x)$$

(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)



**We want an algorithms that find automatically
the best values for the parameters.**

Do you know any naive algorithm for this?

Gradient Descent

(can be used not only for linear regression – useful to minimize functions)

Problem Setup

Gradient Descend it is useful
also to more general function

Have some arbitrary function $J(\theta_0, \theta_1)$

Want an algorithm for

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

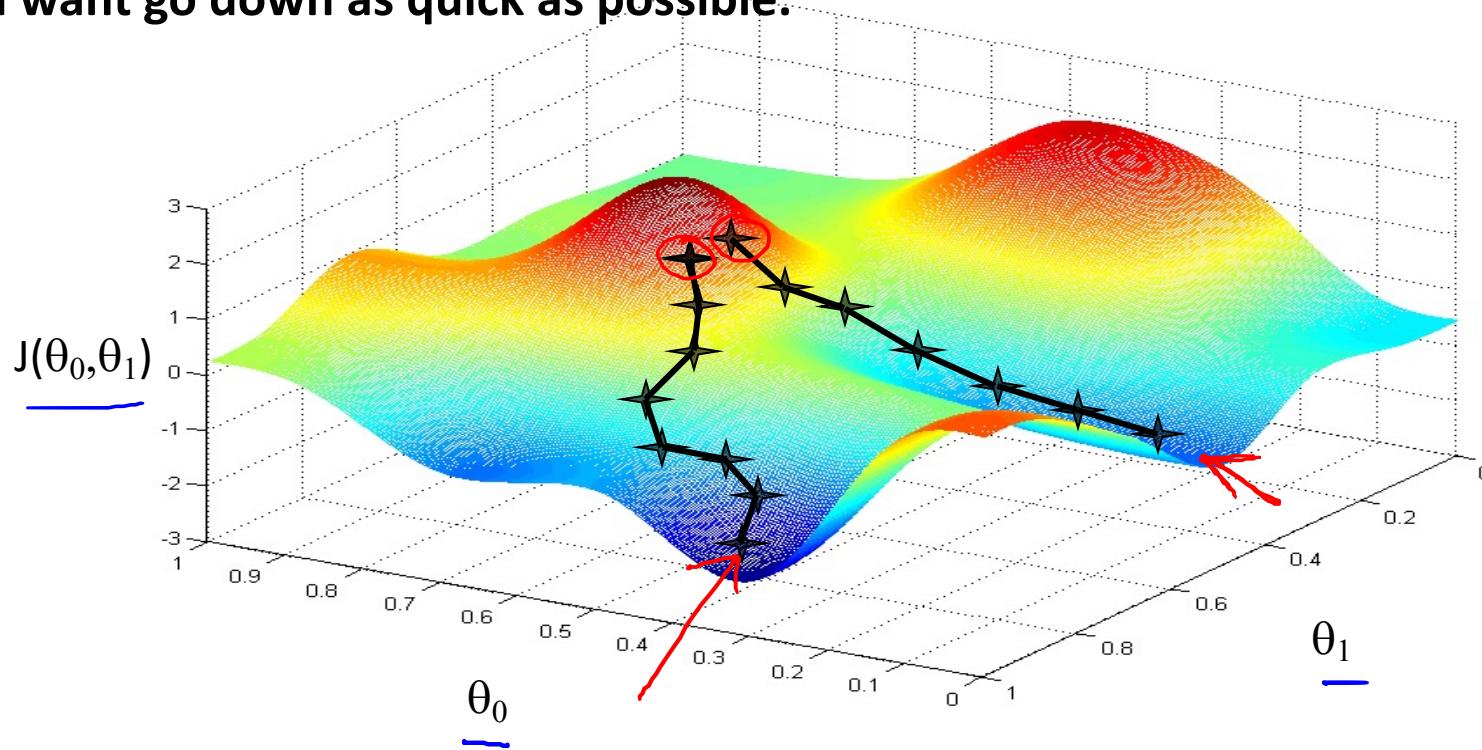
Outline of the Gradient Descent algorithm:

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

Suppose you stand in that point, look around 360°
and choose a direction to perform a step down.

General $J(\theta_1, \theta_2)$ function

You want go down as quick as possible.



Starting from a different point a different minimum is reached.

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

≥ 0

derivative

$:=$ denote assignment operator

Learning rate controls how big is the step

Derivative term give the direction of the step
(we will derive later – do not be scared! ;-)

Gradient descent algorithm

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

Correct: Simultaneous update

```
temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$   
temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$   
 $\theta_0 := \text{temp0}$   
 $\theta_1 := \text{temp1}$ 
```

Incorrect:

```
temp0 :=  $\theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$   
 $\theta_0 := \text{temp0}$   
temp1 :=  $\theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$   
 $\theta_1 := \text{temp1}$ 
```

We have to understand what the gradient descend algorithm is doing and the role of the learning rate and of the derivative term

repeat until convergence {

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

↑ ↑
learning derivative
rate
 ≥ 0

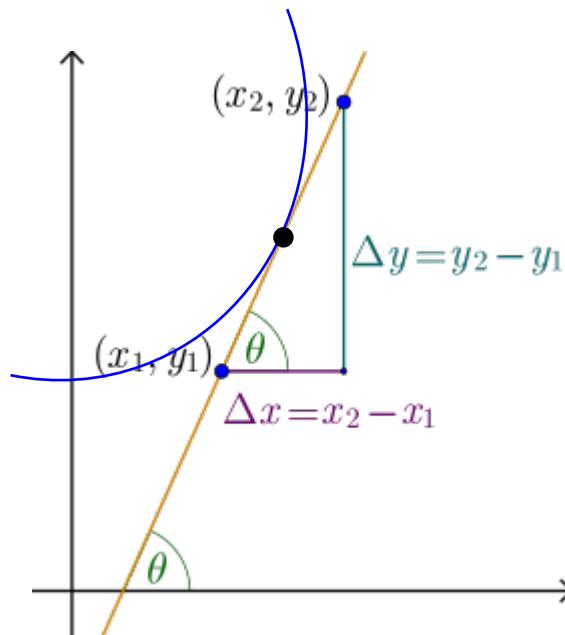
(simultaneously update
 $j = 0$ and $j = 1$)

To figure out this let consider the simplified case:

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

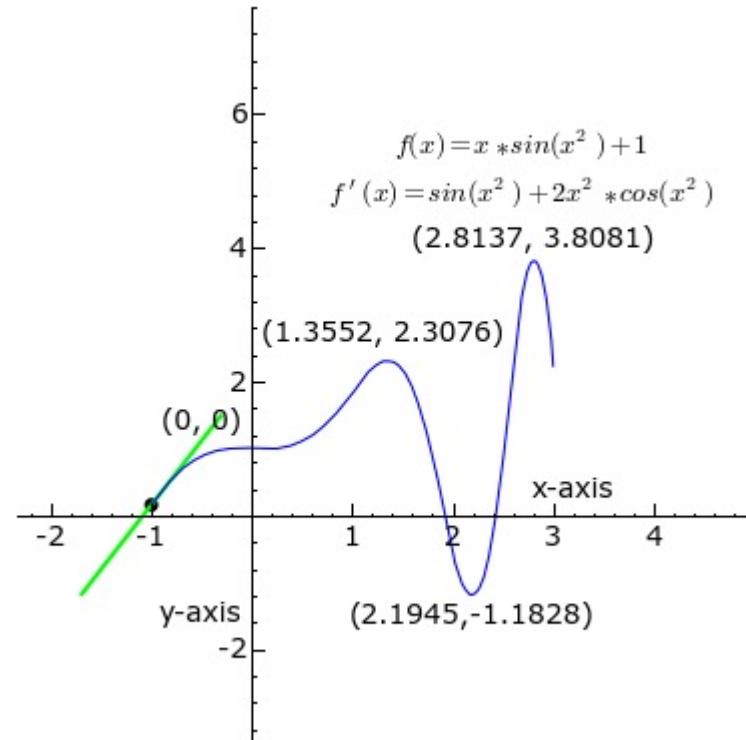
$$\theta_1 \in \mathbb{R}$$

Gradient Descent Intuition (simplified case $J(\theta_1)$) – What is the derivative



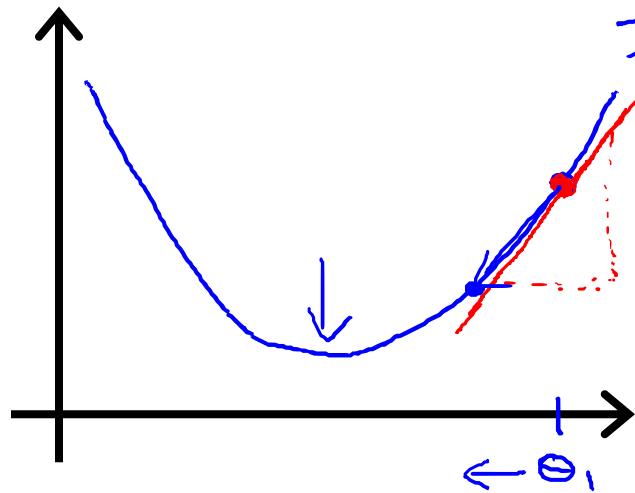
Coefficiente angolare

$$m = \frac{\Delta y}{\Delta x} = \tan(\theta)$$



At each point, the line is always tangent to the curve. Its slope is the derivative; the positivity, negativity and zeroes of the derivative are marked by green, red and black respectively.

Gradient Descent Intuition (simplified case)



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

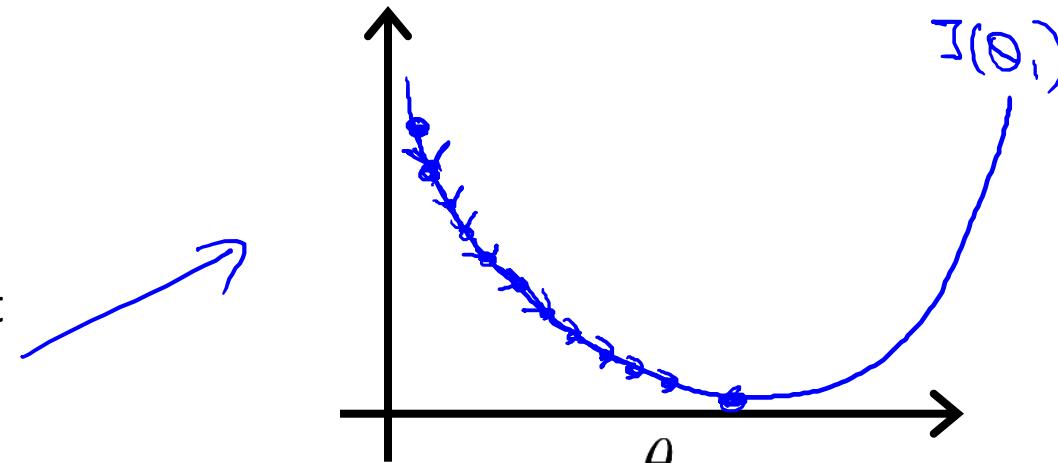
$$\theta_1 := \theta_1 - \cancel{\alpha} \cdot \frac{\partial}{\partial \theta_1} J(\theta_1)$$

≥ 0

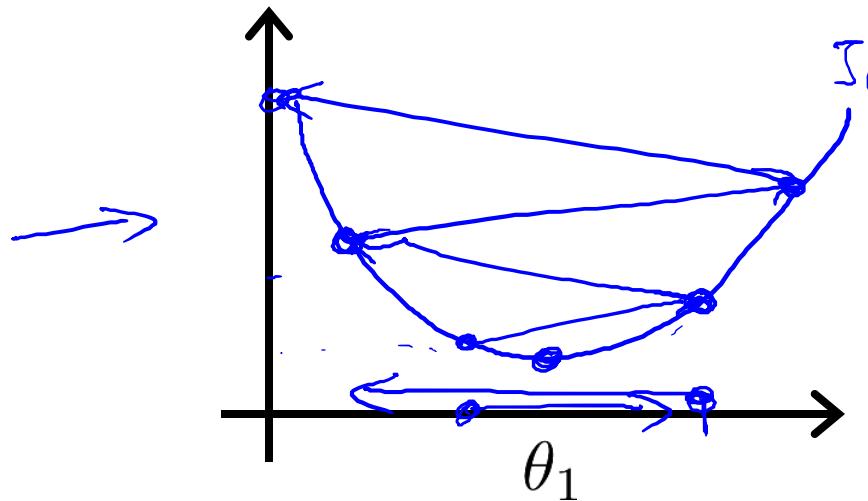
$$\theta_1 := \theta_1 - \underline{\alpha} \cdot (\text{positive number})$$

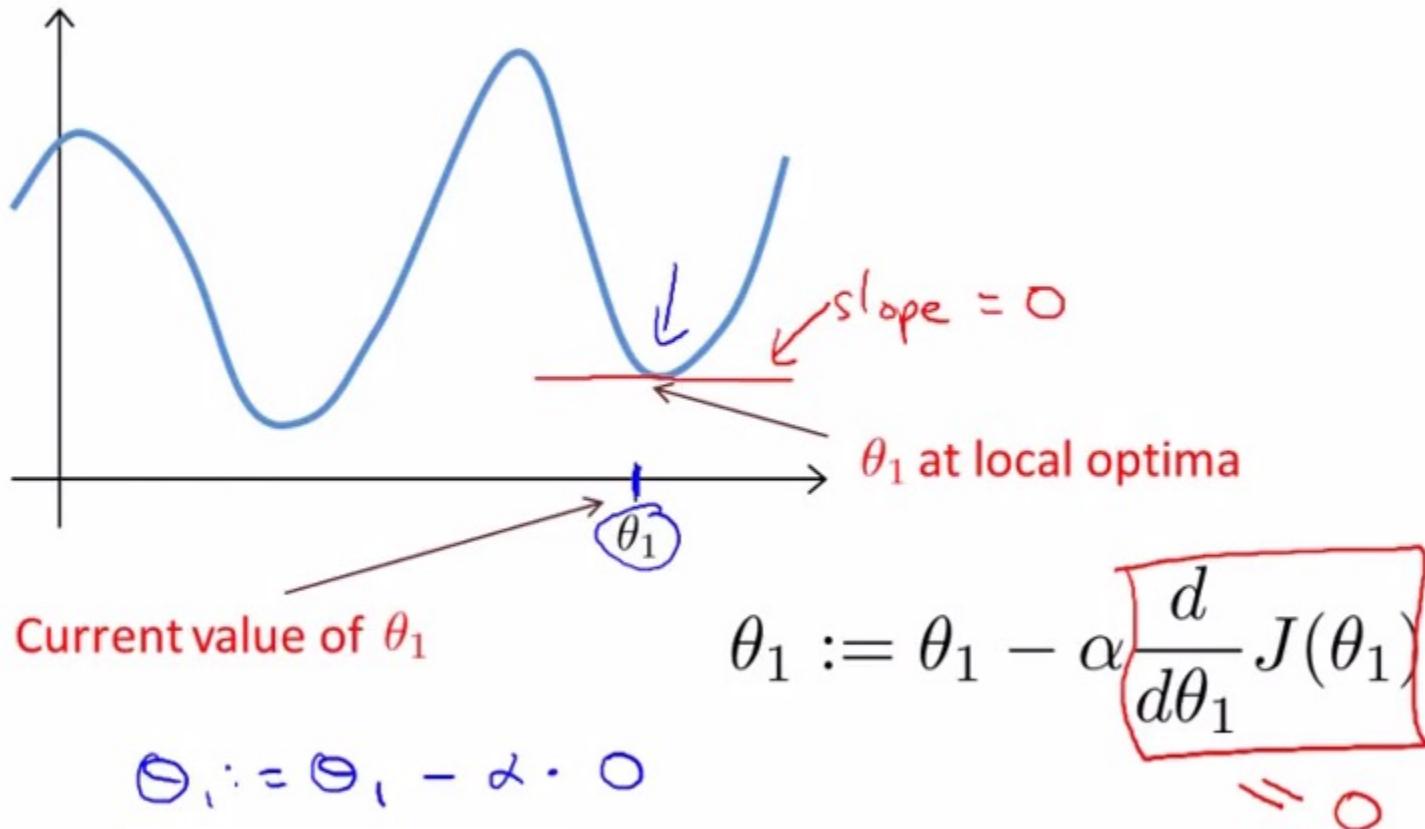
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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.

