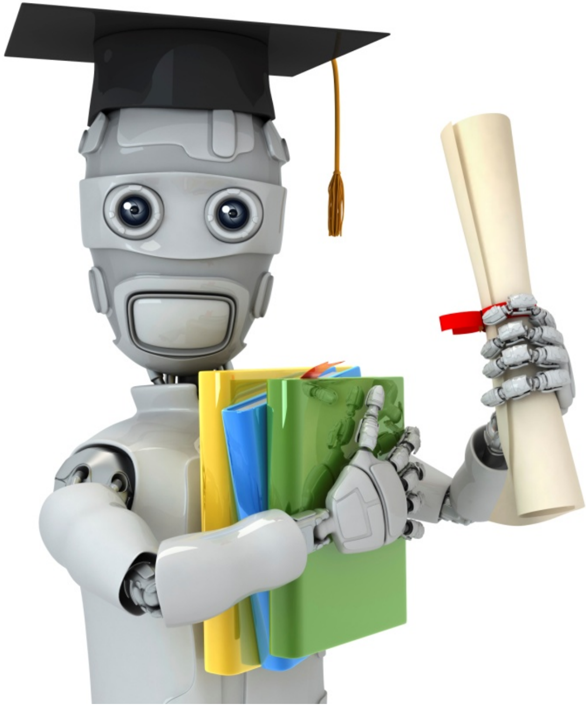
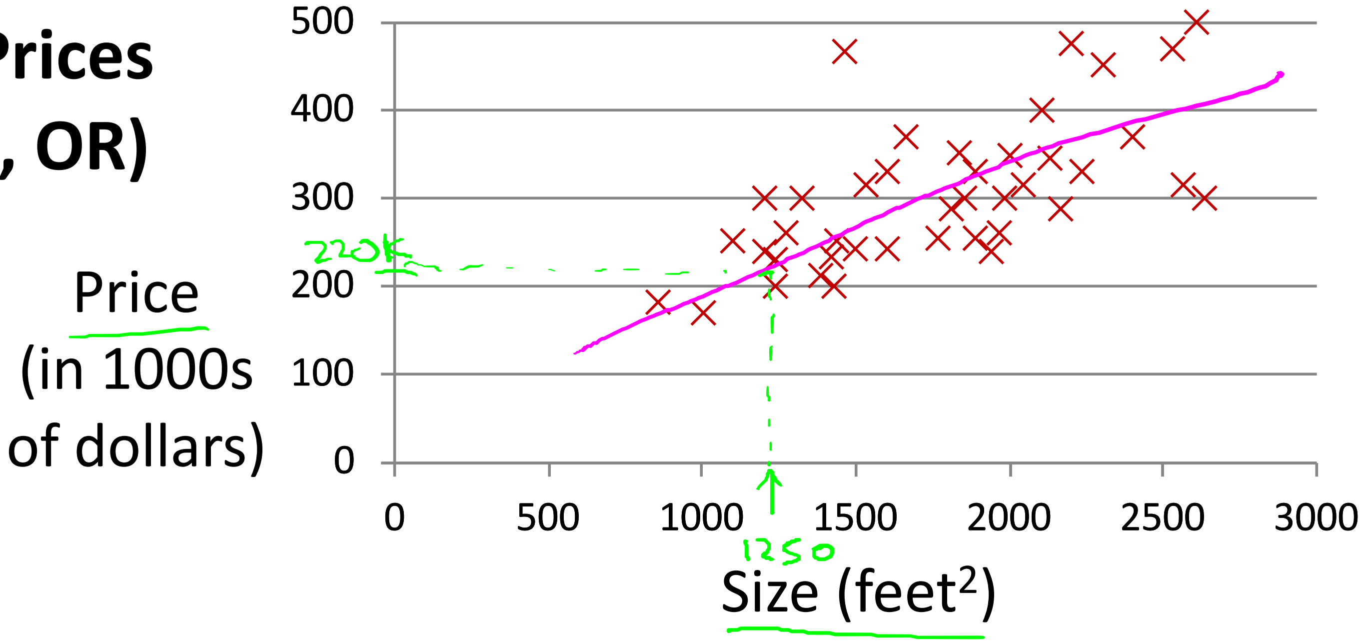


Machine Learning

Linear regression with one variable

Model representation

Housing Prices (Portland, OR)



<u>Supervised Learning</u>

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

<u>Regression Problem</u>

Predict real-valued output

Classification: Discrete-valued output

Andrew Ng

Training set of housing prices (Portland, OR)

Size in feet² (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^{th} training example

$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

Training Set

↓

Learning Algorithm

↓

Size of house → h → Estimated price

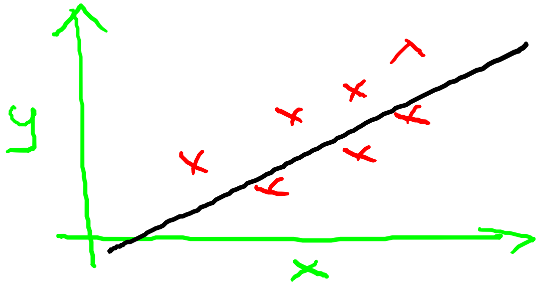
x $\quad$ hypothesis $\quad$ (estimated value of y)

h maps from x's to y's.

How do we represent h ?

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

Shorthand: $h(x)$



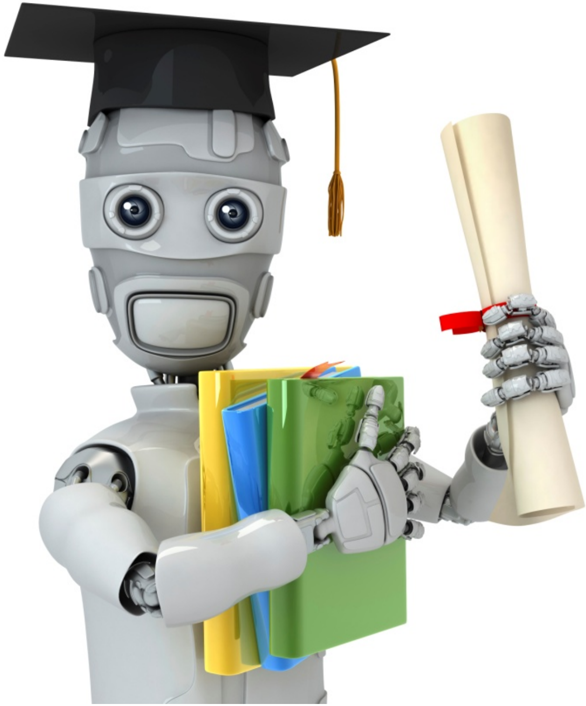
$h(x) = \theta_0 + \theta_1 x$

Linear regression with one variable. (x)

Univariate linear regression.

└ one variable

Andrew Ng



Machine Learning

Linear regression with one variable

Cost function

Training Set

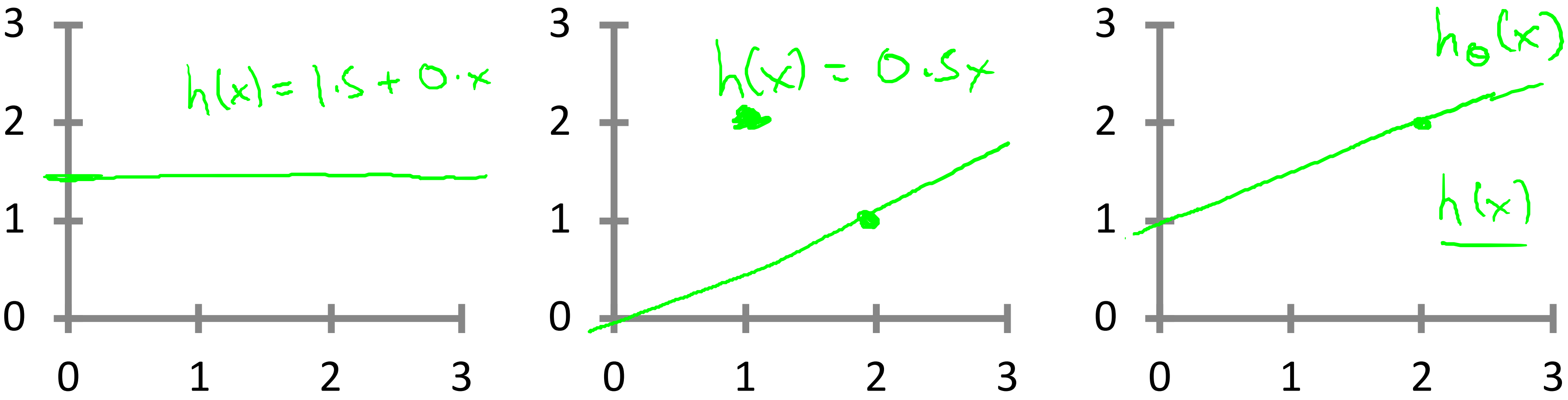
Size in feet^2 (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

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

θ_i's: Parameters

How to choose θ_i's ?

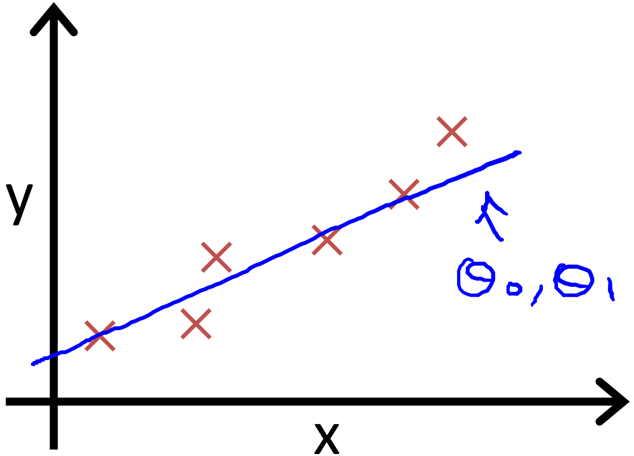
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$

$\theta_0 = 0$
$\theta_1 = 0.5$

$\theta_0 = 1$
$\theta_1 = 0.5$



$(x^{(i)}, y^{(i)})$

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

\# training examples

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

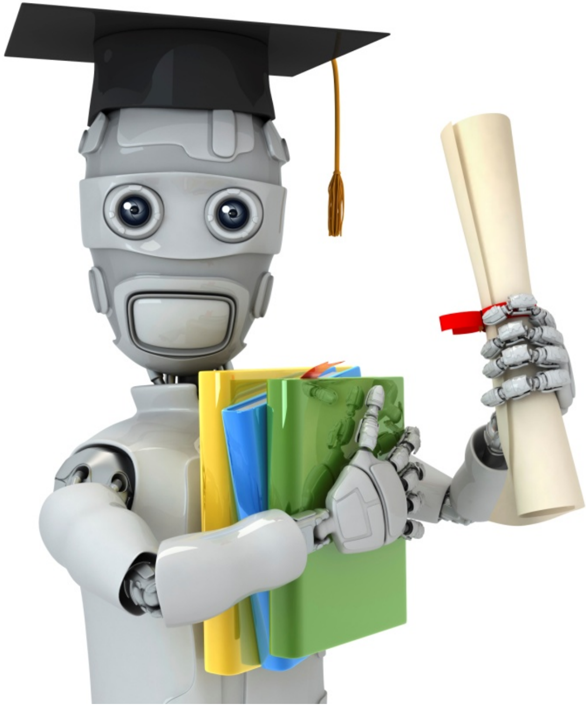
$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

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

$$\underset{\theta_0, \theta_1}{\text{minimize}} \; \underbrace{J(\theta_0, \theta_1)}_{\text{Cost function}}$$

Squared error function

Andrew Ng



Machine Learning

Linear regression with one variable

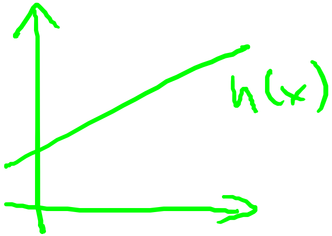
Cost function intuition I

Hypothesis:

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

Parameters:

$$\theta_0, \theta_1$$



Cost Function:

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

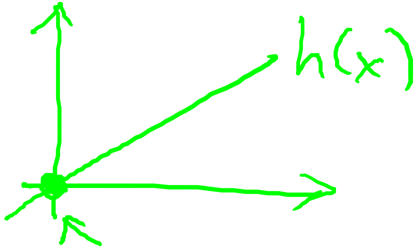
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$

Simplified

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

$\theta_0 = 0$

$$\theta_1$$



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

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

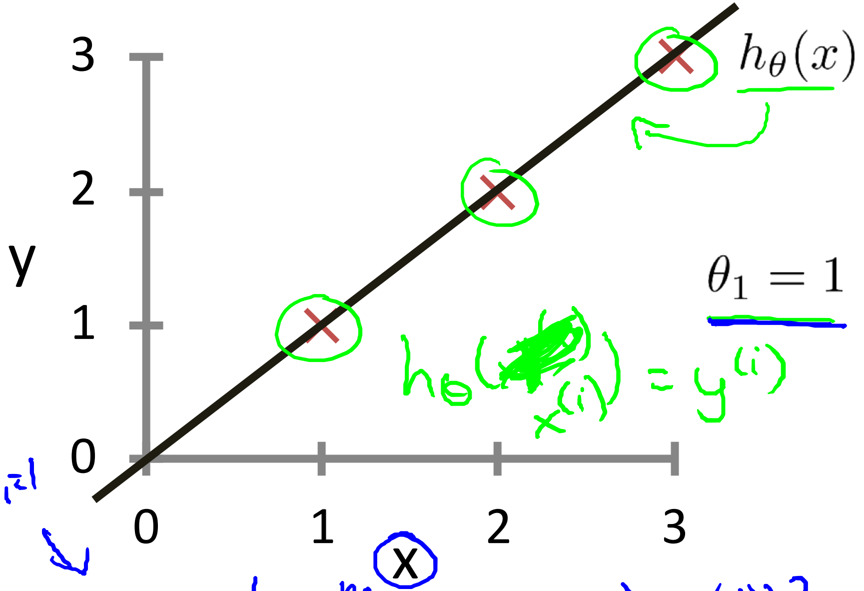
$\underset{\theta_1}{\text{minimize}} \; J(\theta_1)$

$h_\theta(x)$

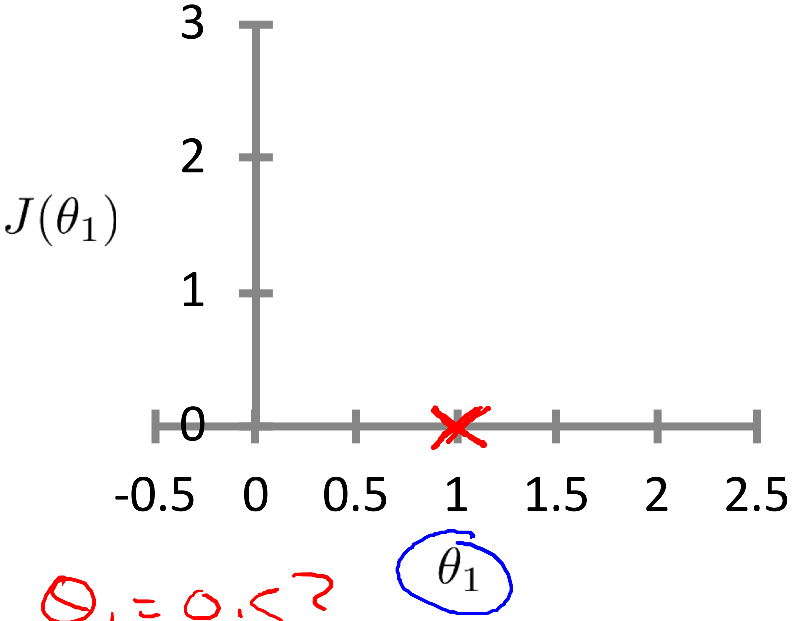
(for fixed θ_1, this is a function of x)

$J(\theta_1)$

(function of the parameter θ_1)



$\theta_1 = 1$



$\theta_1 = 0.5$?

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

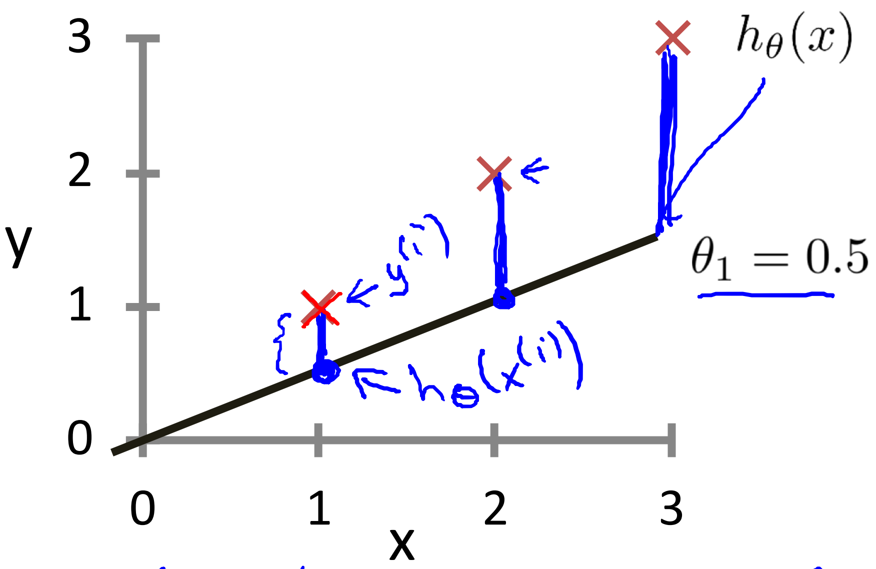
$$= \frac{1}{2m} \sum_{i=1}^{m} \left(\theta_1 x^{(i)} - y^{(i)} \right)^2 = \frac{1}{2m}\left(0^2 + 0^2 + 0^2\right) = 0^2$$

$J(1) = 0$

Andrew Ng

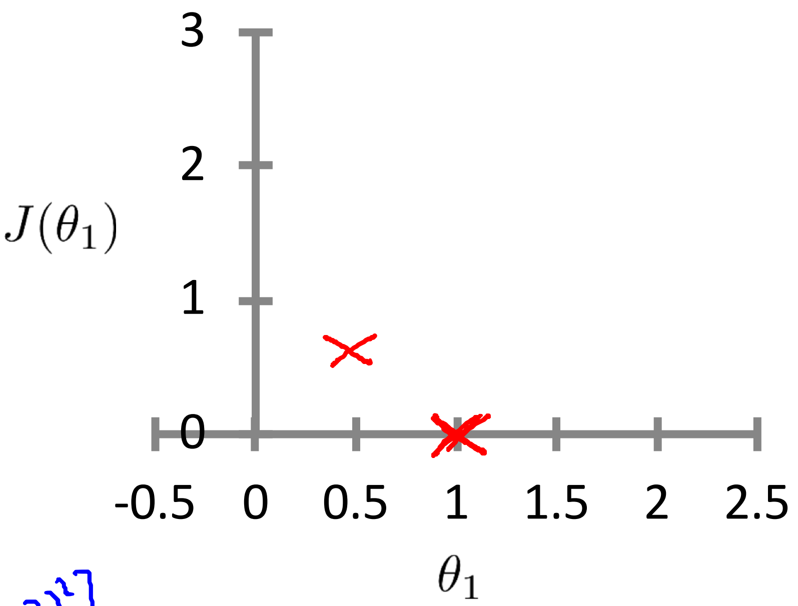
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



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

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

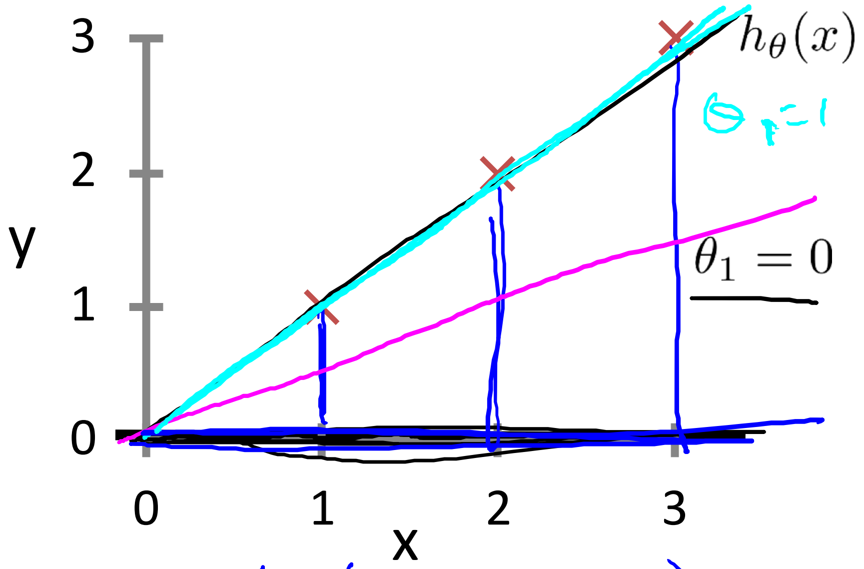
$\theta_1 = 0$?

$J(0) = $?

Andrew Ng

$h_\theta(x)$

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

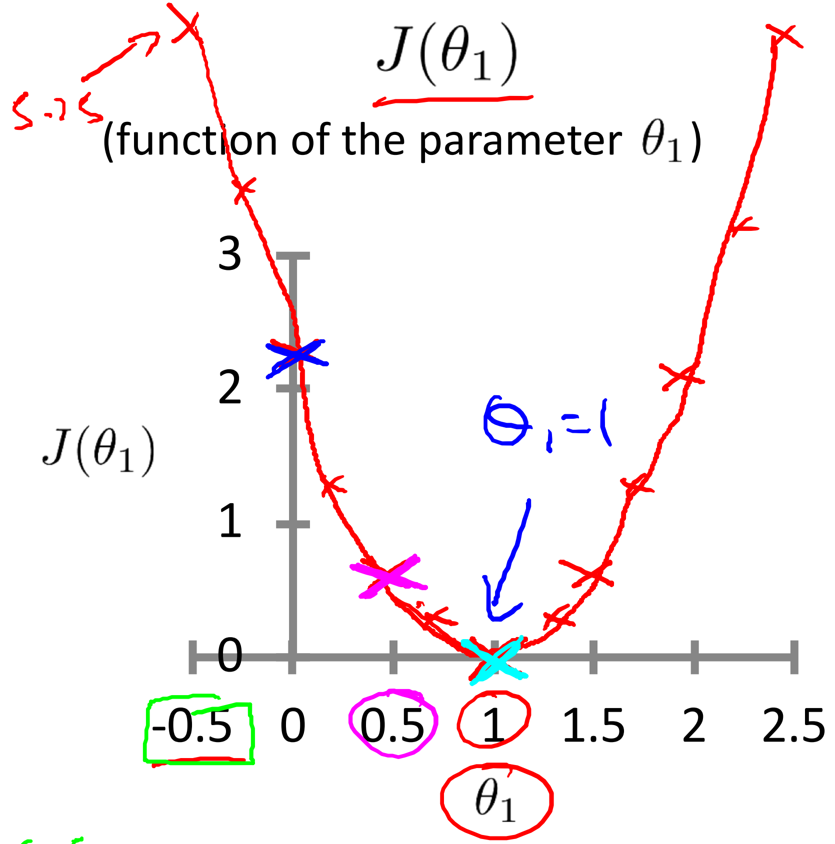


$J(0) = \frac{1}{2m}(1^2 + 2^2 + 3^2)$

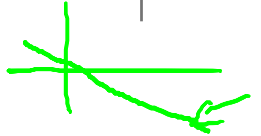
$= \frac{1}{6} \cdot 14 \approx 2.3$

$J(\theta_1)$

(function of the parameter θ_1)

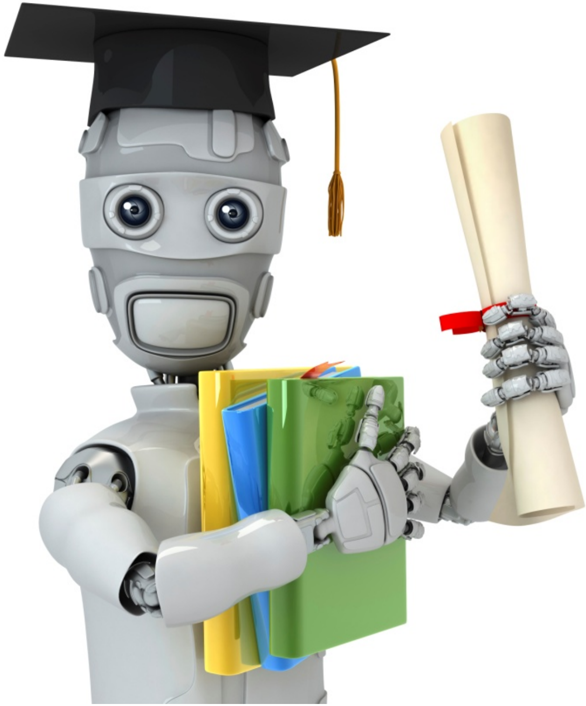


$h(x) = -0.5x$



$\boxed{\text{minimize}_{\theta_1} \; J(\theta_1)}$

Andrew Ng



Machine Learning

Linear regression with one variable

Cost function intuition II

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

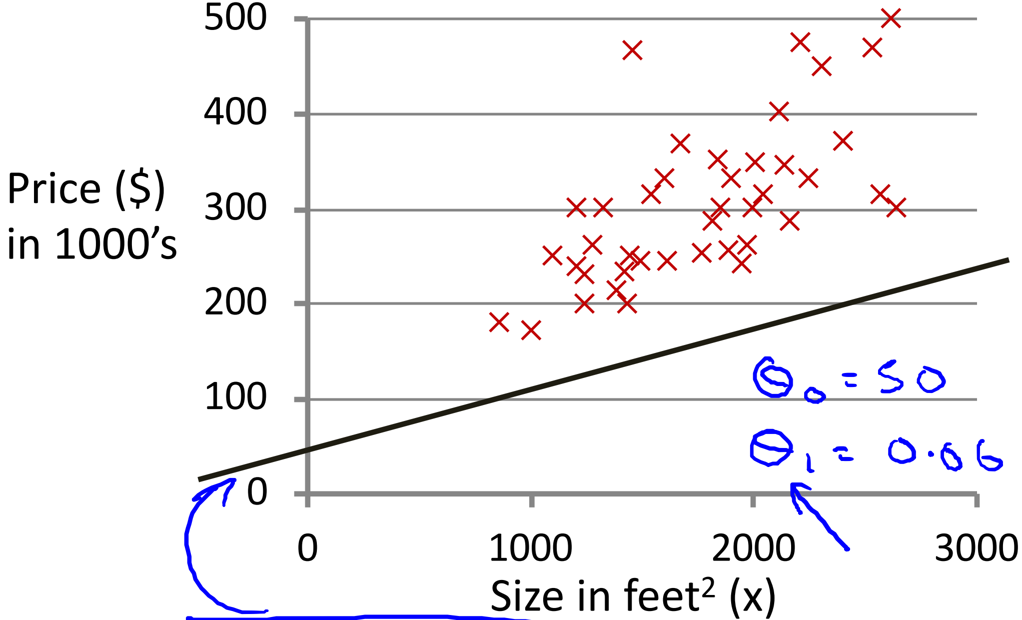
Parameters: θ_0, θ_1

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

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

$$h_\theta(x)$$

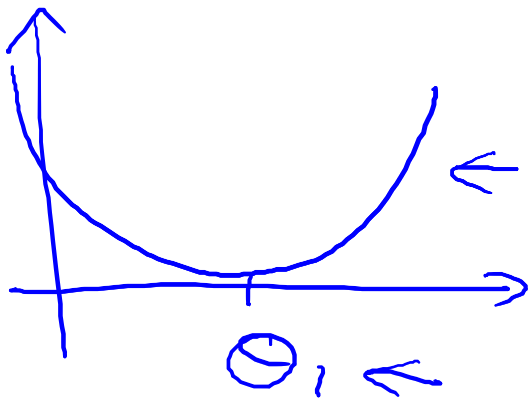
(for fixed θ_0, θ_1, this is a function of x)



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

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

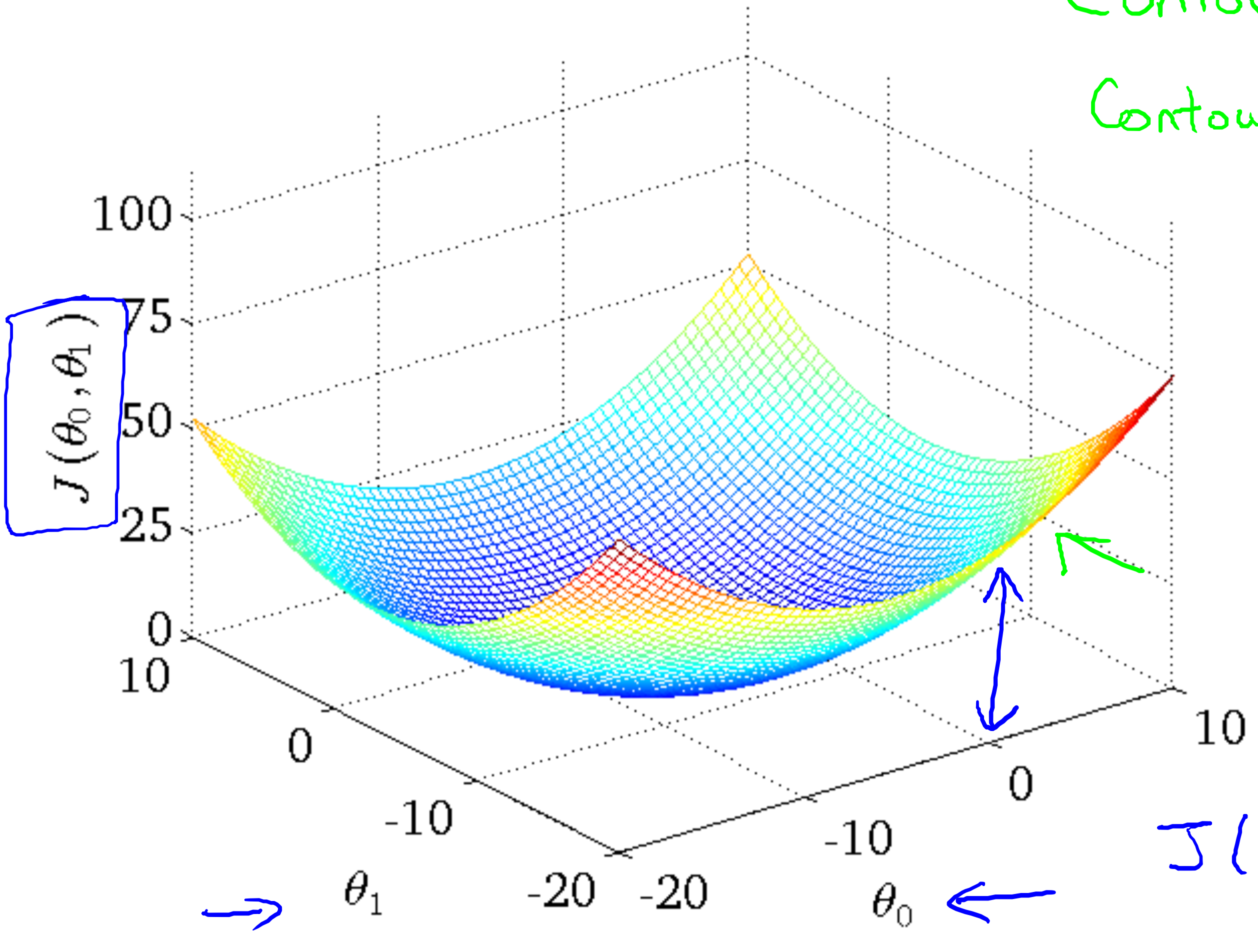
(function of the parameters θ_0, θ_1)



$$\boxed{\theta_0, \theta_1}$$

Contour plots

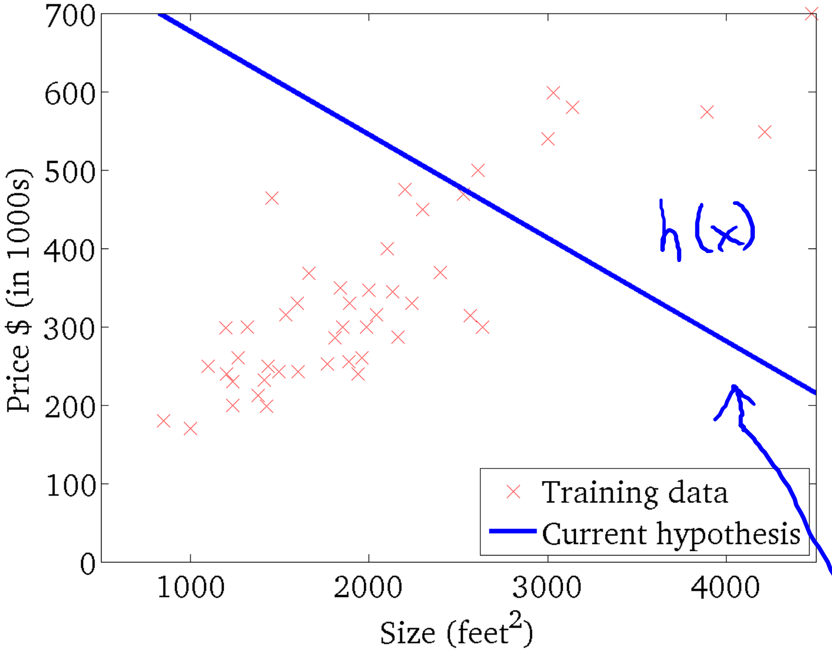
Contour figures -



$J(\theta_0, \theta_1)$

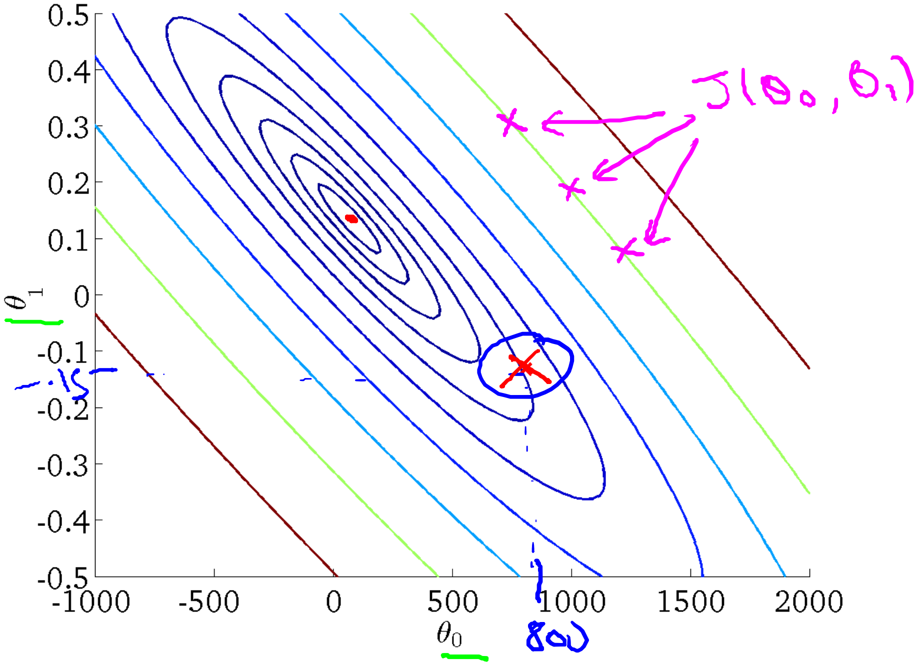
$h_\theta(x)$

(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)

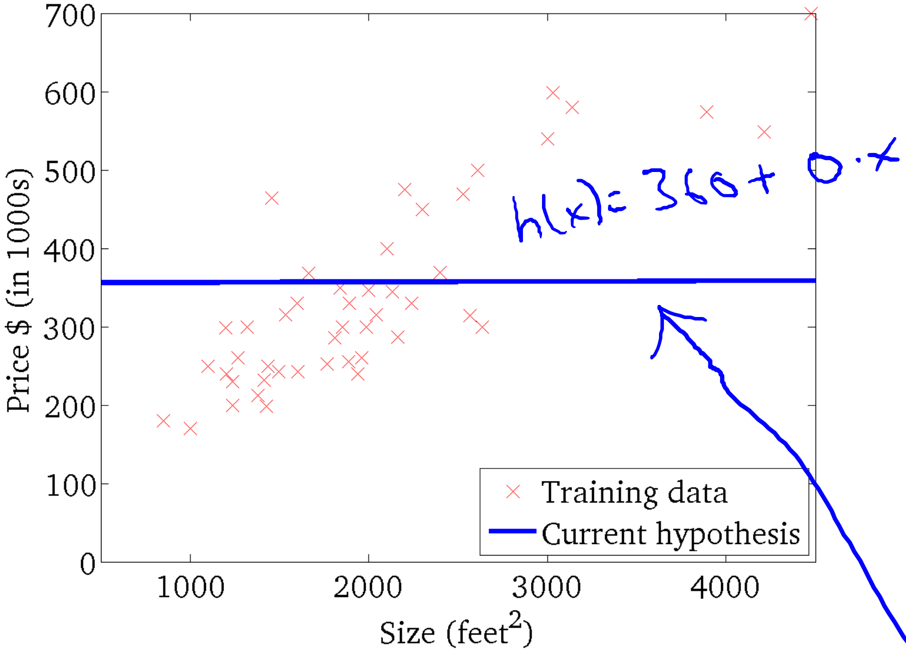


θ_0, θ_1

Andrew Ng

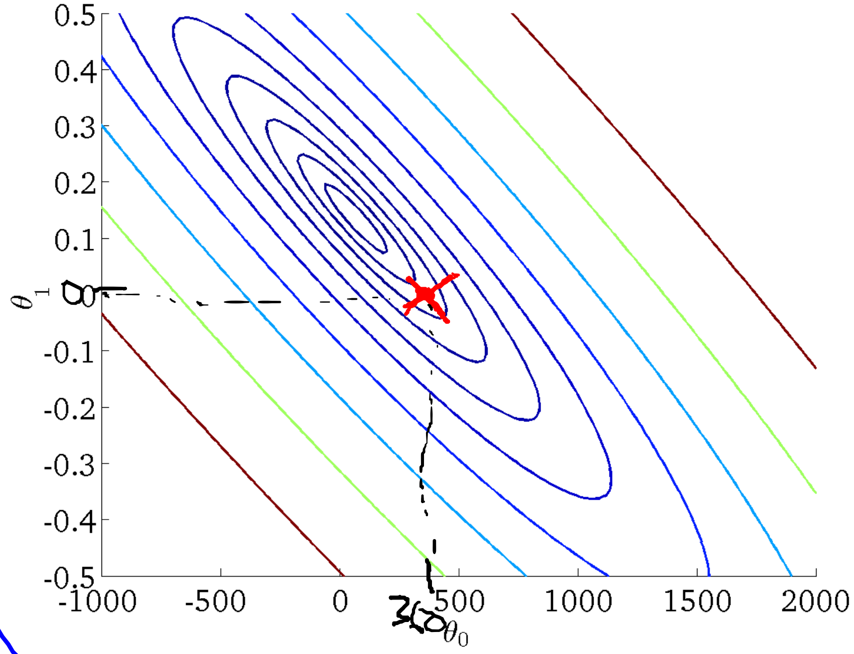
$h_\theta(x)$

(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)



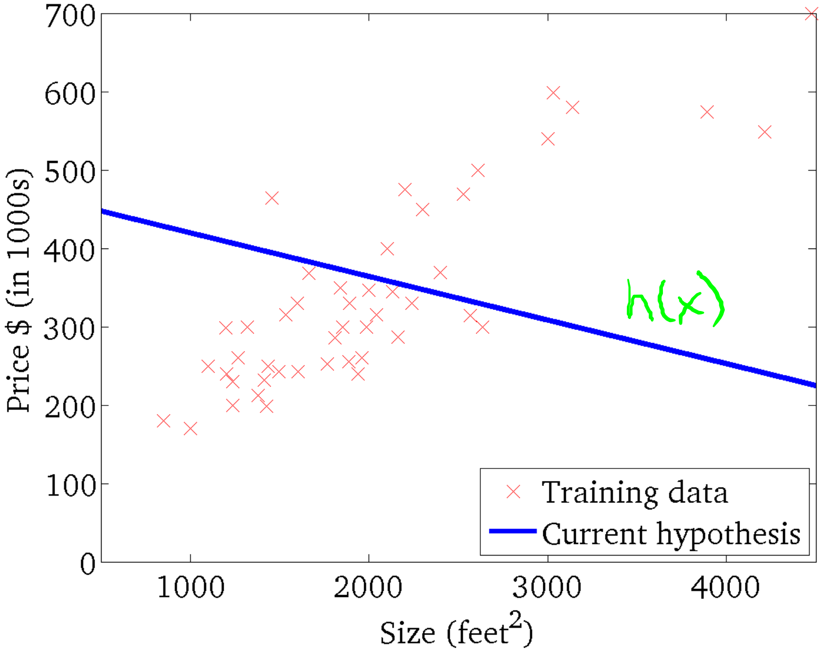
$\theta_0 = 360$

$\theta_1 = 0$

Andrew Ng

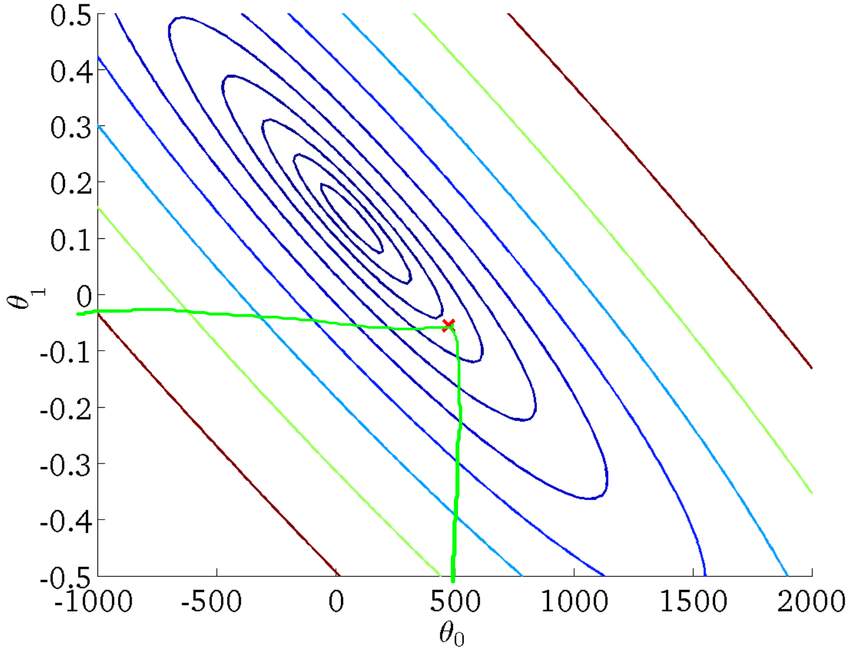
$$h_\theta(x)$$

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



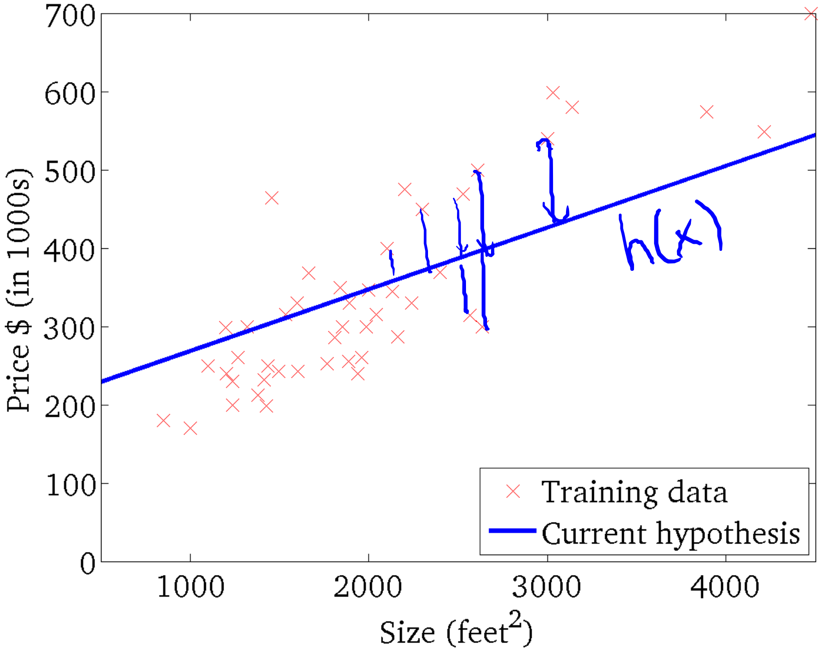
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



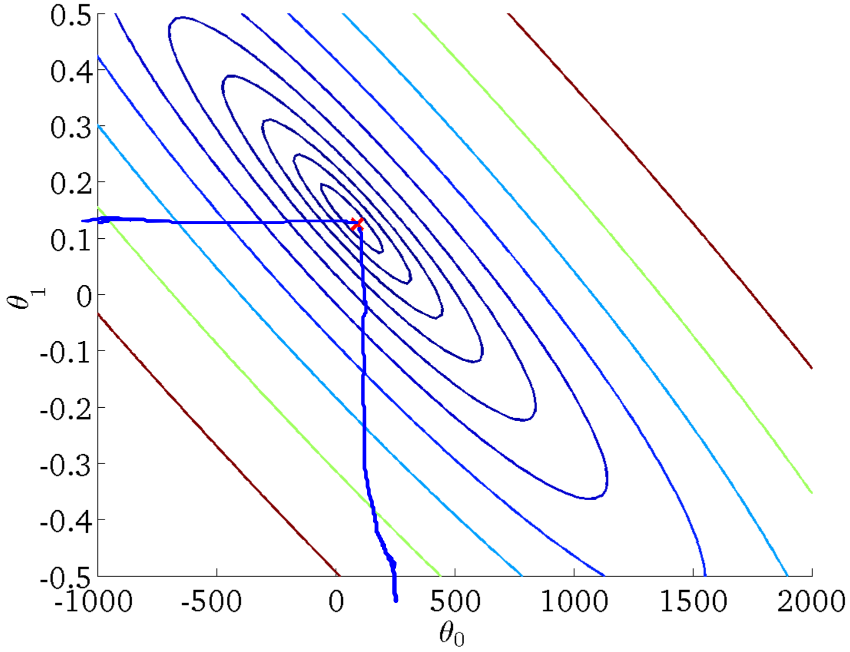
$h_\theta(x)$

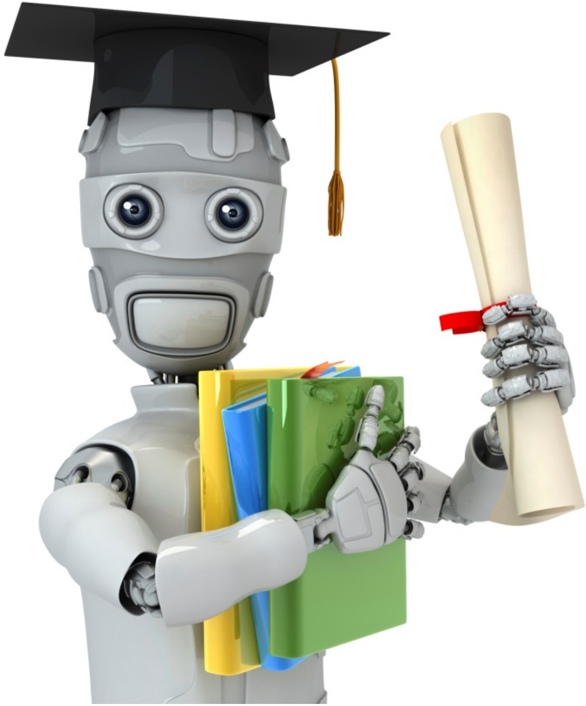
(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)





Machine Learning

Linear regression with one variable

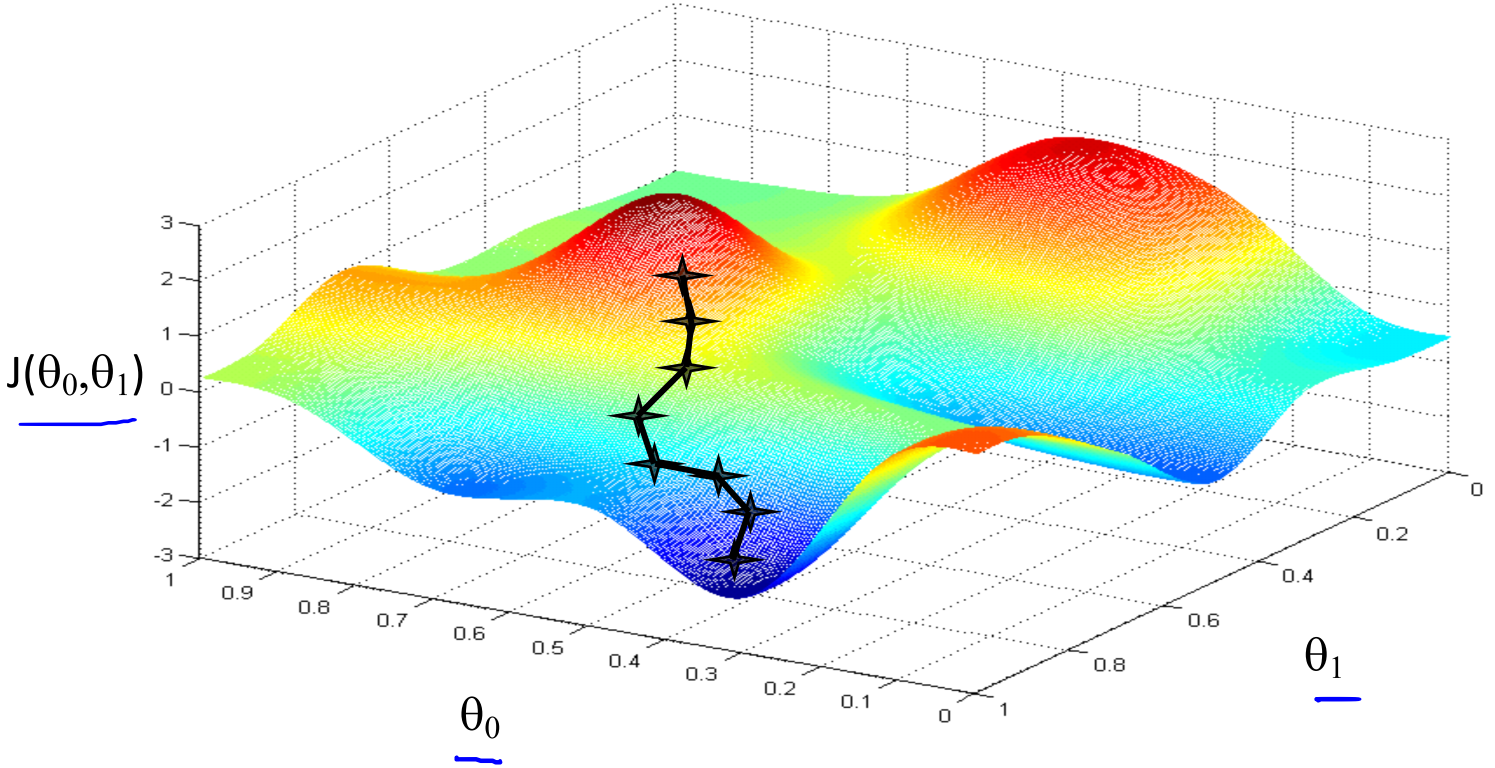
Gradient descent

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

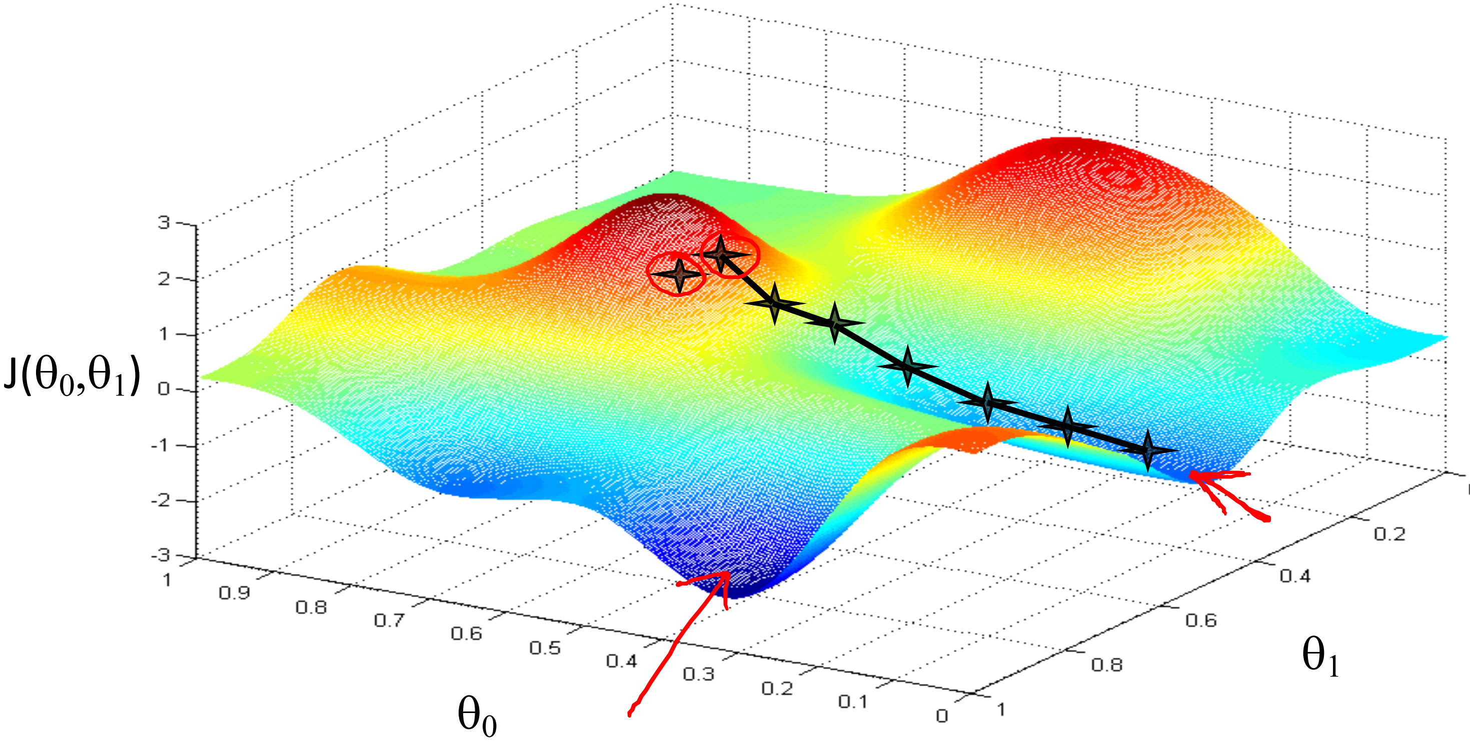
Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0 \dots \theta_n} J(\theta_0, \dots, \theta_n)$

Outline:

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



$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1



J(θ0,θ1)
3
2
1
0
-1
-2
-3
θ0
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ1
0
0.2
0.4
0.6
0.8
1

Gradient descent algorithm

repeat until convergence {

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

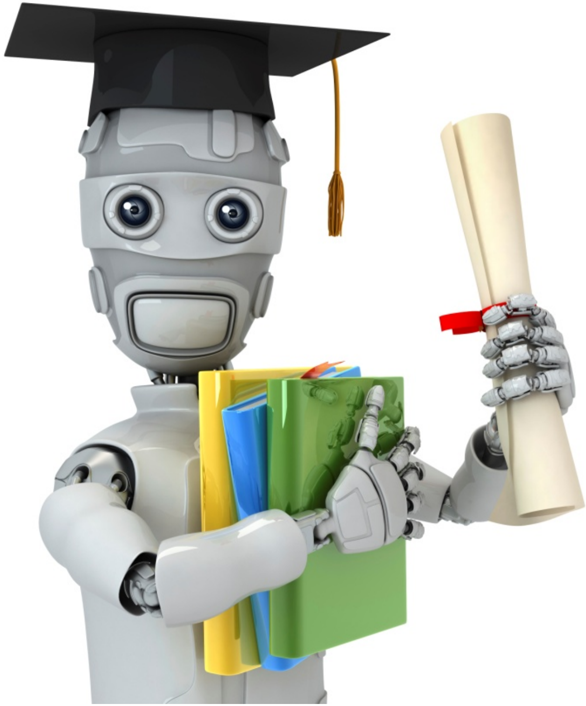
}

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 {

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

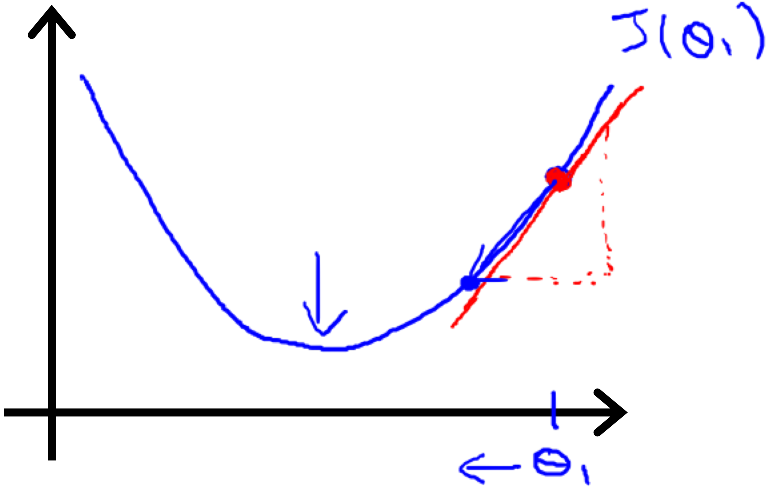
}

learning rate

derivative

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

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

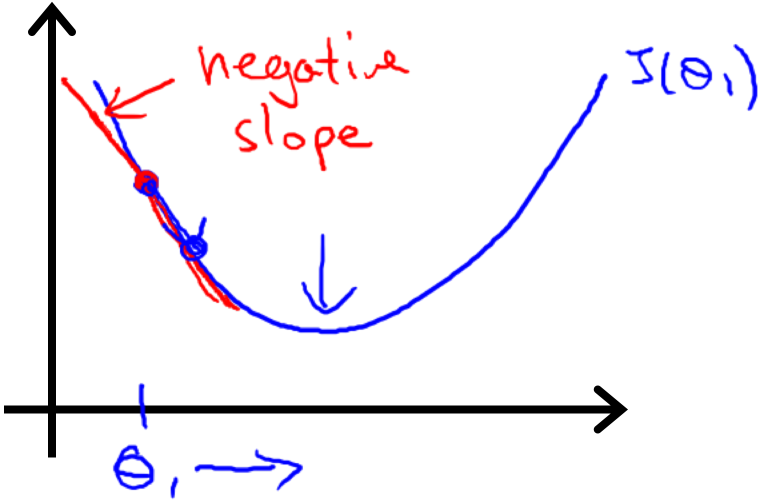


$(\theta_1 \in \mathbb{R})$

$\frac{\partial}{\partial \theta_1}$

$$\theta_1 := \theta_1 - \alpha \underbrace{\boxed{\frac{d}{d\theta_1} J(\theta_1)}}_{\geq 0}$$

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



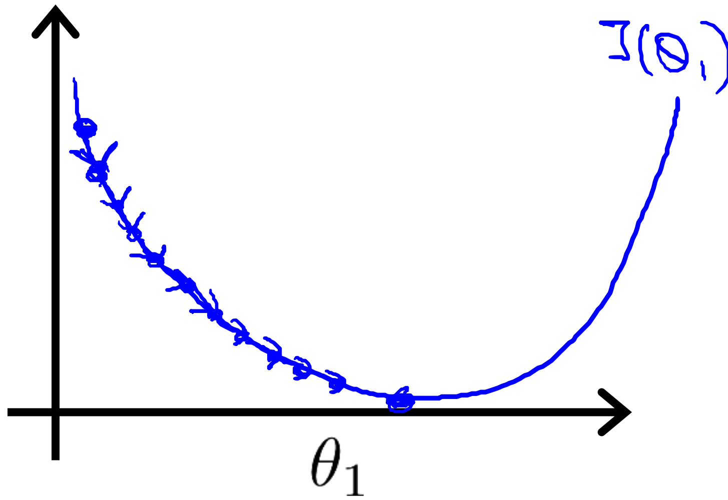
$$\underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\leq 0}$$

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

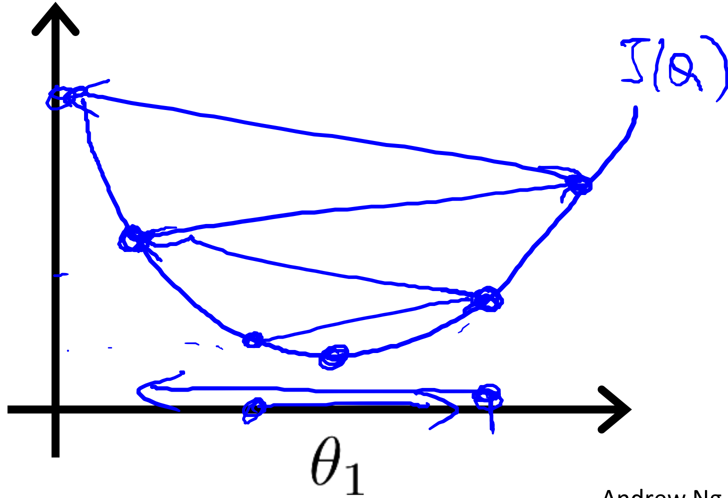
Andrew Ng

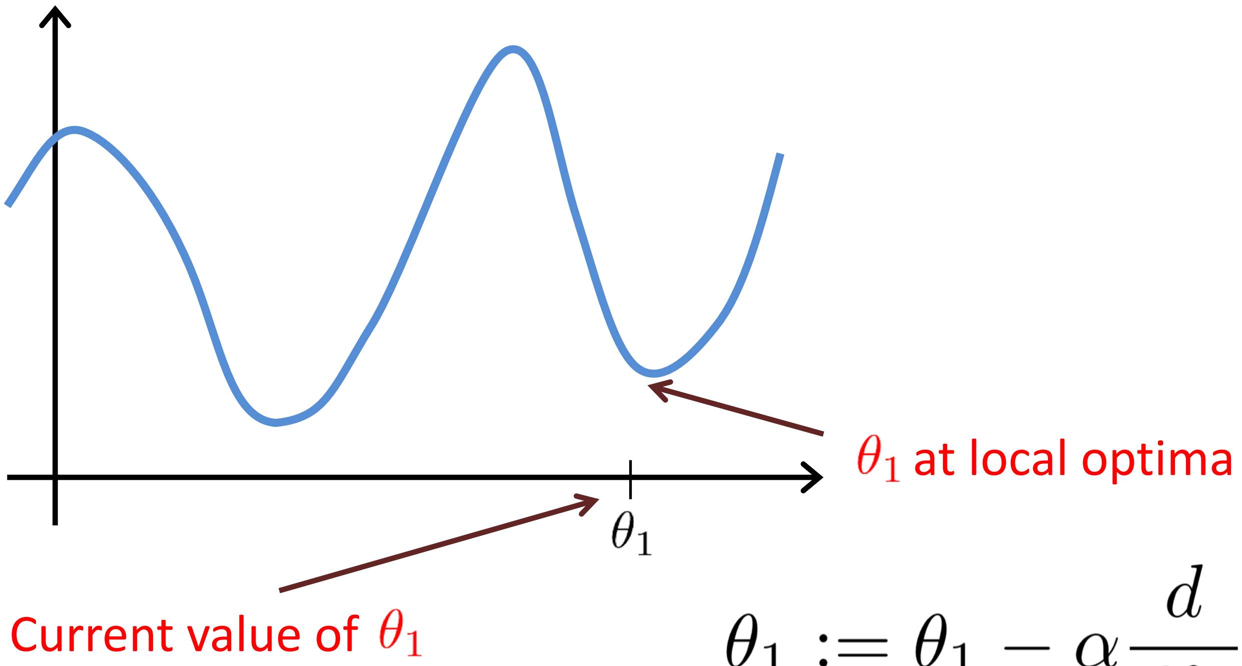
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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.



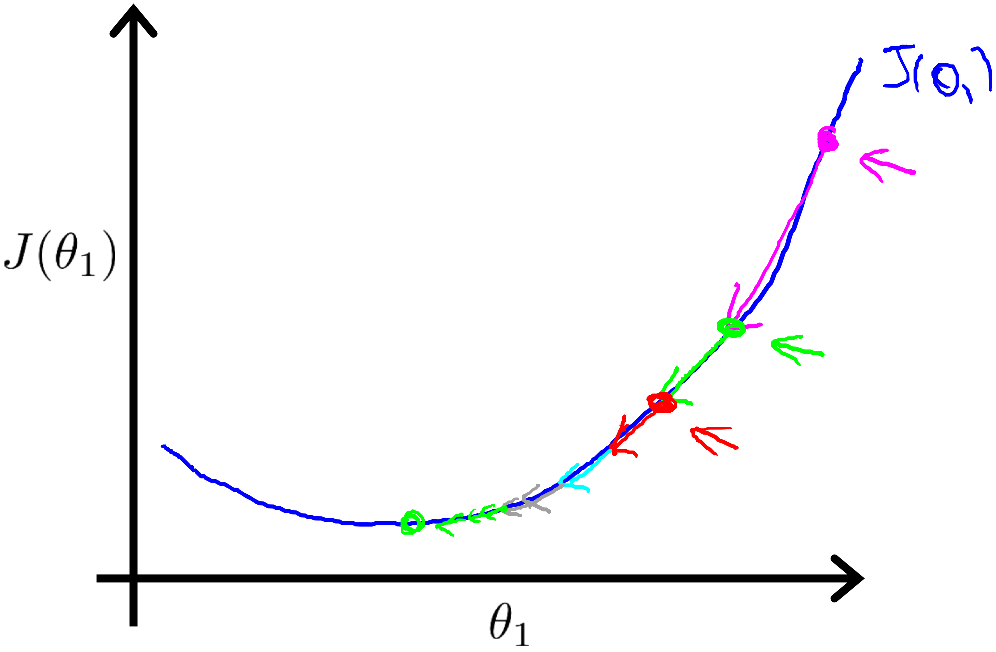


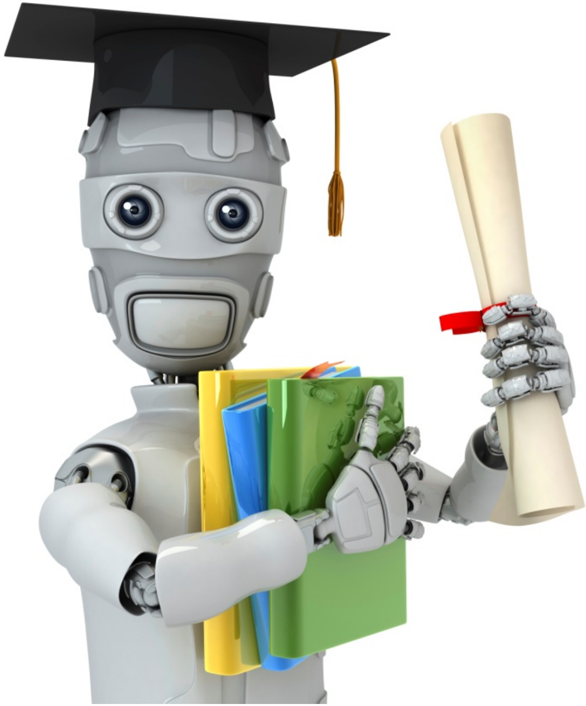
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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

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

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

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

Linear Regression Model

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

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

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

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

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

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

Gradient descent algorithm

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

repeat until convergence {

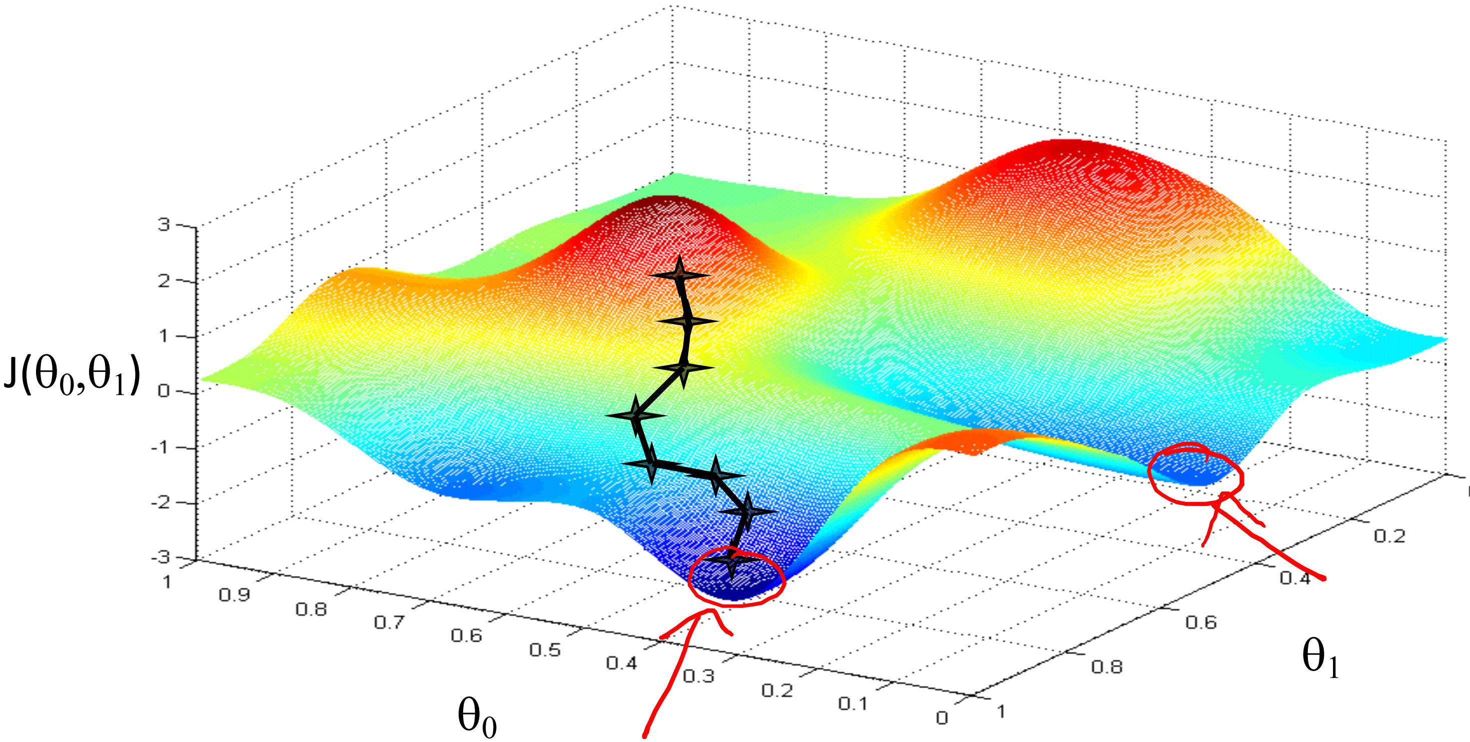
$$\theta_0 := \theta_0 - \alpha \boxed{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)}$$

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

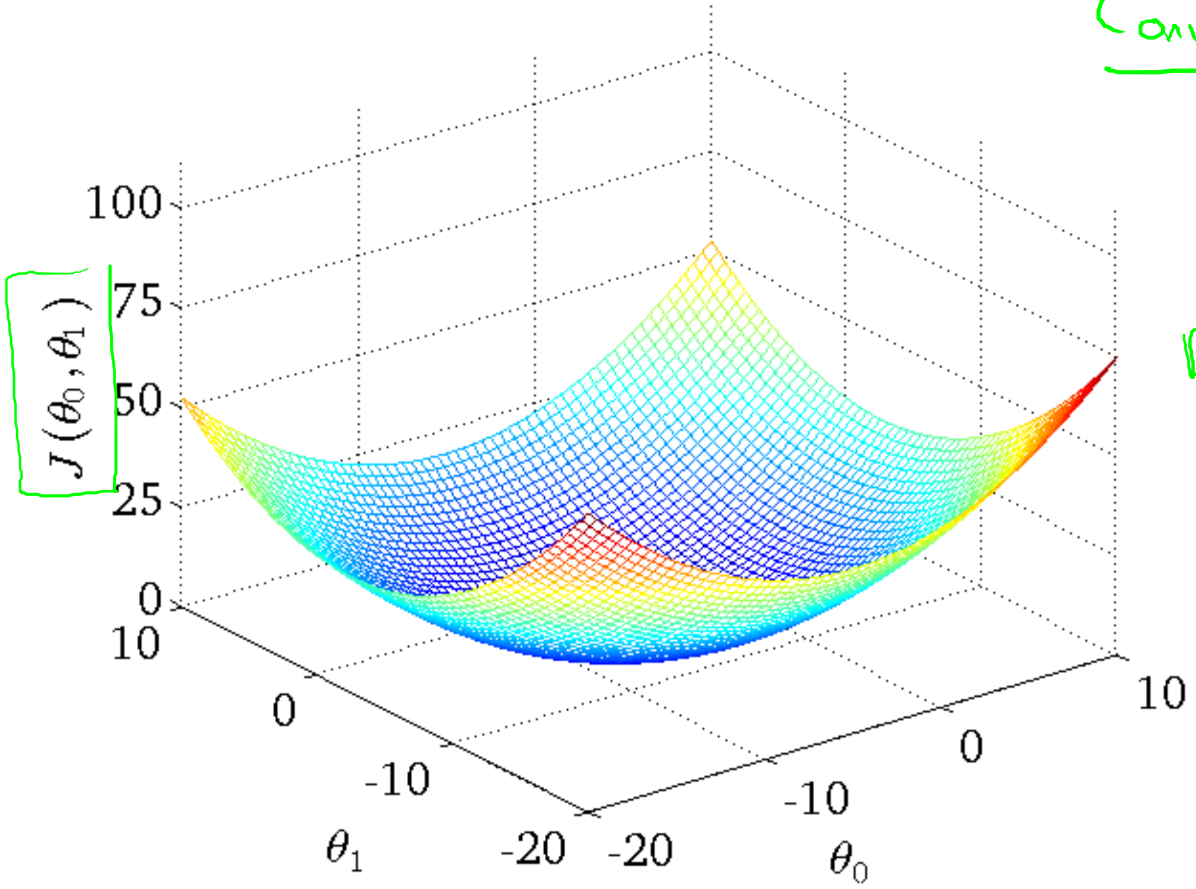
}

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

update θ_0 and θ_1 simultaneously



$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

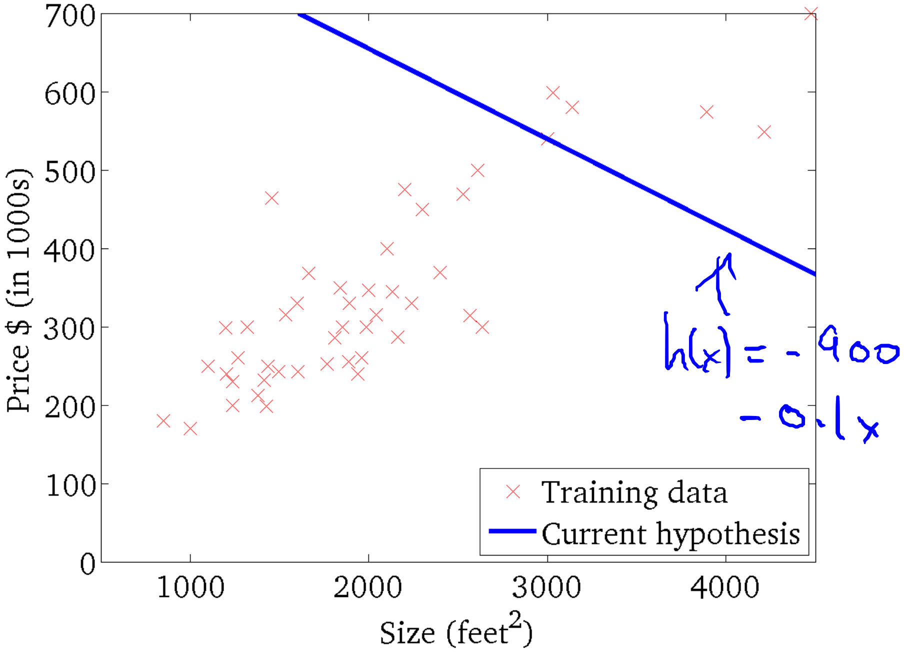


"Convex function"

Bowl-shaped

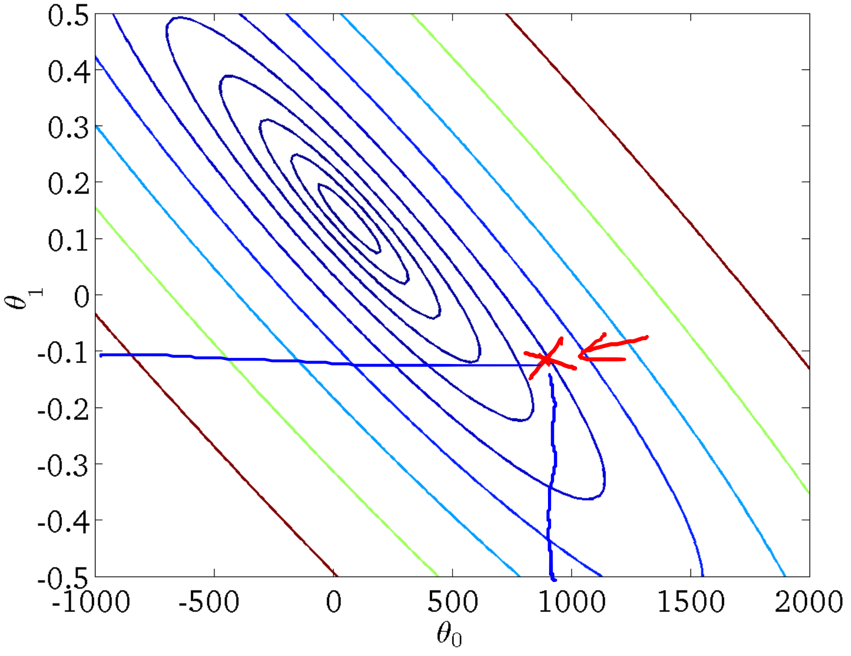
$h_\theta(x)$

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



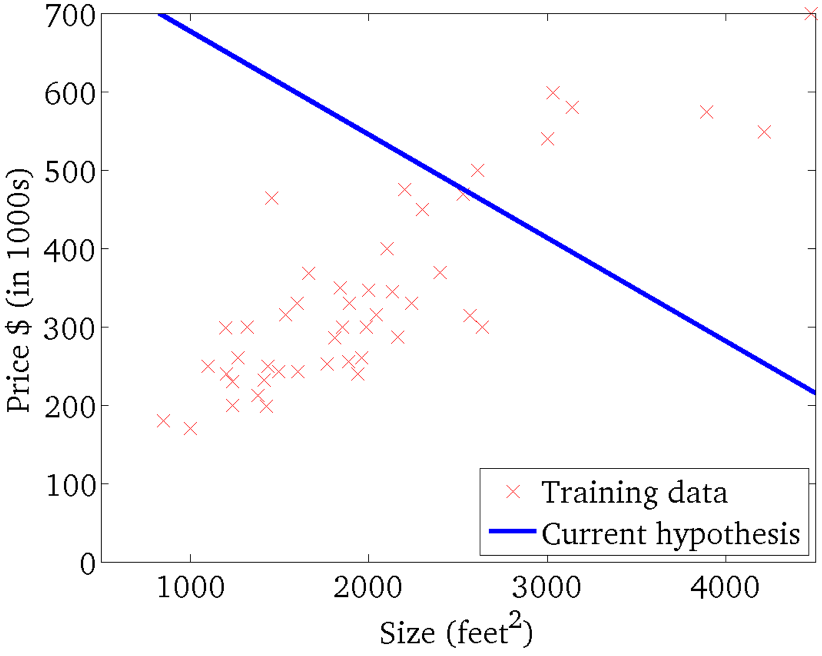
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



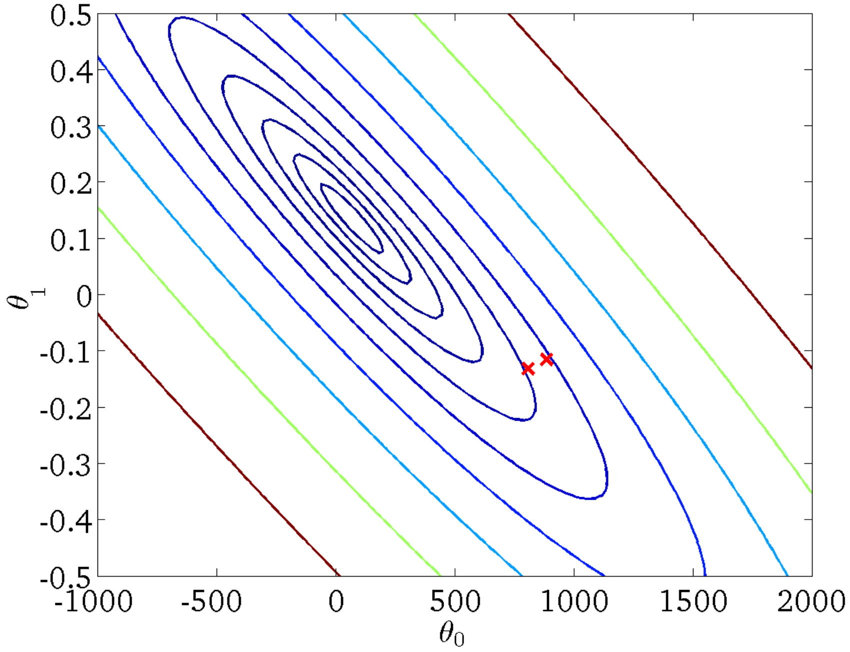
$$h_\theta(x)$$

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



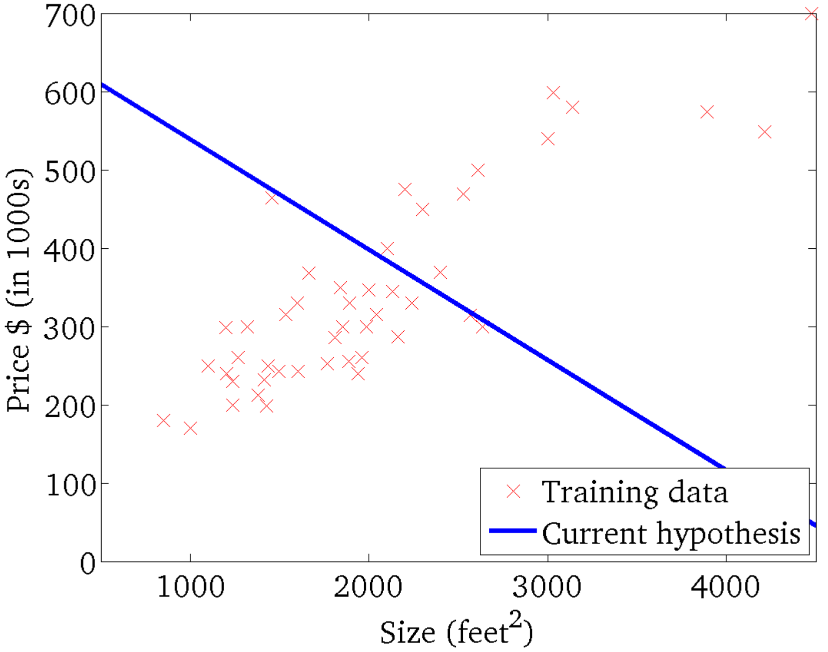
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



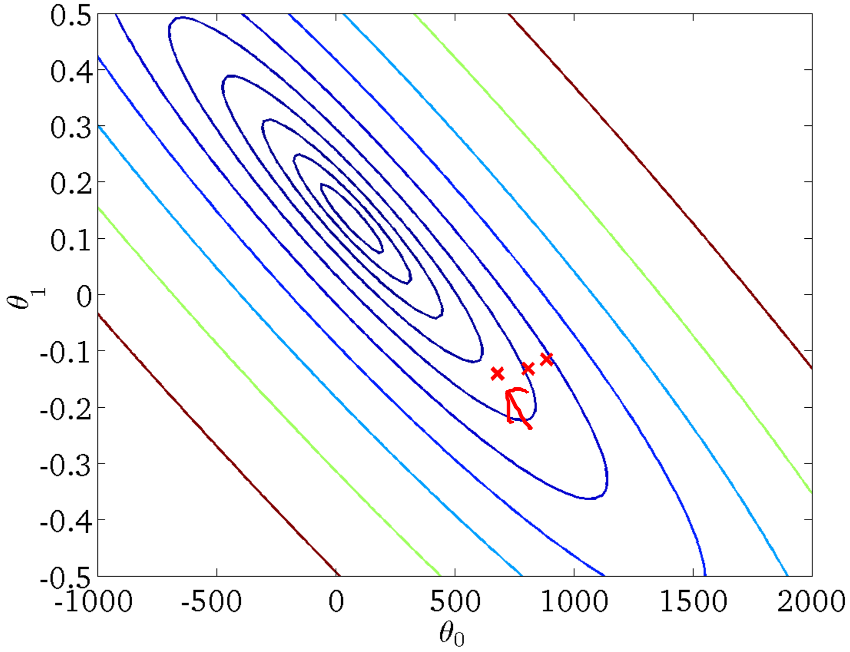
$h_\theta(x)$

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



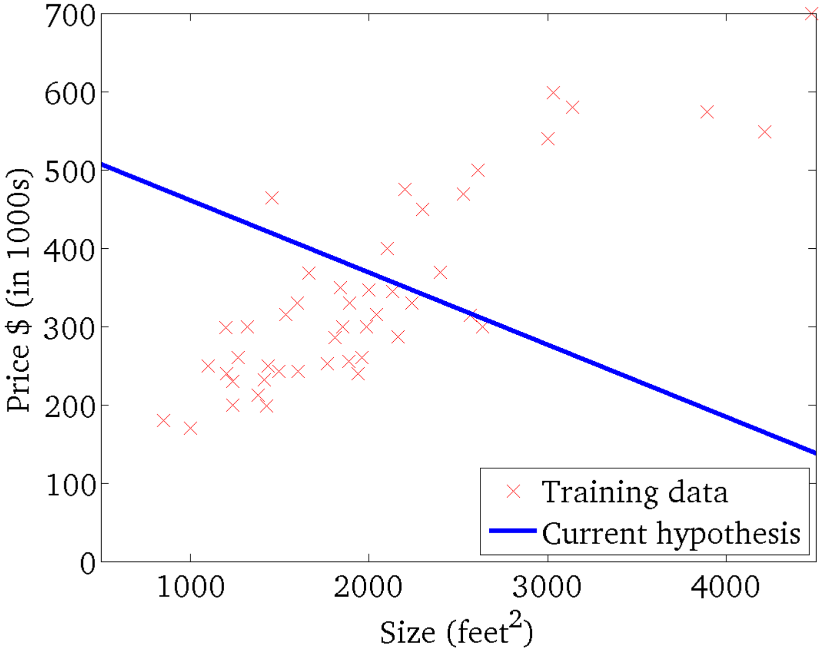
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



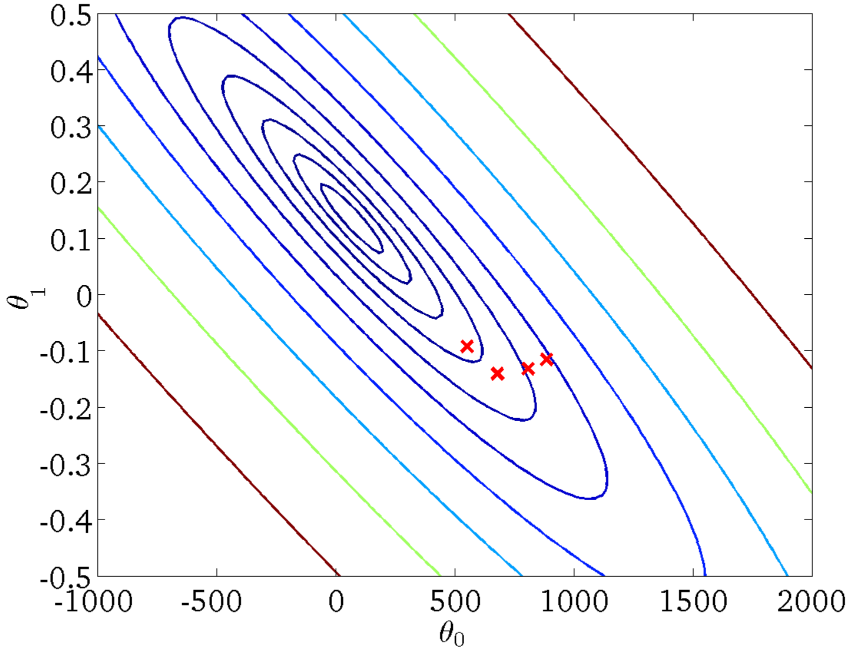
$$h_\theta(x)$$

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



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

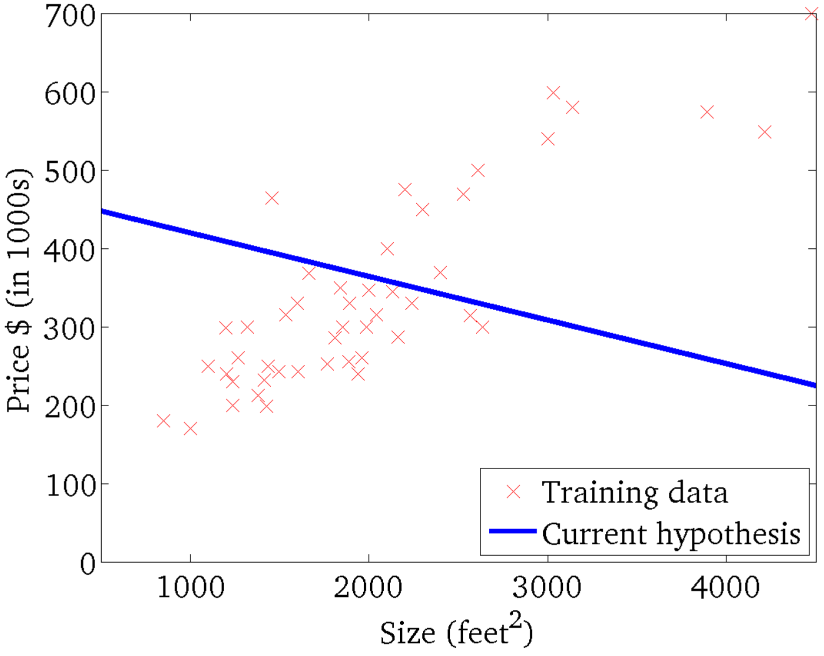
(function of the parameters θ_0, θ_1)



Andrew Ng

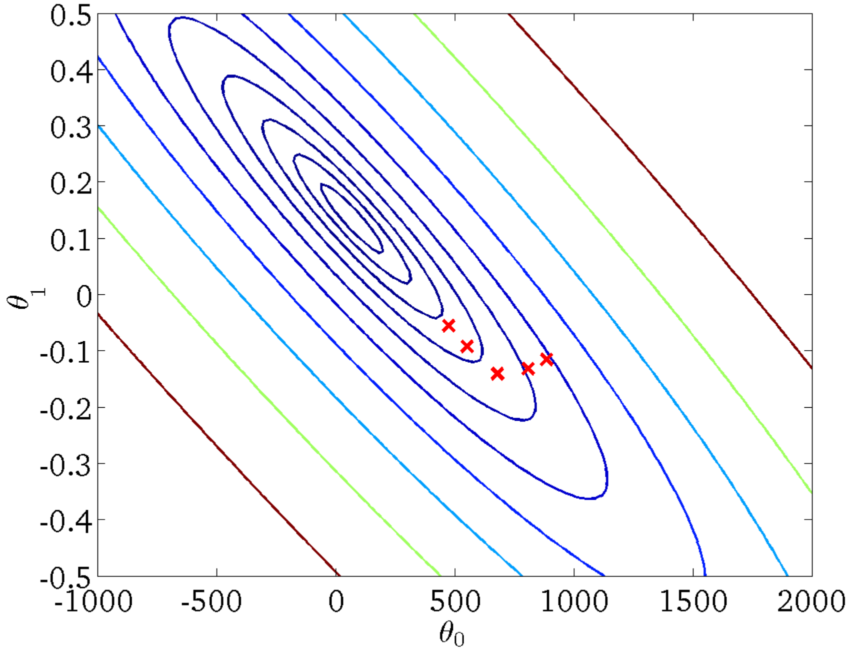
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

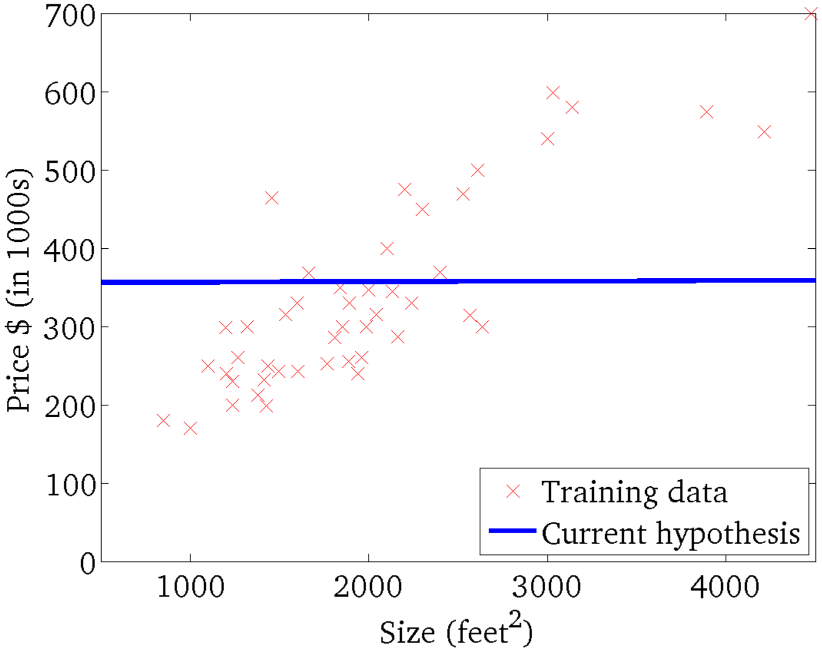
(function of the parameters θ_0, θ_1)



Andrew Ng

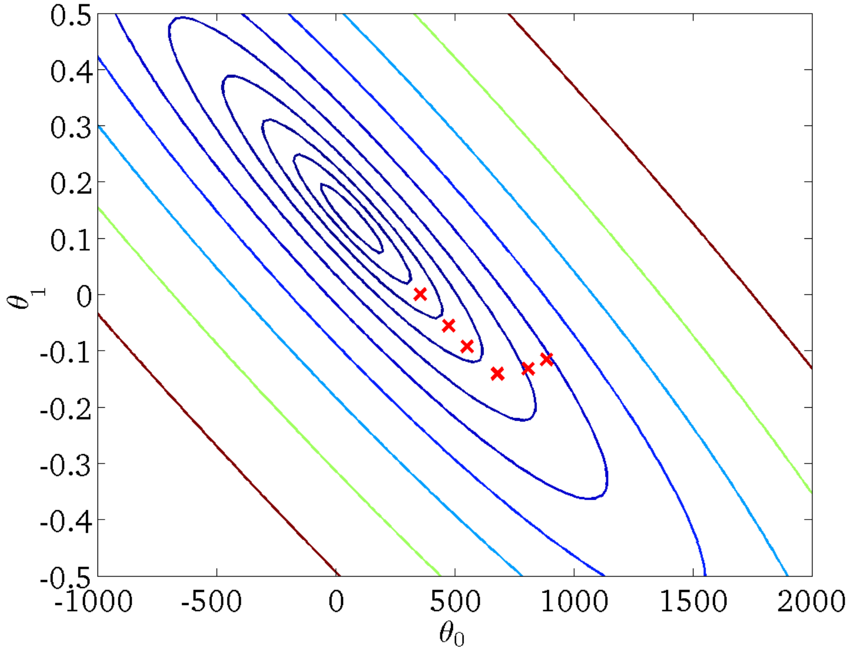
$$h_\theta(x)$$

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



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

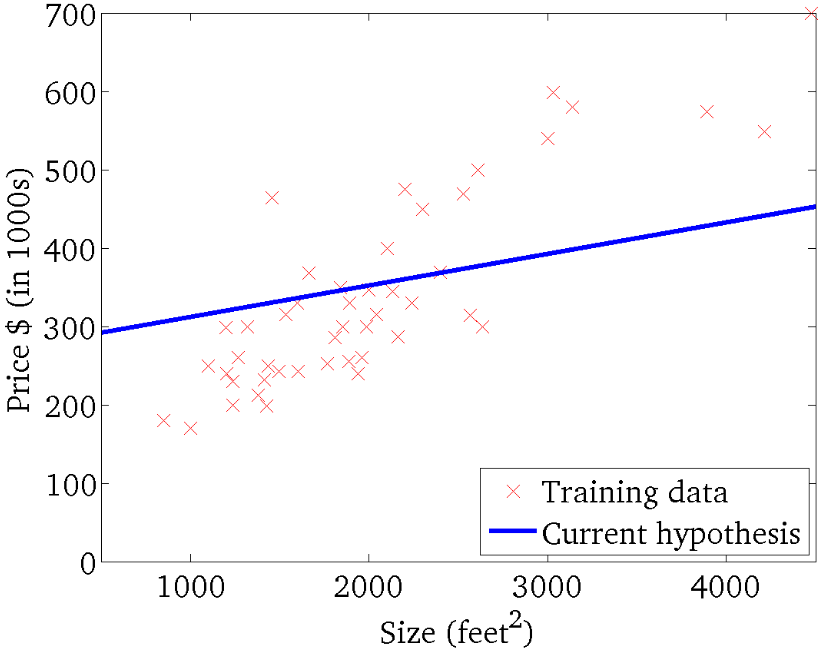
(function of the parameters θ_0, θ_1)



Andrew Ng

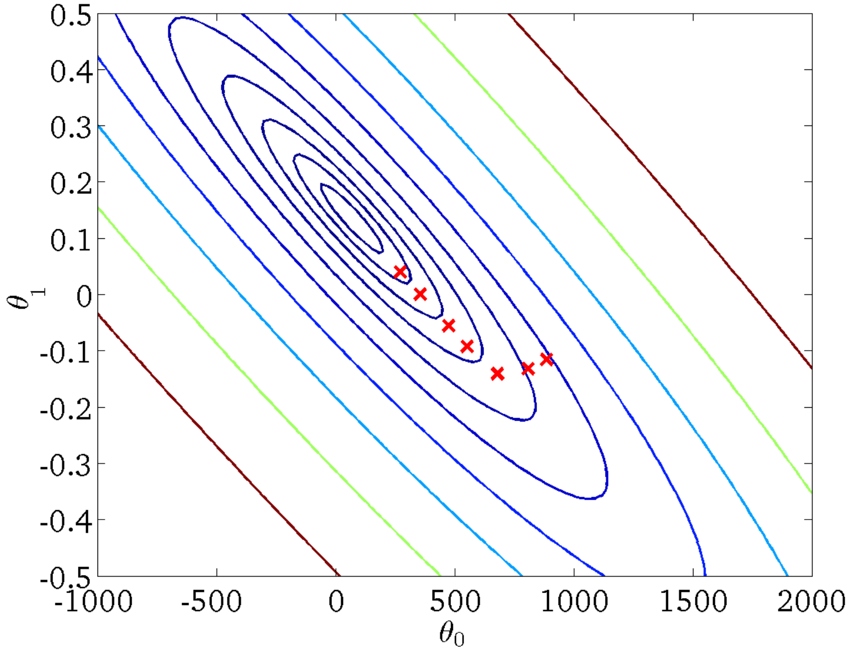
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

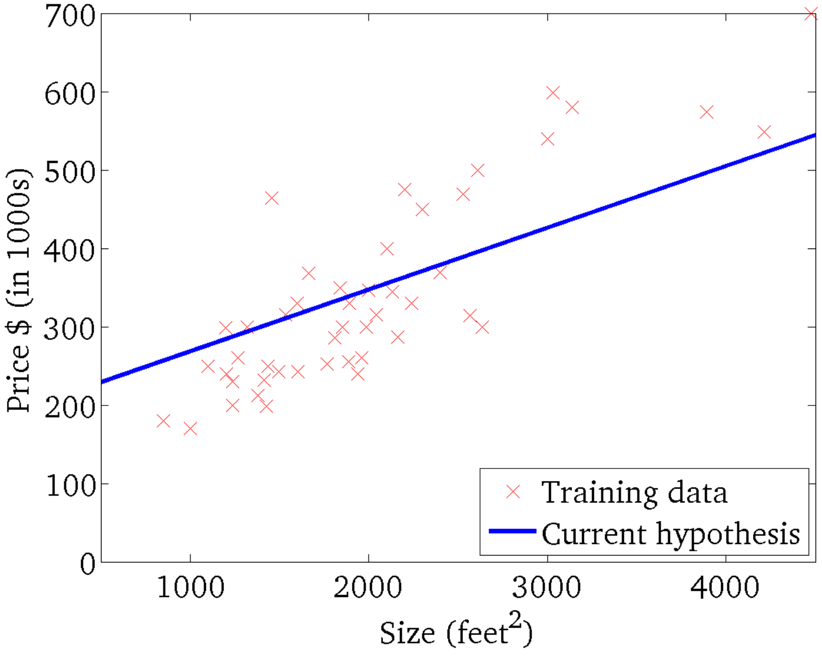
(function of the parameters θ_0, θ_1)



Andrew Ng

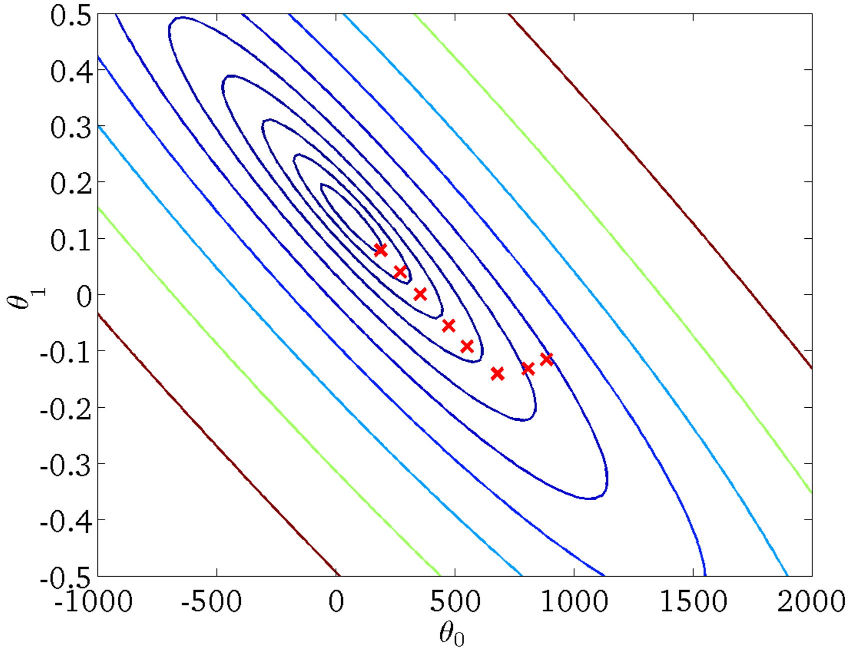
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

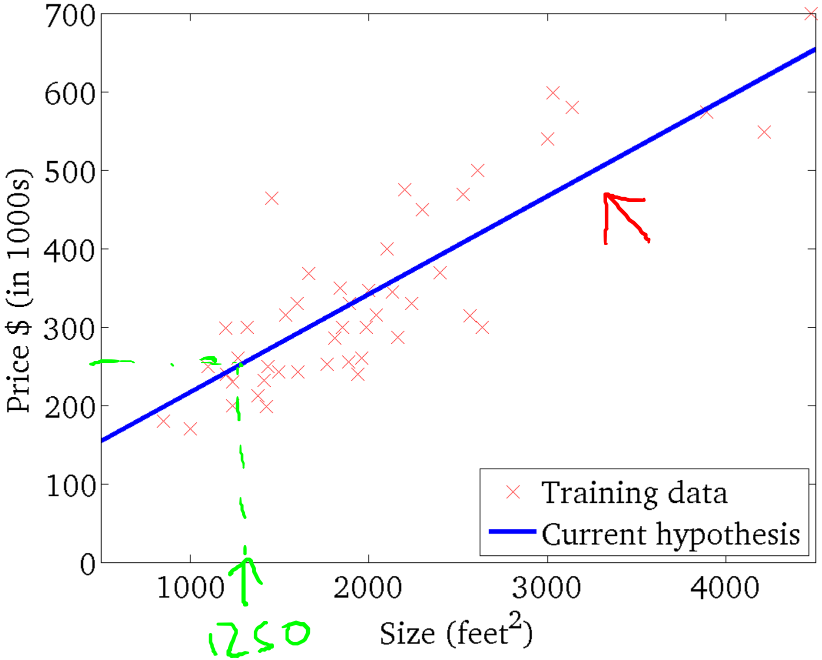
(function of the parameters θ_0, θ_1)



Andrew Ng

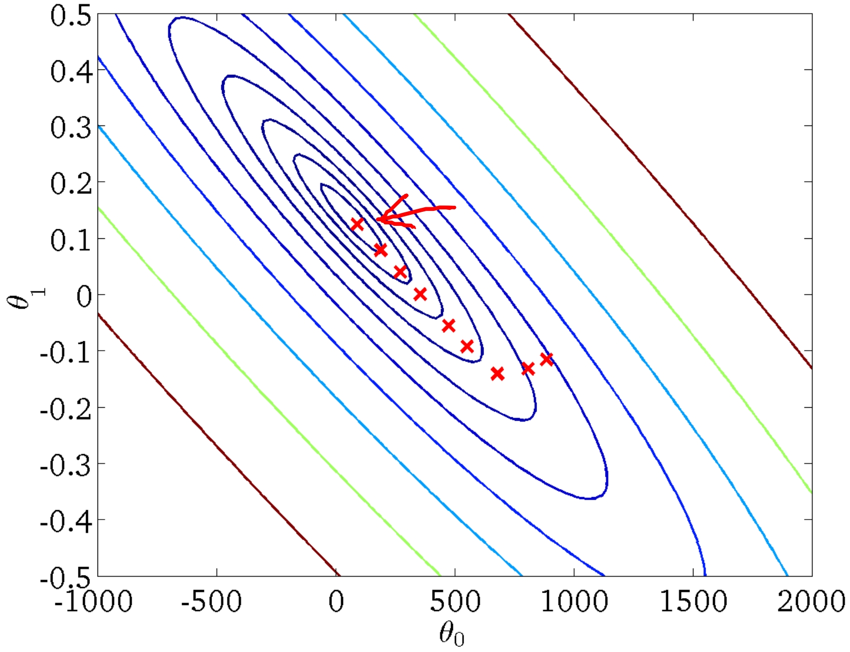
$$h_\theta(x)$$

(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.

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