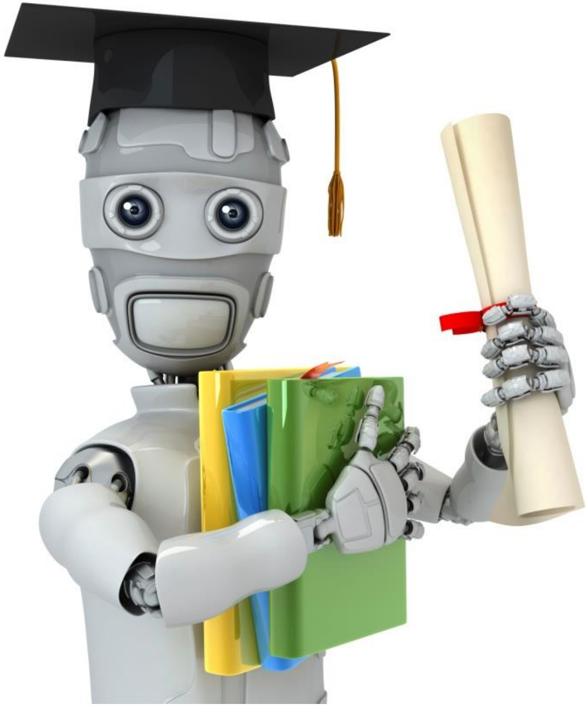
CSE 4621
Machine Learning

Lecture 2

Md. Hasanul Kabir, PhD.

Professor, CSE Department

Islamic University of Technology (IUT)



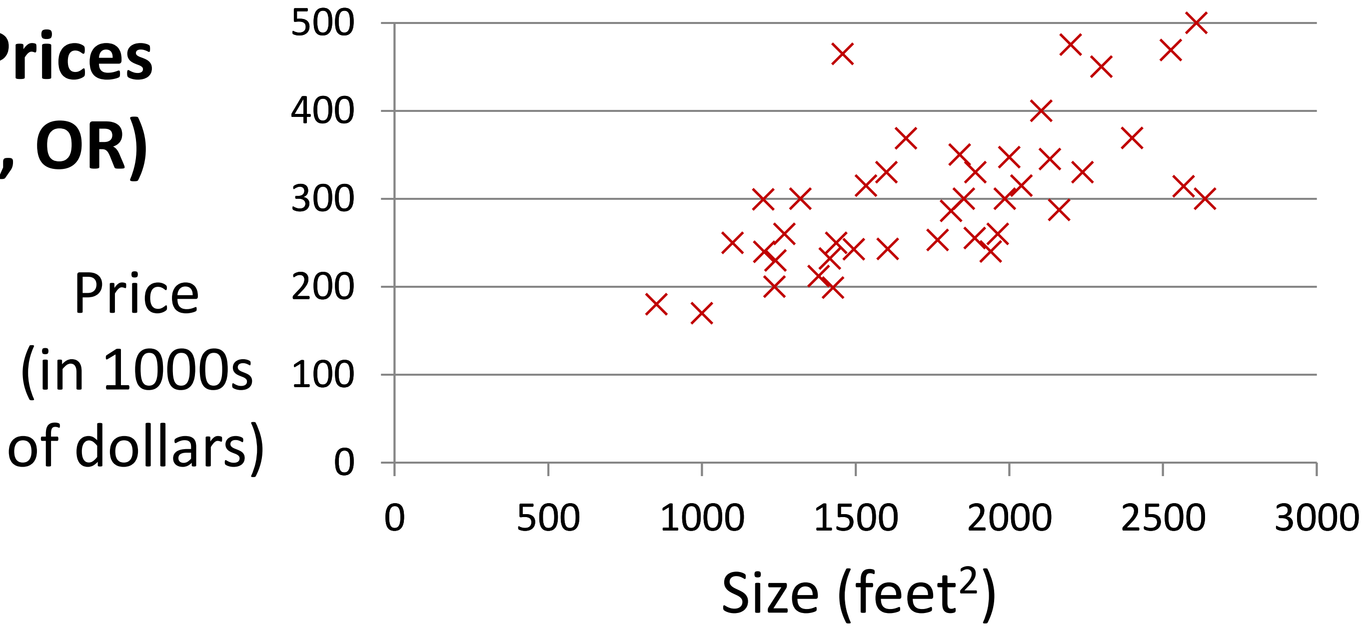
Linear regression with one variable

Model representation

Machine Learning

Source & Special Thanks to Andrew Ng (Coursera) Machine Learning Course

Housing Prices (Portland, OR)



<u>Supervised Learning</u>

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

<u>Regression Problem</u>

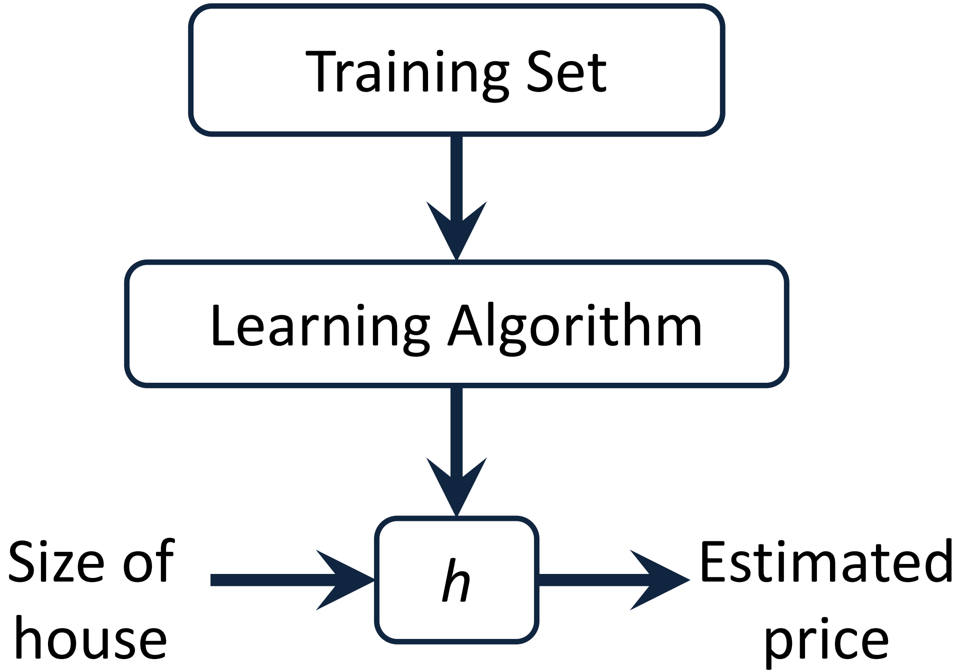
Predict real-valued output

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
…	…

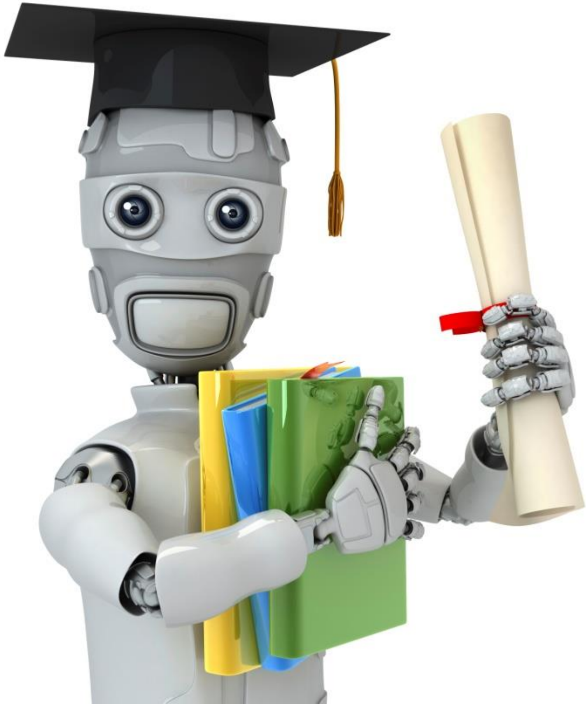
Notation:

m = Number of training examples
x’s = “input” variable / features
y’s = “output” variable / “target” variable



How do we represent *h* ?

Linear regression with one variable.
Univariate linear regression.



Machine Learning

Linear regression with one variable

Cost function

Training Set

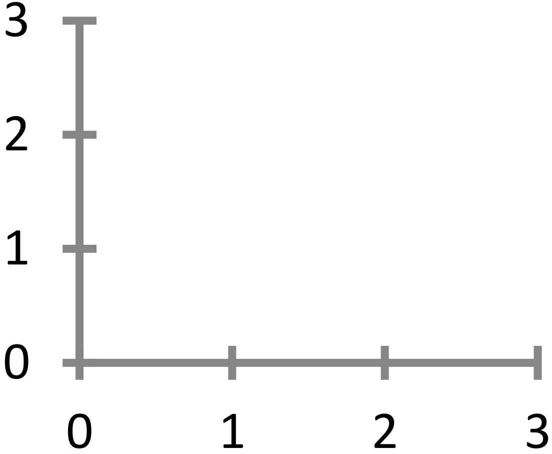
Size in feet^2 (x)	**Price ($) in 1000's (y)**
2104	460
1416	232
1534	315
852	178
…	…

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

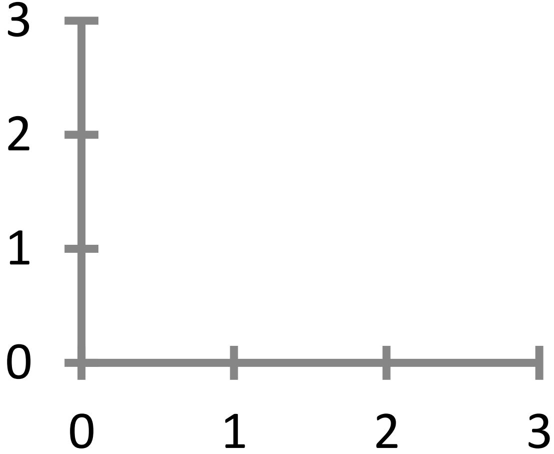
θ_i's: Parameters

How to choose θ_i's ?

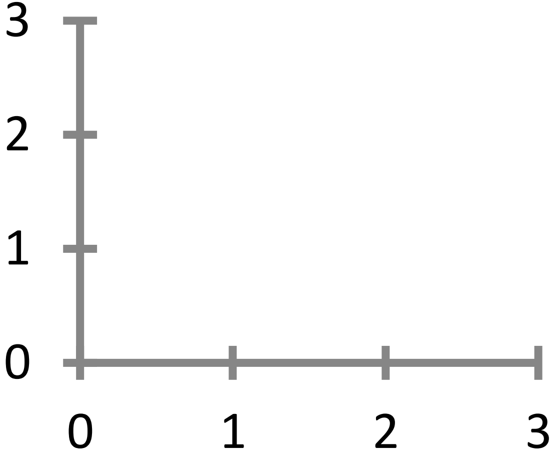
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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

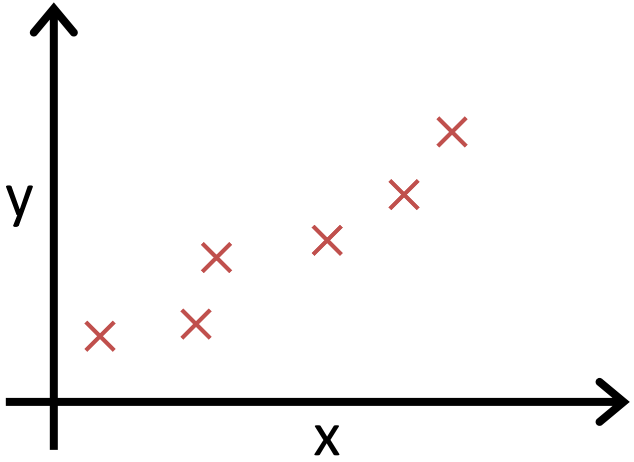


$$\theta_0 = 0$$
$$\theta_1 = 0.5$$

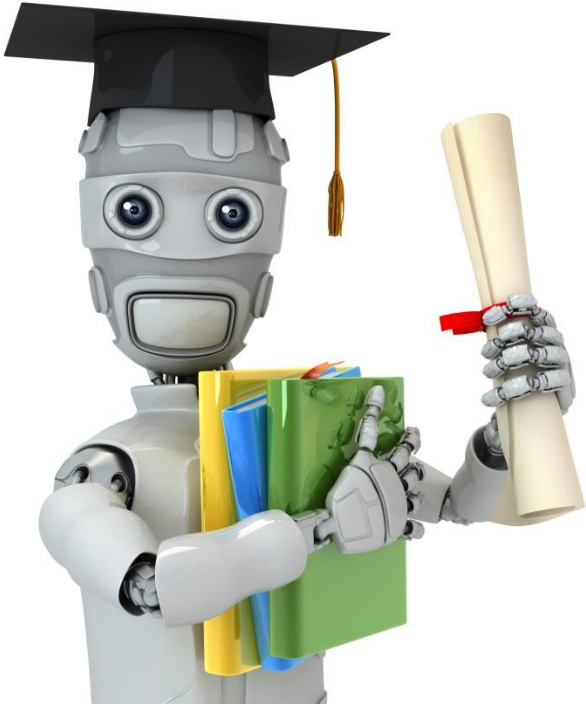


$$\theta_0 = 1$$
$$\theta_1 = 0.5$$

Andrew Ng



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



Machine Learning

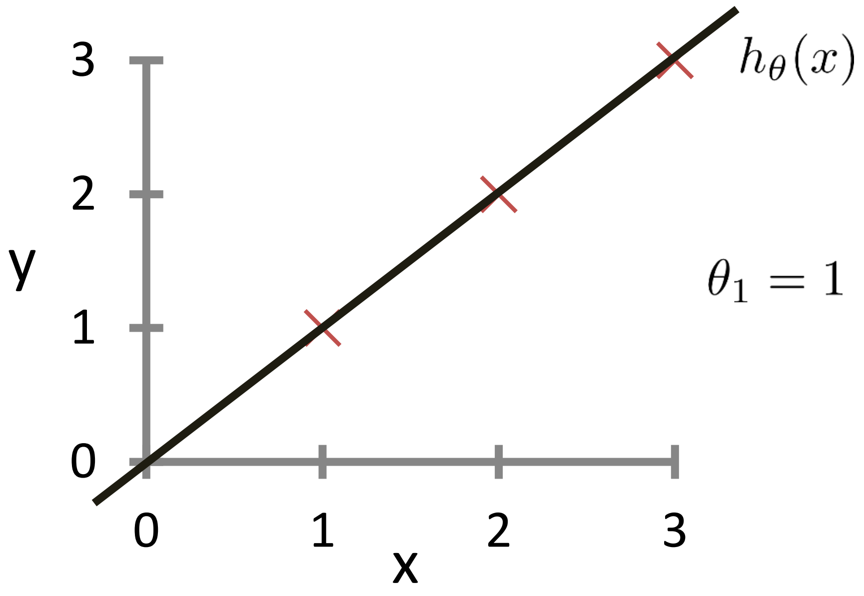
Linear regression with one variable

Cost function intuition I

	Simplified
Hypothesis:	
$h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) = \theta_1 x$
Parameters:	
θ_0, θ_1	θ_1
Cost Function:	
$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$	$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$
Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$	$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

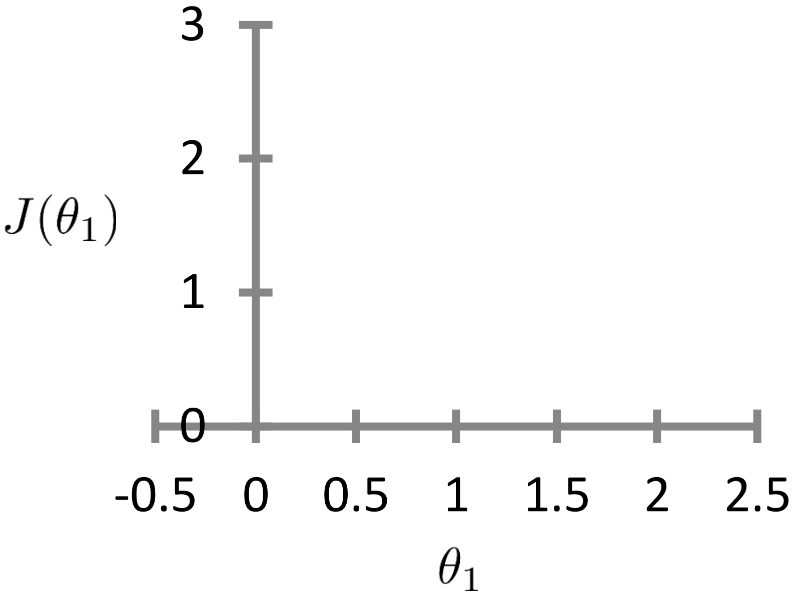
$h_\theta(x)$

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



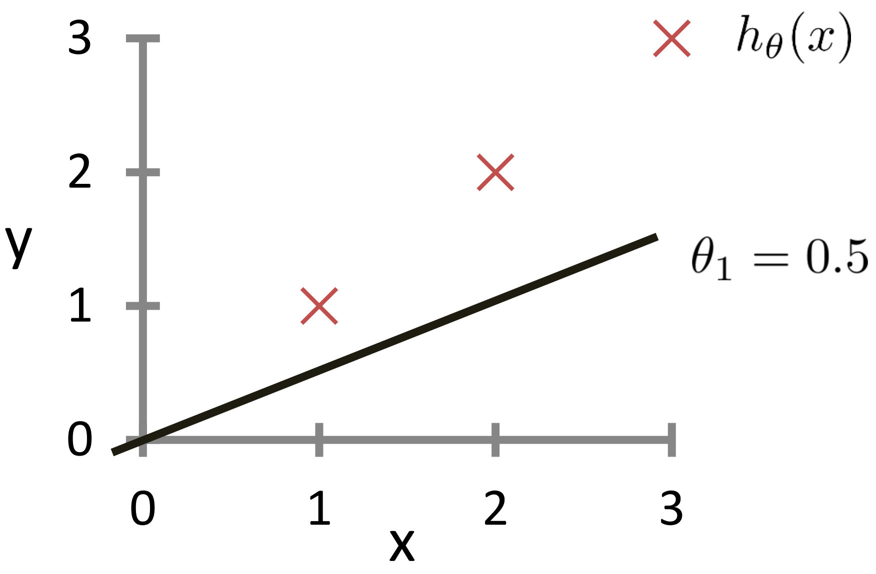
$J(\theta_1)$

(function of the parameter θ_1)



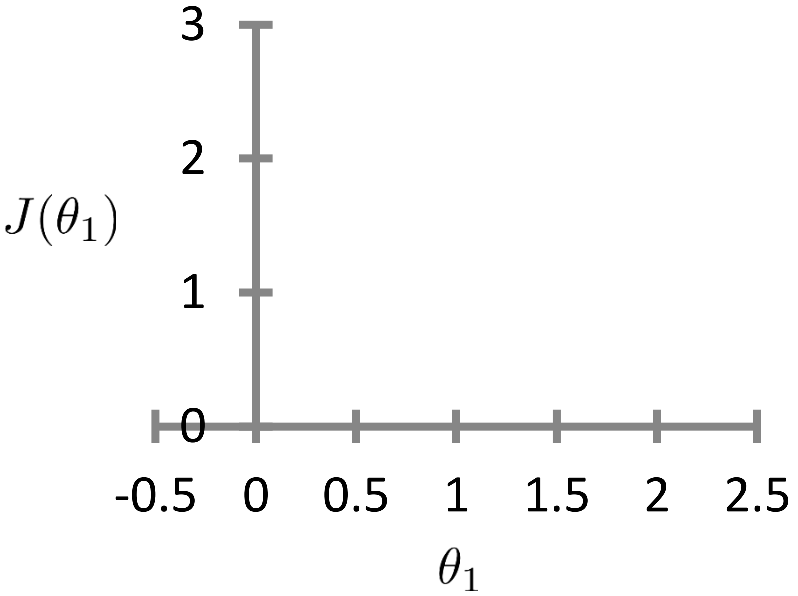
$h_\theta(x)$

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



$J(\theta_1)$

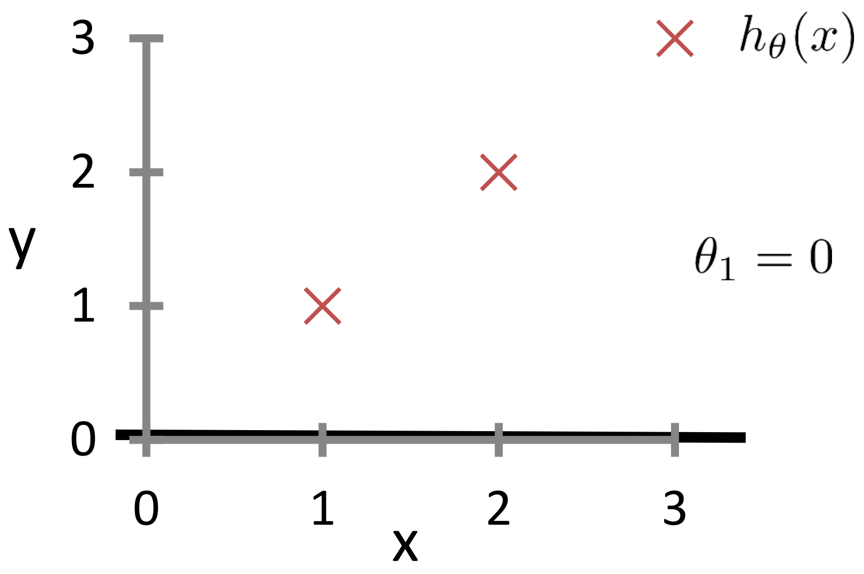
(function of the parameter θ_1)



Andrew Ng

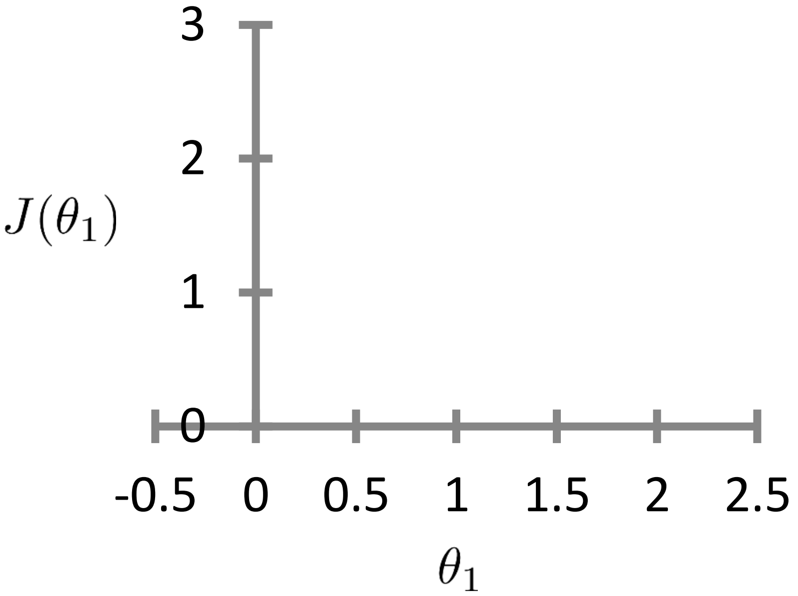
$$h_\theta(x)$$

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

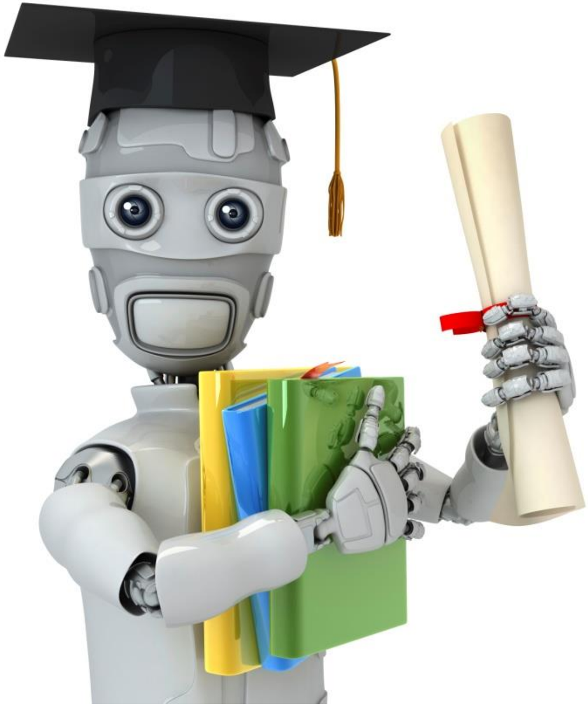


$$J(\theta_1)$$

(function of the parameter θ_1)



Andrew Ng



Machine Learning

Linear regression with one variable

Cost function intuition II

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

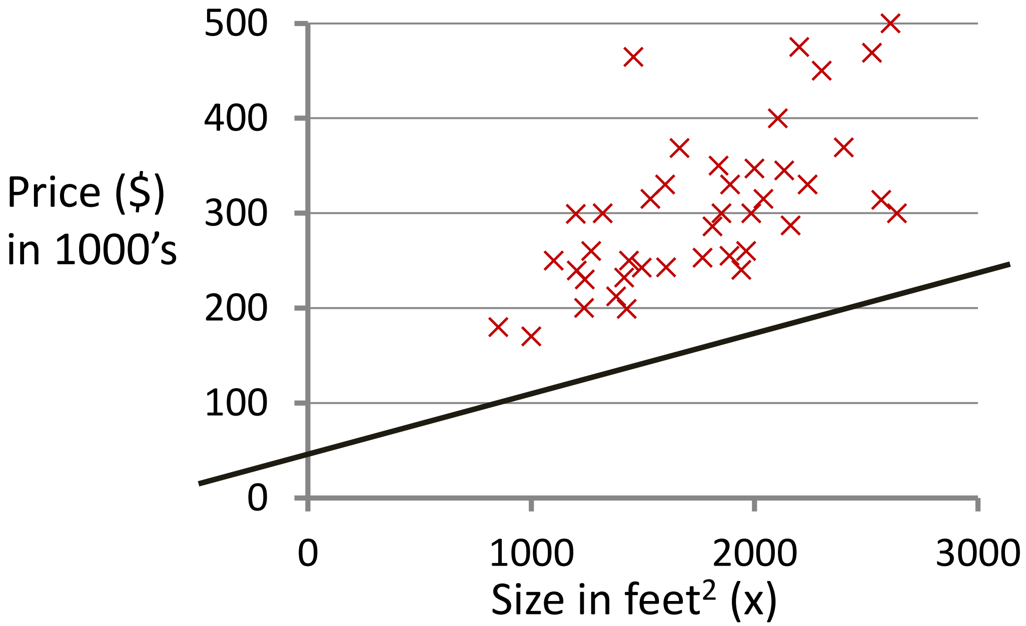
Parameters: θ_0, θ_1

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

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

$h_\theta(x)$

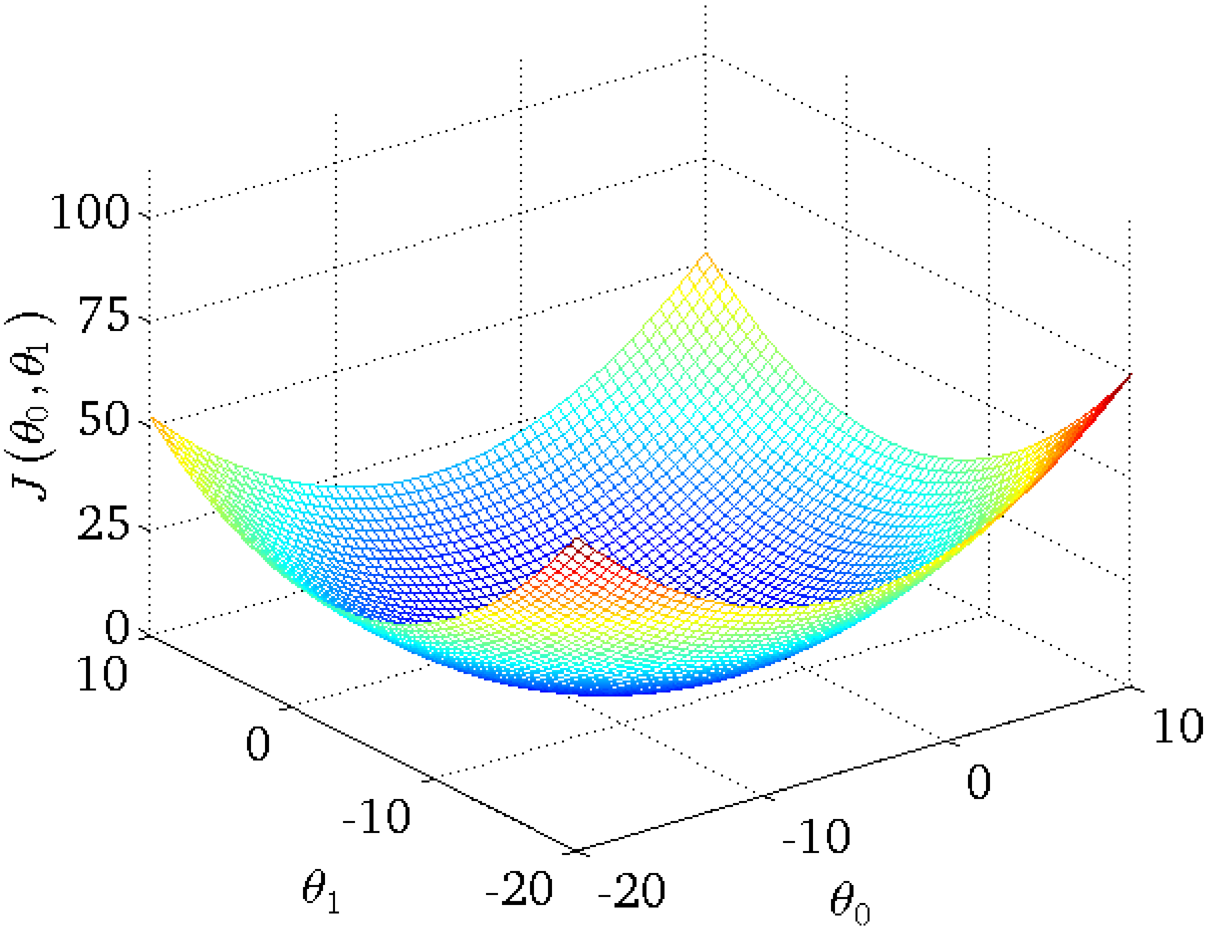
(for fixed θ_0, θ_1, this is a function of x)



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

$J(\theta_0, \theta_1)$

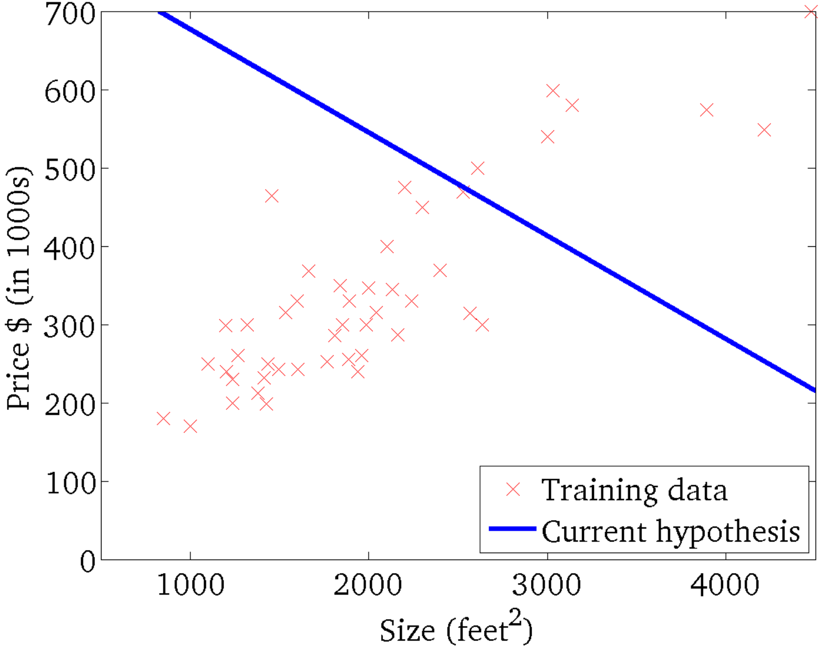
(function of the parameters θ_0, θ_1)



100
75
50
25
0
$J(\theta_0,\theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

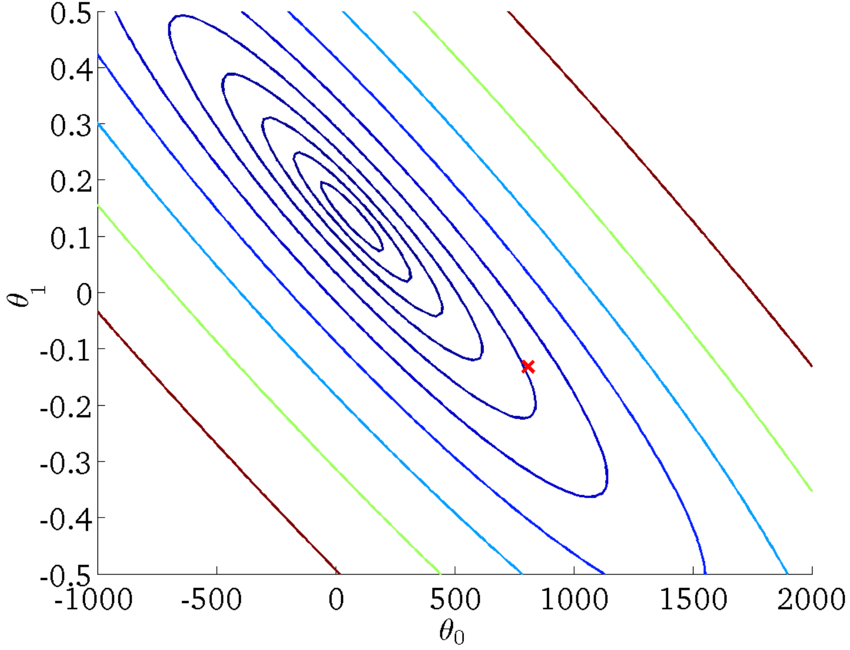
$$h_\theta(x)$$

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



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

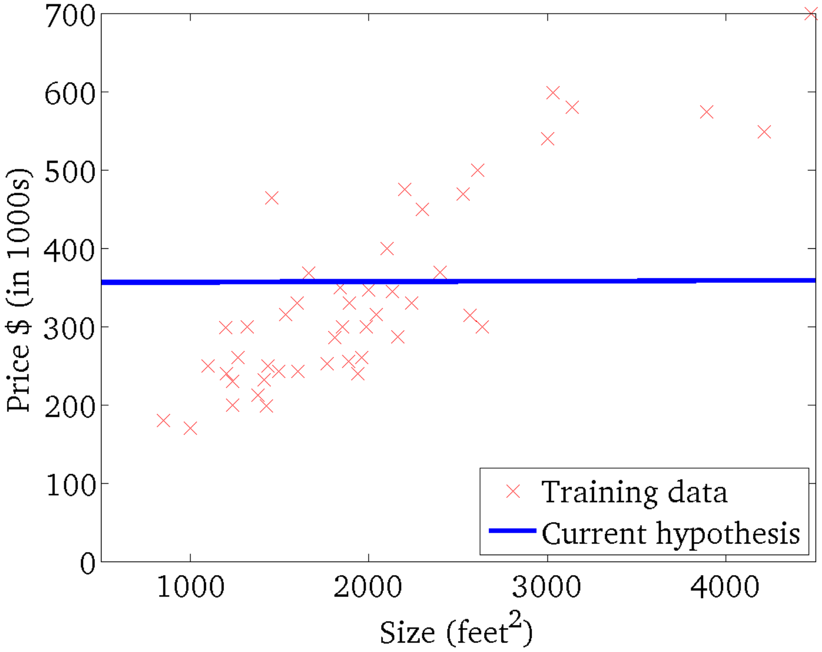
(function of the parameters θ_0, θ_1)



Andrew Ng

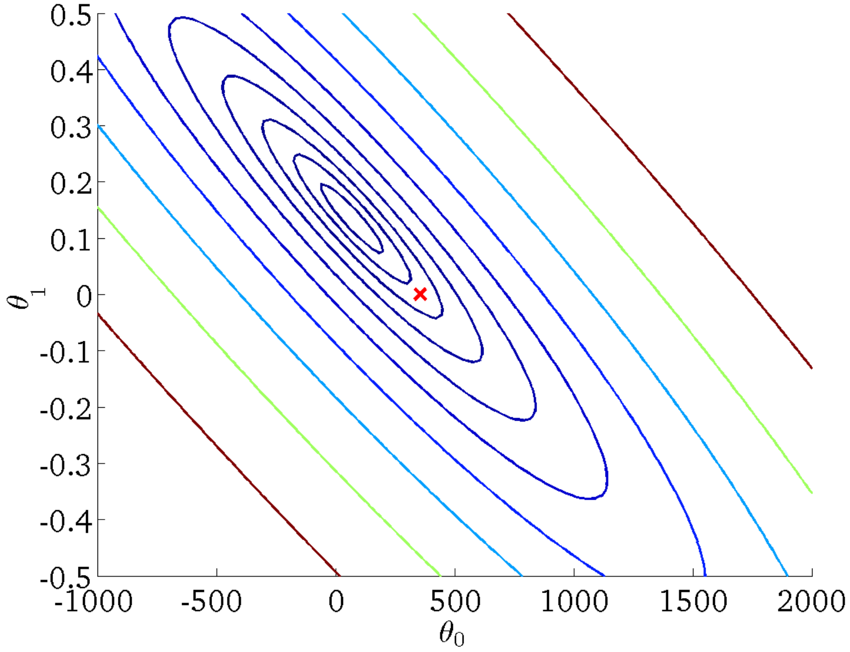
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

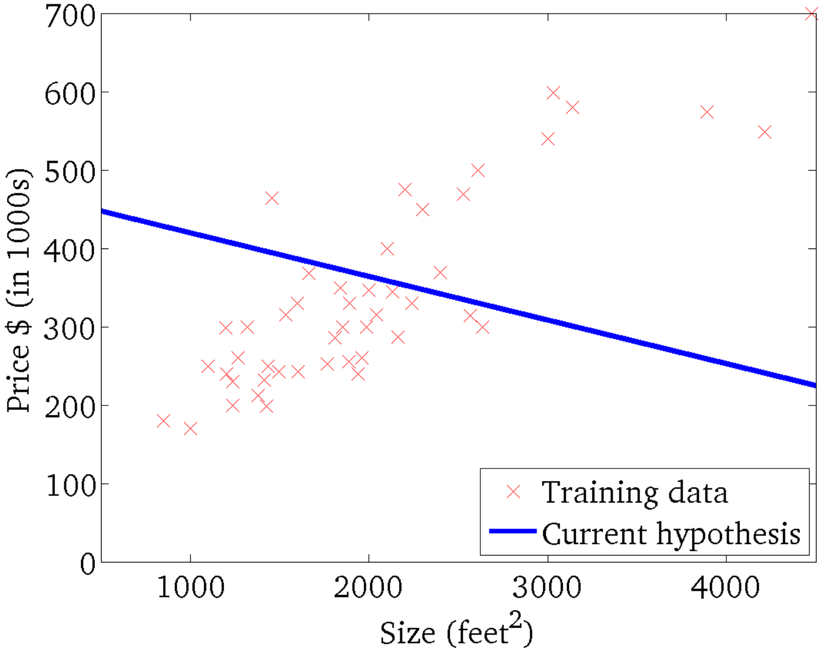
(function of the parameters θ_0, θ_1)



Andrew Ng

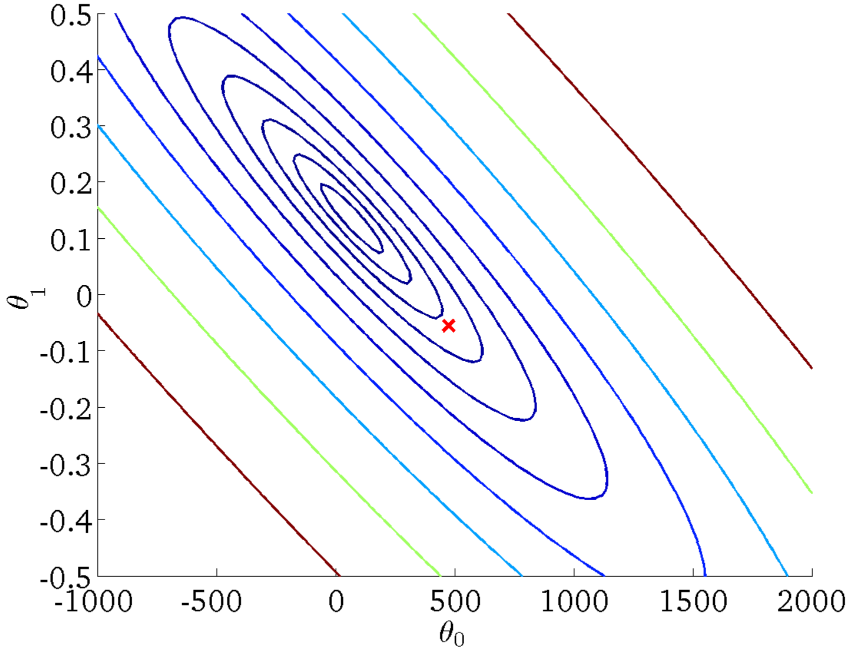
$$h_\theta(x)$$

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



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

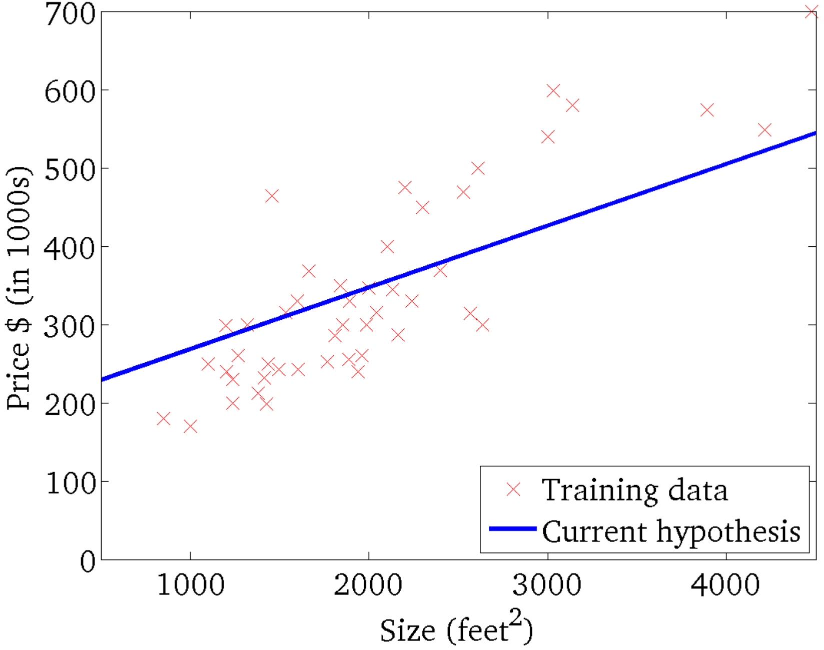
(function of the parameters θ_0, θ_1)



Andrew Ng

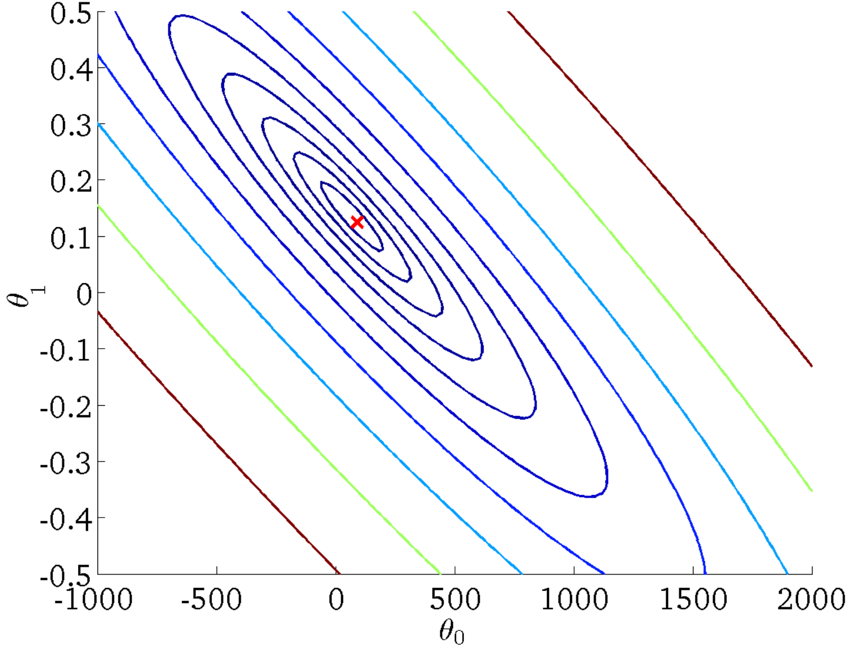
$h_\theta(x)$

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

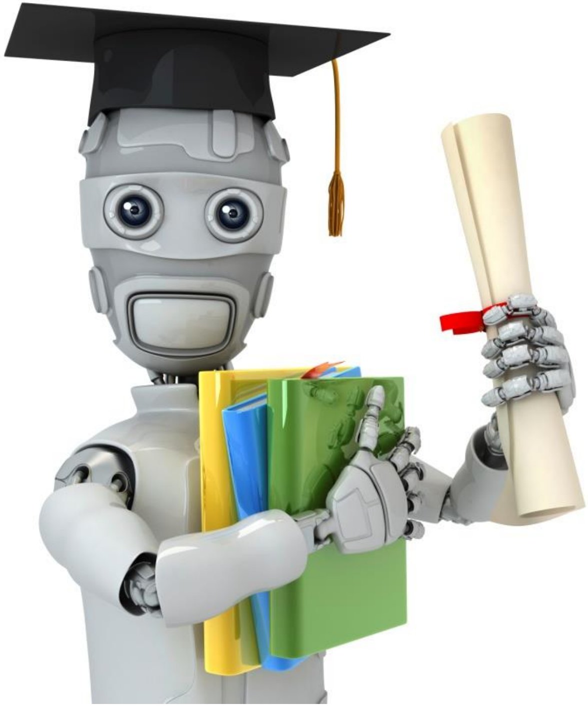


$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Andrew Ng



Machine Learning

Linear regression with one variable

Gradient descent

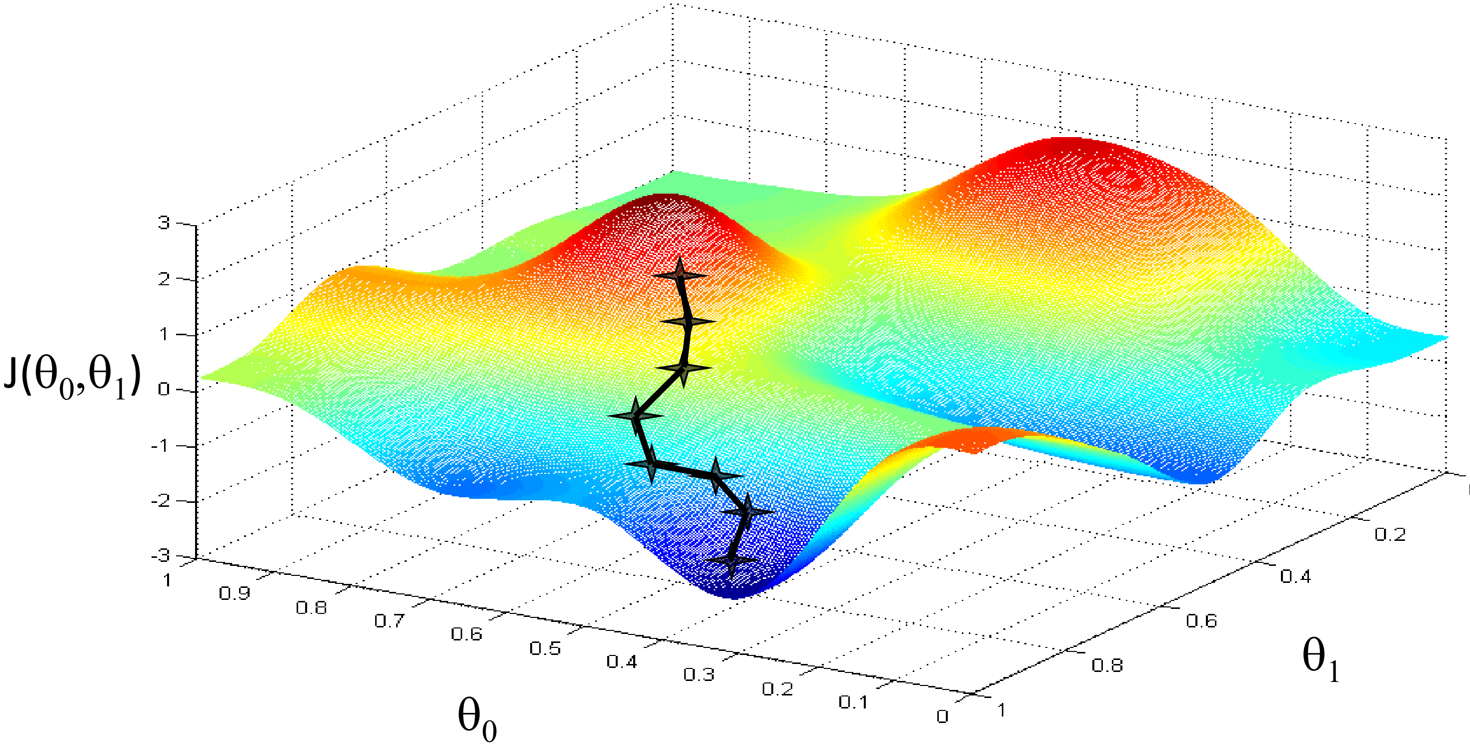
Have some function $J(\theta_0, \theta_1)$

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

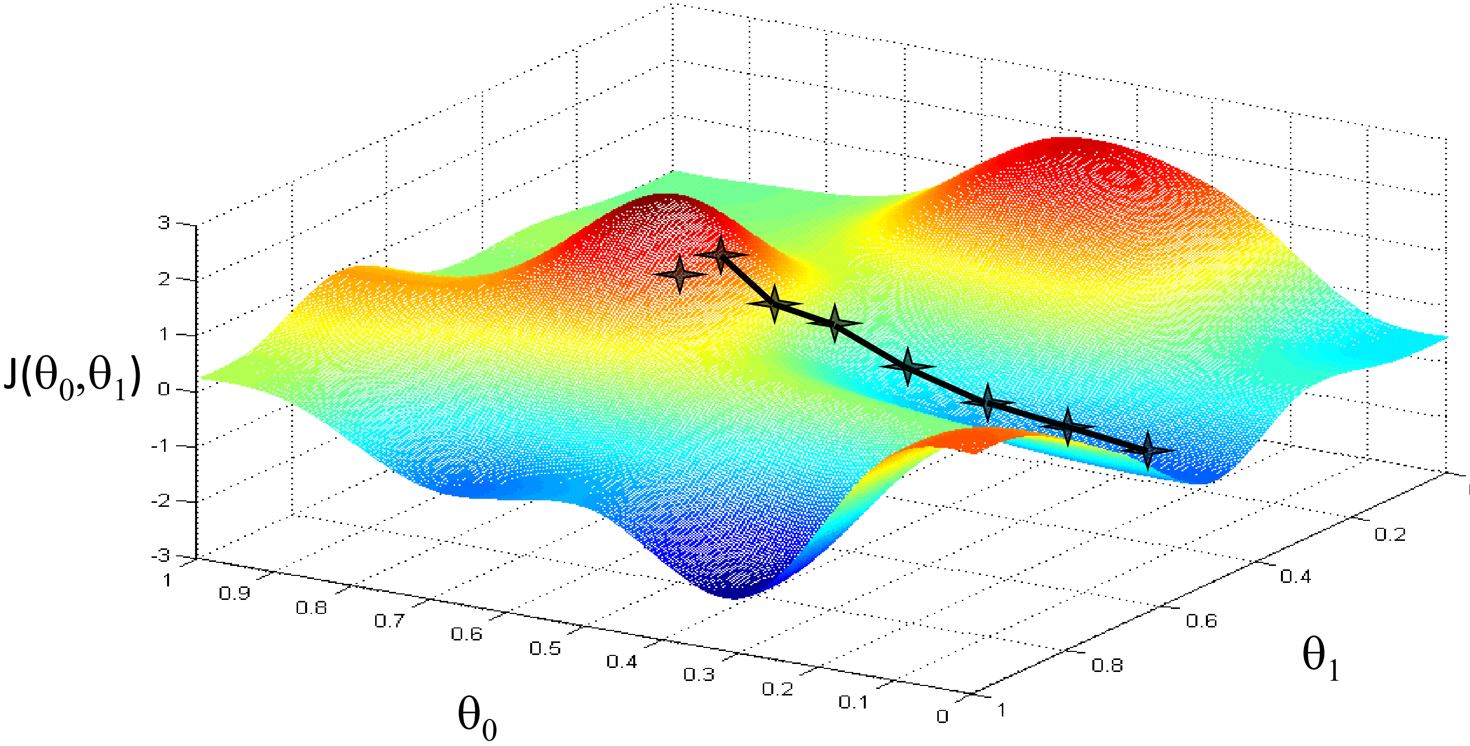
Outline:

- 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



$J(\theta_0,\theta_1)$
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



$J(\theta_0,\theta_1)$
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

Gradient descent algorithm

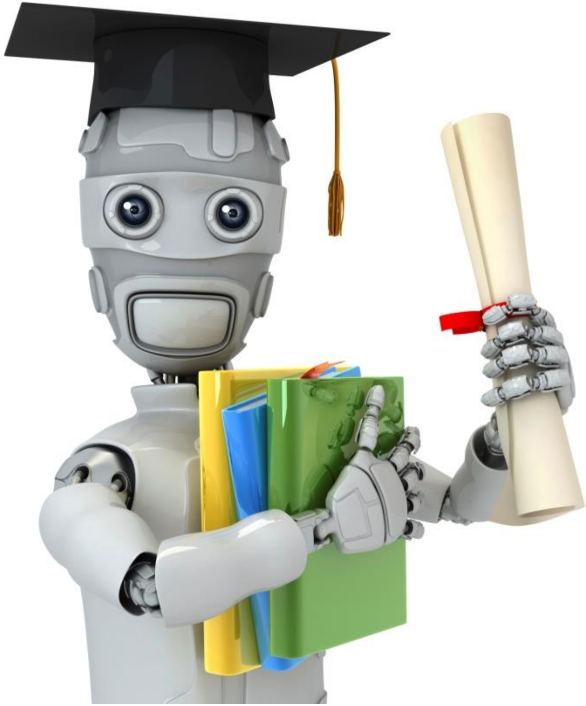
$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1) \\
&\}
\end{aligned}
$$

Correct: Simultaneous update

$$
\begin{aligned}
\text{temp0} &:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
\text{temp1} &:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
\theta_0 &:= \text{temp0} \\
\theta_1 &:= \text{temp1}
\end{aligned}
$$

Incorrect:

$$
\begin{aligned}
\text{temp0} &:= \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \\
\theta_0 &:= \text{temp0} \\
\text{temp1} &:= \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \\
\theta_1 &:= \text{temp1}
\end{aligned}
$$



Machine Learning

Linear regression with one variable

Gradient descent intuition

Gradient descent algorithm

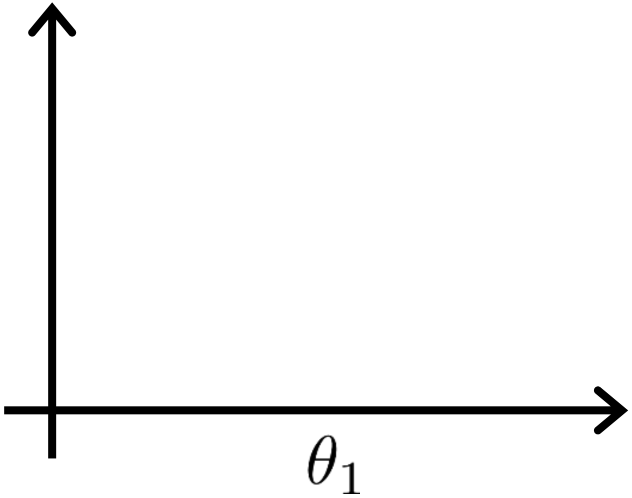
$$
\text{repeat until convergence } \{
$$

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

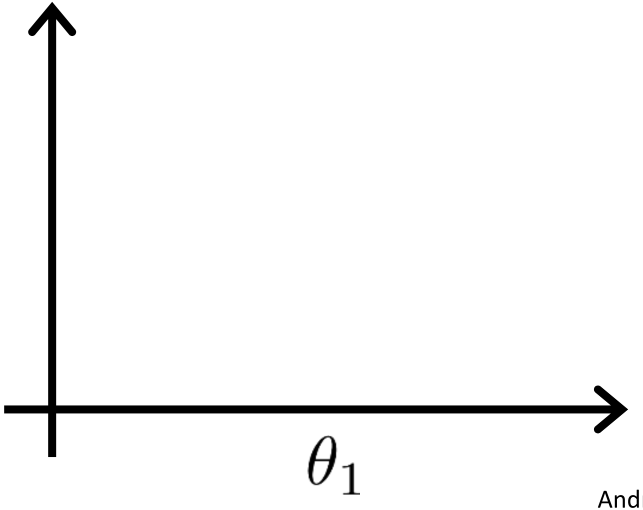
$$
\}
$$

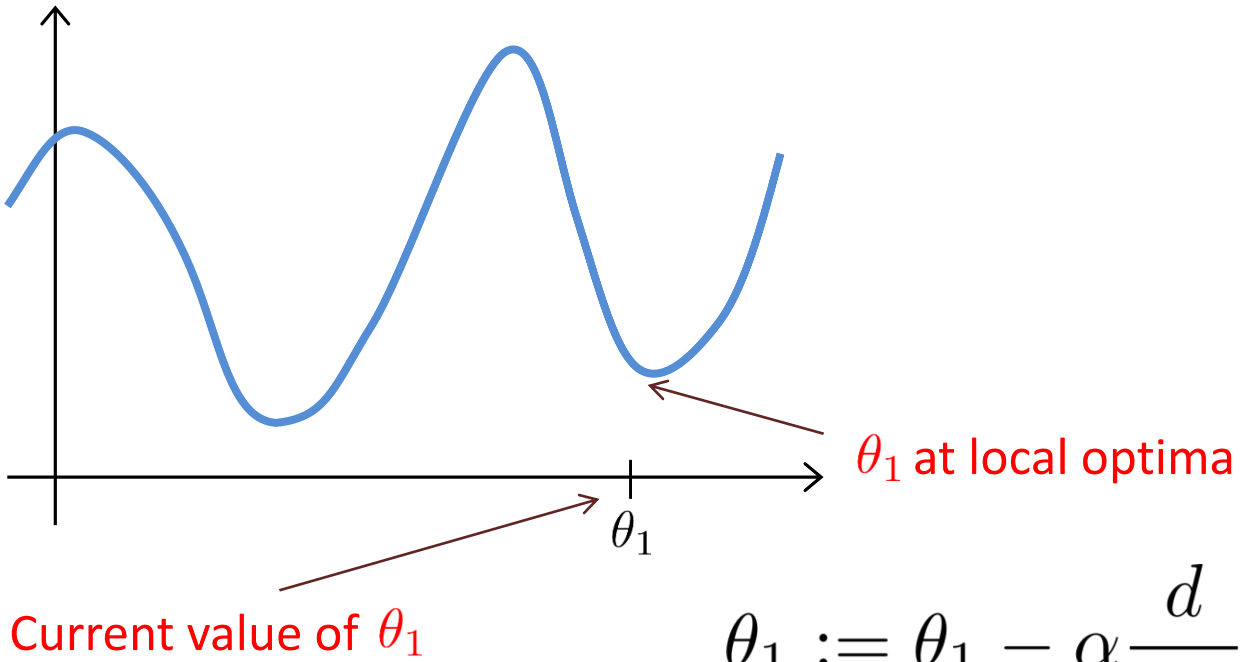
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

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.



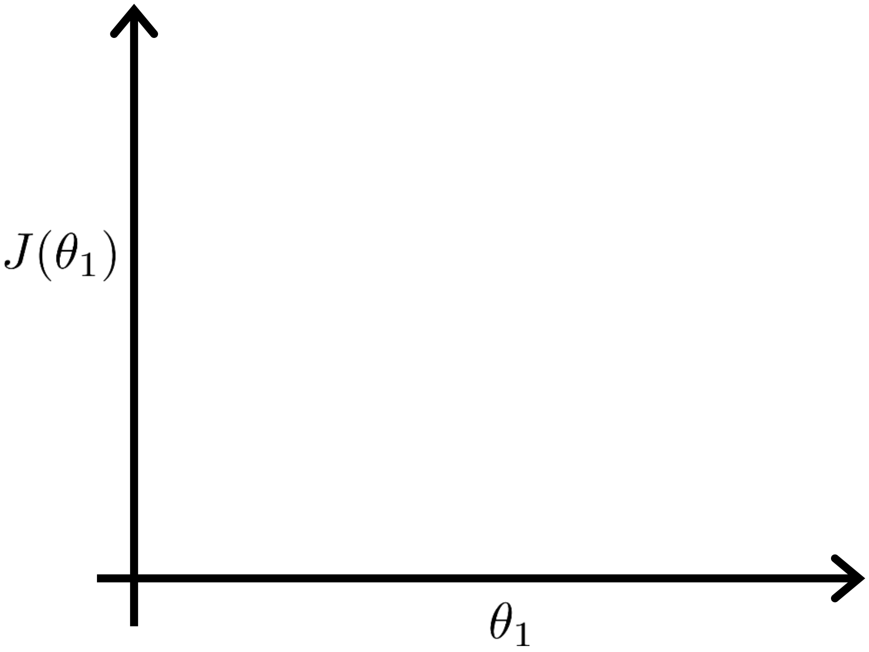


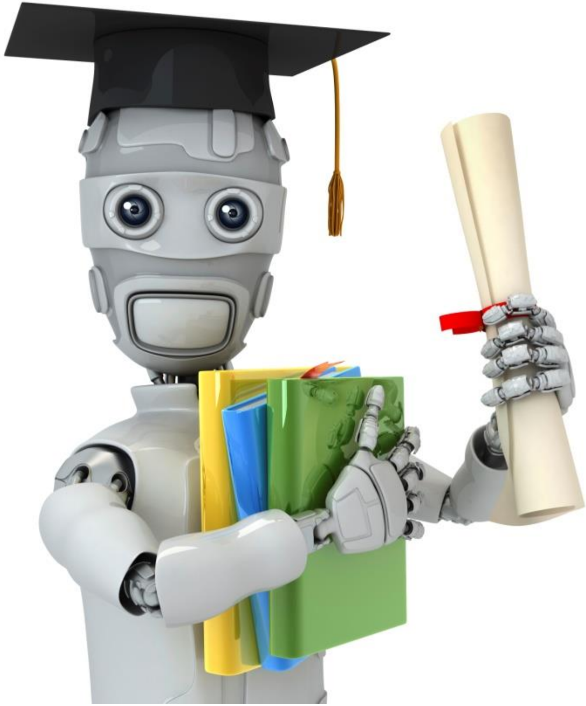
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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

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

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

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

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

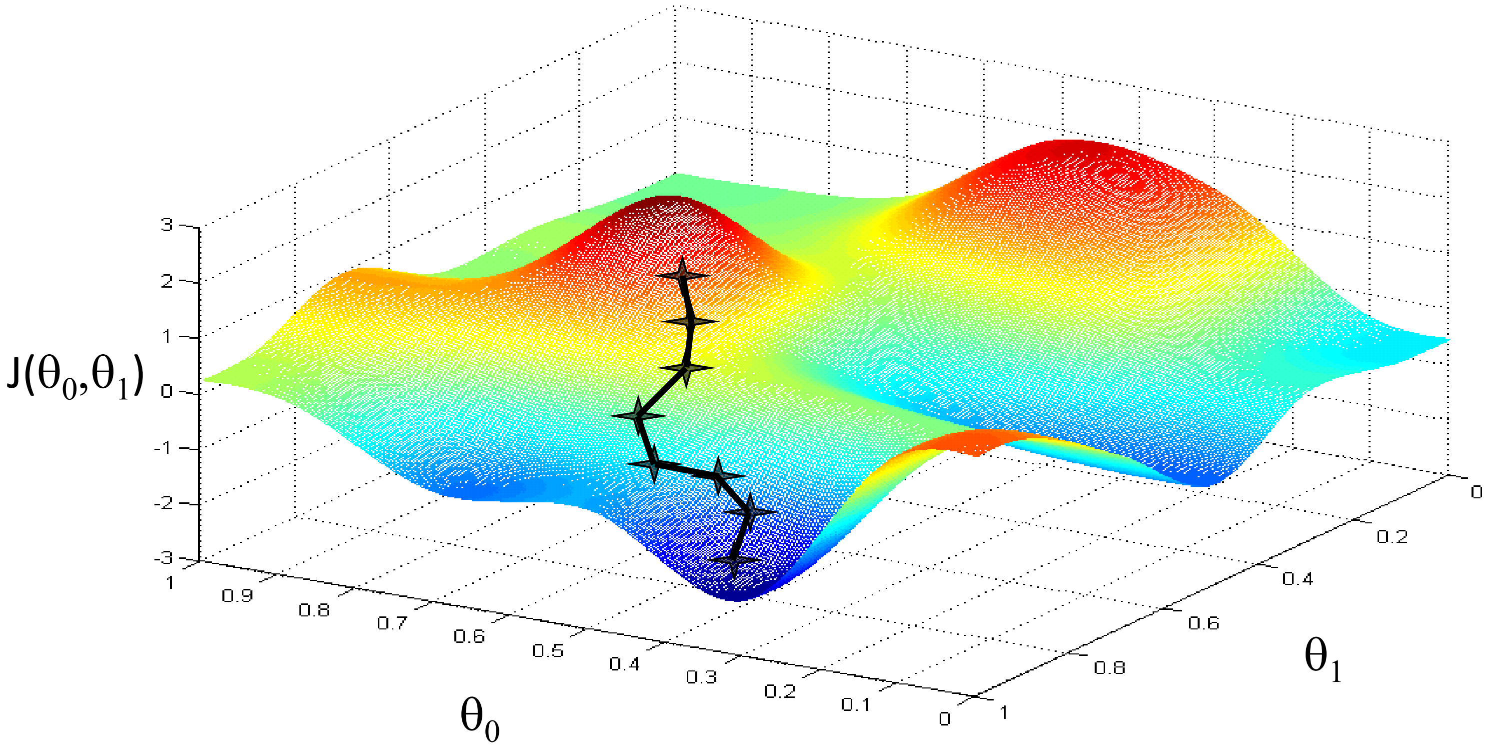
$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) =$$

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) =$$

$$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) =$$

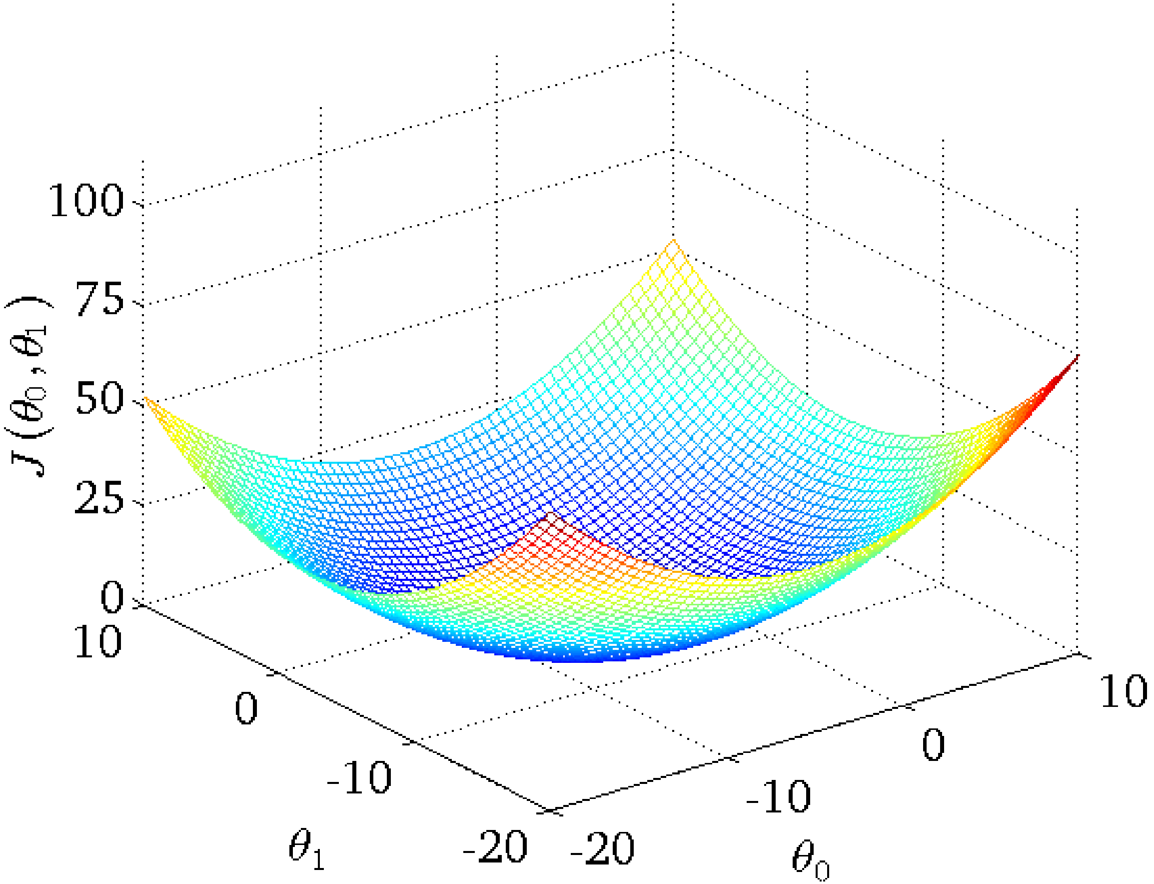
Gradient descent algorithm

$$
\left.
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \\
&\quad \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)} \\
&\}
\end{aligned}
\right\}
\text{ update } \theta_0 \text{ and } \theta_1 \text{ simultaneously}
$$



$J(\theta_0,\theta_1)$
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

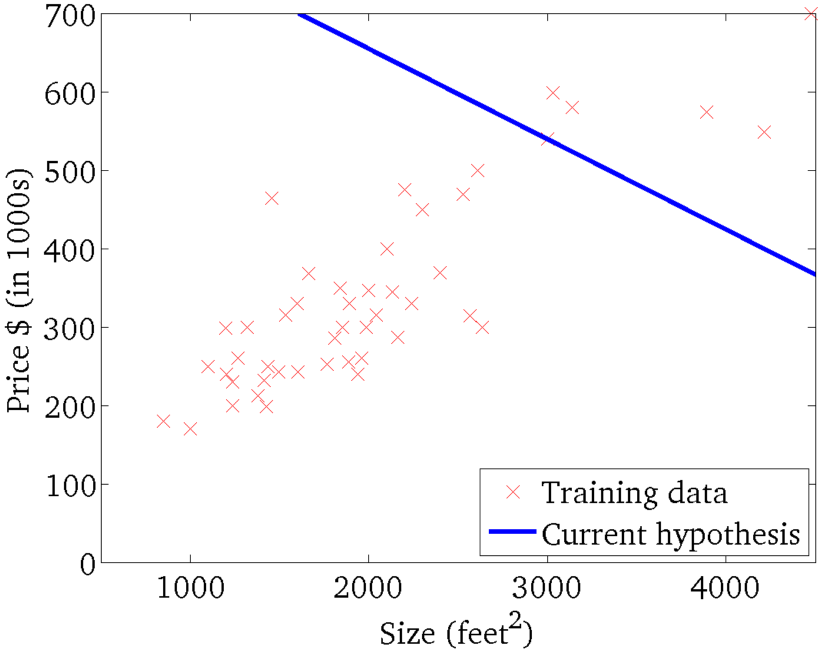
Andrew Ng



100
75
50
25
0
$J(\theta_0, \theta_1)$
10
0
-10
-20
θ_1
-20
-10
0
10
θ_0

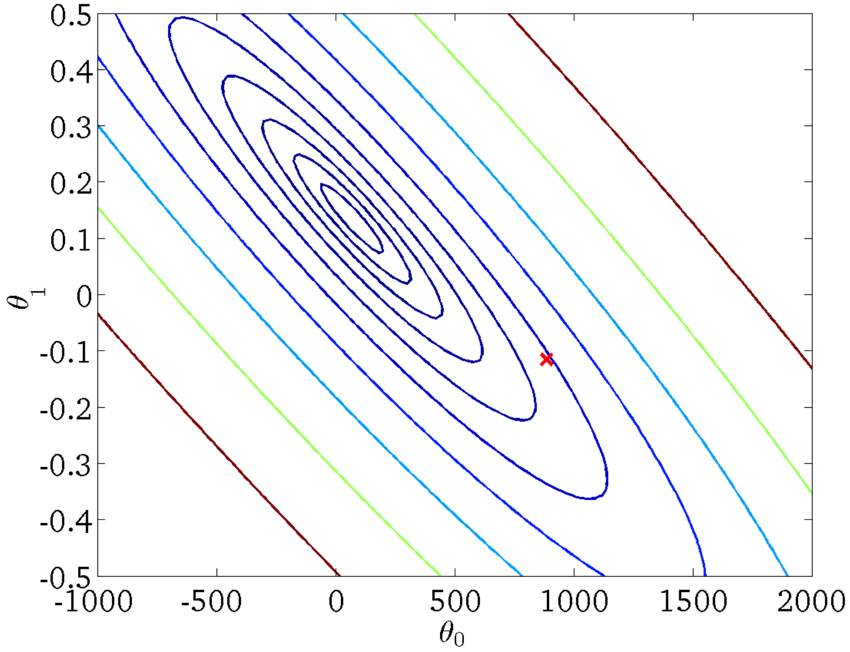
$$h_\theta(x)$$

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



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

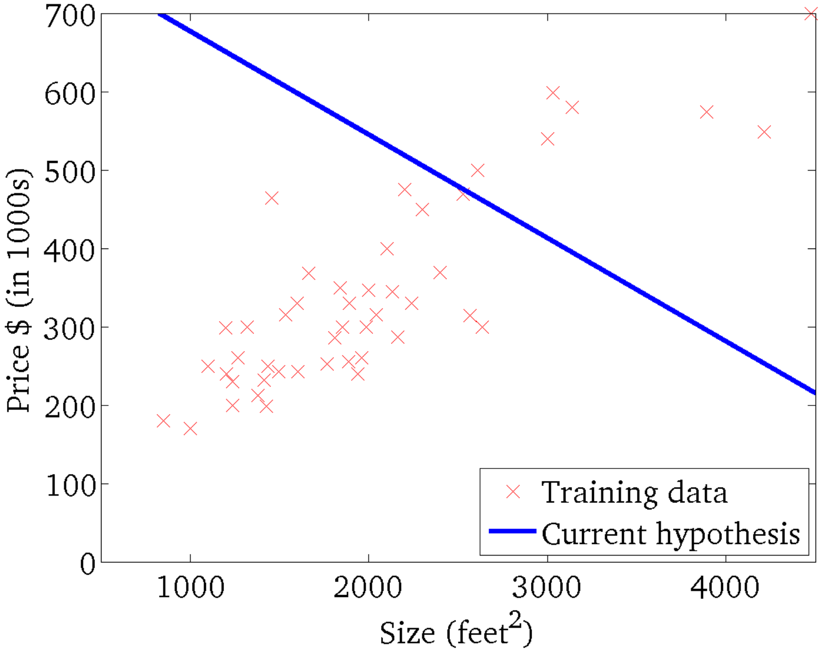
(function of the parameters θ_0, θ_1)



Andrew Ng

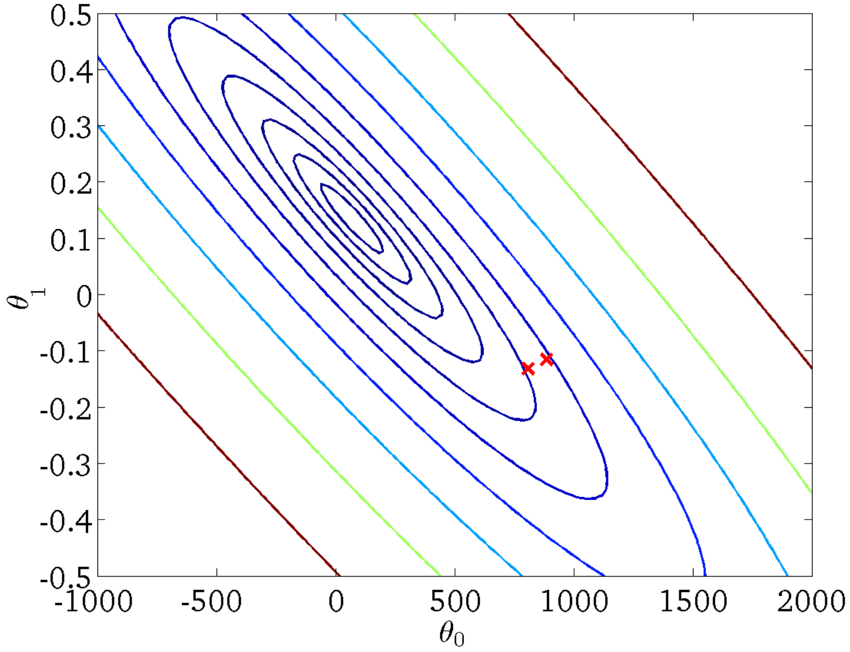
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

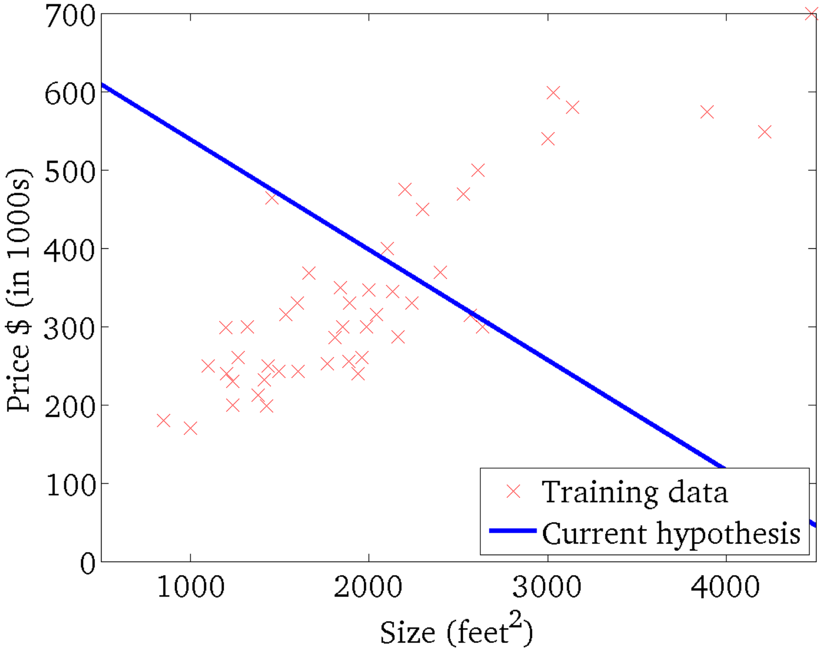
(function of the parameters θ_0, θ_1)



Andrew Ng

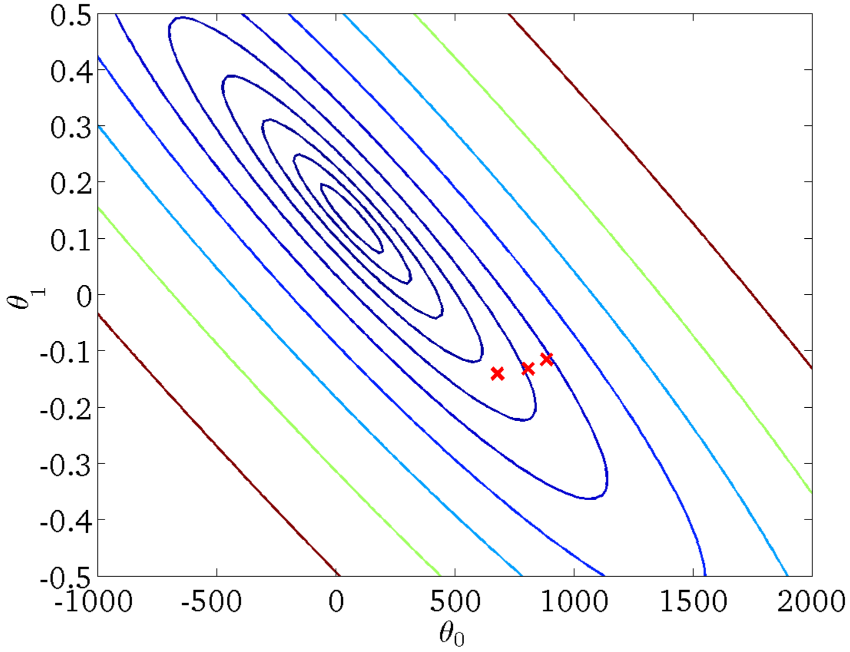
$$h_\theta(x)$$

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



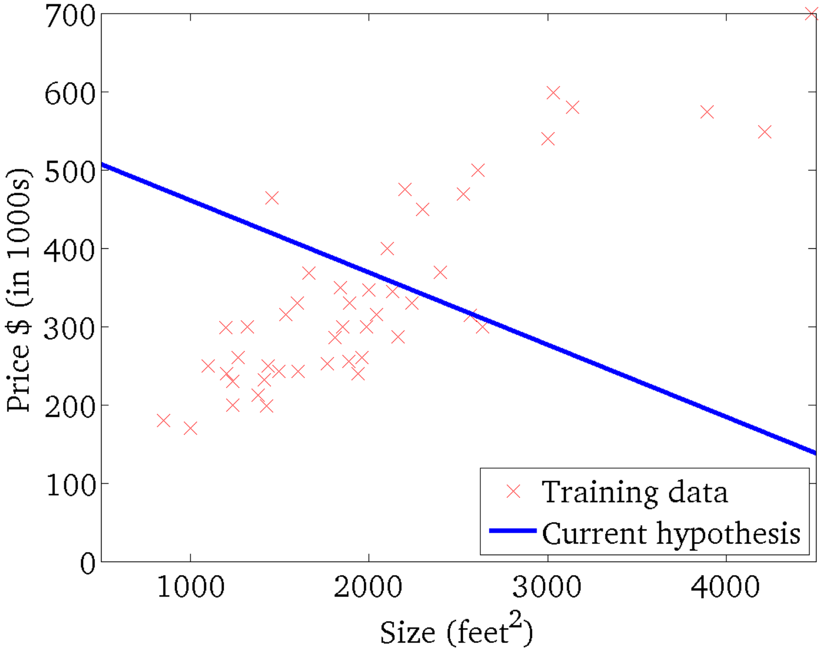
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



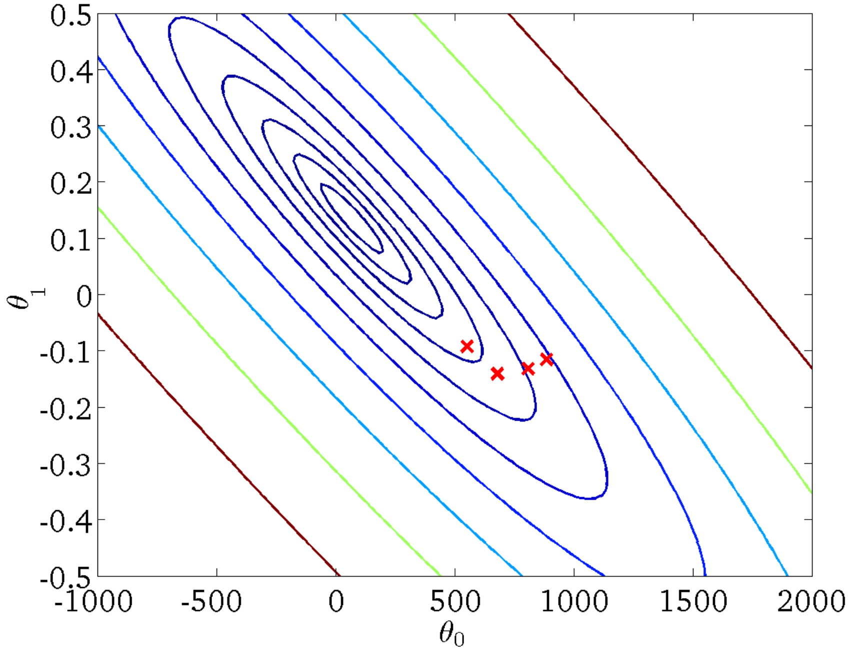
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

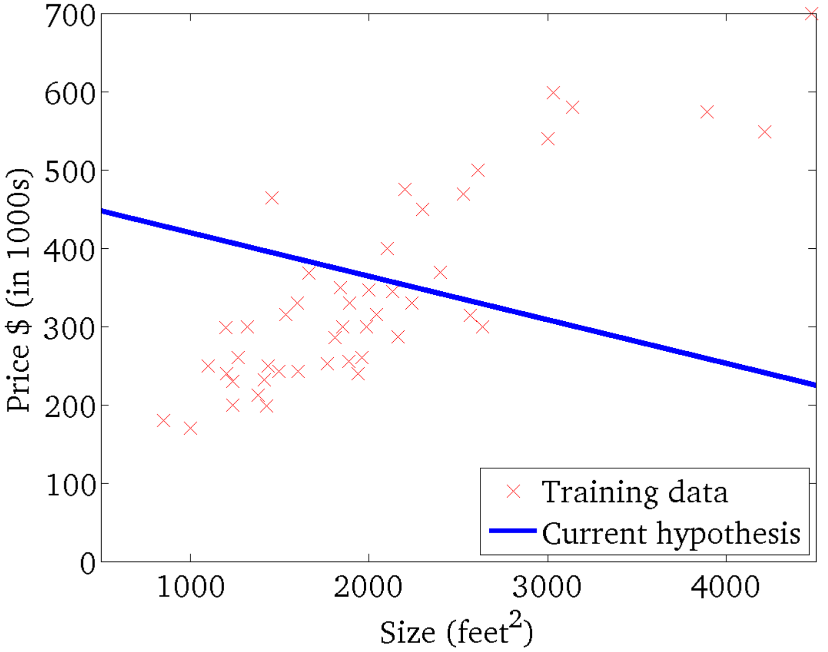
(function of the parameters θ_0, θ_1)



Andrew Ng

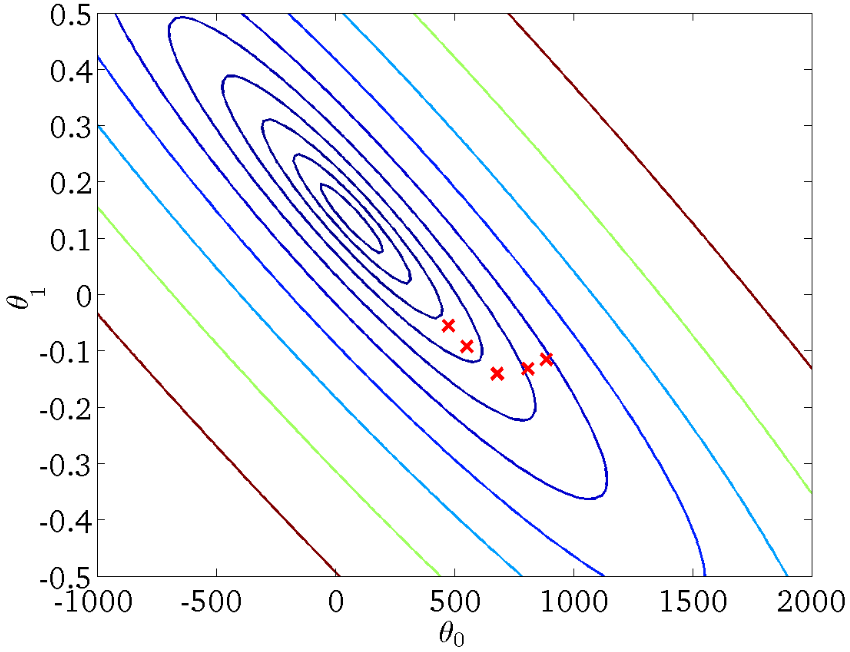
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

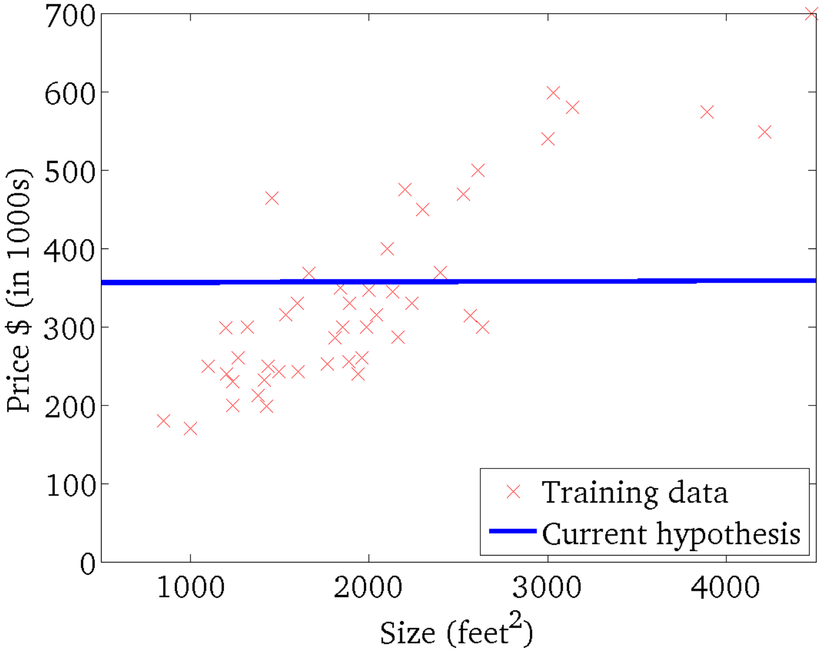
(function of the parameters θ_0, θ_1)



Andrew Ng

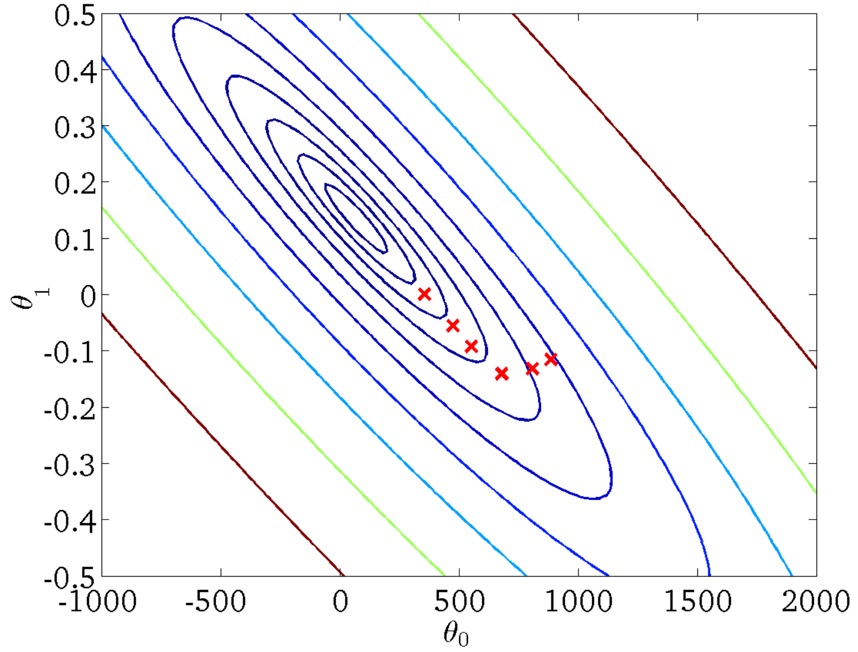
$$h_\theta(x)$$

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



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

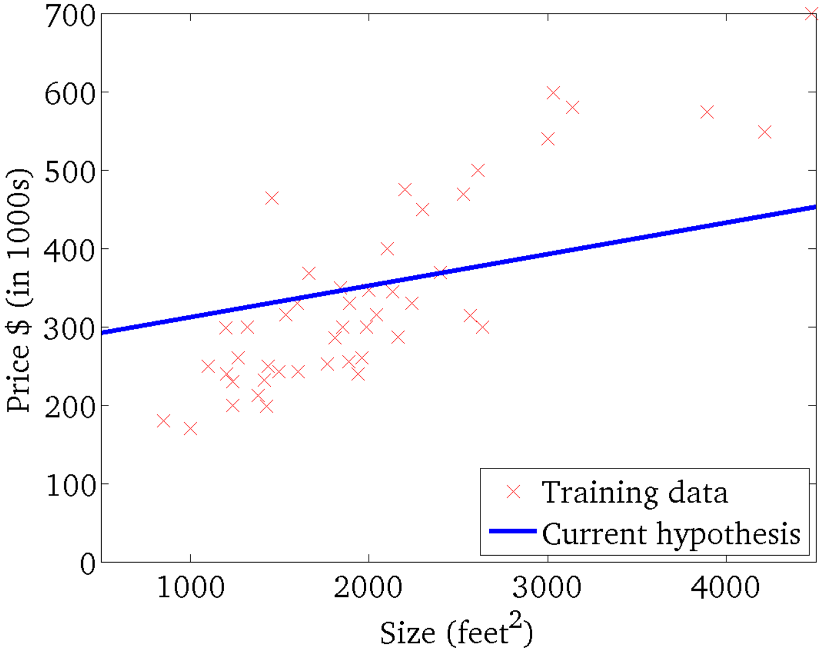
(function of the parameters θ_0, θ_1)



Andrew Ng

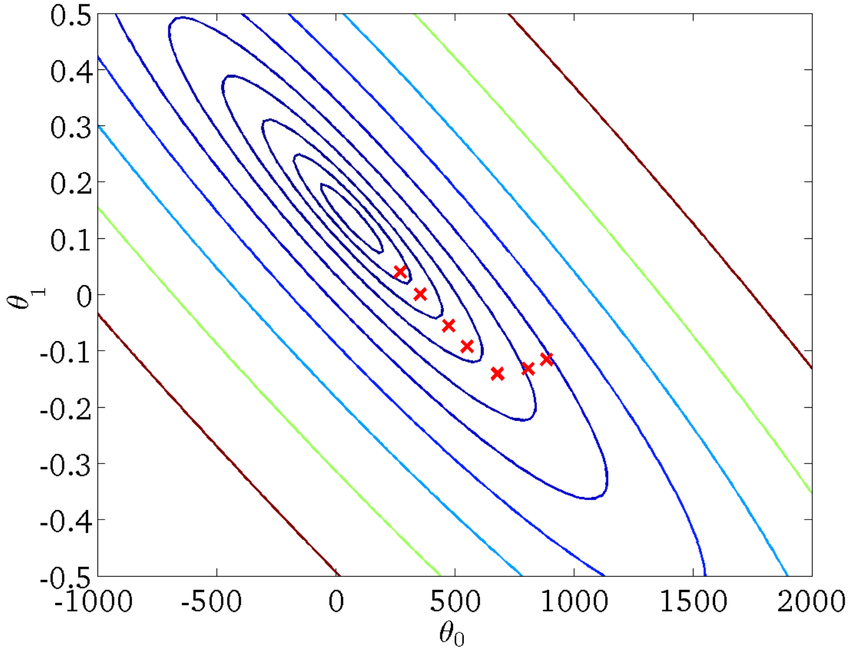
$$h_\theta(x)$$

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



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

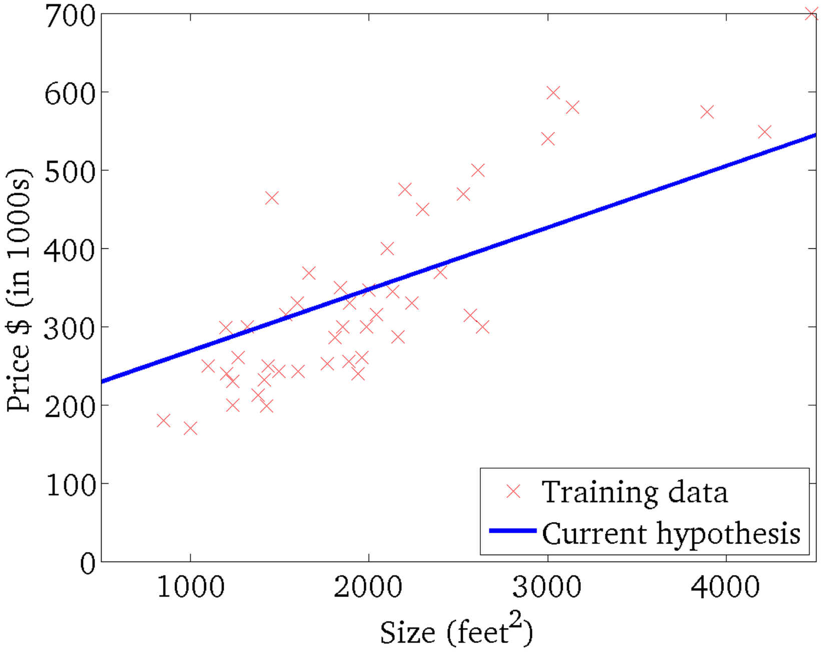
(function of the parameters θ_0, θ_1)



Andrew Ng

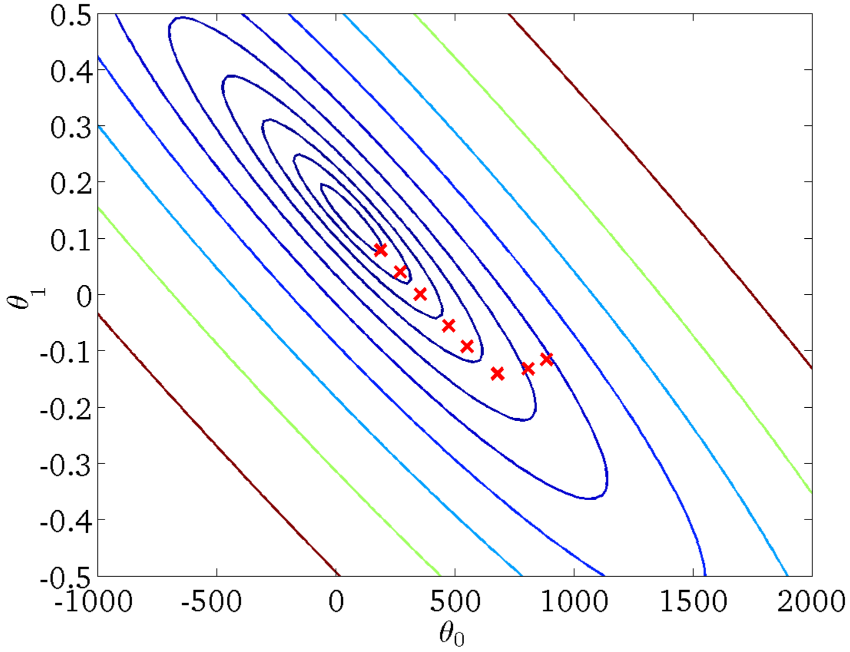
$$h_\theta(x)$$

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



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

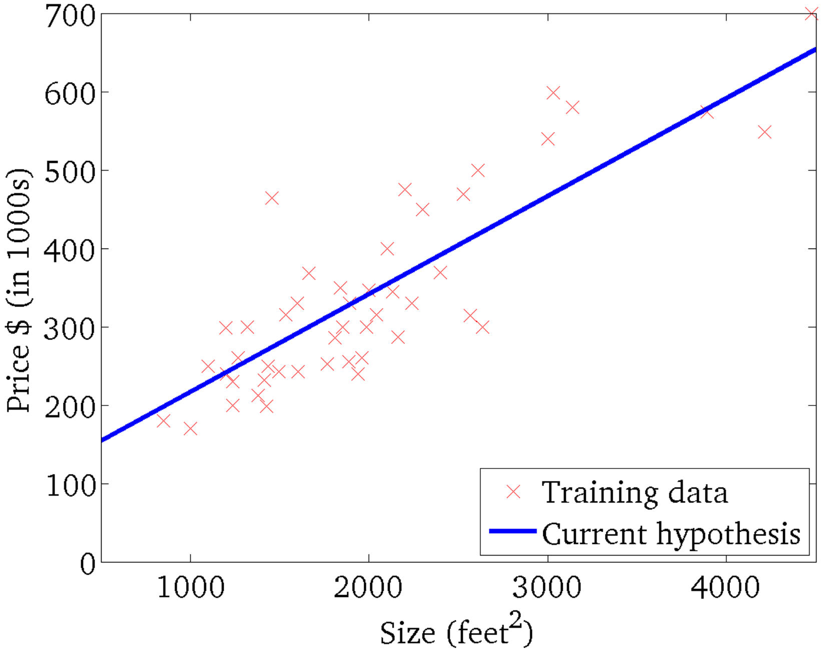
(function of the parameters θ_0, θ_1)



Andrew Ng

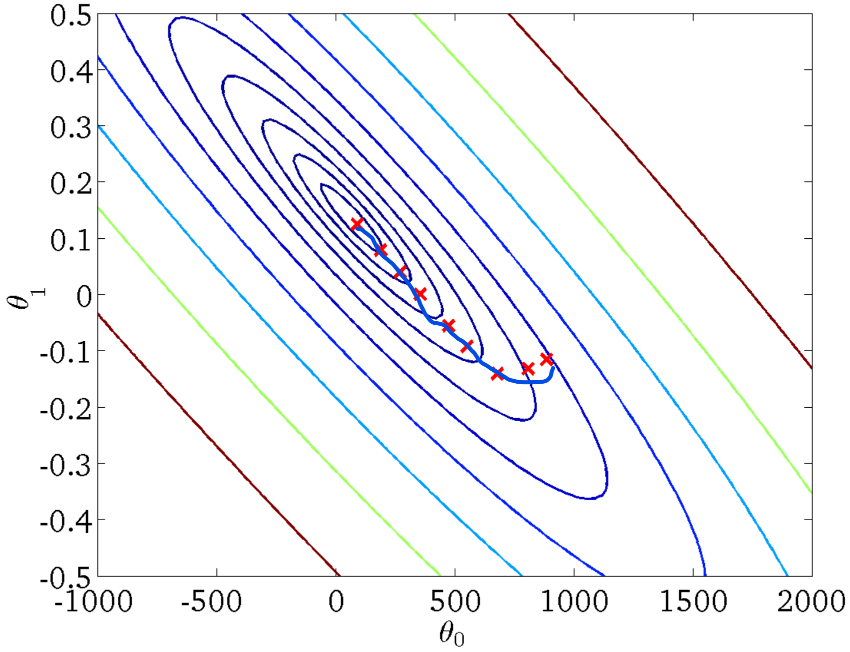
$$h_\theta(x)$$

(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.

- **“Batch” Gradient Descent**

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

- **“Stochastic” Gradient Descent**

“Single”: Each step of gradient descent uses one training example and moves to the next.