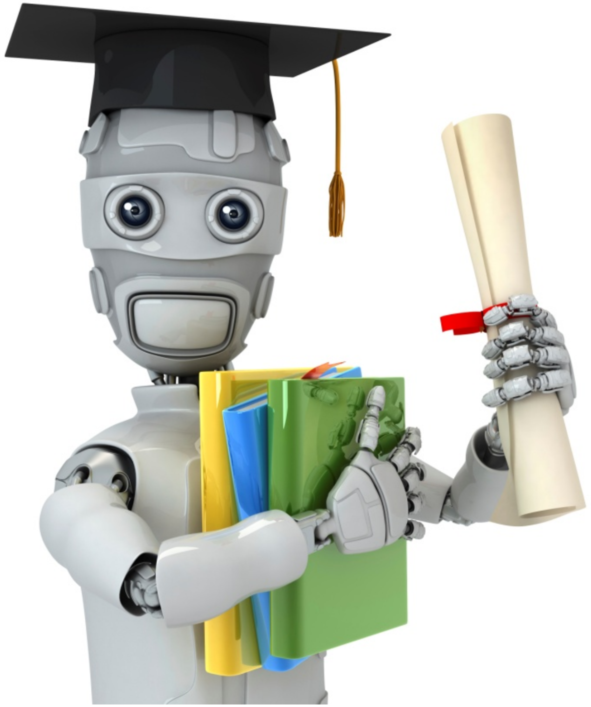
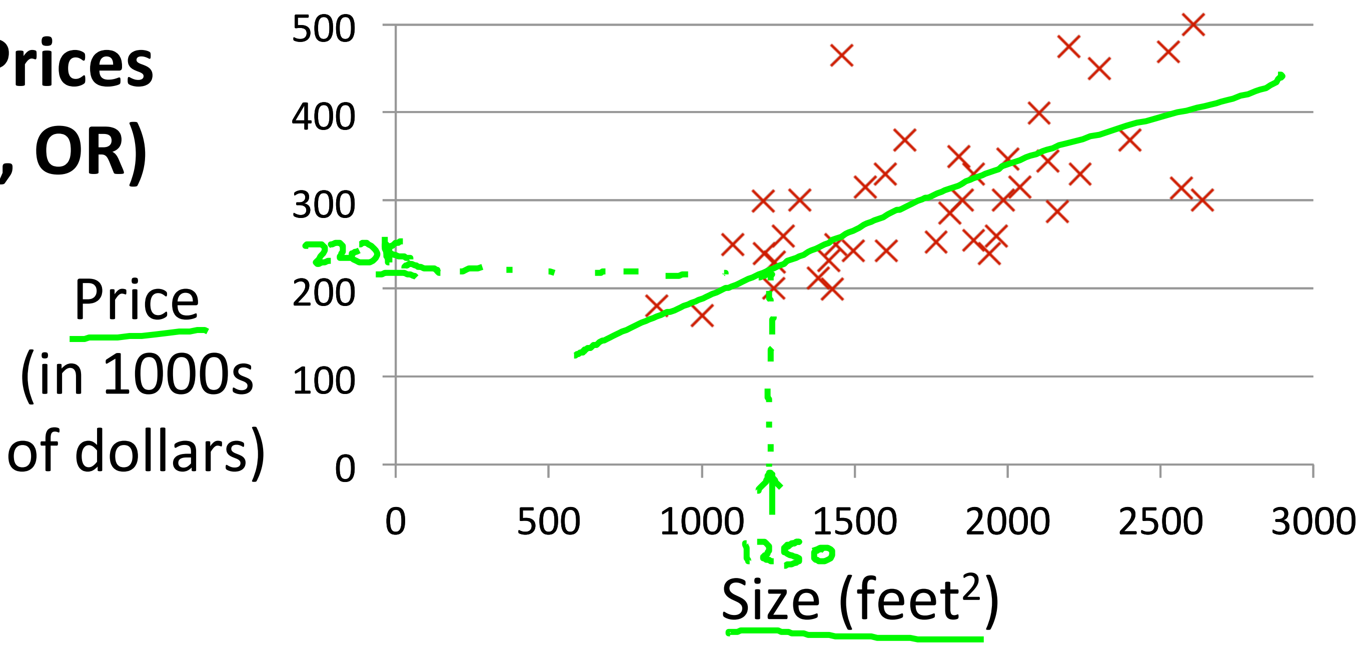


Machine Learning

Linear regression with one variable

Model representation

Housing Prices (Portland, OR)



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)

first training example (x,y)

Size in feet^2 (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

$x^{(1)} = 2104$

$x^{(2)} = 1416$

$y^{(1)} = 460$

Andrew Ng

$$\begin{tikzcd} & \text{Training Set} \arrow[d] & \\ & \text{Learning Algorithm} \arrow[d] & \\ \text{Size of house} \arrow[r] & \boxed{h} \arrow[r] & \text{Estimated price} \end{tikzcd}$$

$\underline{x}$ $\qquad$ $\underline{\text{hypothesis}}$ $\qquad$ $\underline{\text{(estimated value of } y)}$

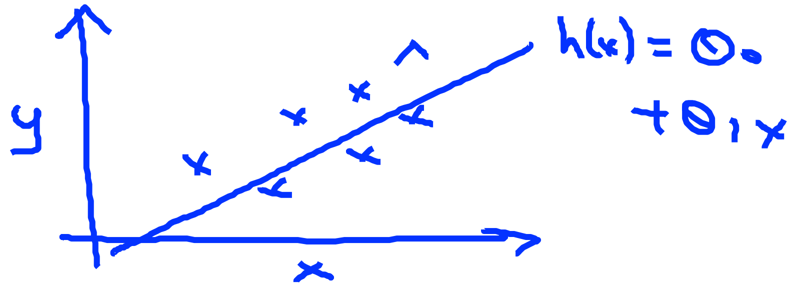
h maps from x's to y's.

h is a function that matches x to y to predict
(not great name hypothesis for this function)

How do we represent *h* ?

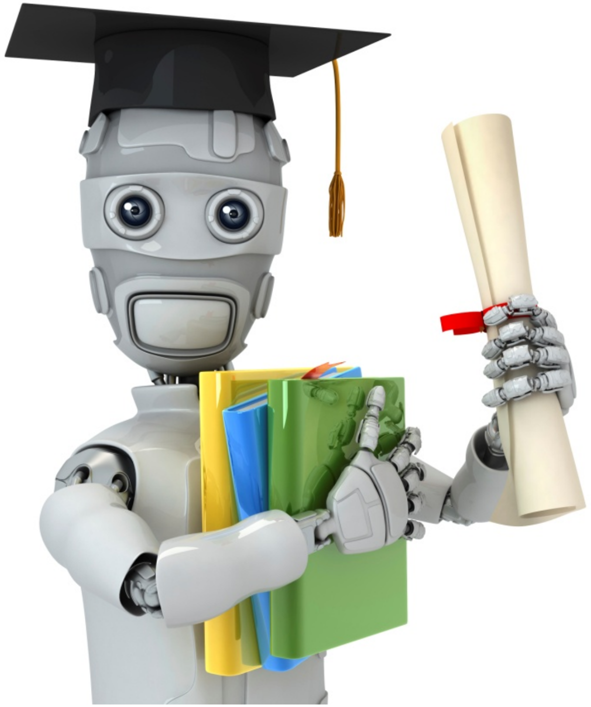
$h_{\underline{\theta}}(x) = \underline{\theta_0 + \theta_1 x}$

$\bar{\text{S}}$horthand: $h(x)$



Linear regression with one variable. (x)
$\underline{\text{Univariate}}$ 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
…	…

$m = 47$

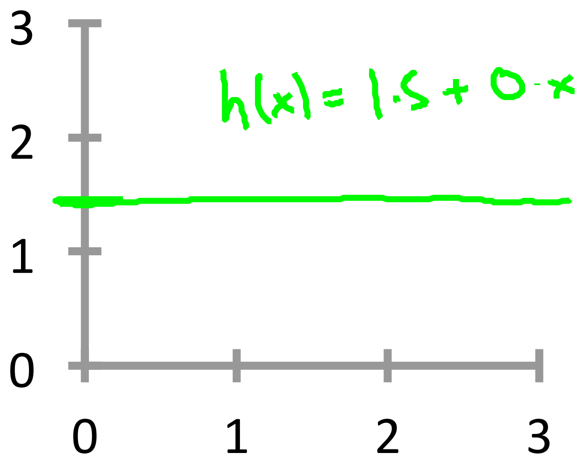
Hypothesis: $\underline{h_\theta(x) = \underset{\uparrow}{\textcircled{\theta_0}} + \underset{\uparrow}{\textcircled{\theta_1}} x}$

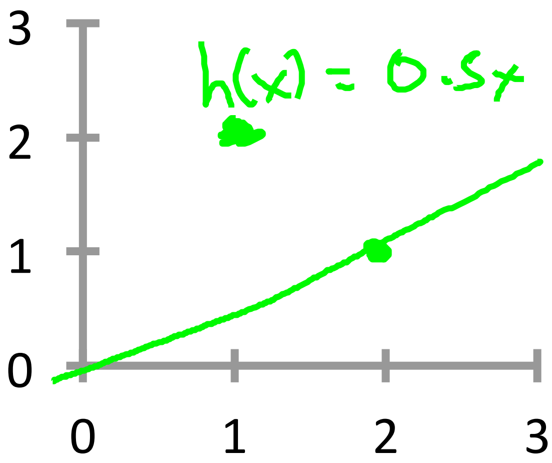
θ_i's: $\underline{\text{Parameters}}$

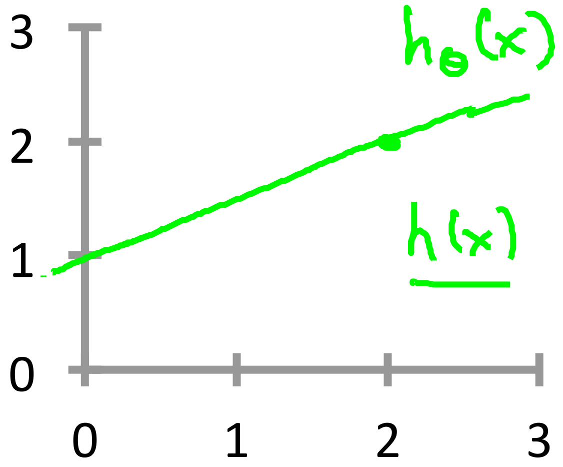
How to choose θ_i's ?

Andrew Ng

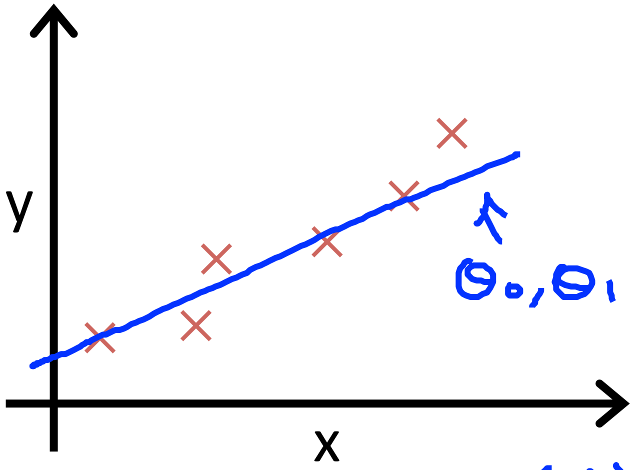
$$\underline{h_\theta(x)} = \theta_0 + \theta_1 x$$







$\rightarrow \theta_0 = 1.5$	$\rightarrow \theta_0 = 0$	$\rightarrow \theta_0 = 1$
$\rightarrow \theta_1 = 0$	$\rightarrow \theta_1 = 0.5$	$\rightarrow \theta_1 = 0.5$



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

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

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

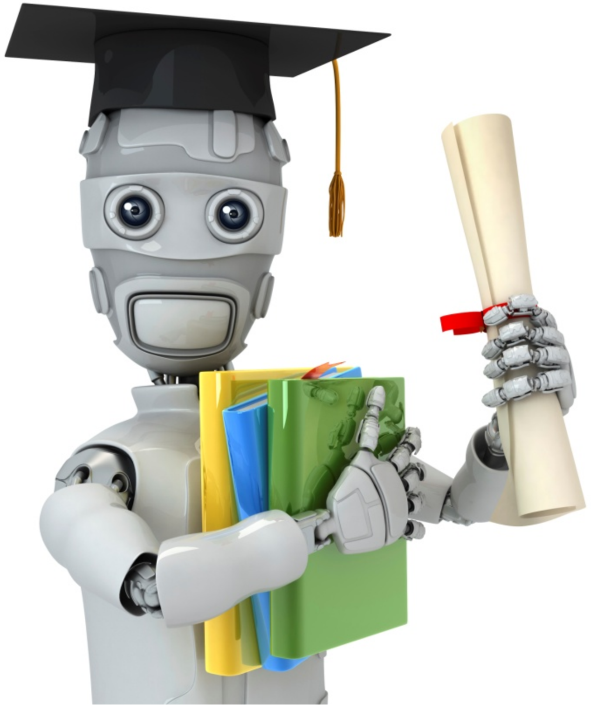
$$\underline{h_\theta(x^{(i)}) = \underline{\theta_0} + \underline{\theta_1} x^{(i)}}$$

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

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

Squared error function

Andrew Ng



Machine Learning

Linear regression with one variable

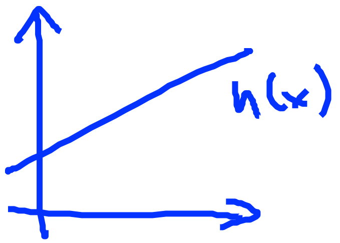
Cost function intuition I

Hypothesis:

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

Parameters:

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



Cost Function:

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

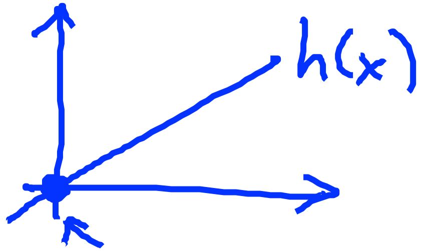
Goal: $\nearrow \; \underset{\theta_0, \theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$

Simplified

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

$$\theta_0 = 0$$

$$\underline{\theta_1}$$

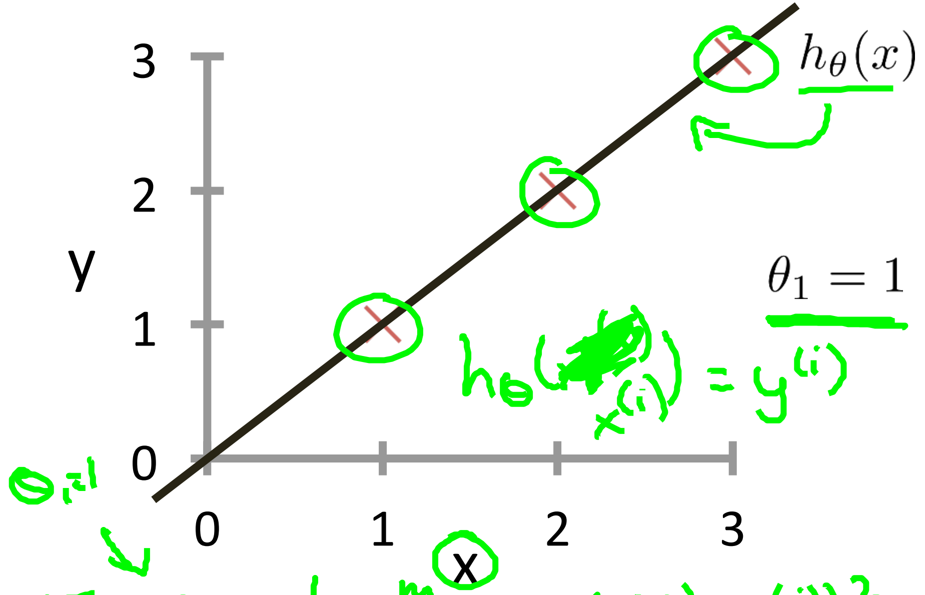


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(\underbrace{h_\theta(x^{(i)})}_{\theta, x^{(i)}} - y^{(i)}\right)^2$$

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

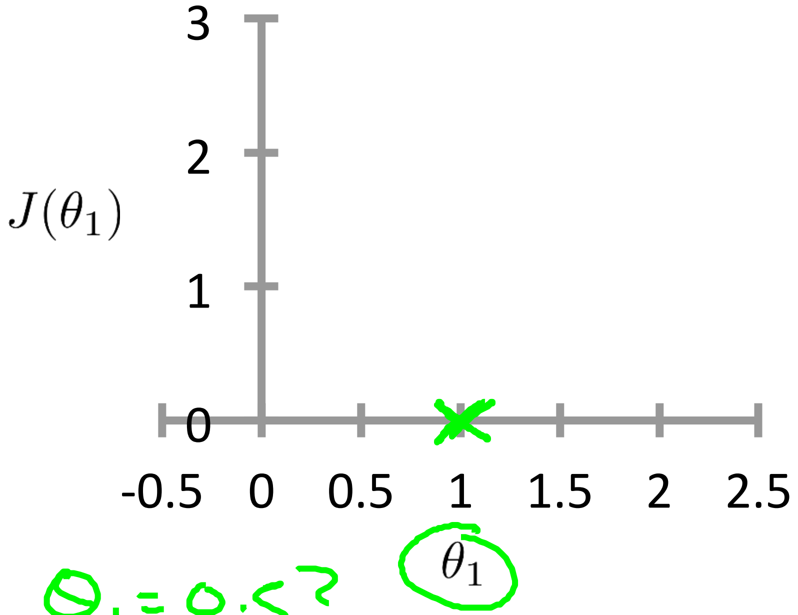
$\rightarrow h_\theta(x)$

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



$\rightarrow J(\theta_1)$

(function of the parameter θ_1)



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

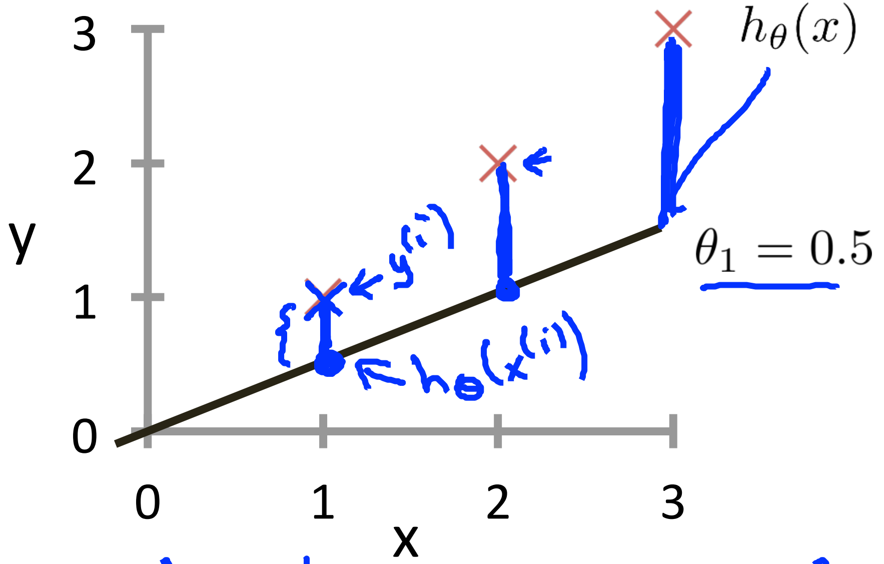
$$= \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 x^{(i)} - y^{(i)}\right)^2 = \frac{1}{2m}\left(0^2 + 0^2 + 0^2\right) = 0^2$$

$\theta_1 = 0.5?$

$J(1) = 0$

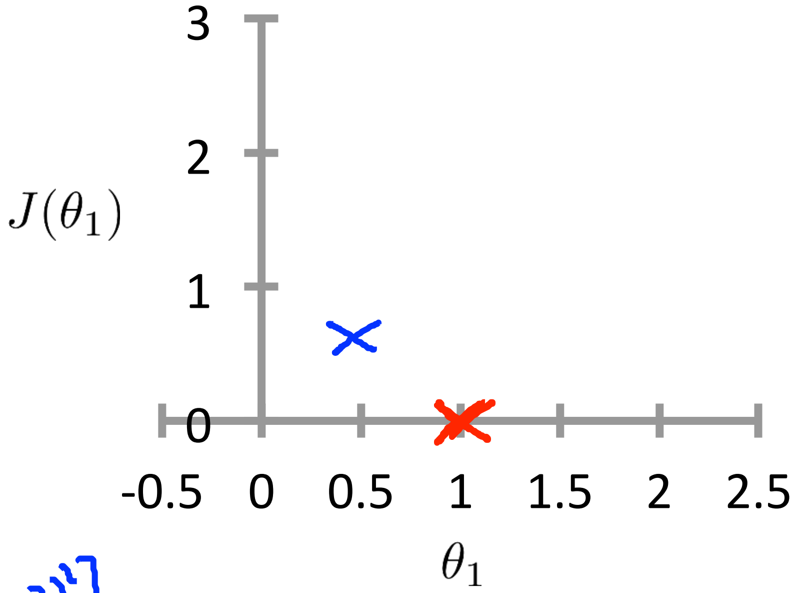
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



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

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

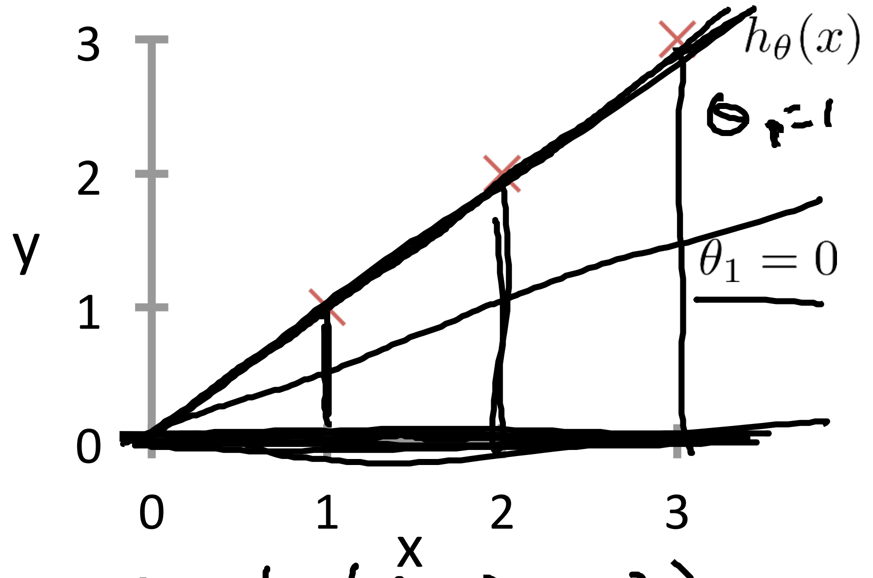
$\theta_1 = 0?$

$J(0) = ?$

Andrew Ng

$h_\theta(x)$

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

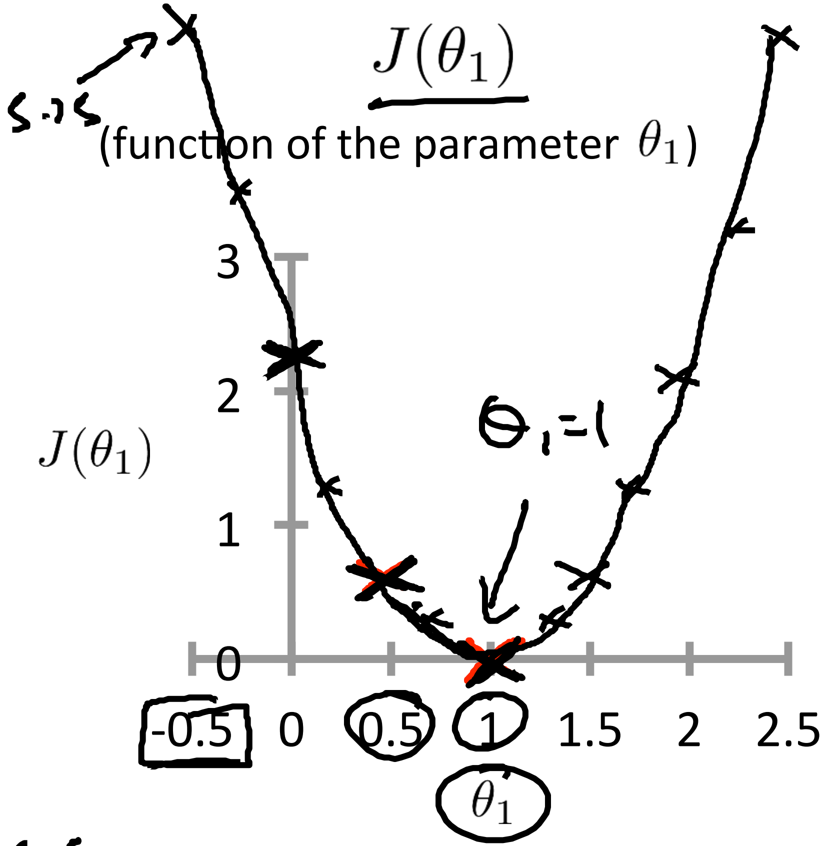


$J(0) = \frac{1}{2m}(1^2+2^2+3^2)$

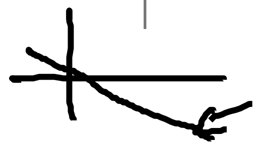
$= \frac{1}{6} \cdot 14 \approx 2.3$

$J(\theta_1)$

(function of the parameter θ_1)

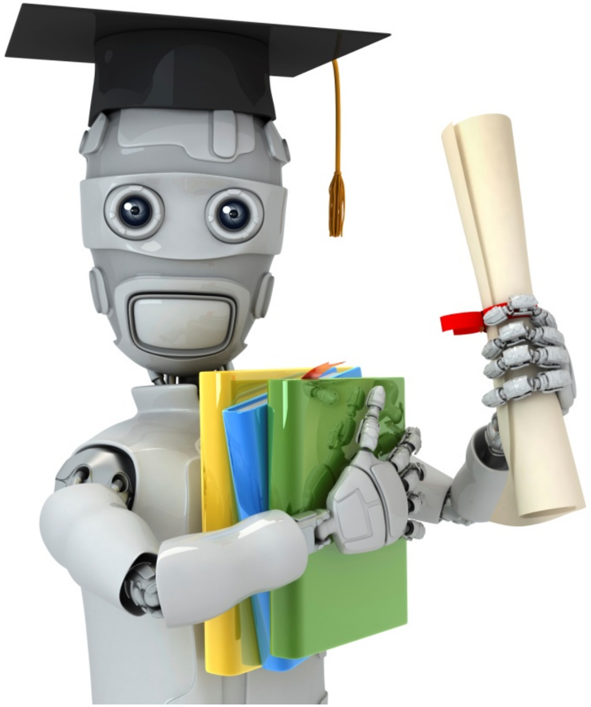


h(x) = -0.5x



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

Andrew Ng



Machine Learning

Linear regression with one variable

Cost function intuition II

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

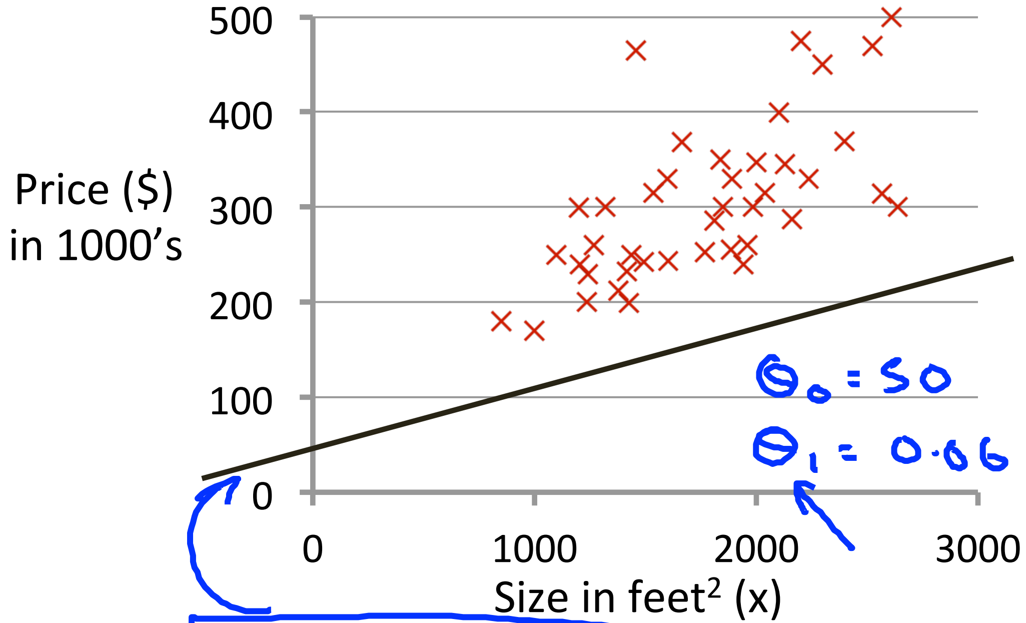
Parameters: $\underline{\theta_0, \theta_1}$

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

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

$h_\theta(x)$

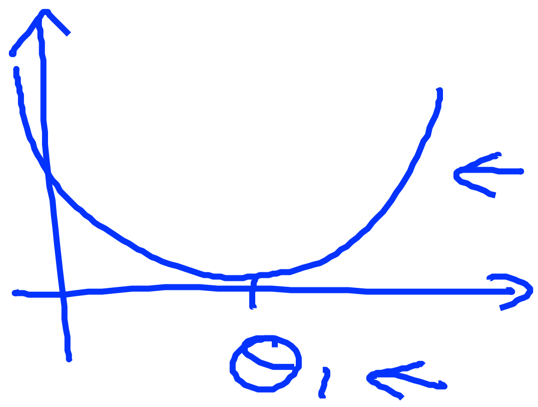
(for fixed θ_0, θ_1, this is a function of x)



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

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

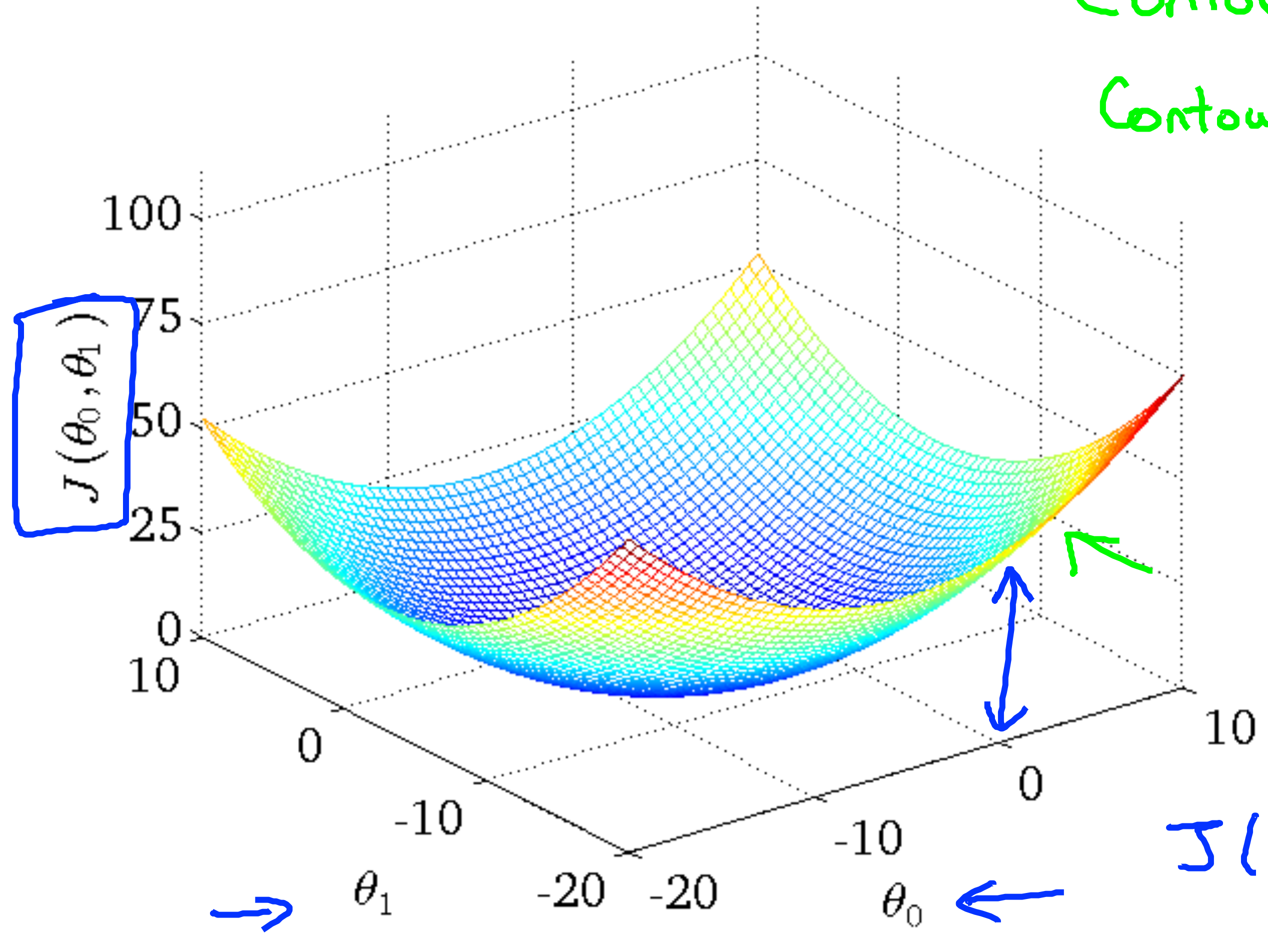


$$\boxed{\theta_0, \theta_1}$$

Andrew Ng

Contour plots

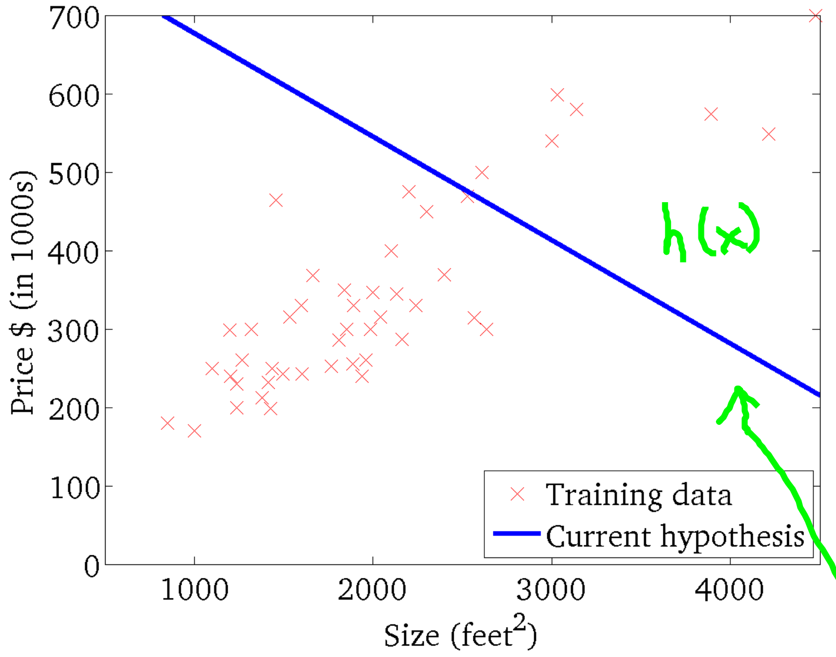
Contour figures.



$J(\theta_0, \theta_1)$

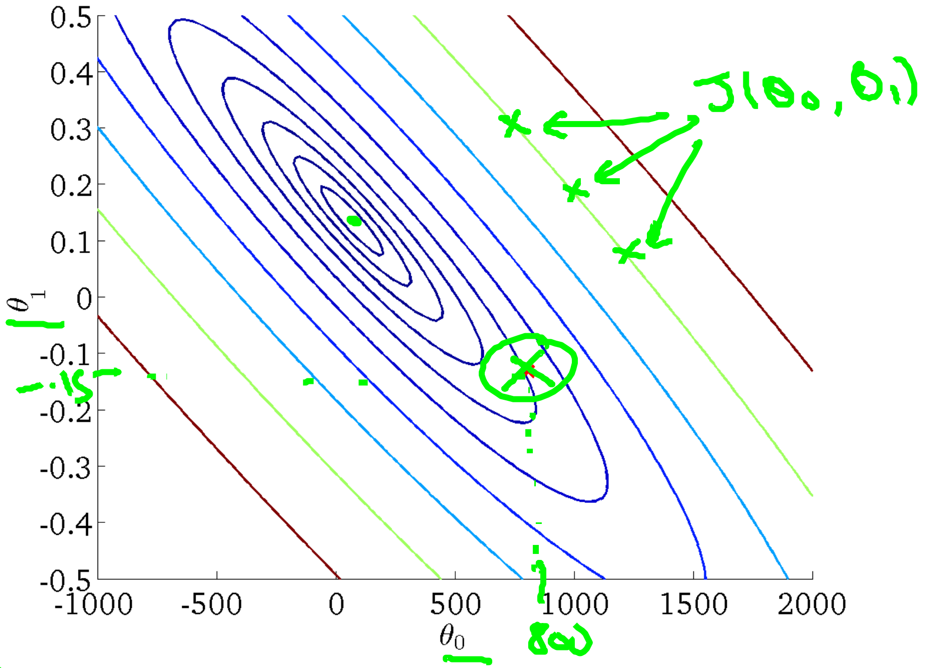
$h_\theta(x)$

(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)



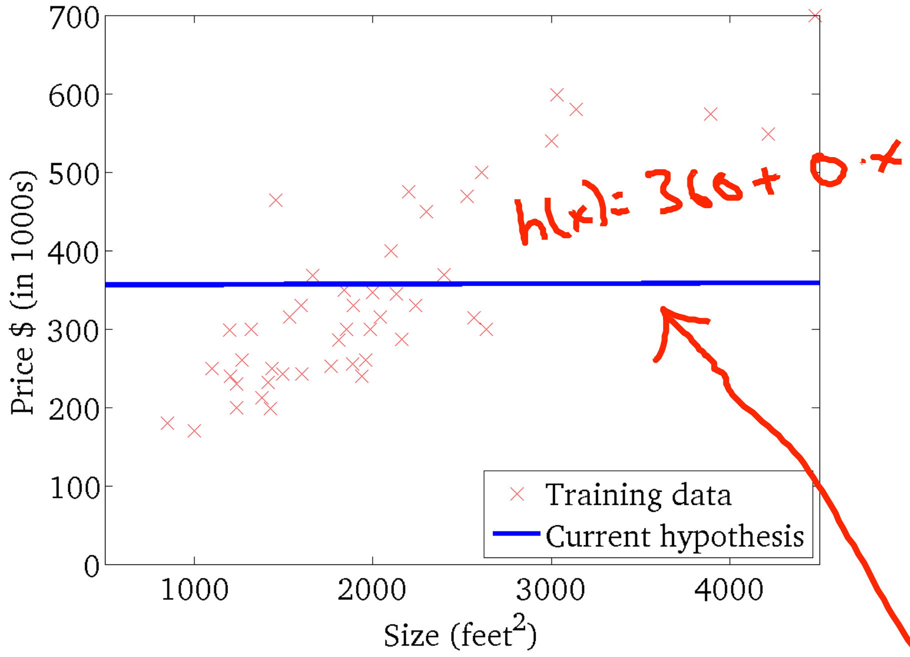
$\underline{\theta_0, \theta_1}$

This is a Contow plot. prettu cool hah

Andrew Ng

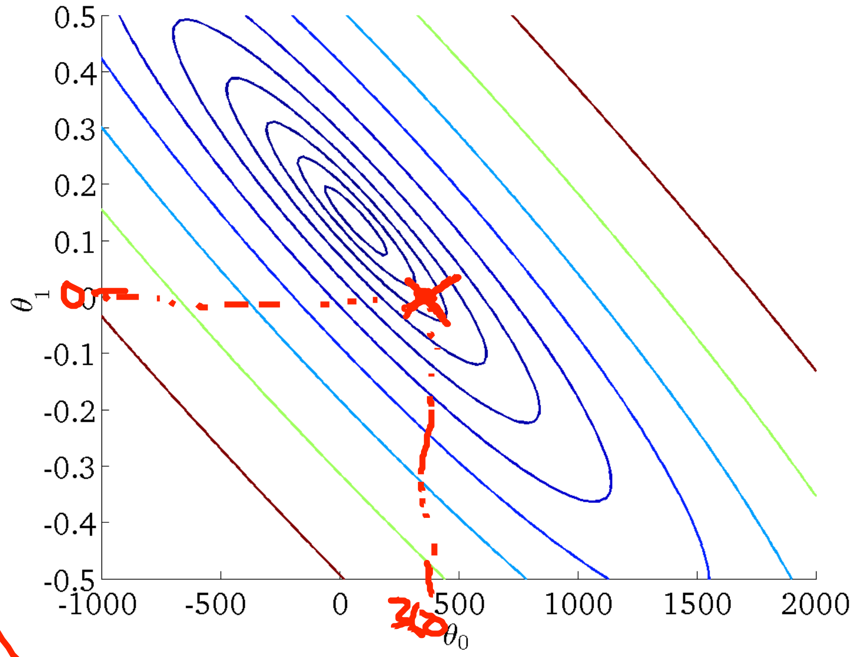
$h_\theta(x)$

(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)



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

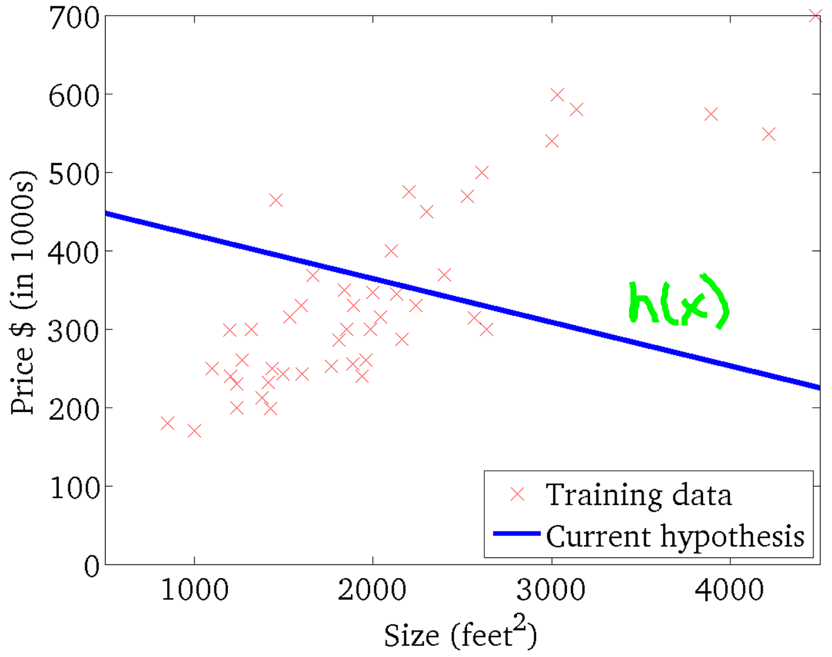
Andrew Ng

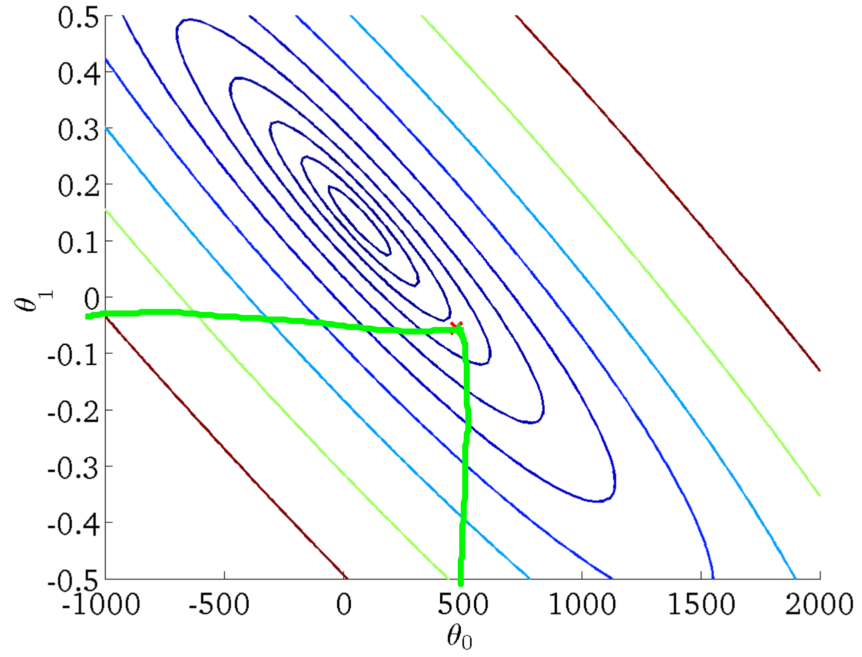
$$h_\theta(x)$$

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

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

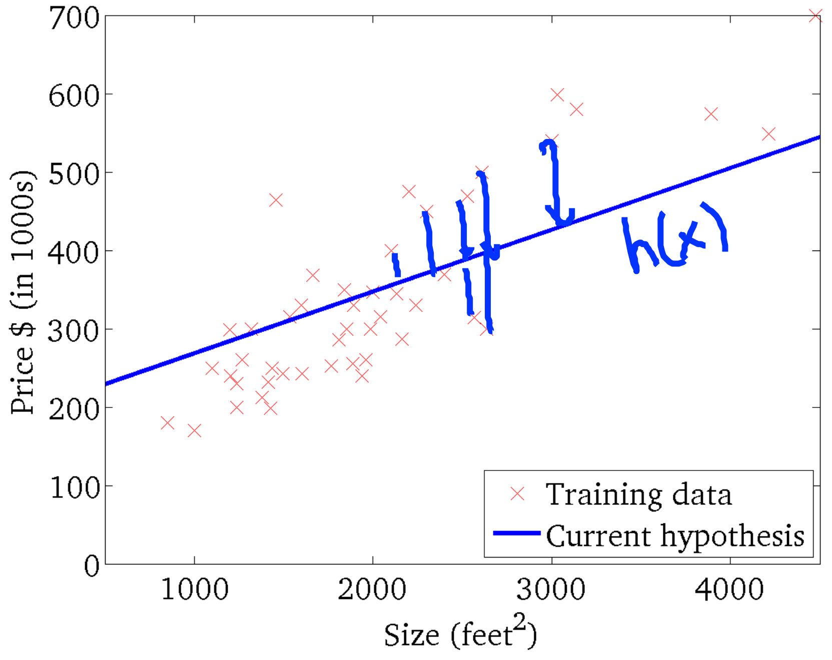
(function of the parameters θ_0, θ_1)





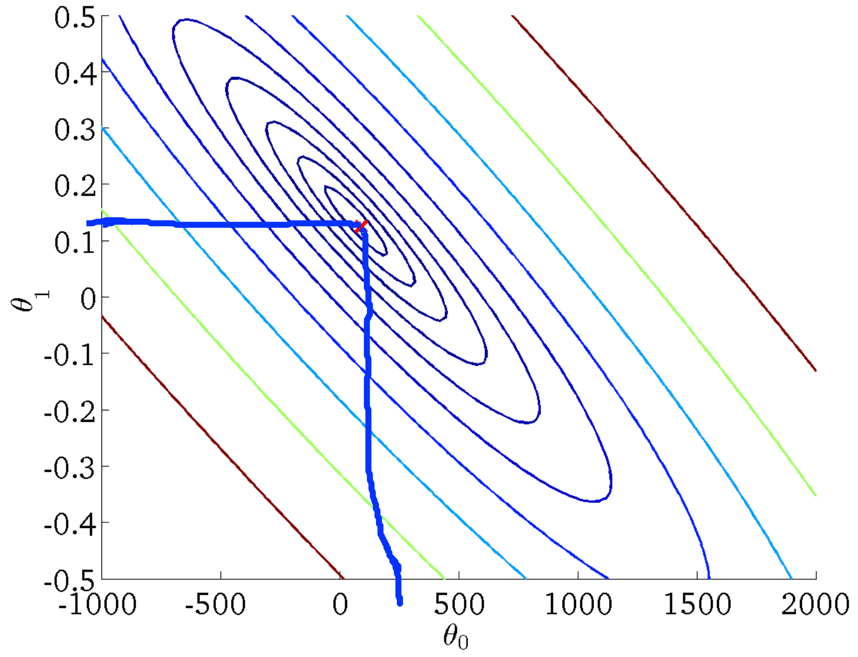
$$h_\theta(x)$$

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

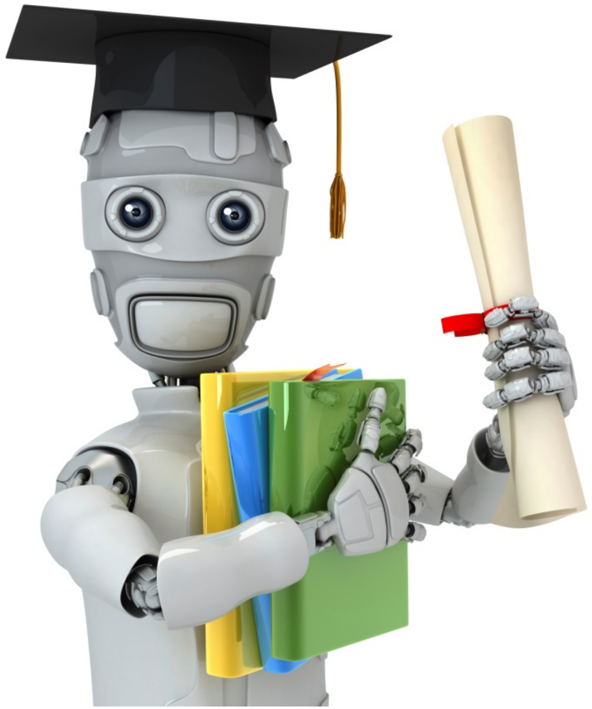


$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



Andrew Ng



Machine Learning

Linear regression with one variable

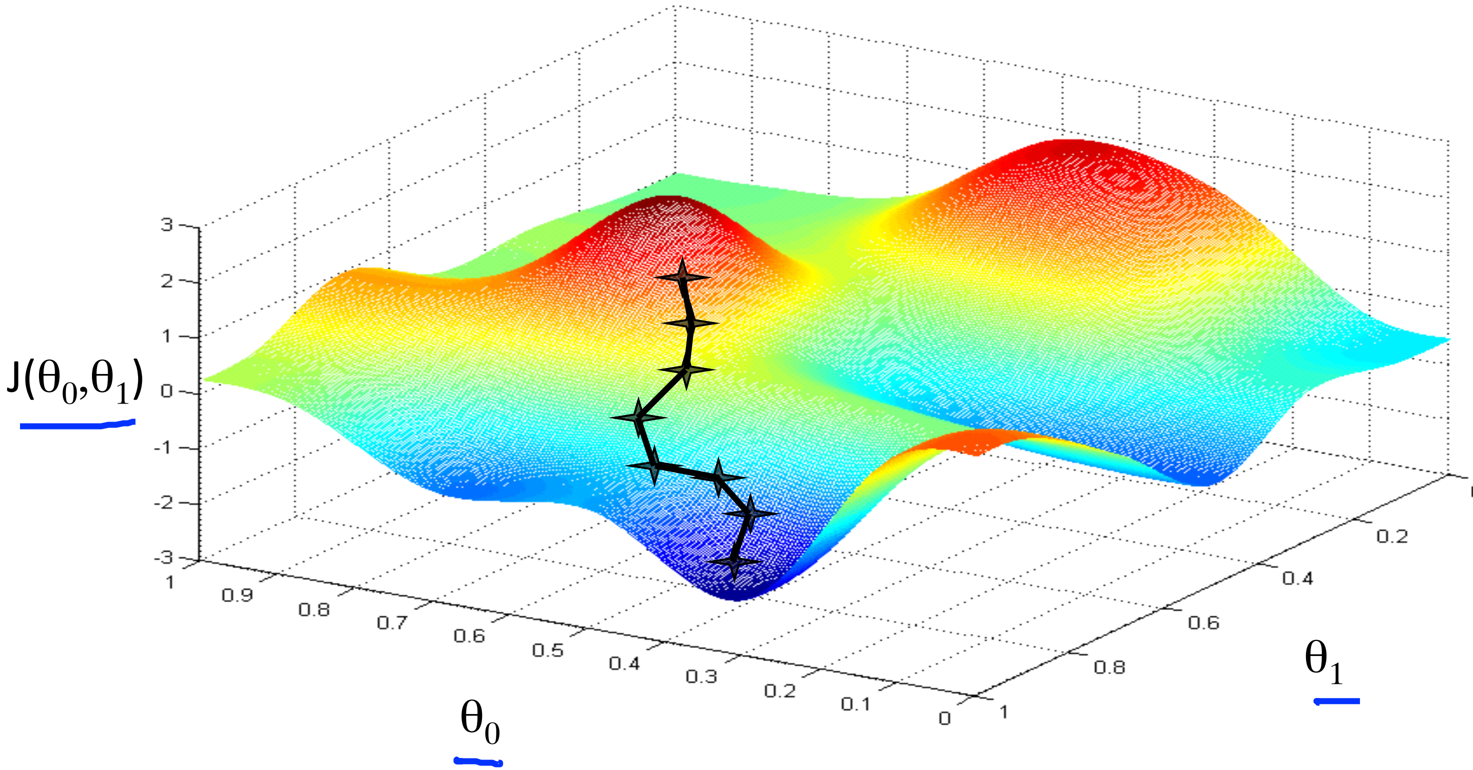
Gradient descent

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

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

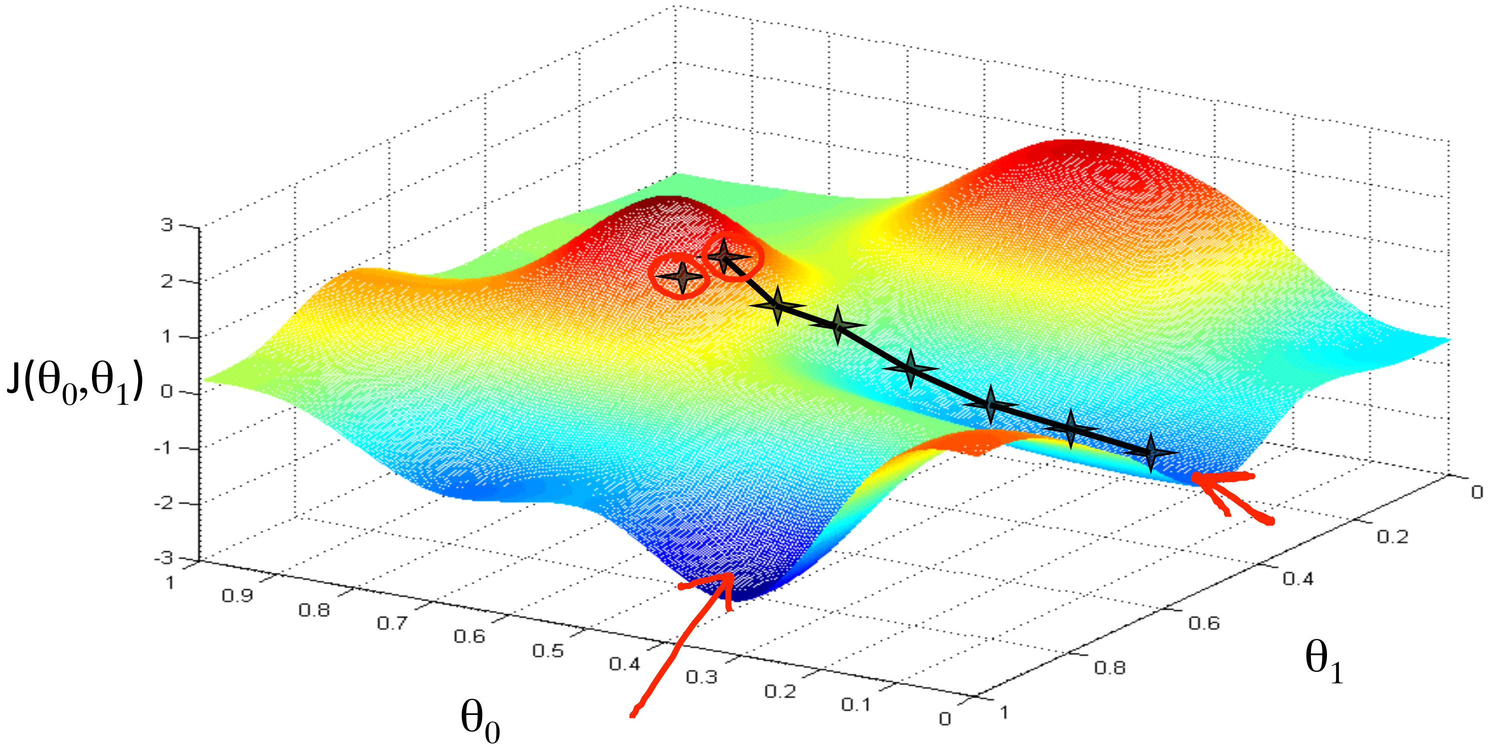
Outline:

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



$J(\theta_0,\theta_1)$
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



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

Gradient descent algorithm

Assignment

$a := b$

$a := a + 1$

Truth assertion

$a = b$

$\cancel{a = a + 1}$

θ_0, θ_1

repeat until convergence {

$$\underline{\theta_j} := \theta_j - \underbrace{\alpha}_{\text{learning rate}} \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \underline{(\text{for } j = 0 \text{ and } j = 1)}$$

}

we will talk about this derivativ term later.

: how big the step we are taking down the hill

Simultaneously update θ_0 and θ_1

Correct: Simultaneous update

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

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

$\theta_0 := \text{temp0}$

$\theta_1 := \text{temp1}$

Incorrect:

(Not a simultaneous update)

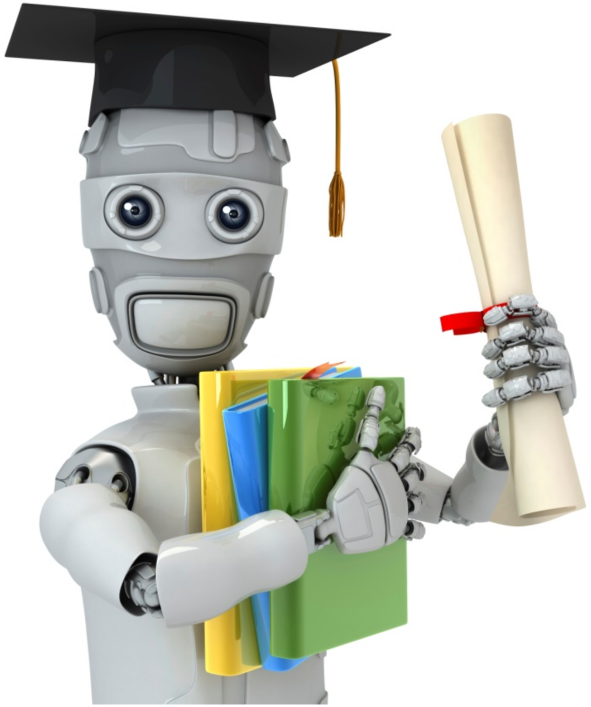
$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$\theta_0 := \text{temp0}$

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

$\theta_1 := \text{temp1}$

Andrew Ng



Machine Learning

Linear regression with one variable

Gradient descent intuition

Gradient descent algorithm

repeat until convergence {

$$\rightarrow \underline{\theta_j} := \underline{\theta_j} - \underset{\text{learning rate}}{\underbrace{\alpha}} \underset{\text{derivative}}{\underbrace{\boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}}}$$

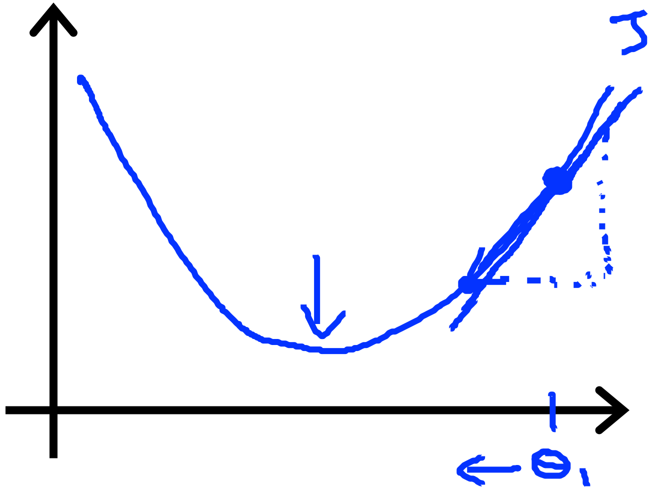
}

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

Example of where we want to:

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

Andrew Ng

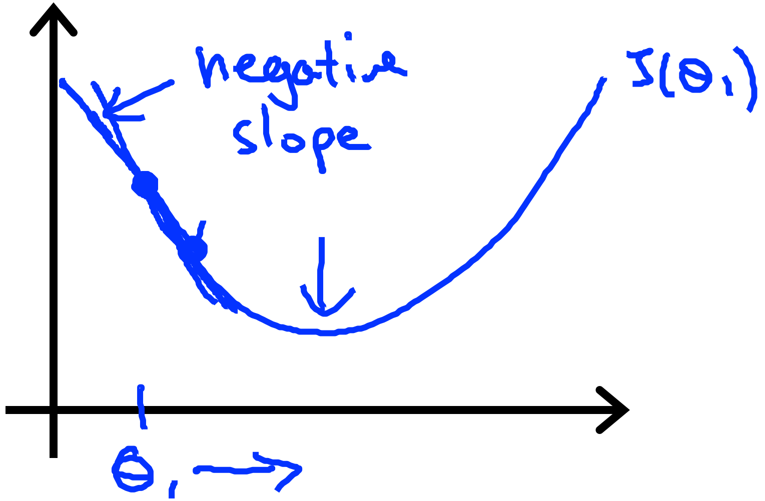


$(\theta_1 \in \mathbb{R})$

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

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

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

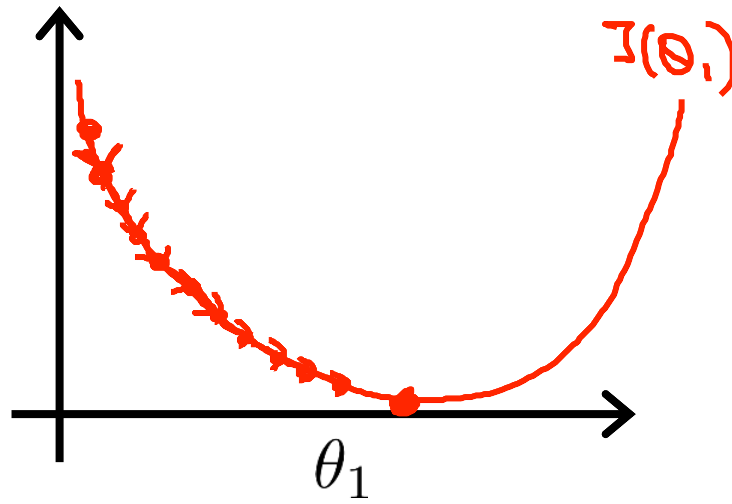


$$\underbrace{\frac{d}{d\theta_1} J(\theta_1)}_{\leq 0}$$

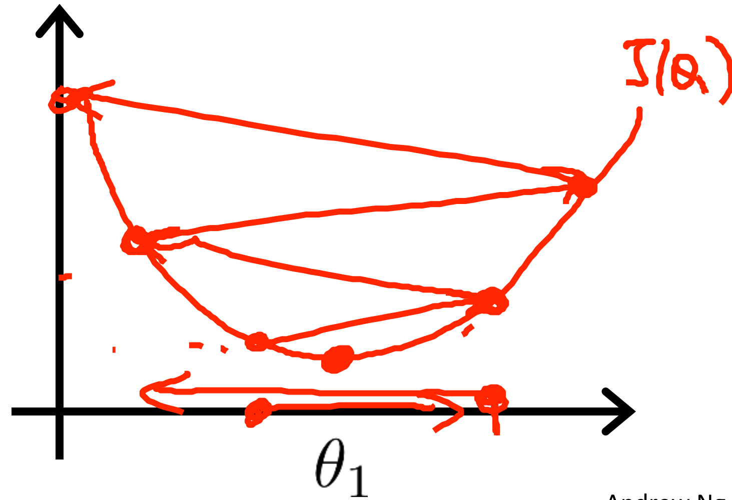
$$\theta_1 := \underline{\theta_1} \underset{\uparrow}{-} \alpha \, \underset{\uparrow}{(}\text{negative number})$$

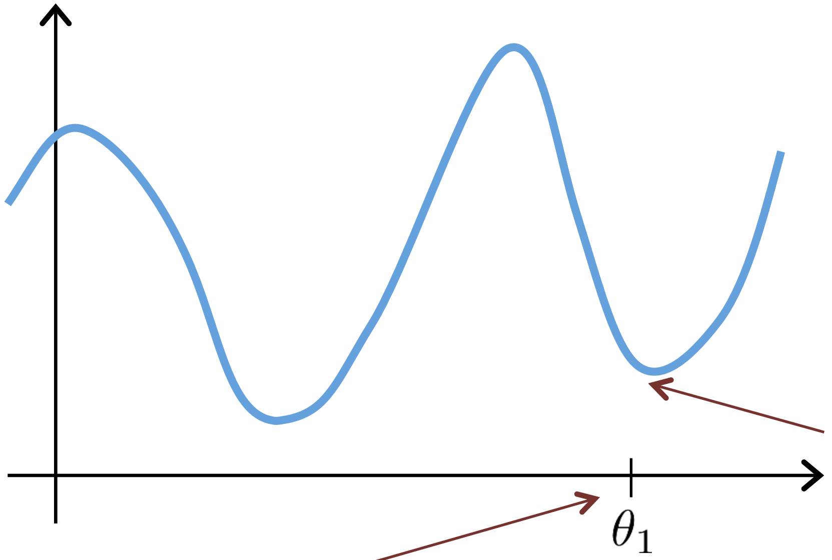
$$\boxed{\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)}$$

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.





θ_1 at local optima

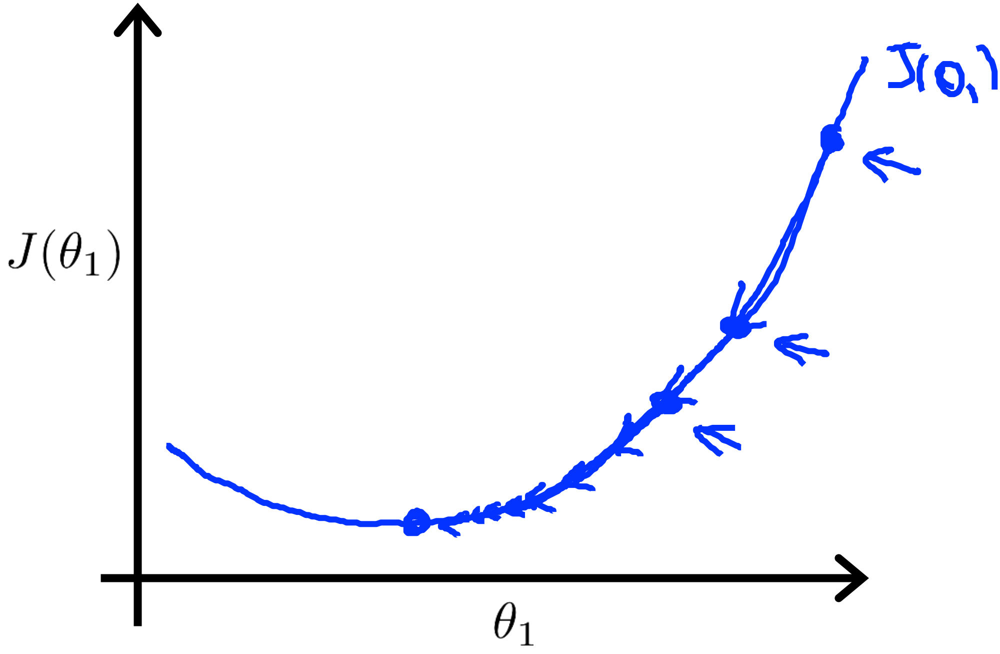
Current value of θ_1

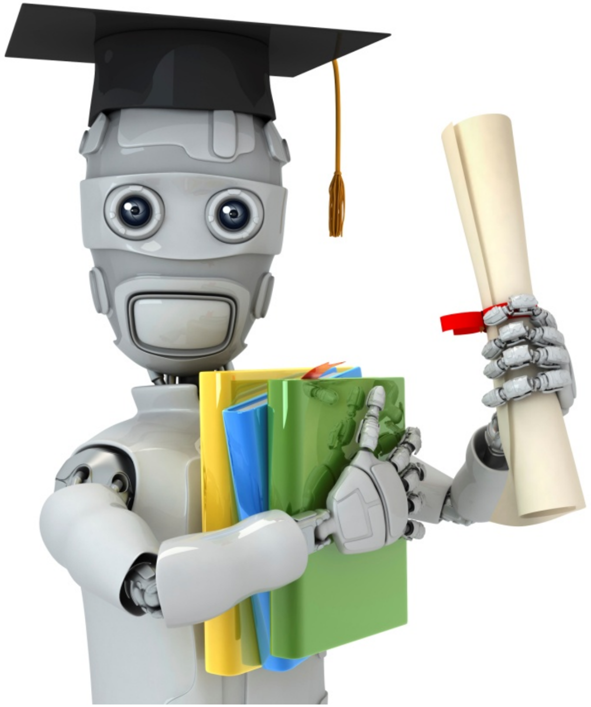
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

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

```
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

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

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

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

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

So this is the results of that partial derivativ. (not the derivation, just the results)

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

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

Andrew Ng

Gradient descent algorithm

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

repeat until convergence {

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

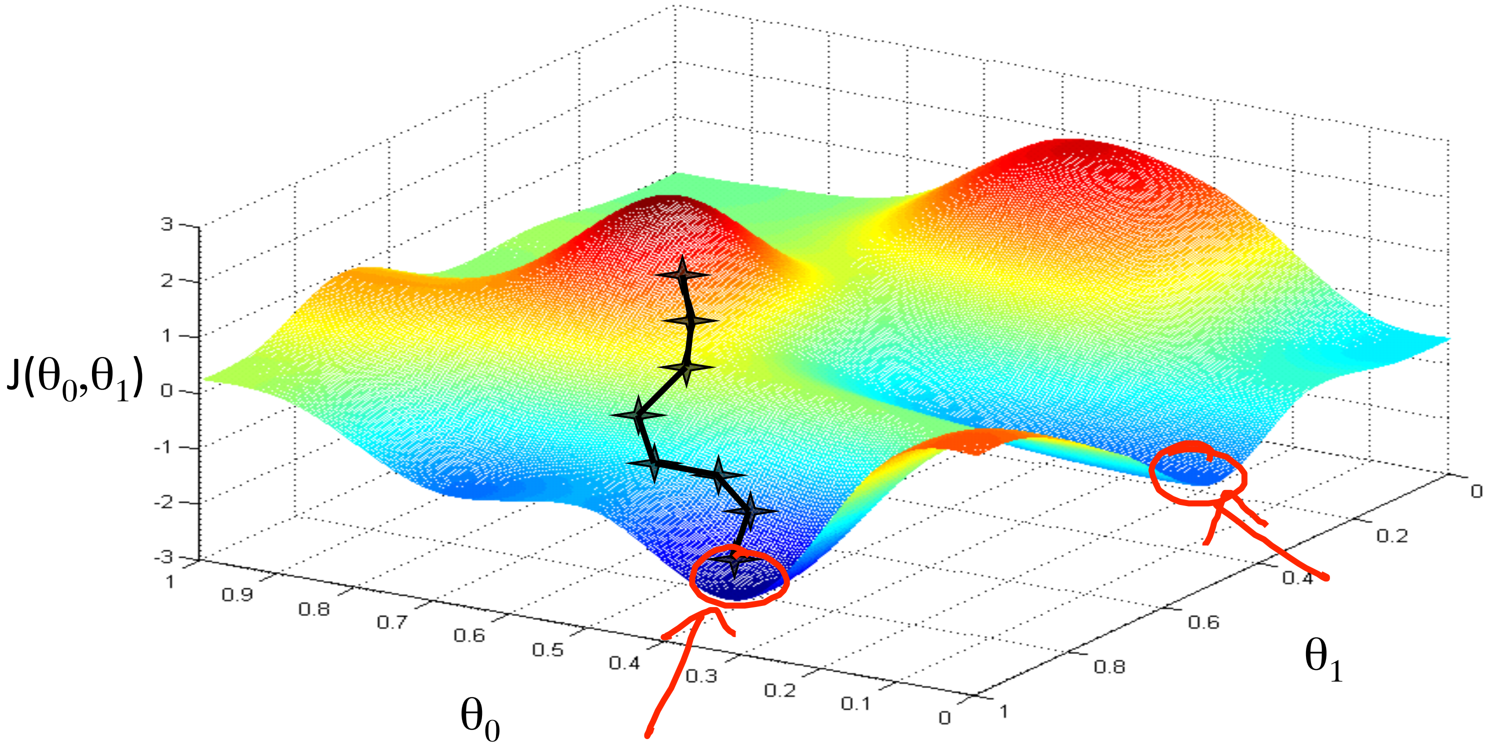
$$\theta_1 := \theta_1 - \alpha \boxed{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}}$$

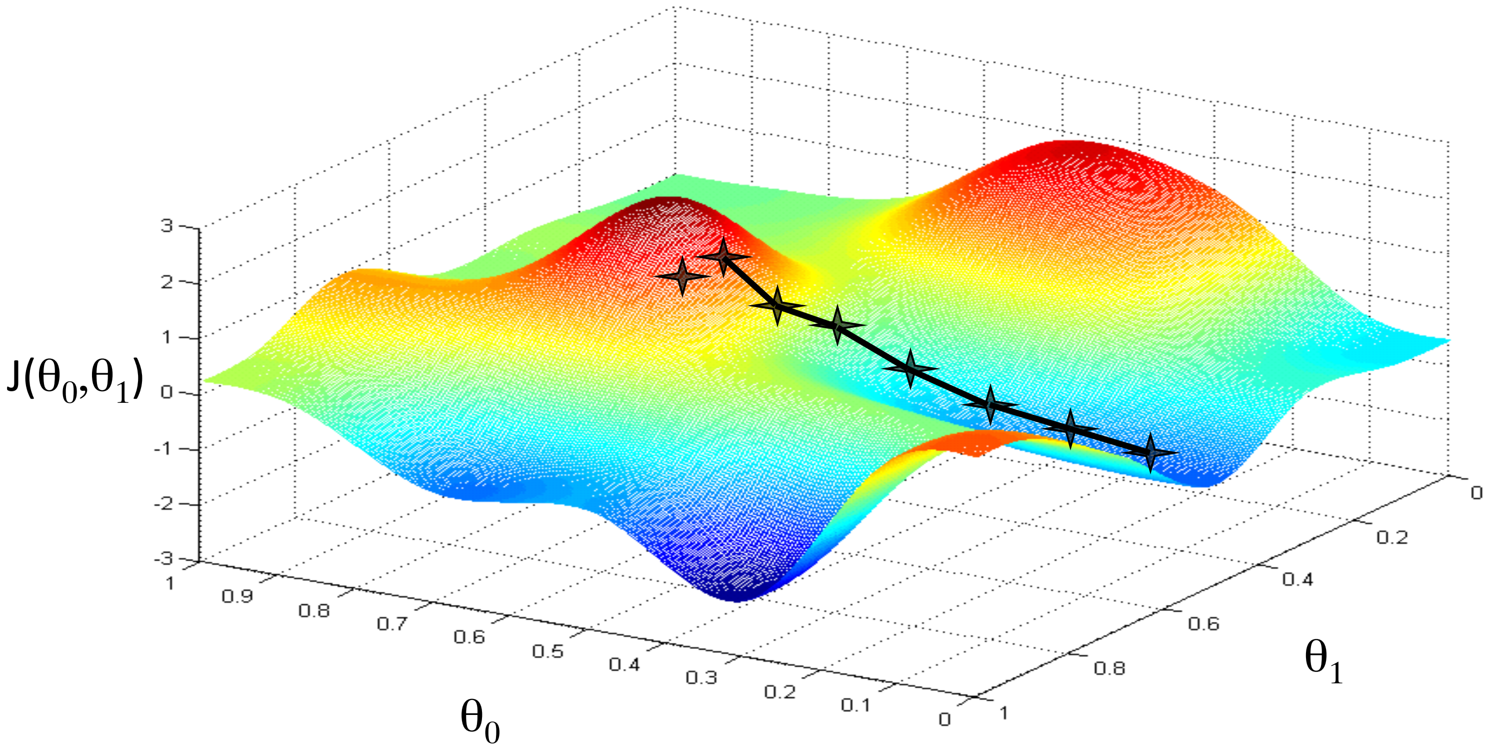
}

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

update θ_0 and θ_1 simultaneously

Its dependent on where you initialize it, it can end up in different local optimals



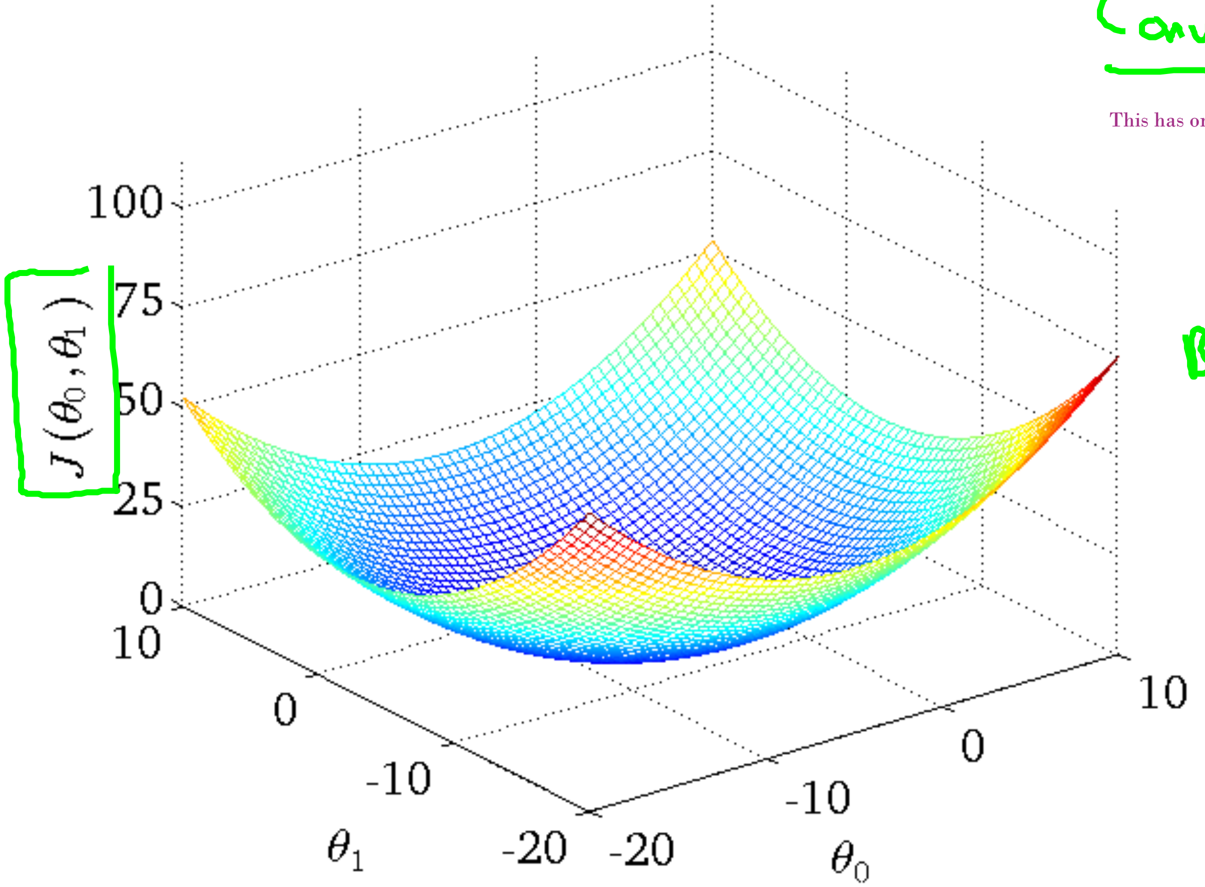


$J(\theta_0,\theta_1)$
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1

Cost function for linear regression is allways going to be a convex function(Bowl-shape fct) which has only one local optimal.

"Convex function"

This has only one local optimal



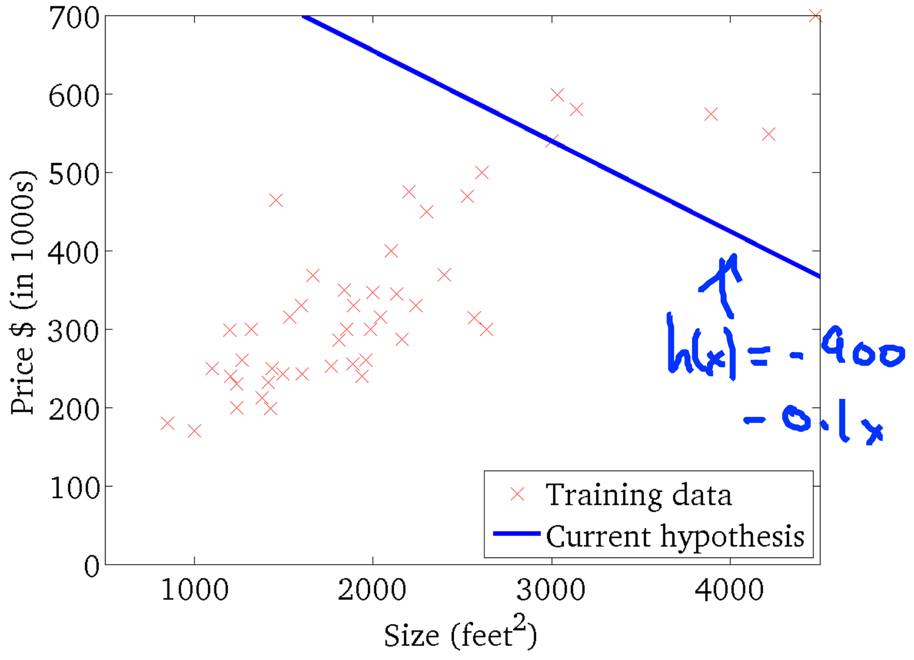
Bowl-shaped

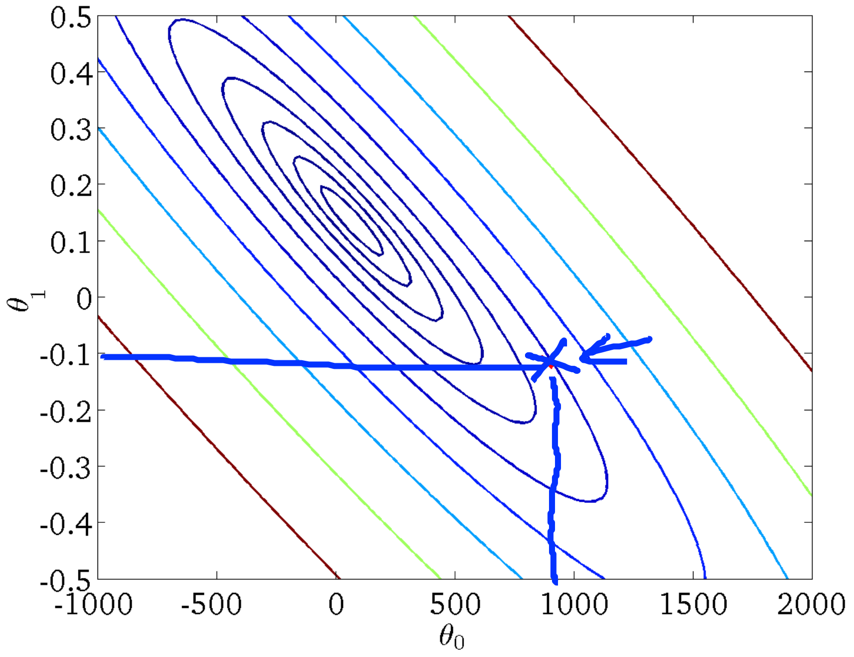
$\underline{h_\theta(x)}$

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

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

(function of the parameters θ_0, θ_1)

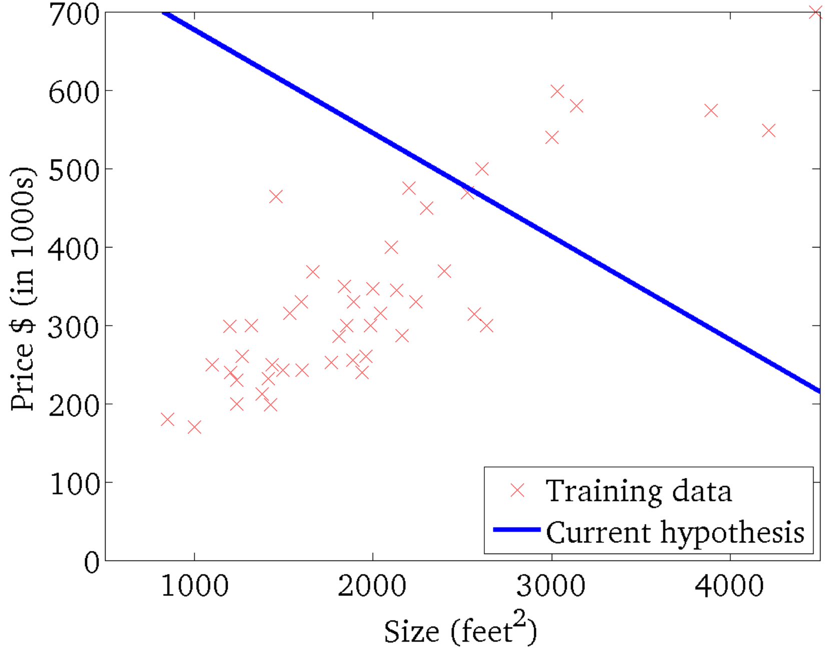




Andrew Ng

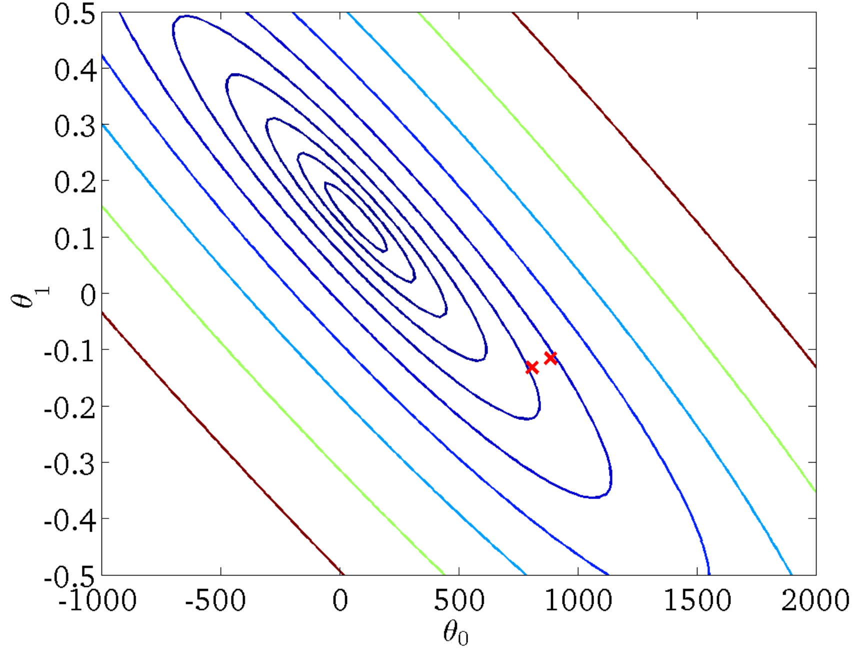
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

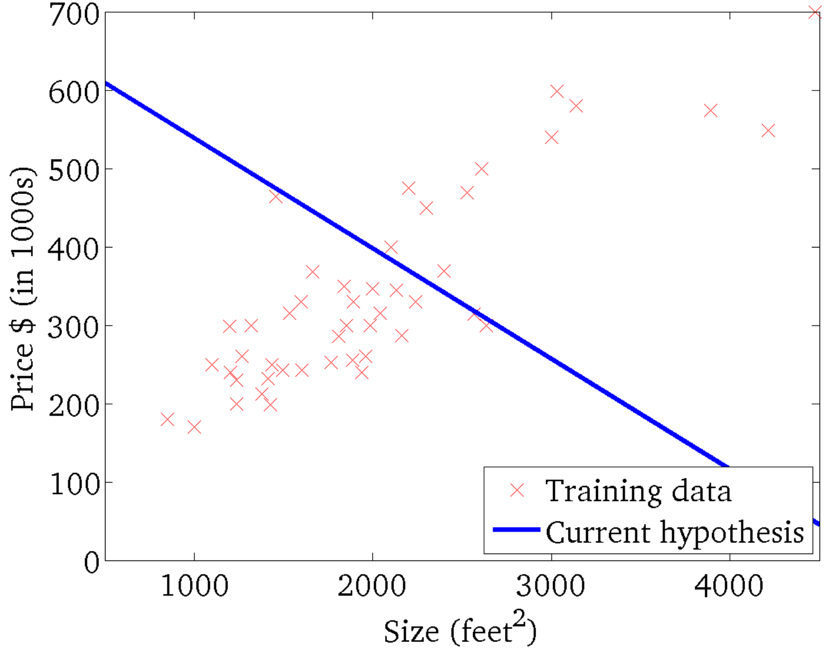
(function of the parameters θ_0, θ_1)



Andrew Ng

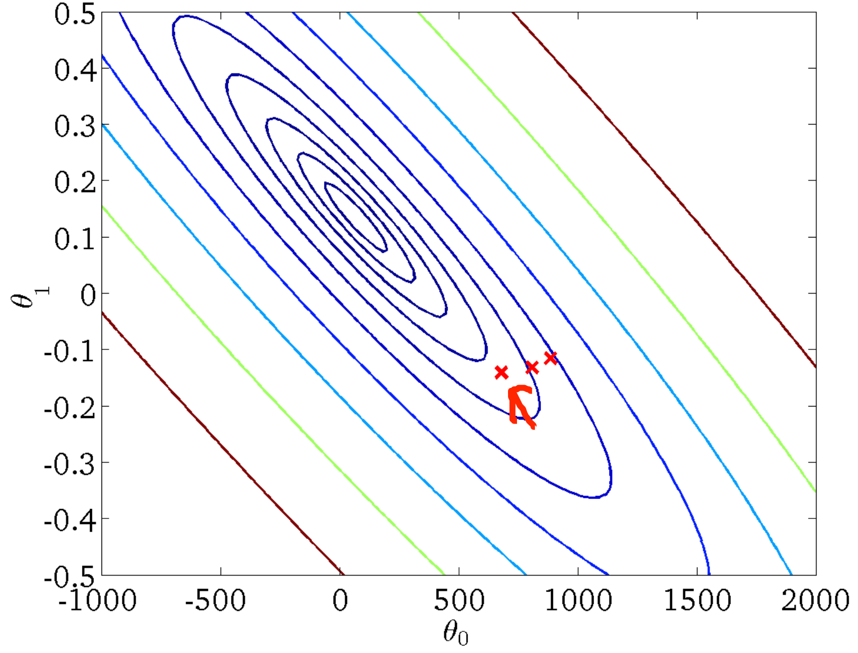
$h_\theta(x)$

(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)



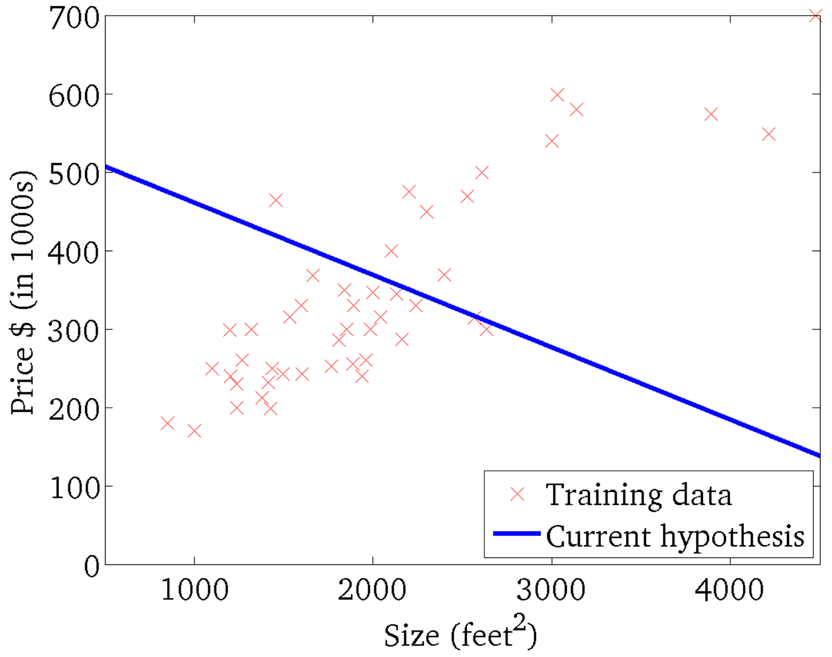
Andrew Ng

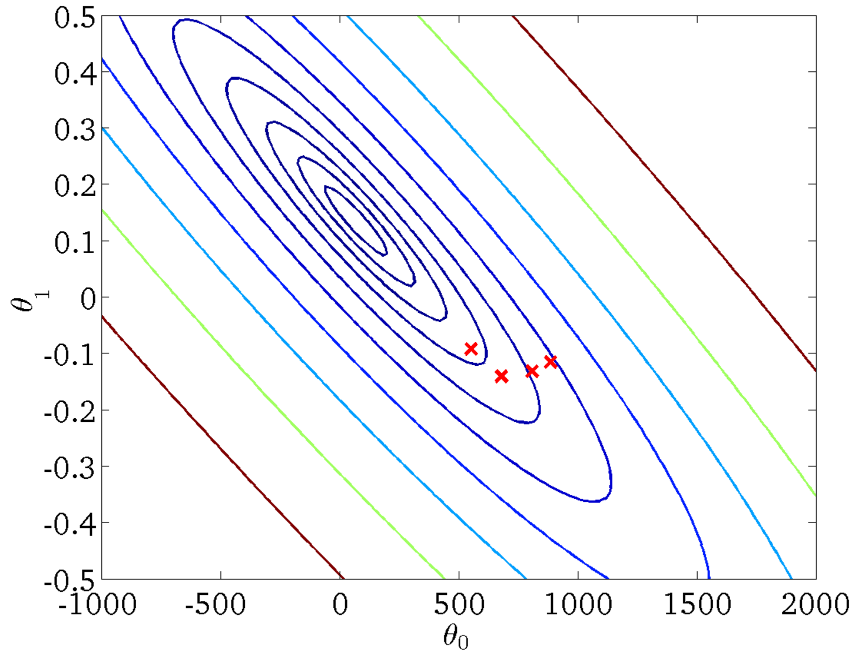
$$h_\theta(x)$$

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

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

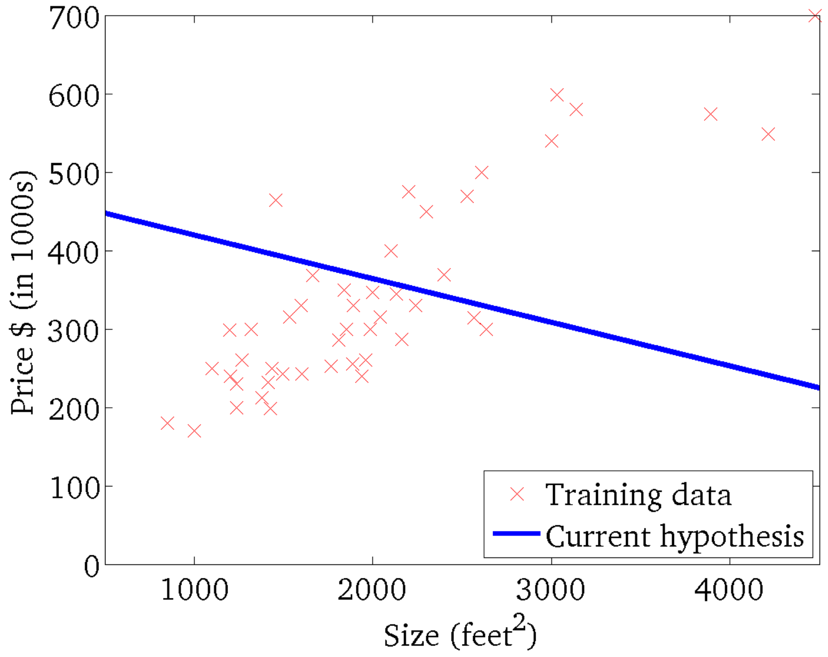
(function of the parameters θ_0, θ_1)





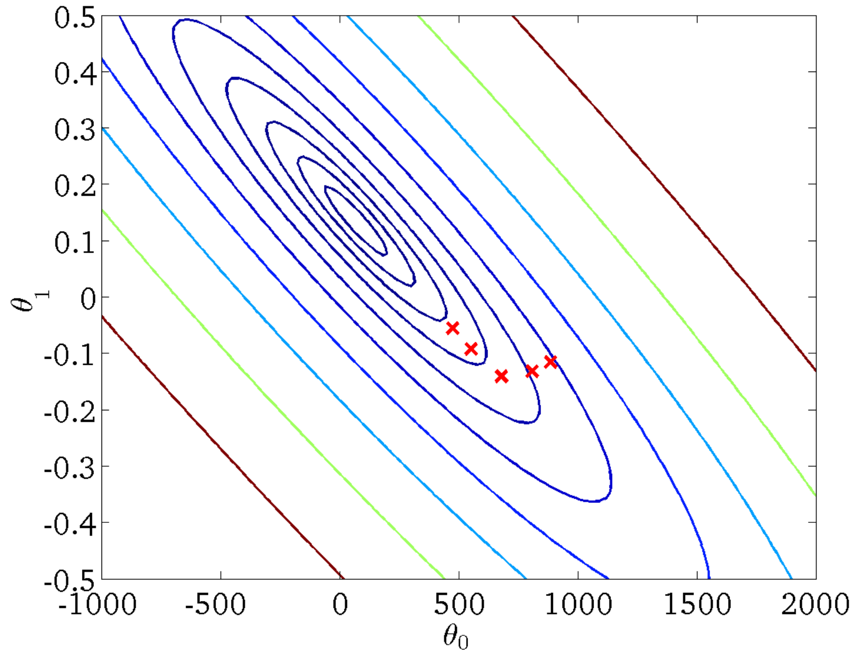
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

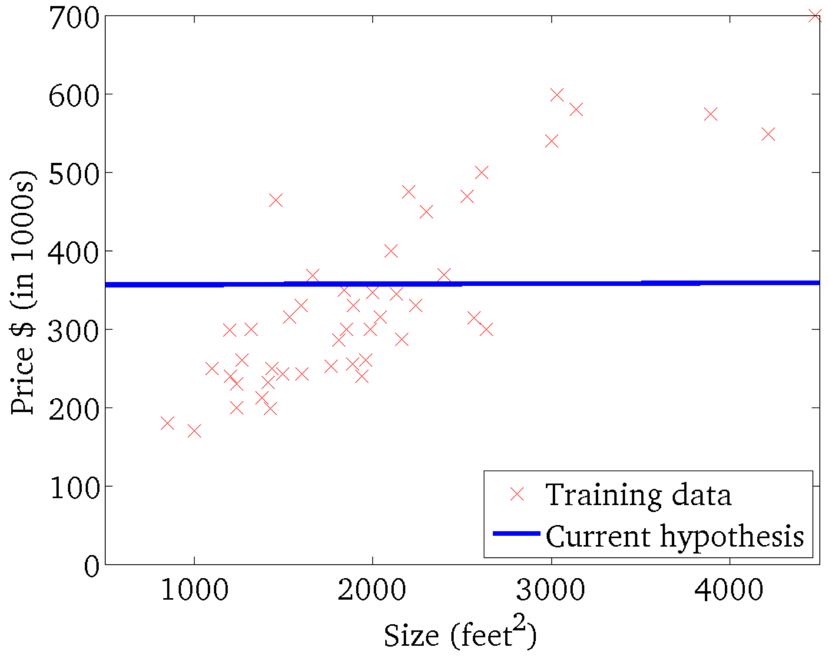
(function of the parameters θ_0, θ_1)



Andrew Ng

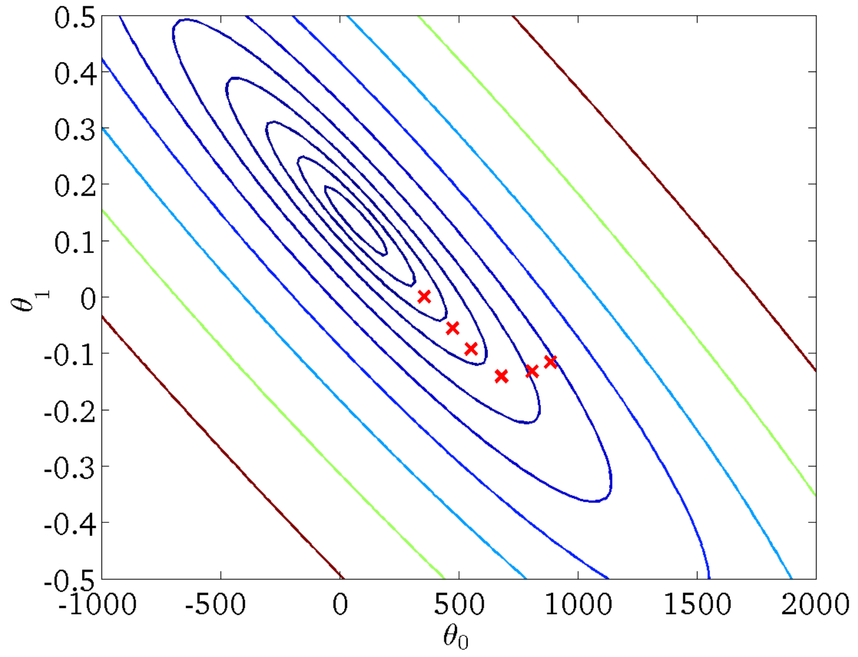
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

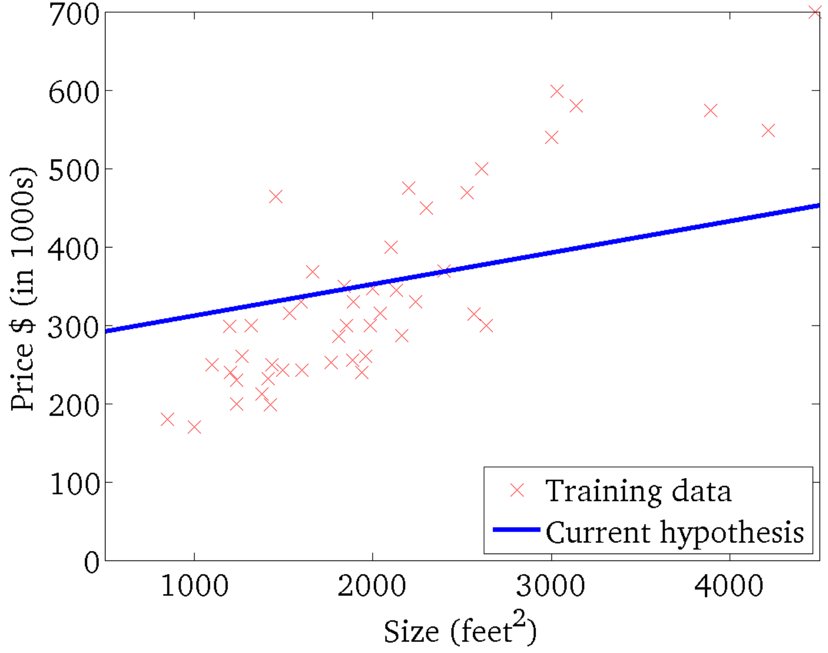
(function of the parameters θ_0, θ_1)



Andrew Ng

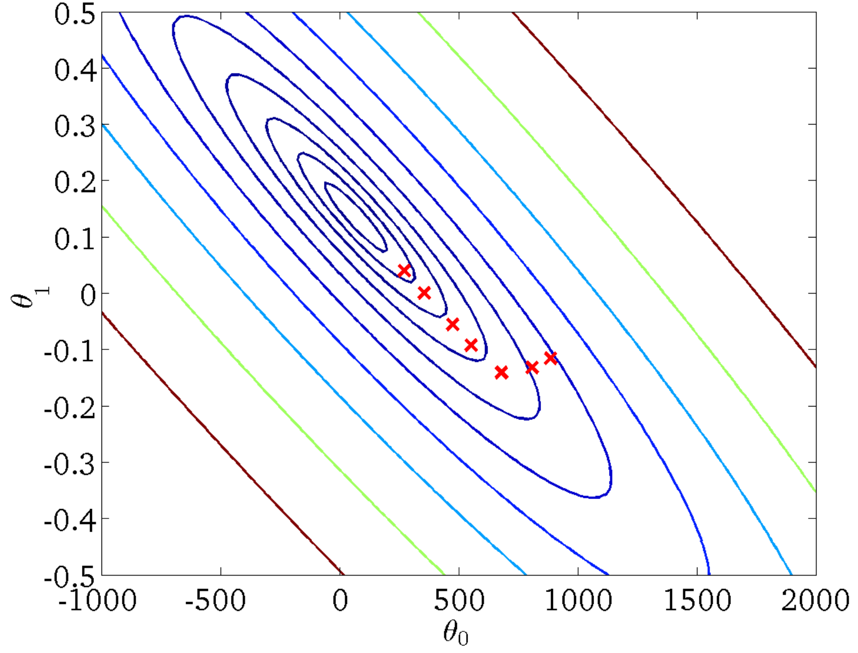
$$h_\theta(x)$$

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



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

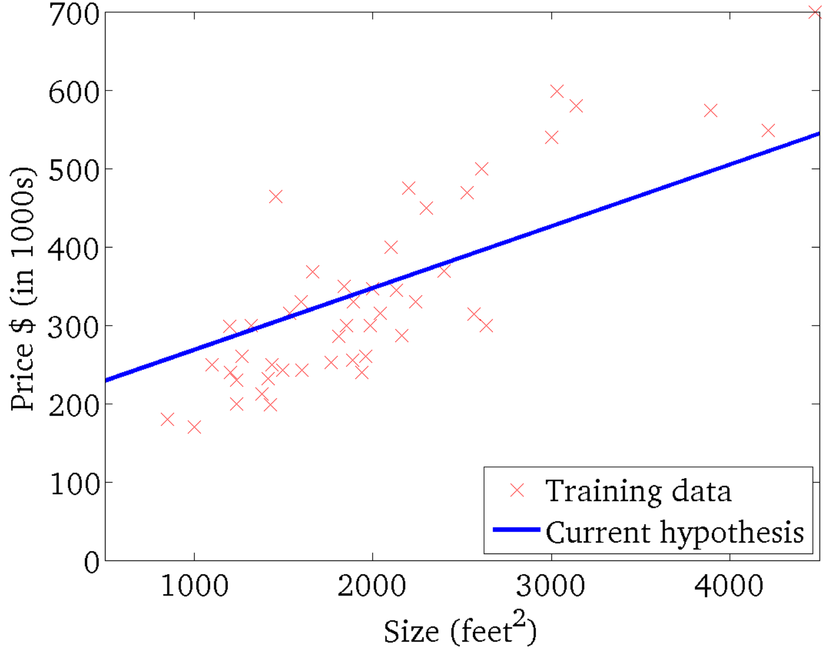
(function of the parameters θ_0, θ_1)



Andrew Ng

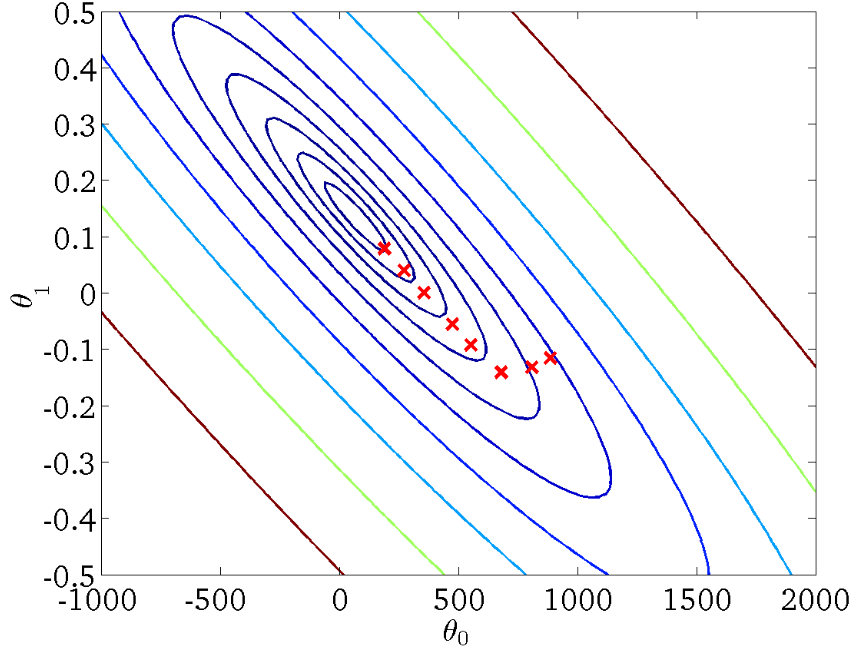
$h_\theta(x)$

(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)



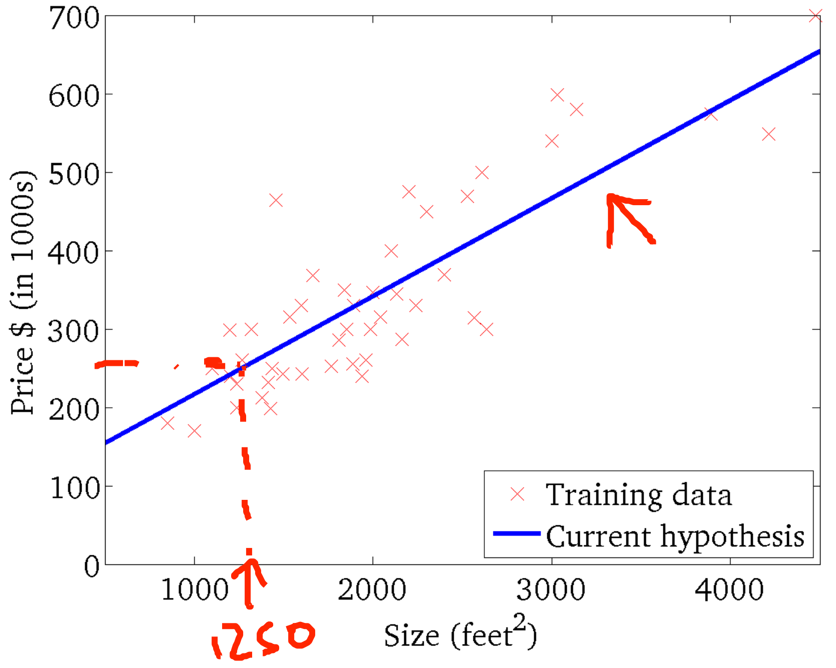
Andrew Ng

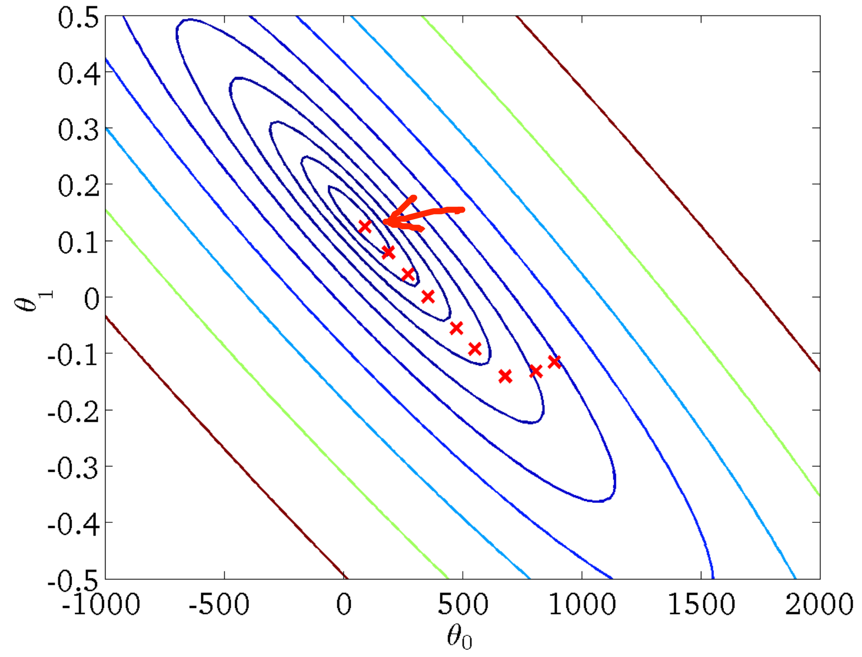
$h_\theta(x)$

(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)





So this is grading decent to fit a regression to the data.

Andrew Ng

The algorithm we just whent over is also called the Batch Gradient Descent

"Batch" Gradient Descent

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

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