Computer Vision (CS 419/619)
Linear Regression and Gradient Descent

Anup Kumar Gupta

Under supervision of
Dr. Puneet Gupta

Linear Regression

Supervised Learning

- Given the “right answer” for each example in the data.

Regression Problem

- Predict real-valued output

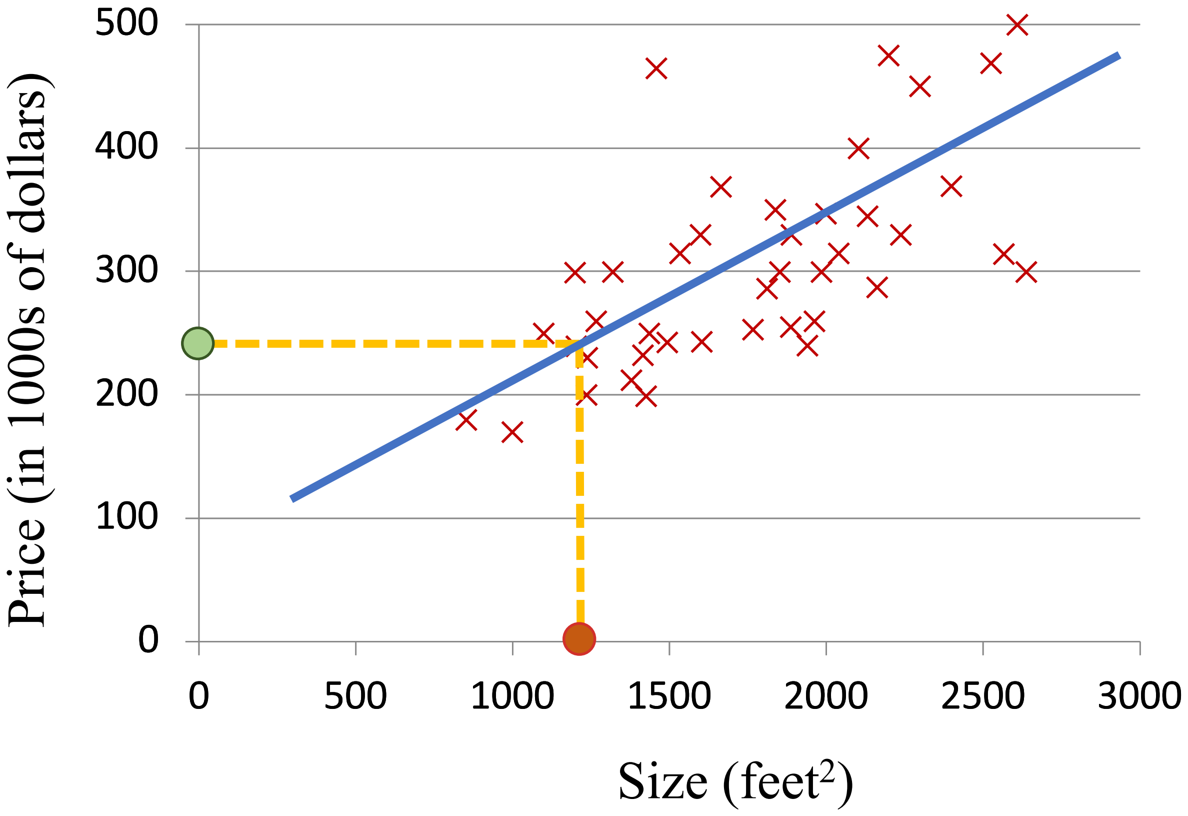


Fig. 1: Housing prices vs Size

Training set

Notations

- $m =$ number of training examples
- x = "input" variable / features
- y = "output" variable / "target" variable

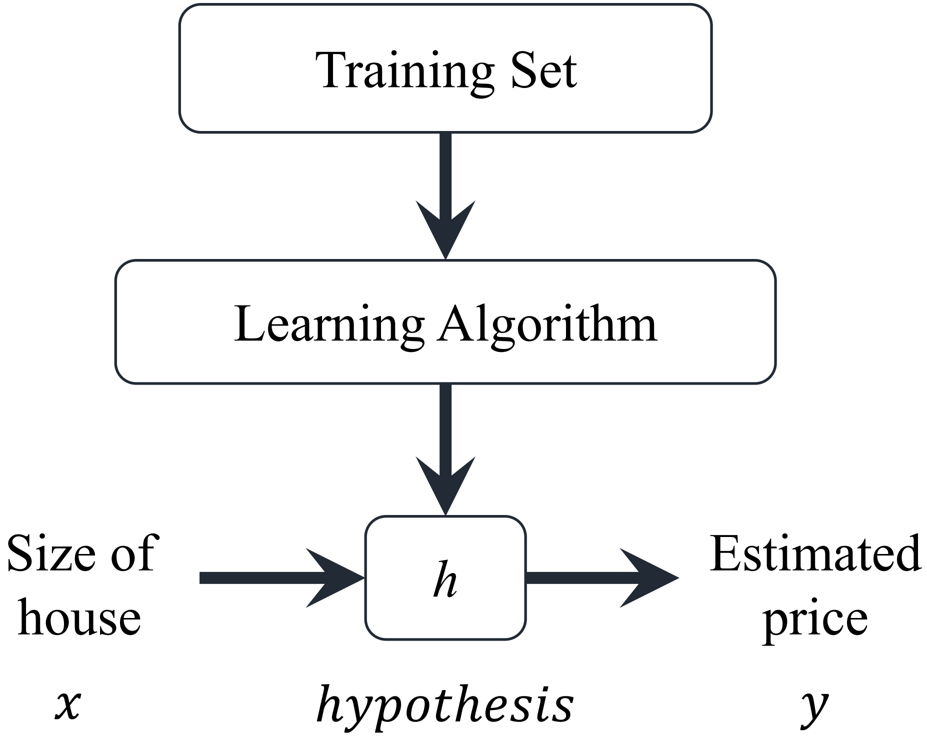
- We will use (x, y) to denote one training example.
- We will use $\left(x^{(i)}, y^{(i)}\right)$ to denote the i^{th} training example.
- For example: $x^{(1)} = 2104$ and $y^{(1)} = 460$

- Note: superscript denotes index, and not power.

Size in feet² (x)	**Price in 1000 dollars (y)**
2104	460
1416	232
1534	315
852	178
…	…

Tab. 1: Training Samples

Training Procedure



h maps the input x to the output y

How do we represent *h* ?

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

As a shorthand we often remove the θ notation:

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

y is a linear function of x. It resembles:

$y = c + m \cdot x$

Equation of straight line.

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

Training procedure – How to compute the θs

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

- θ_1 and θ_2 or in general θ_i = parameters
- How to choose or find these θ_is ?

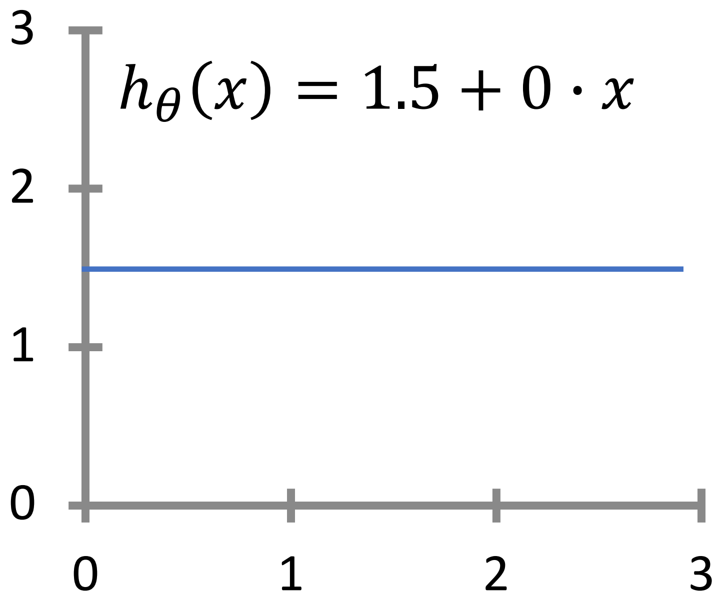
- If we have these θ_is and a new data point x is given to us,
- then we can calculate y using these parameters.

Size in feet² (x)	**Price in 1000 dollars (y)**
2104	460
1416	232
1534	315
852	178
…	…

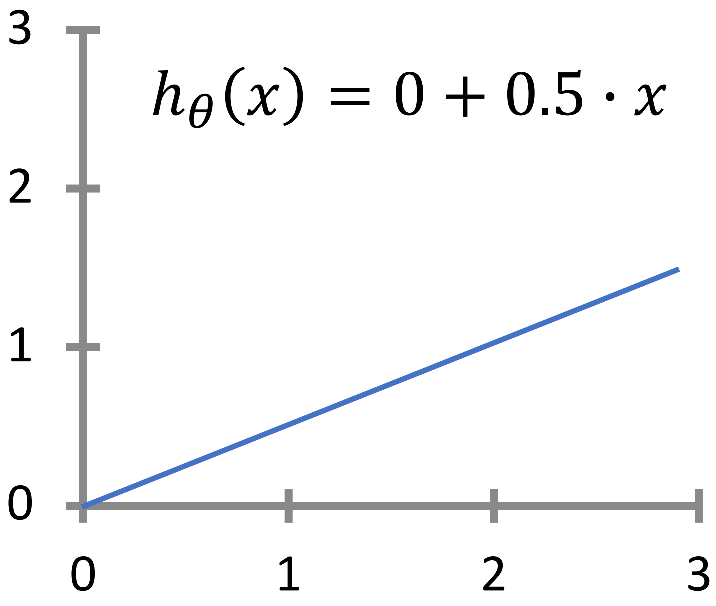
Tab. 1: Training Samples

Variations of h_θ with variations in θ_i

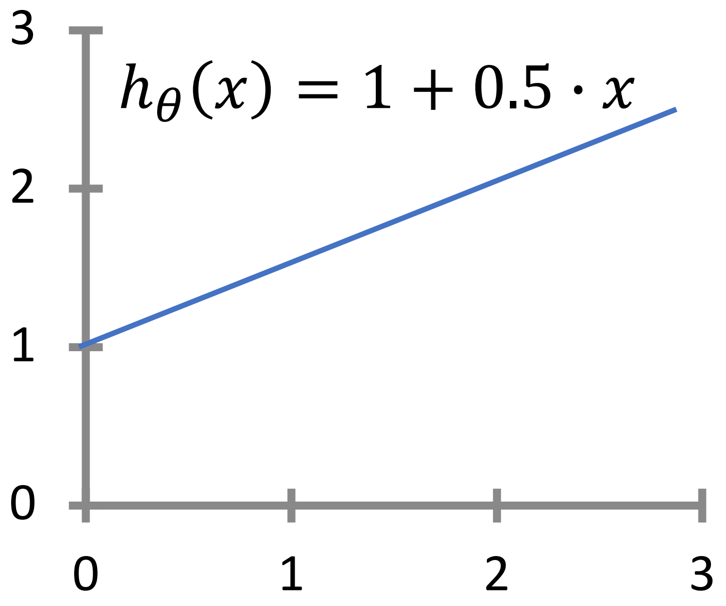
$$h_\theta(x) = \theta_0 + \theta_1 \cdot x$$



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

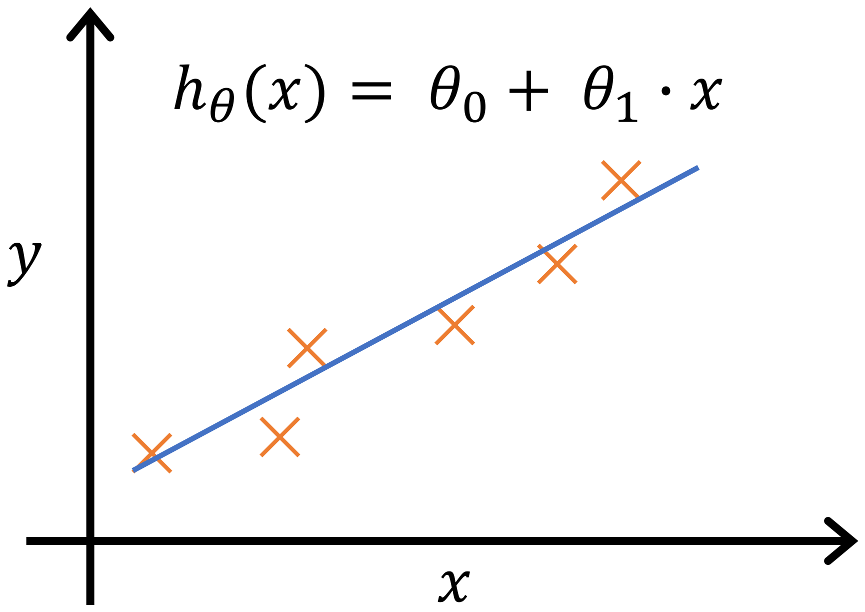


$\theta_0 = 0$
$\theta_1 = 0.5$



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

Deciding and Choosing the parameters θ_is



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

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

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

The idea is to choose θ_0 and θ_1 such that $h_\theta(x)$ is close to y for given training sample pairs (x, y)

$J(\theta_0, \theta_1)$ is called as cost function.
It is also called as squared error function.

Hypothesis function (Simplified)

- Hypothesis

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

- Parameters

$$\theta_0, \theta_1$$

- Cost function

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

- Goal

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

- Simplified Hypothesis ($\theta_0 = 0$)

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

- Parameter

$$\theta_1$$

- Cost function

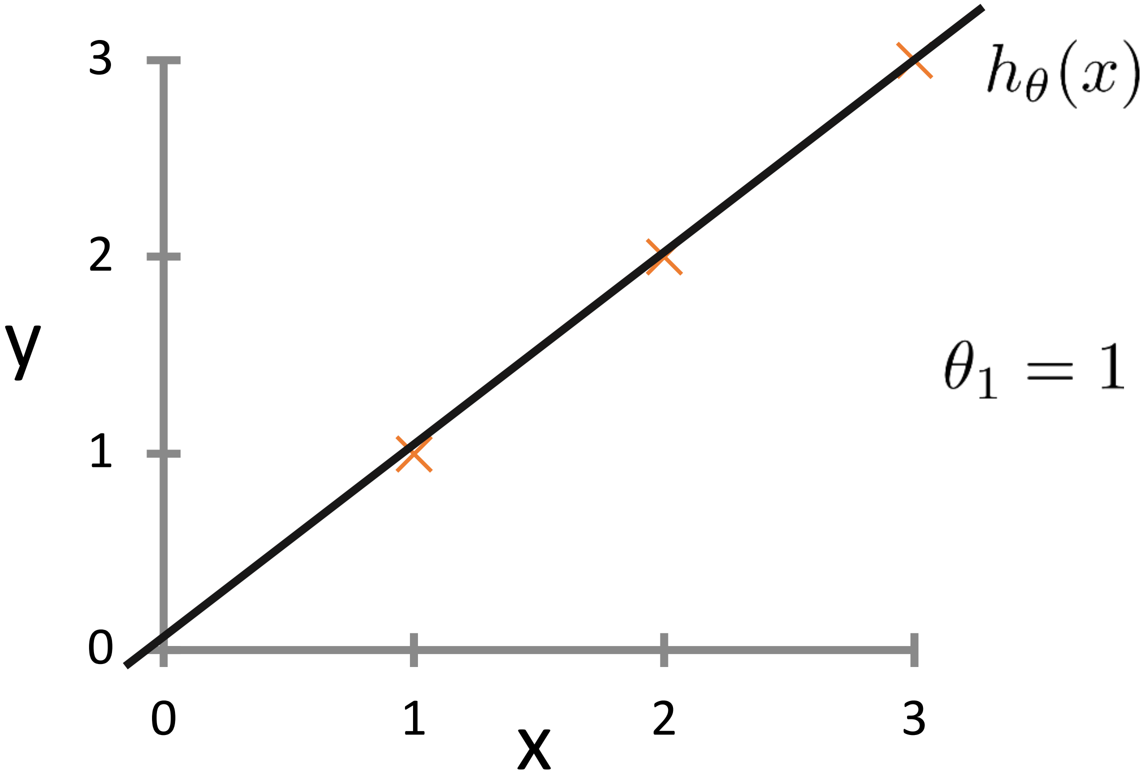
$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- Goal

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

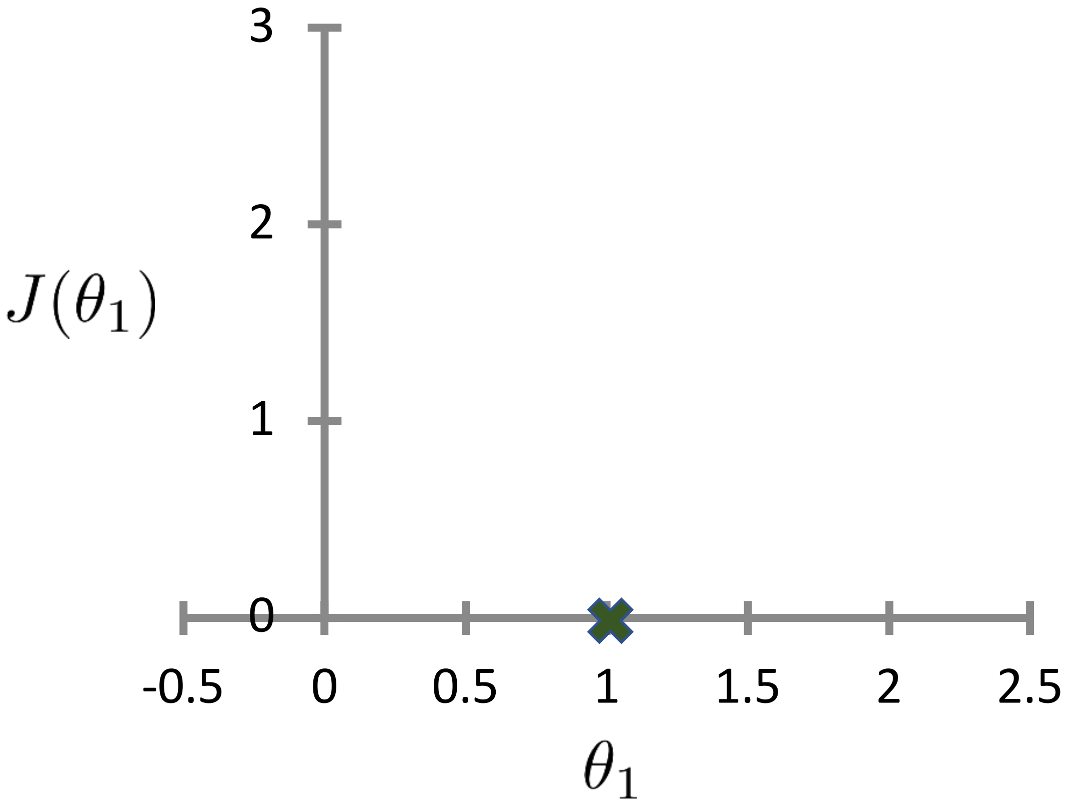
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



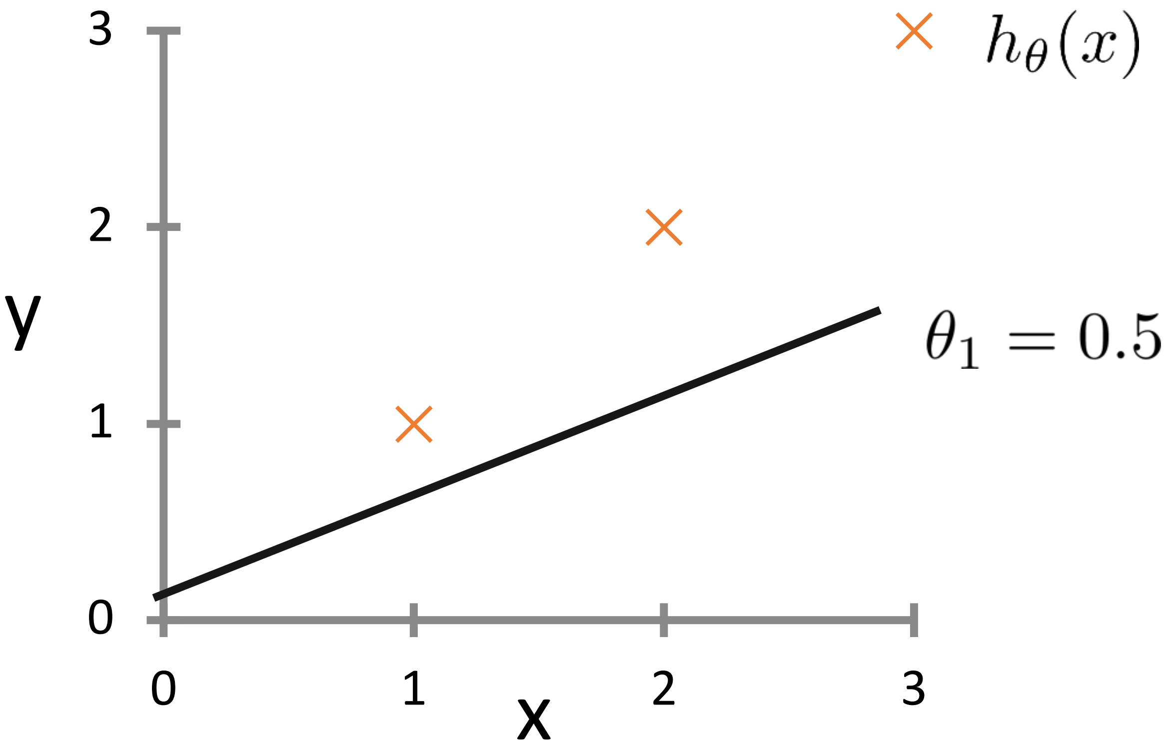
$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 \cdot x^{(i)} - y^{(i)}\right)^2$$

$$J(\theta_1 = 1) = \frac{1}{2*3}\left\{(1\times 1 - 1)^2 + (1\times 2 - 2)^2 + (1\times 3 - 3)^2\right\}$$

$$J(\theta_1 = 1) = 0$$

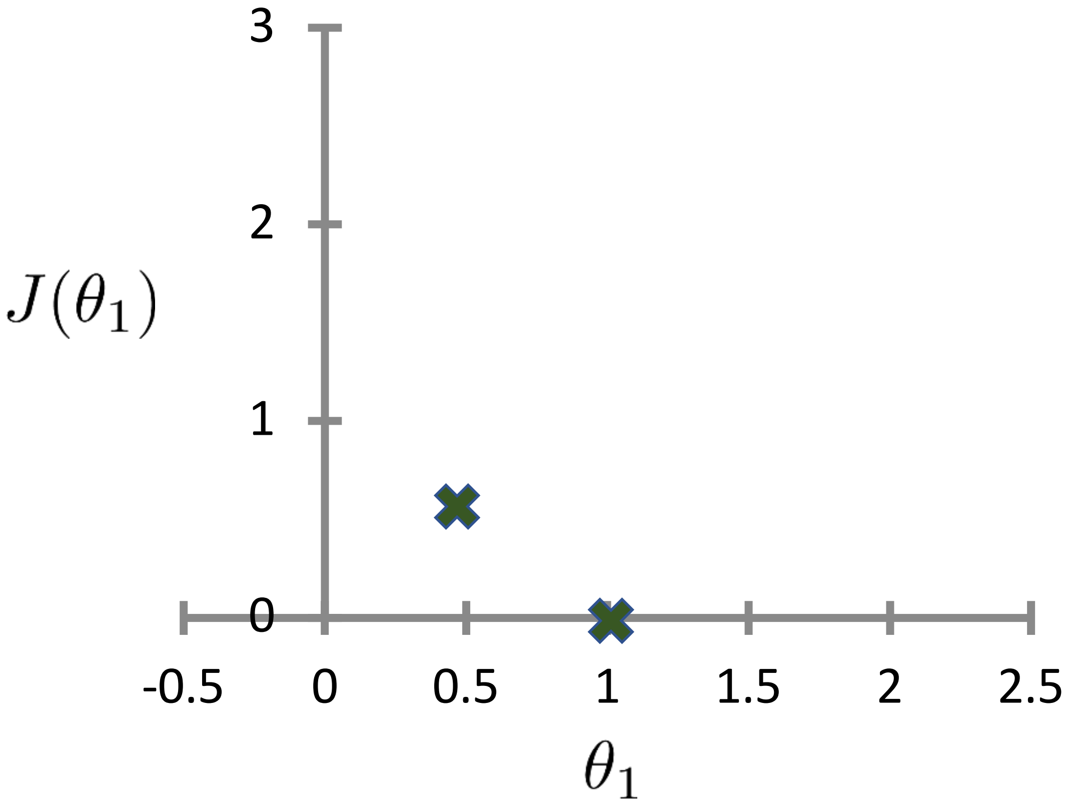
$$h_\theta(x)$$

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



$$J(\theta_1)$$

(function of the parameter θ_1)



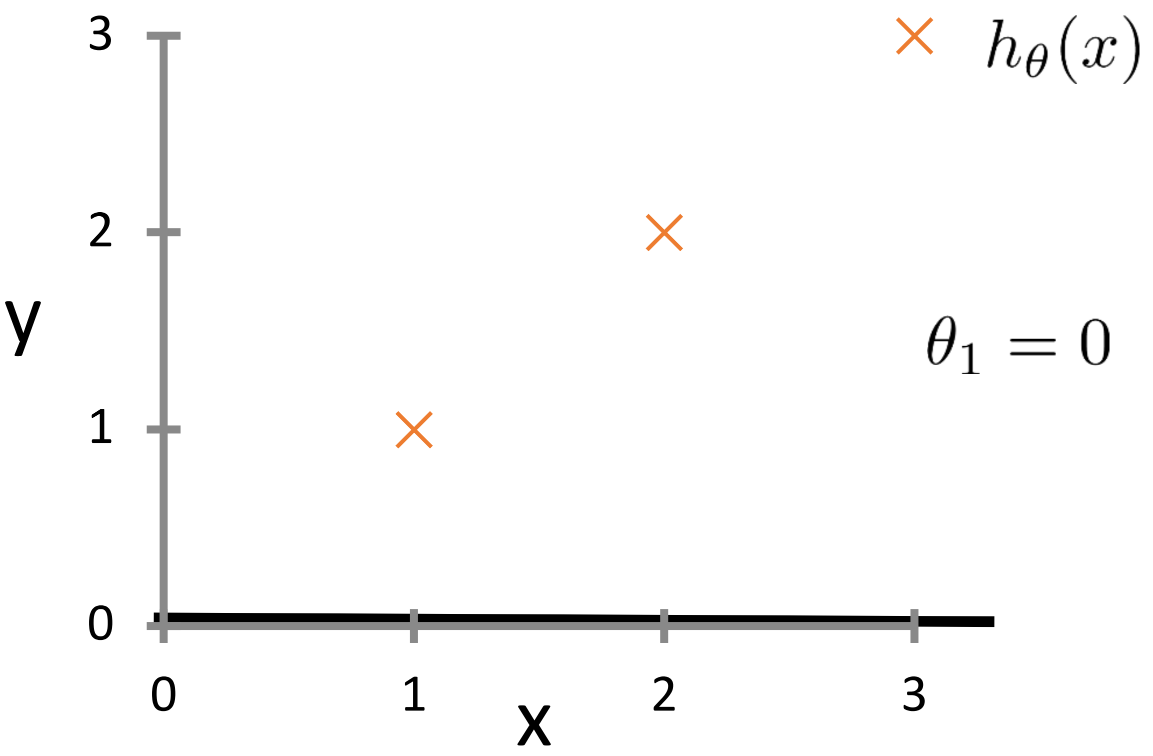
$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 \cdot x^{(i)} - y^{(i)}\right)^2$$

$$J(\theta_1 = 0.5) = \frac{1}{2*3}\{(0.5\times1 - 1)^2 + (0.5\times2 - 2)^2 + (0.5\times3 - 3)^2\}$$

$$J(\theta_1 = 0.5) = 3.5/6 = 0.58$$

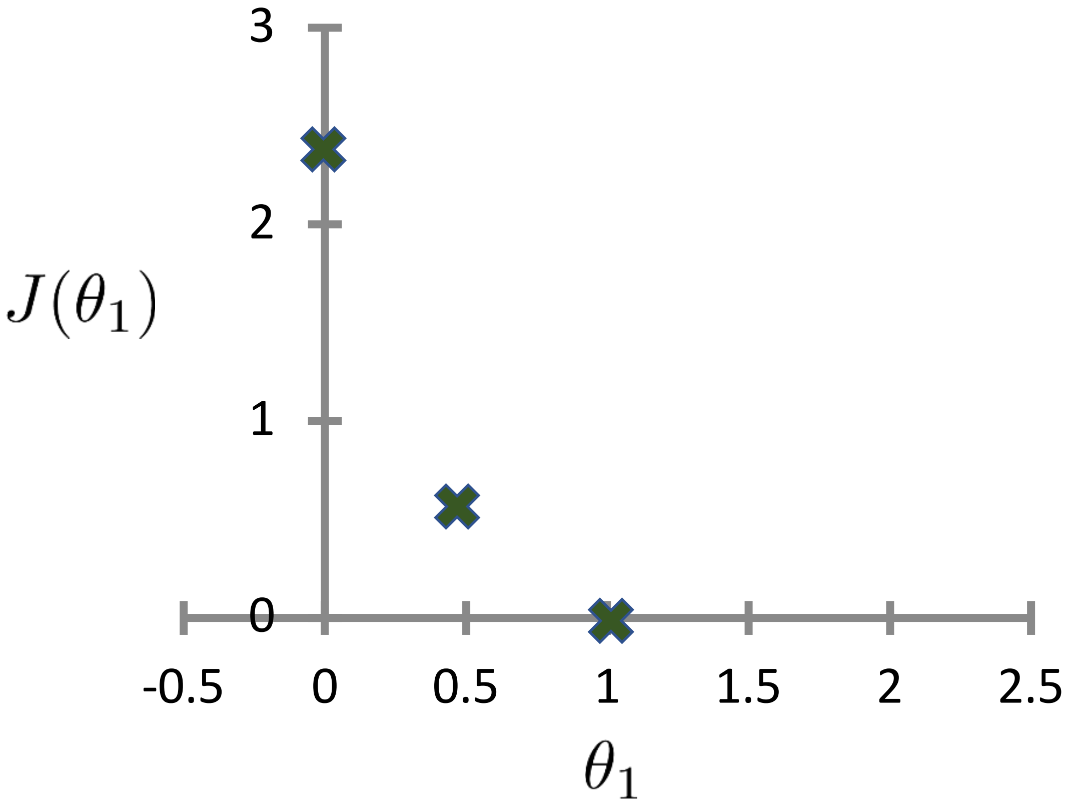
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)

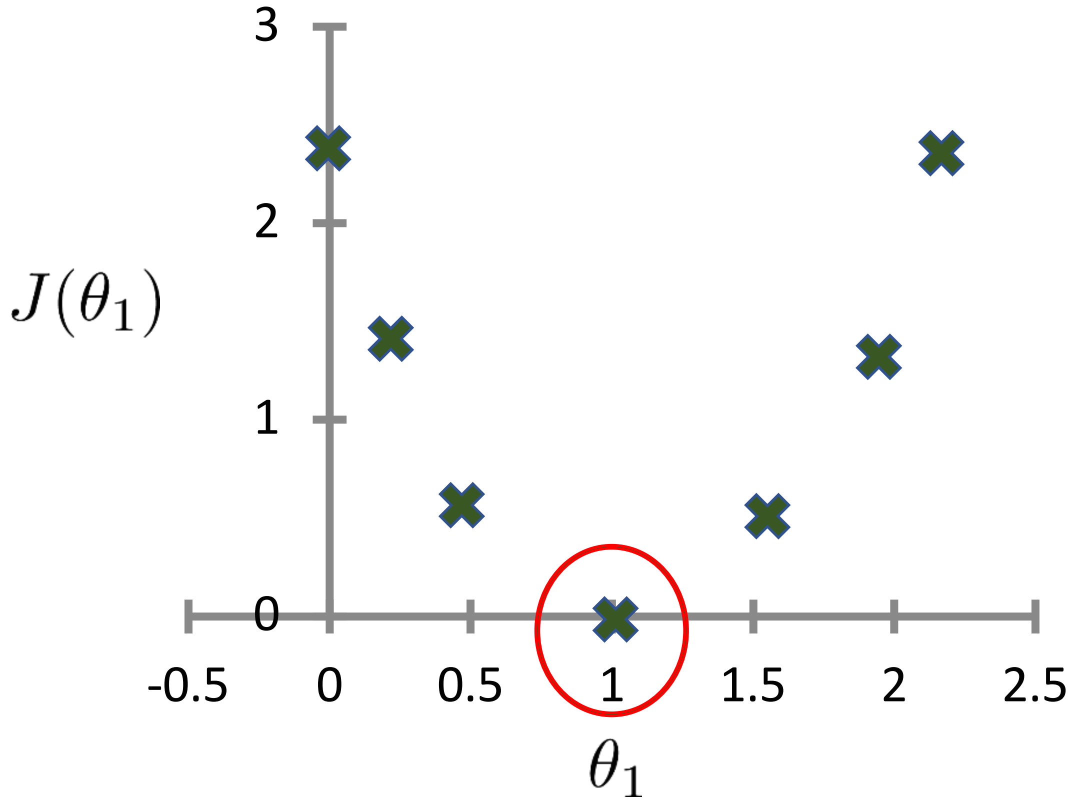


$$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 = \frac{1}{2m}\sum_{i=1}^{m}\left(\theta_1 \cdot x^{(i)} - y^{(i)}\right)^2$$

$$J(\theta_1 = 0) = \frac{1}{2 * 3}\{(0 - 1)^2 + (0 - 2)^2 + (0 - 3)^2\}$$

$$J(\theta_1 = 0) = 14/6 = 2.3$$

Which one to choose?



The value of θ_1 which gives minimum $J(\theta_1)$

Hypothesis function (Summary)

- Hypothesis

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

- Parameters

$$\theta_0, \theta_1$$

- Cost function

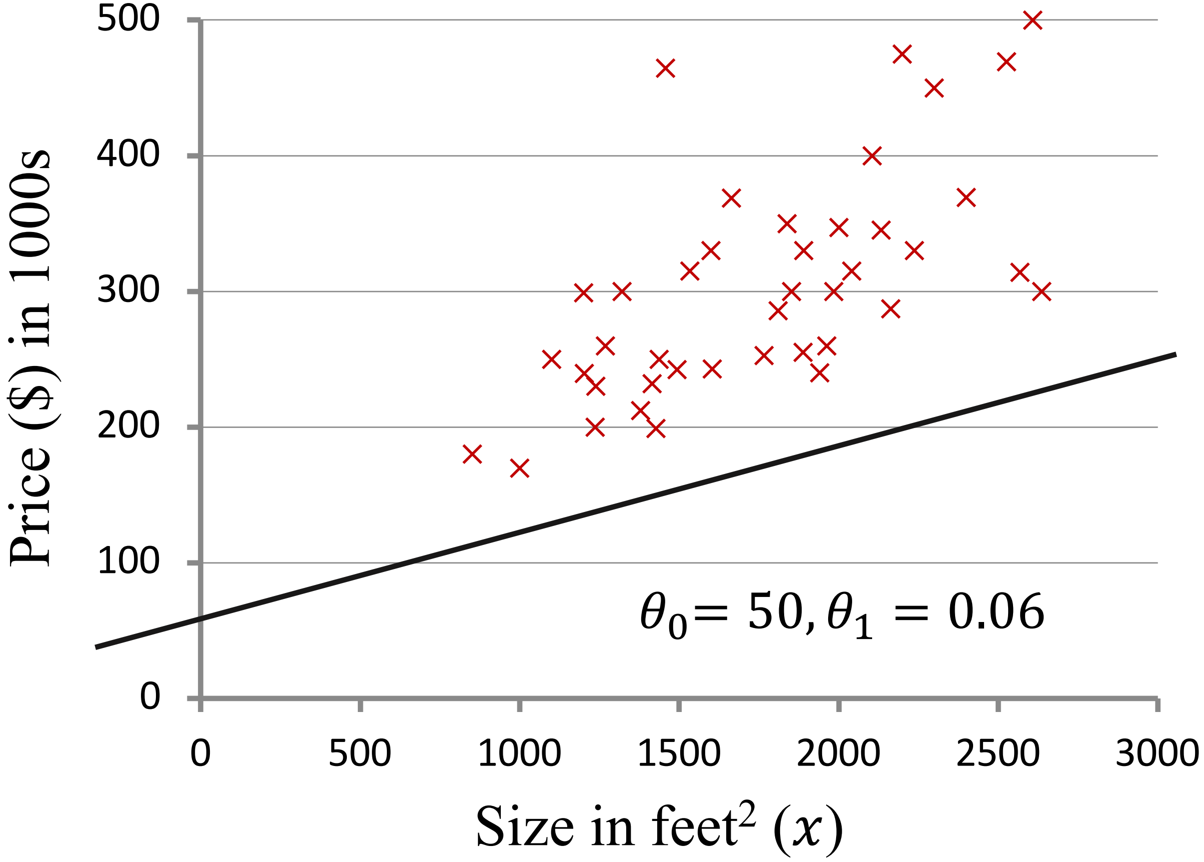
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

- Goal

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

$h_\theta(x)$

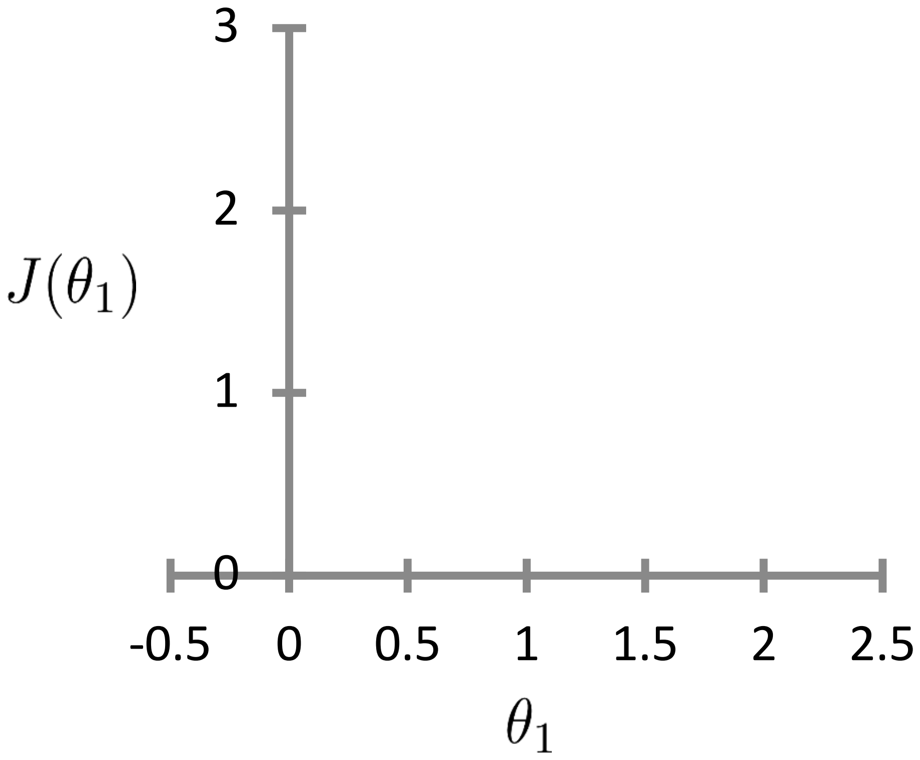
(for fixed θ_0, θ_1, this is a function of x)



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

$J(\theta_0, \theta_1)$

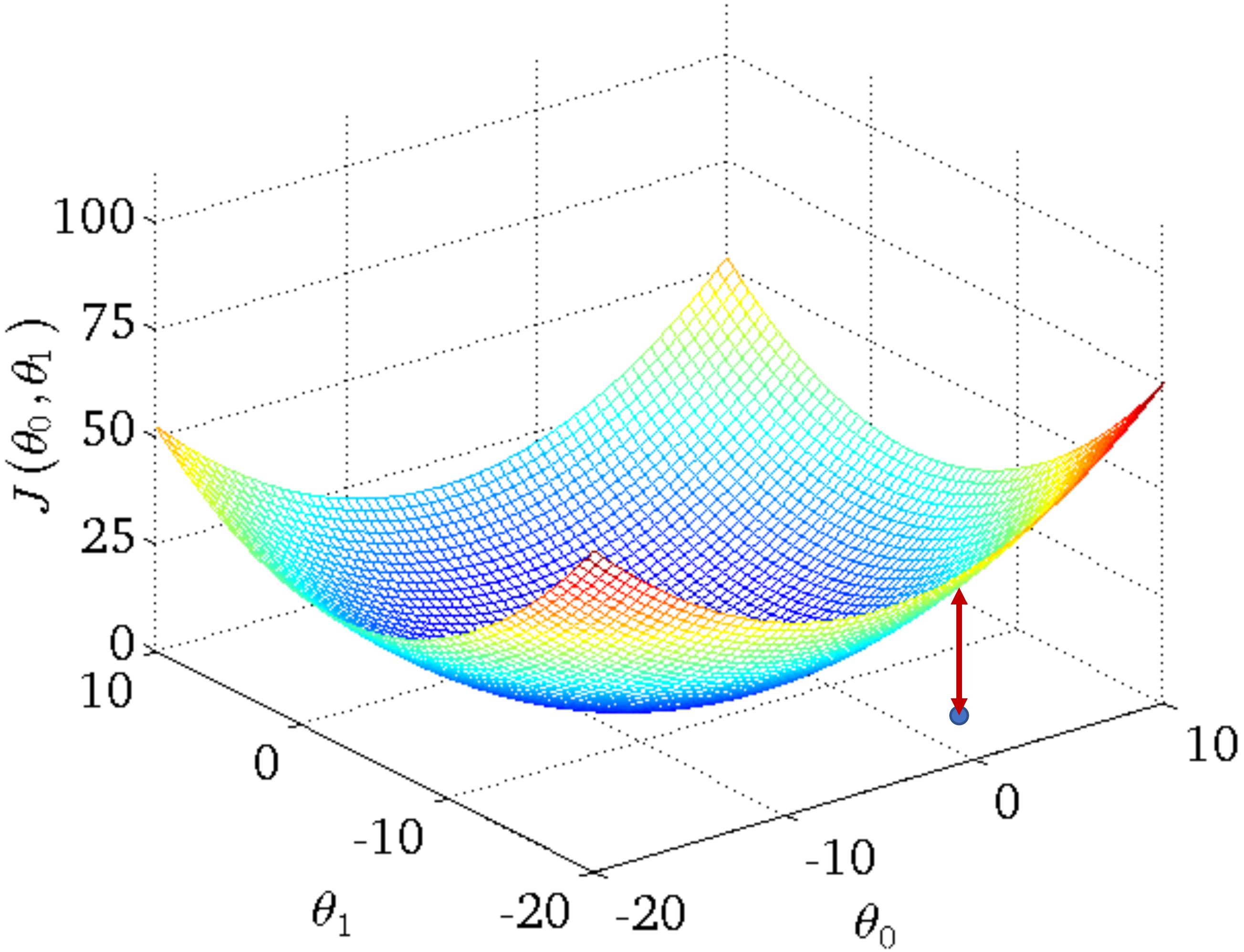
(function of the parameters θ_0, θ_1)



- But now we have two parameters θ_0, θ_1.
- Hence, we need to add an extra axis to the plot.

Loss Function in 3-D (Example)

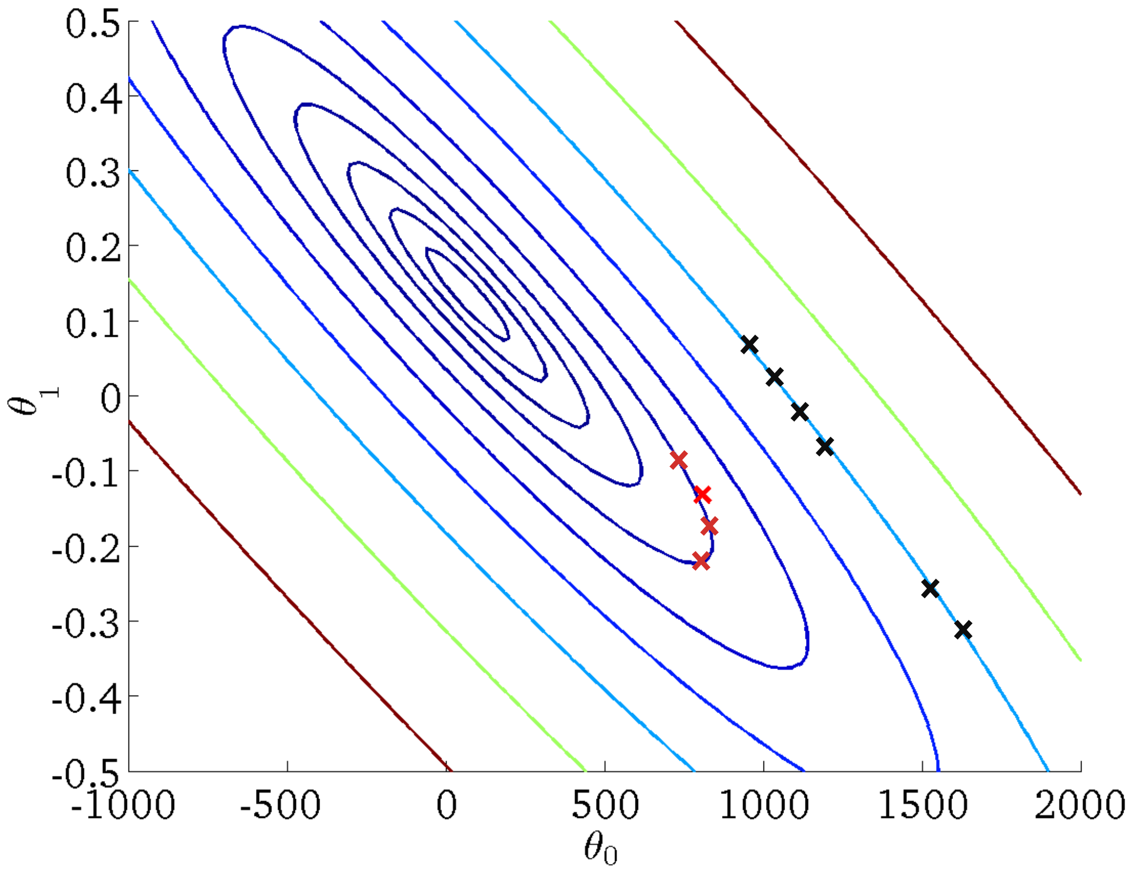
- This is a 3D surface plot
- The axes denotes:
  - both the parameters (θ_0, θ_1)
  - the loss function $J(\theta_0, \theta_1)$
- As we vary the values of θ_0 and θ_1 we get different values of the loss function $J(\theta_0, \theta_1)$
- *[Informal]* The height of the surface from the (θ_0, θ_1) denotes the value of loss at that point.



Demo: https://academo.org

Contour Plots

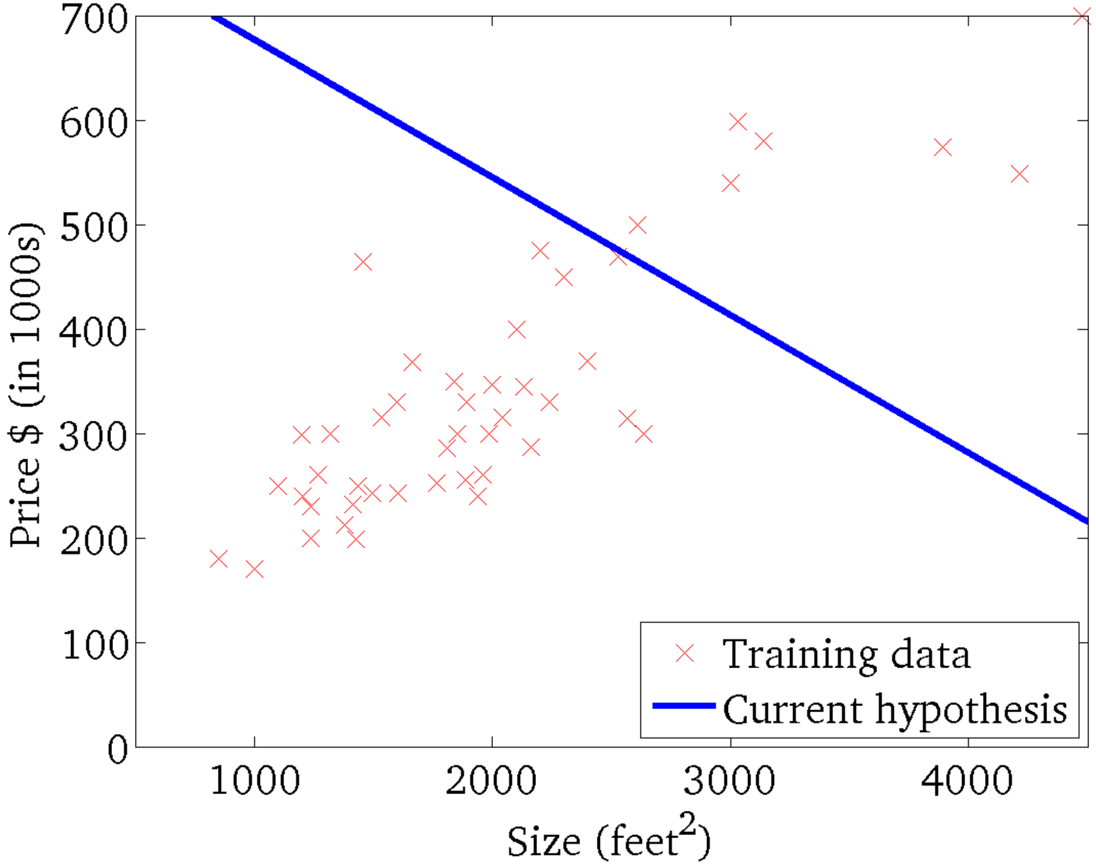
- This is a contour plot of a 3D surface.
- The axes denotes the parameters θ_0 and θ_1.
- An ellipse (or oval) is a set of points that takes on the same value.
- *[Generally]* A color towards the shade of blue (cold color) denotes a lower value, whereas a color towards the red shade (warm color) denotes a higher value.
- For example:



- These all points will have same value as they lie on the same ellipse

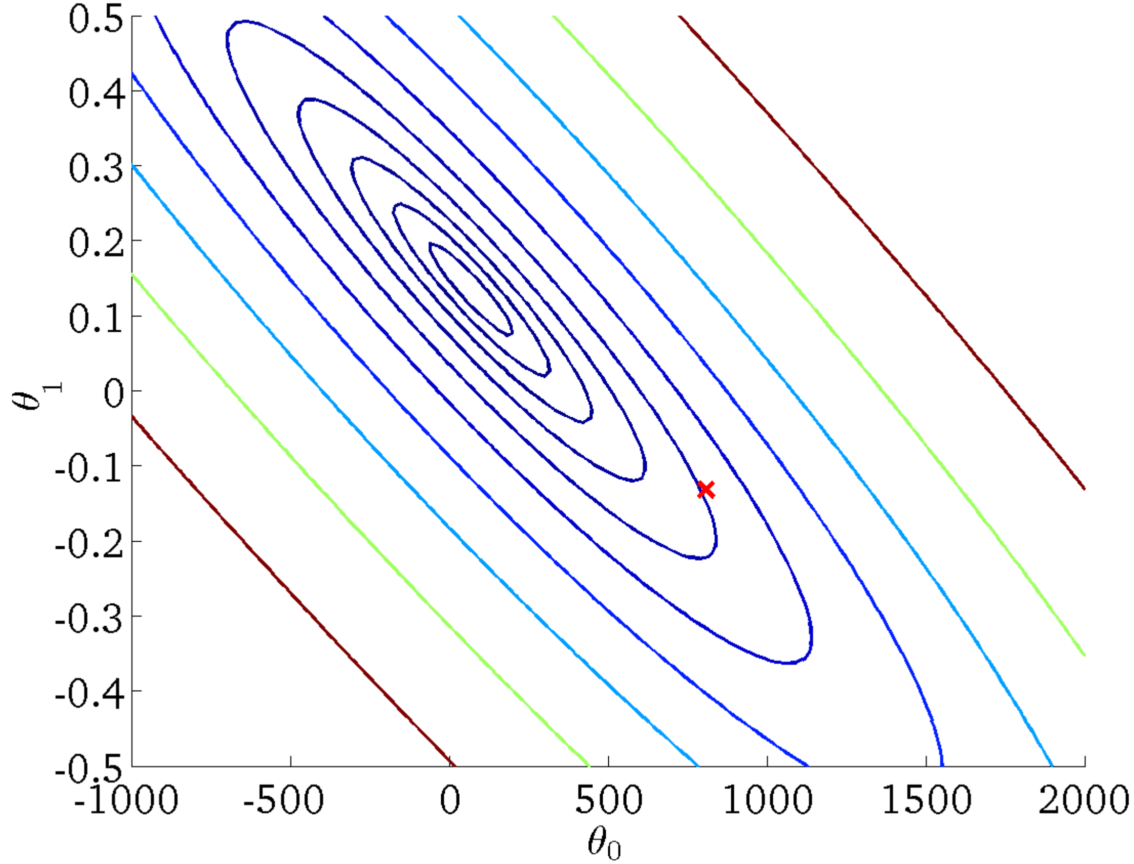
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

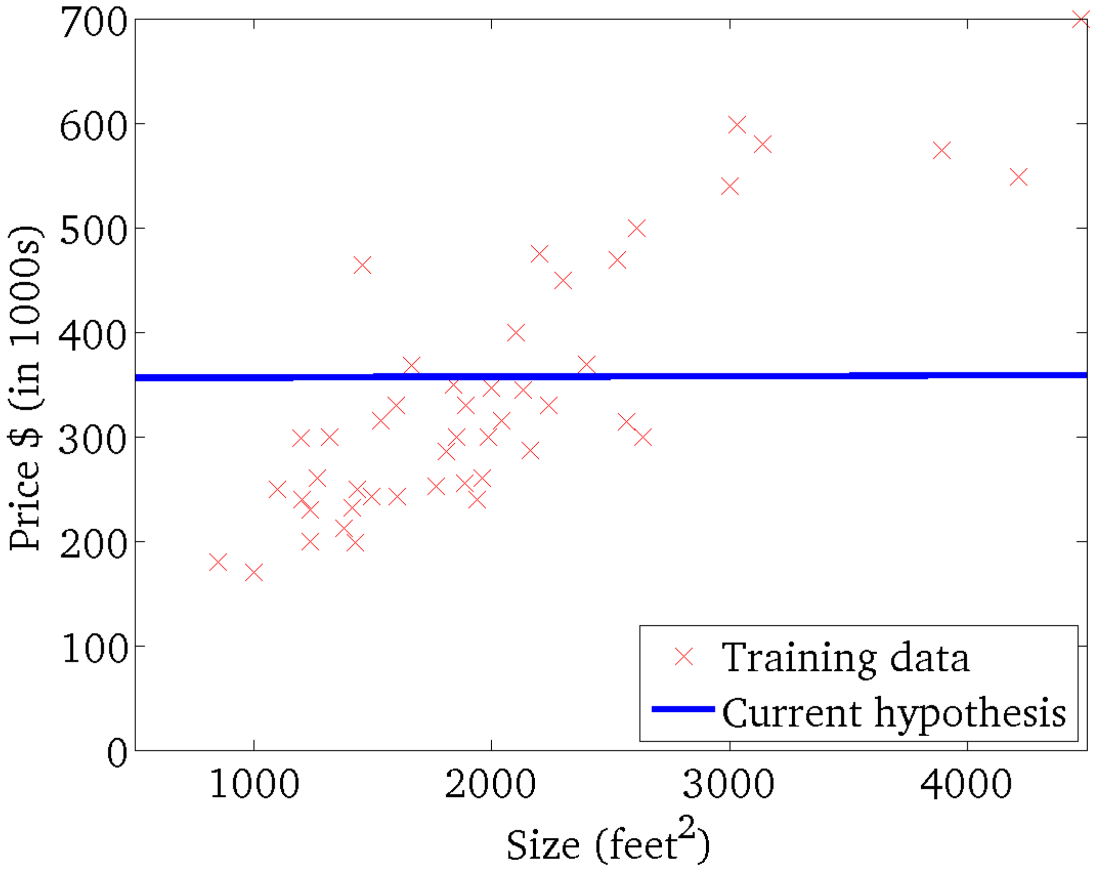
(function of the parameters θ_0, θ_1)



$\theta_0 \approx 800$ and $\theta_1 \approx -0.15$

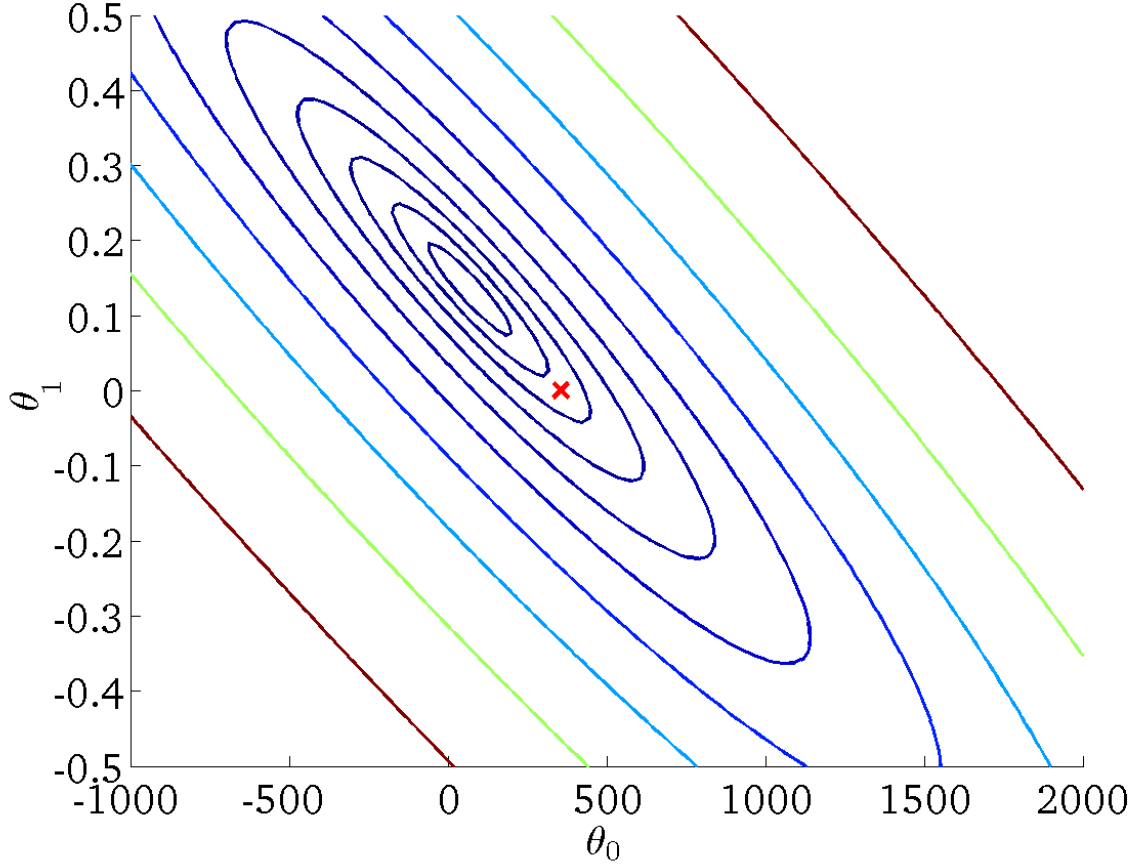
$h_\theta(x)$

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



$J(\theta_0, \theta_1)$

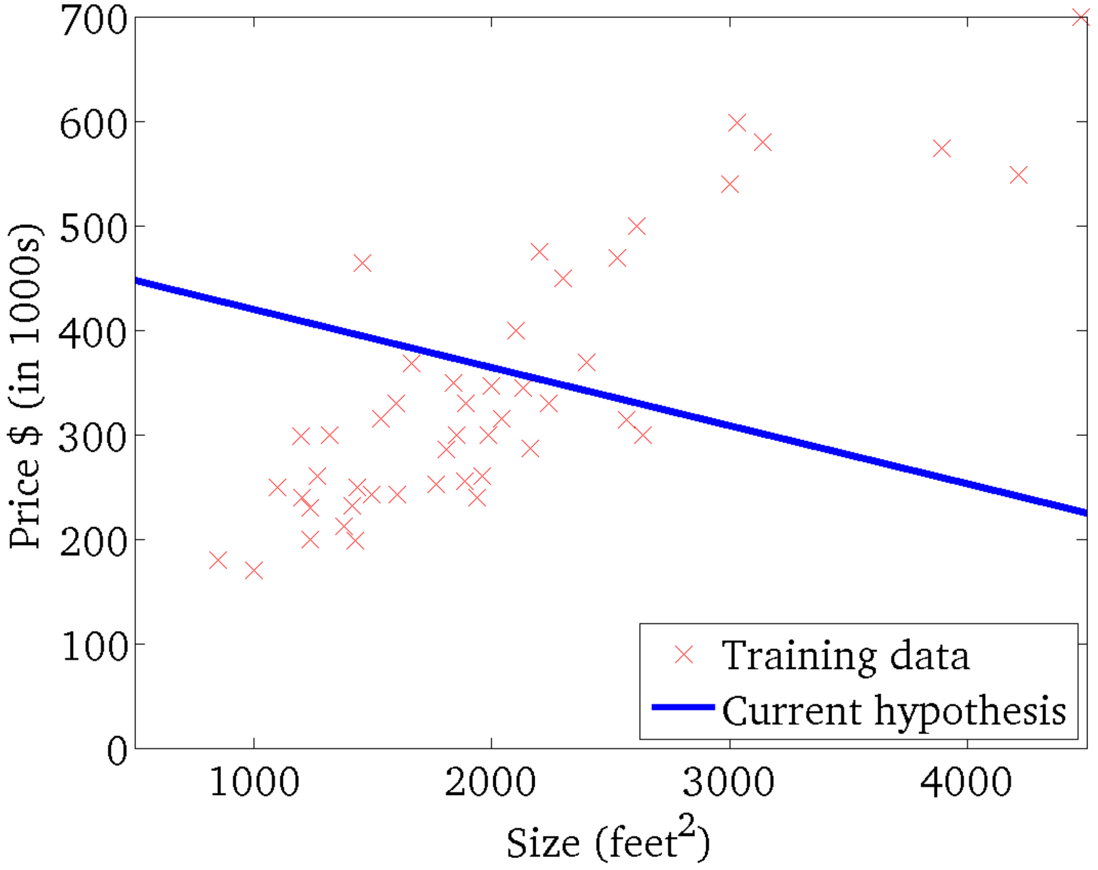
(function of the parameters θ_0, θ_1)



$\theta_0 \approx 360$ and $\theta_1 \approx 0$

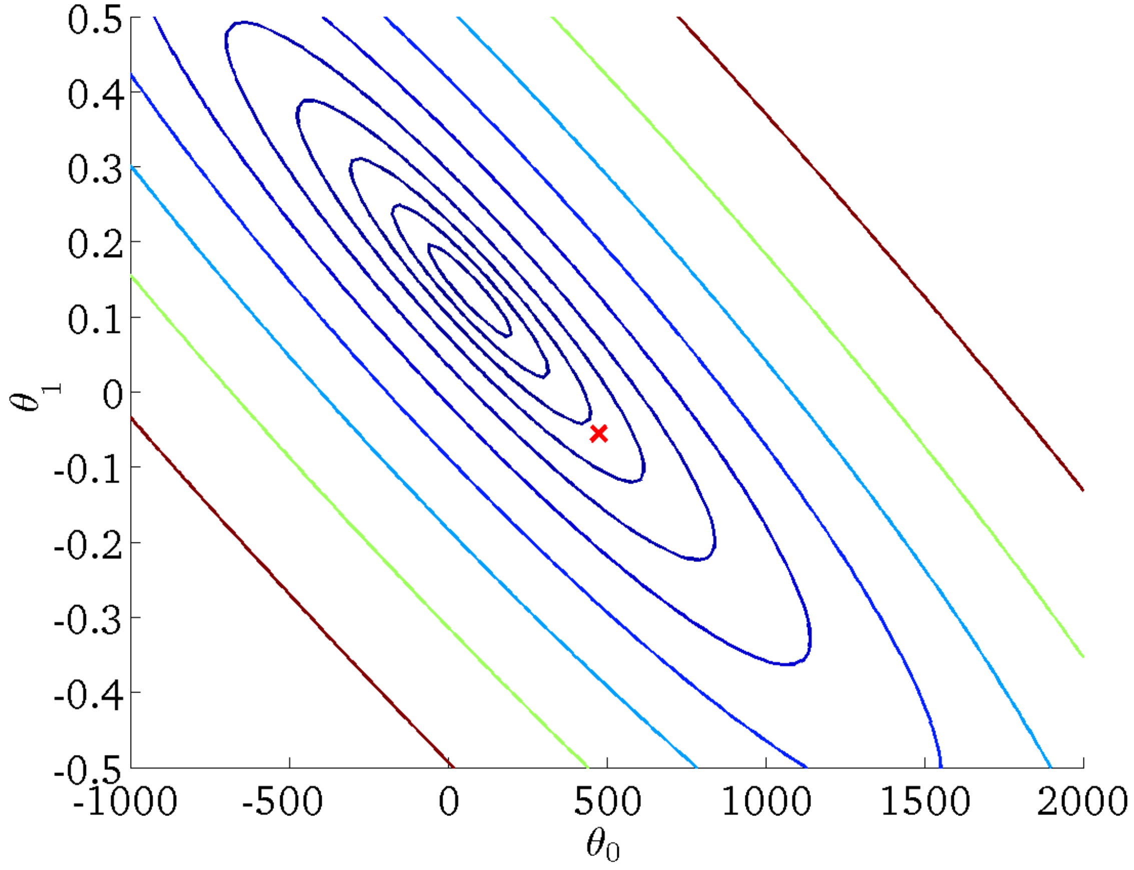
$h_\theta(x)$

(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)



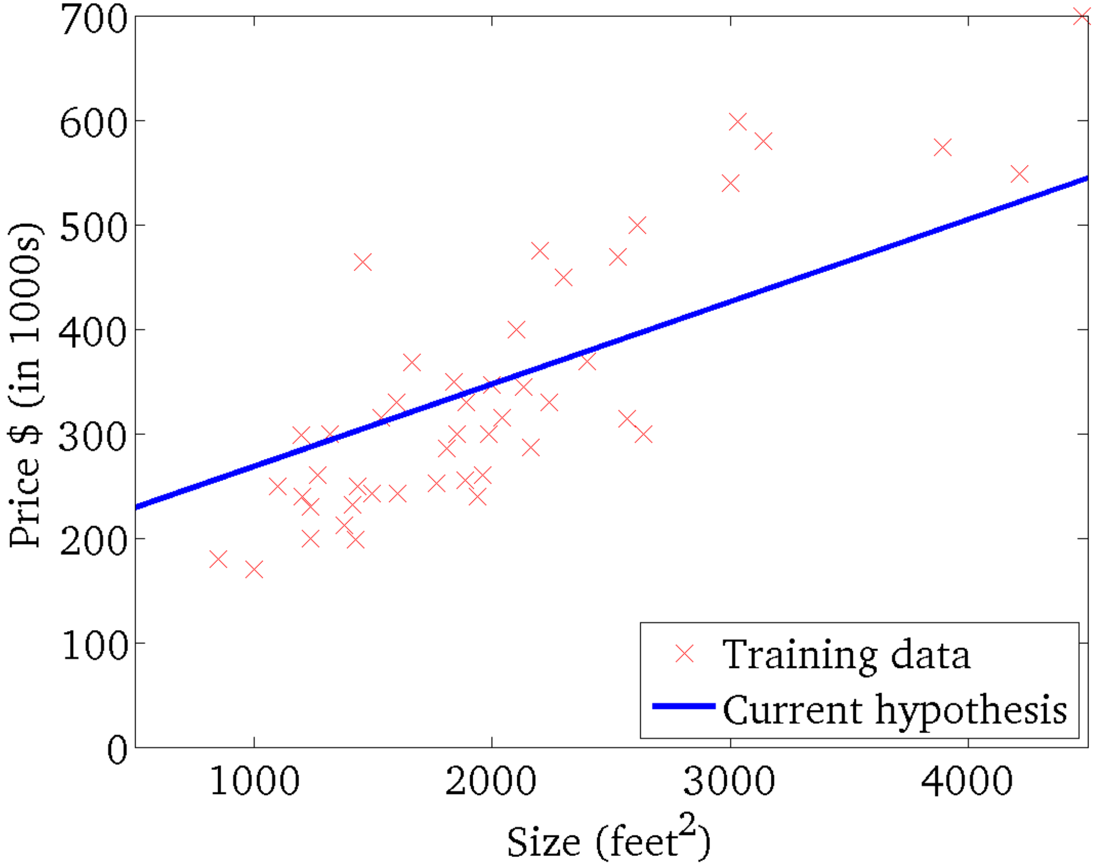
$\theta_0 \approx 510$ and $\theta_1 \approx -0.02$

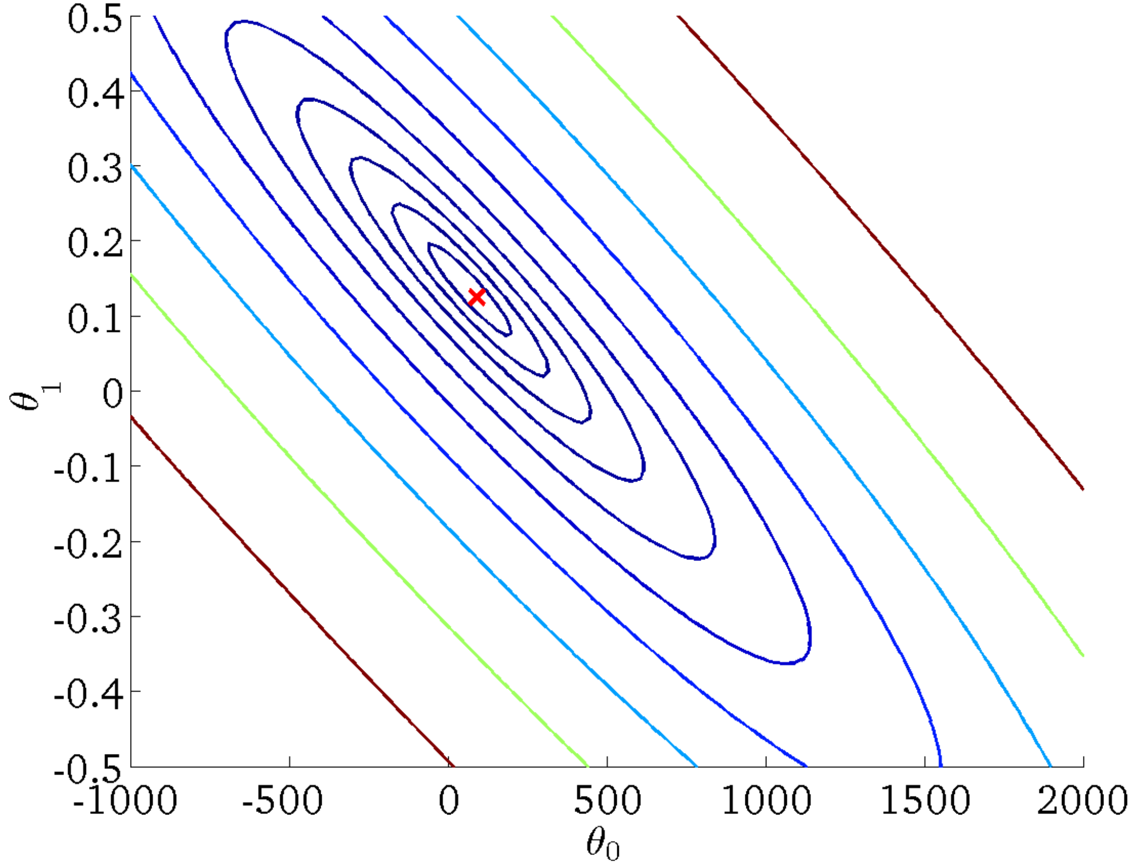
$h_\theta(x)$

(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)





$\theta_0 \approx 480$ and $\theta_1 \approx 0.14$

Not minimum, but close to minimum.

Gradient Descent Algorithm

- Cost function

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

- Goal

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

Algorithm

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

- Generalized Cost function

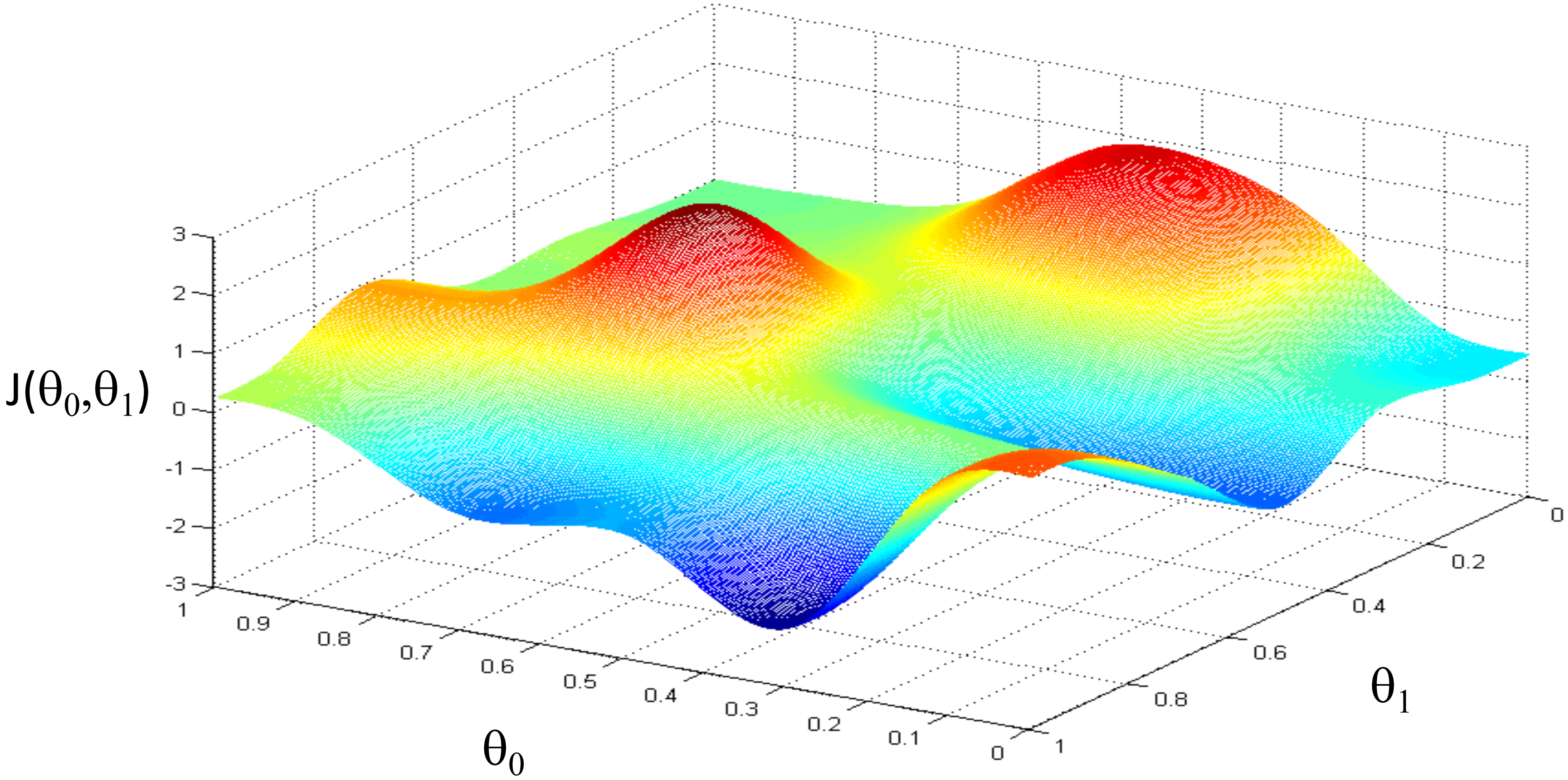
$$J(\theta_0, \theta_1, \theta_2\ \ldots \theta_n)$$

- Goal

$$\underset{\theta_0 \ldots \theta_n}{minimize}\ J(\theta_0, \theta_1, \theta_2\ \ldots \theta_n)$$

Algorithm

1. Start with some $\theta_0, \theta_1, \theta_2\ \ldots \theta_n$.
2. Keep changing $\theta_0, \theta_1, \theta_2\ \ldots \theta_n$ to reduce $J(\theta_0, \theta_1, \theta_2\ \ldots \theta_n)$ until we hopefully end up at a minimum.

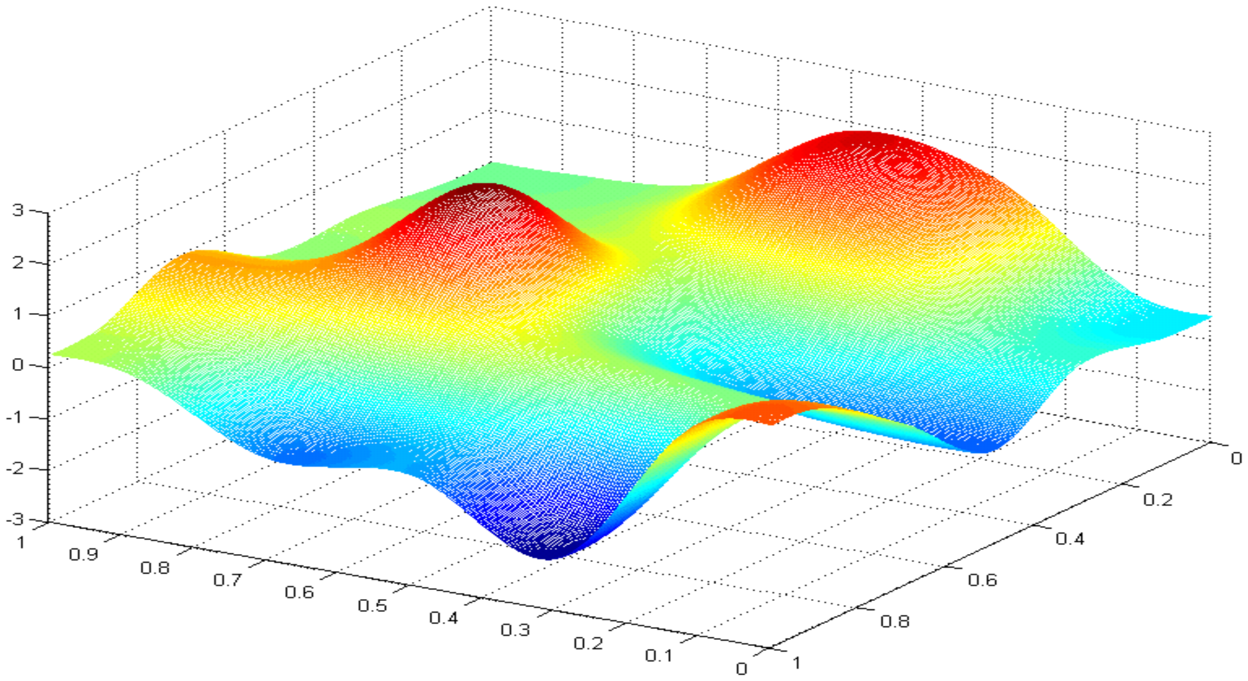


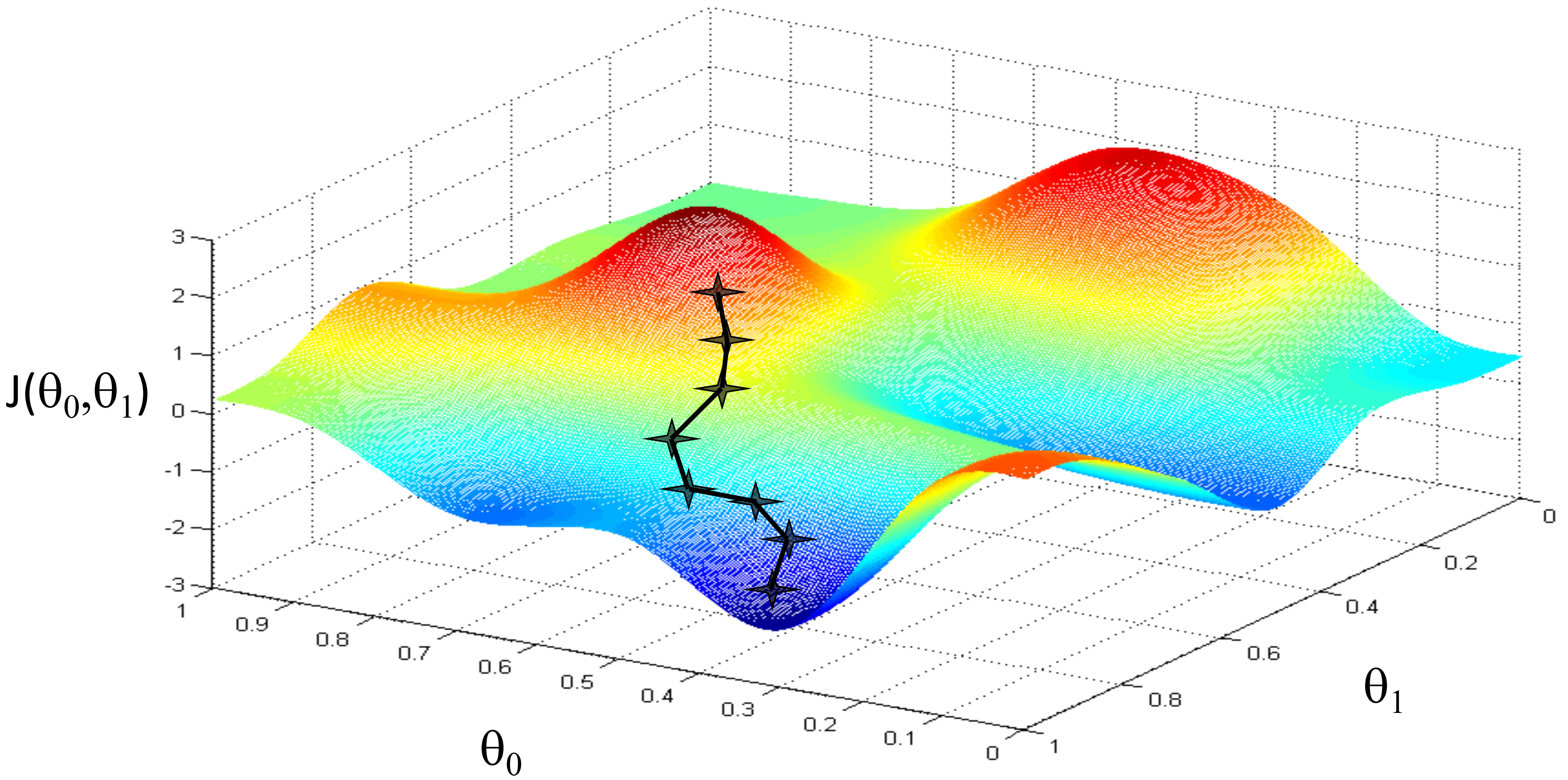
3
2
1
0
-1
-2
-3
J(θ0,θ1)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Gradient Descent Algorithm – Intuitive Approach

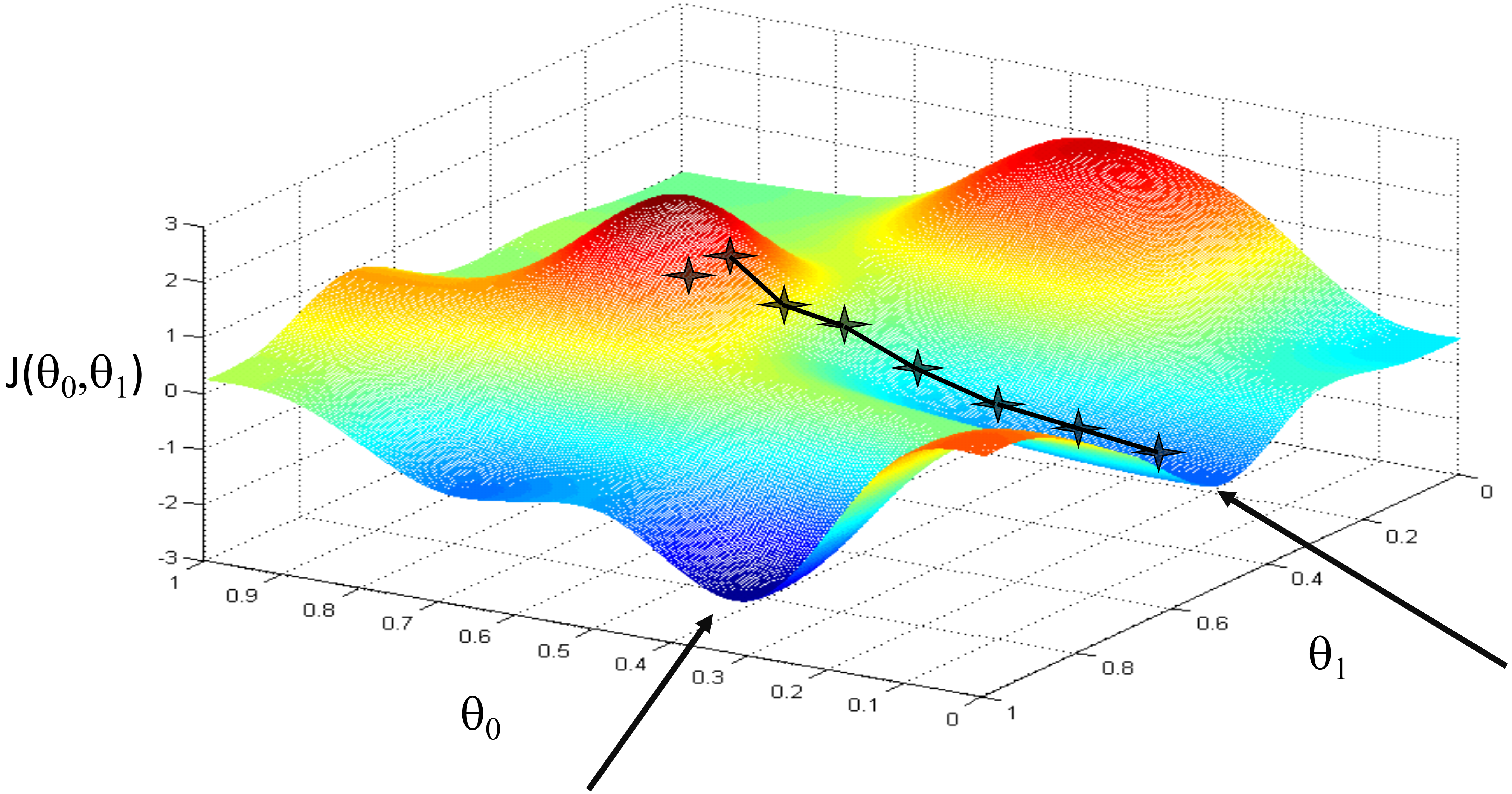
- Assume we are in some grassy park, with two hill like structures.
- Assume that we are physically standing at that point on the hill, on this little red hill in the park.
- In gradient descent, we take a 360 degrees spin and ponder:

"If we were to take a little steps in some direction, so as we want to go downhill as quickly as possible, What direction should we choose" ?




J(θ0,θ1)
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
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Gradient Descent Algorithm – Initial point matters



Gradient Descent Algorithm

repeat until convergence {

for $j = 0$ and $j = 1${

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

}

}

- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ – gradient or derivative
- α – learning rate (positive constant)

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:

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

Gradient Descent Algorithm – 1-Dimensional

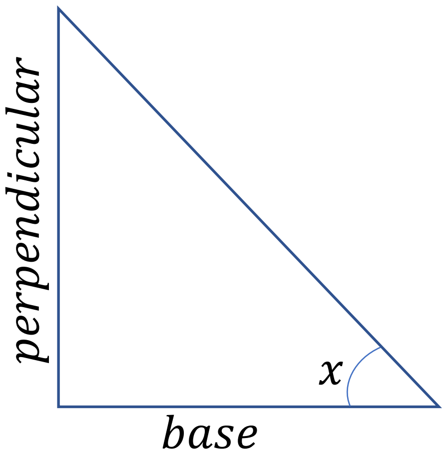
repeat until convergence {

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

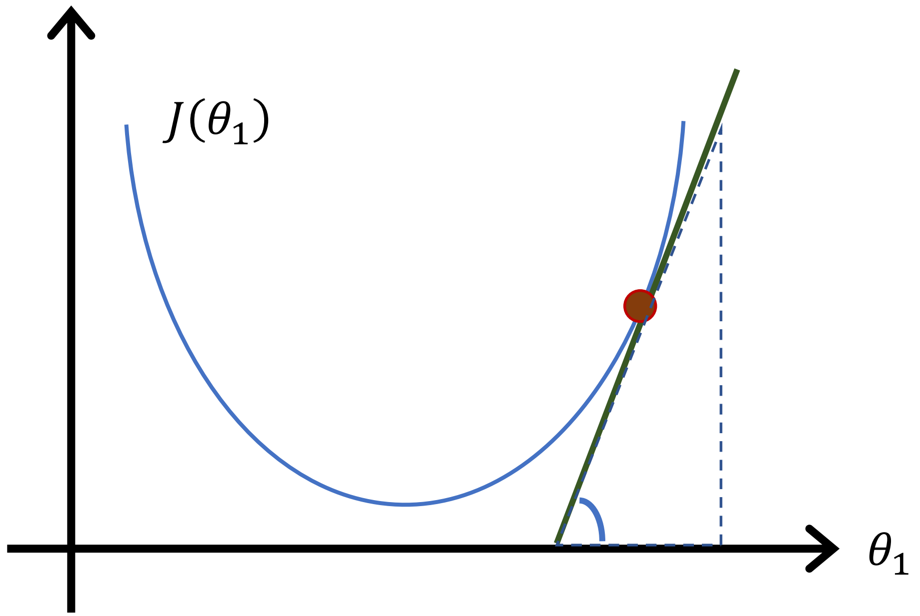
}

- $\frac{\partial}{\partial \theta_j}$ – partial derivative
- $\frac{d}{d\theta_j}$ – derivative

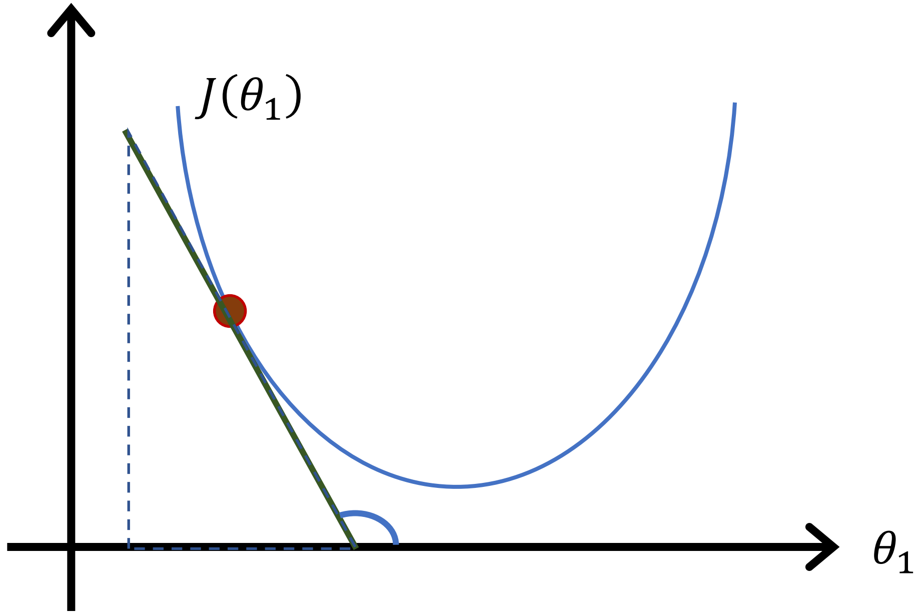
Few more pointers:



- $\tan x = \frac{\text{perpendicular}}{\text{base}}$
- $\tan x$ is positive in 1st and 3rd quadrant.
- $\tan x$ is positive when $0° < x < 90°$ and $180° < x < 270°$.



- $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_j} J(\theta_1)$
- $\frac{d}{d\theta_j} J(\theta_1)$ is positive
- $\theta_1 := \theta_1 - \alpha$ (positive)
- θ_1 decreases

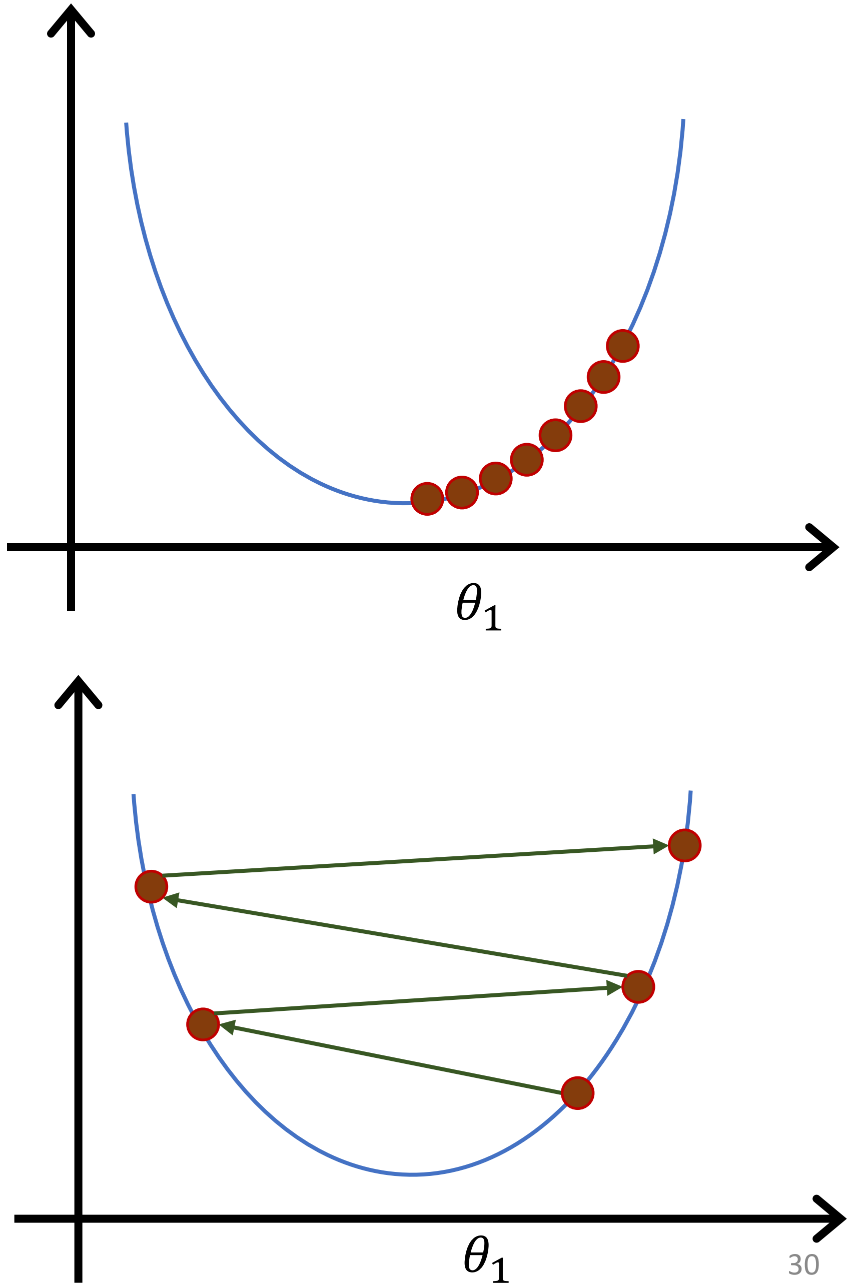


- $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_j} J(\theta_1)$
- $\frac{d}{d\theta_j} J(\theta_1)$ is negative
- $\theta_1 := \theta_1 - \alpha$ (negative)
- θ_1 increases

Effect of Learning Rate (α)

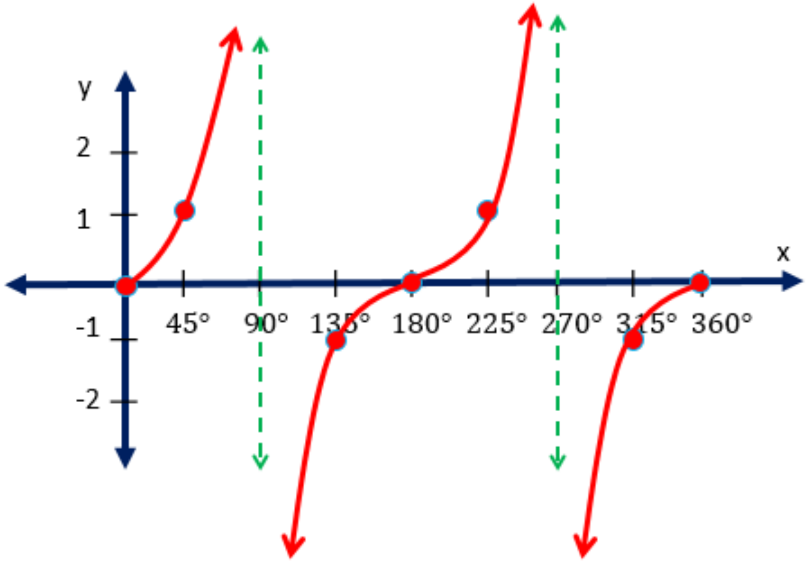
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_j} J(\theta_1)$$

- 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.

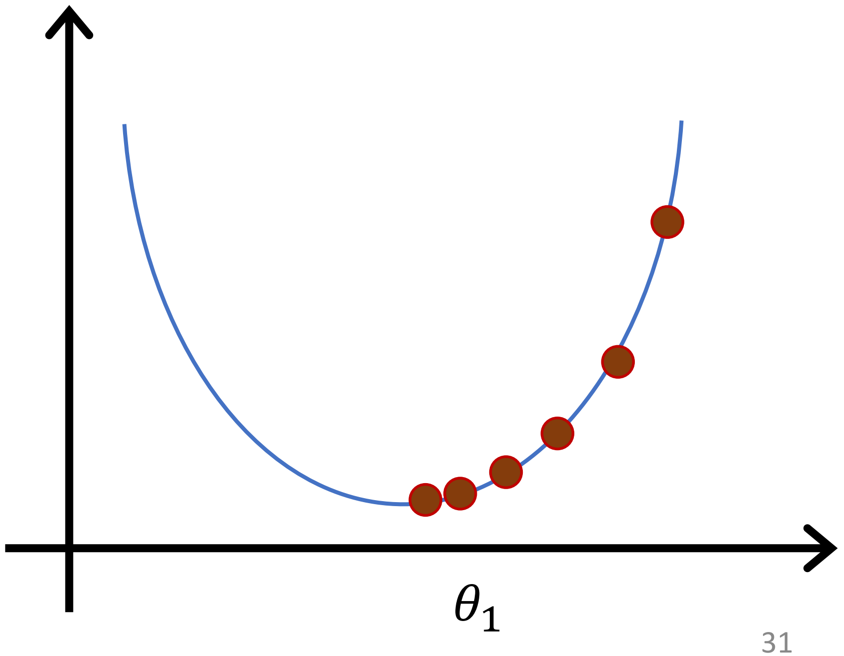


Effect of Learning Rate (α)

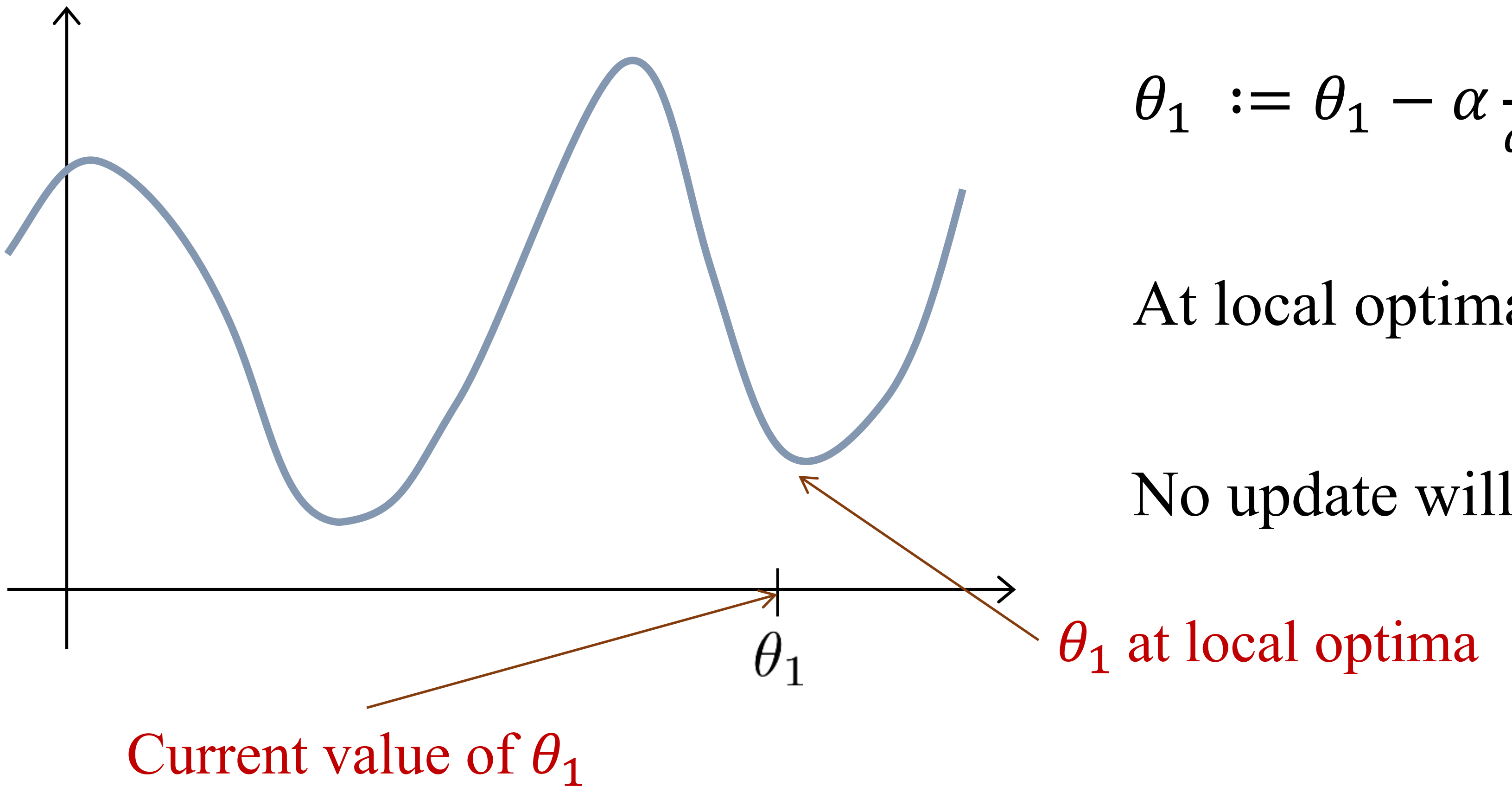
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_j} J(\theta_1)$$



- Gradient descent can converge to a local minimum, even with the learning rate α fixed.
- As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.



Behavior at Local Minima



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

At local optima: $\frac{d}{d\theta_j} J(\theta_1) = 0$

No update will occur

Linear Regression and Gradient Descent Algorithm

Gradient Descent Algorithm

repeat until convergence {

for $j = 0$ and $j = 1${

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

}

}

Linear Regression

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

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

We will use the Gradient Descent Algorithm to

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

Calculating the Gradients

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

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

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

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

Gradient descent algorithm on Linear Regression

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

repeat until convergence {

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

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

}

update θ_0, θ_1 simultaneously

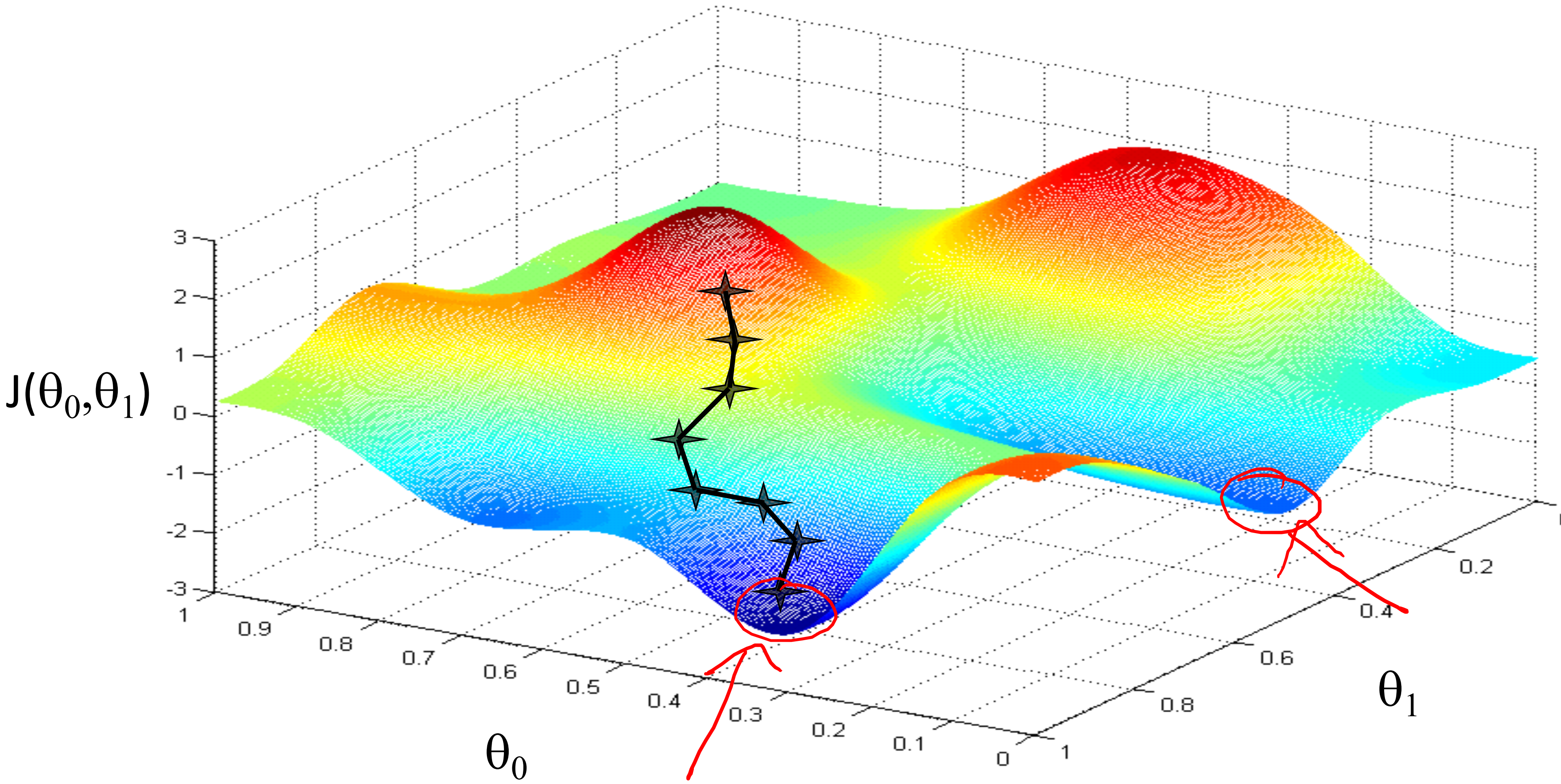
$$\frac{\partial}{\partial\theta_1} J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x^{(i)}$$

repeat until convergence {

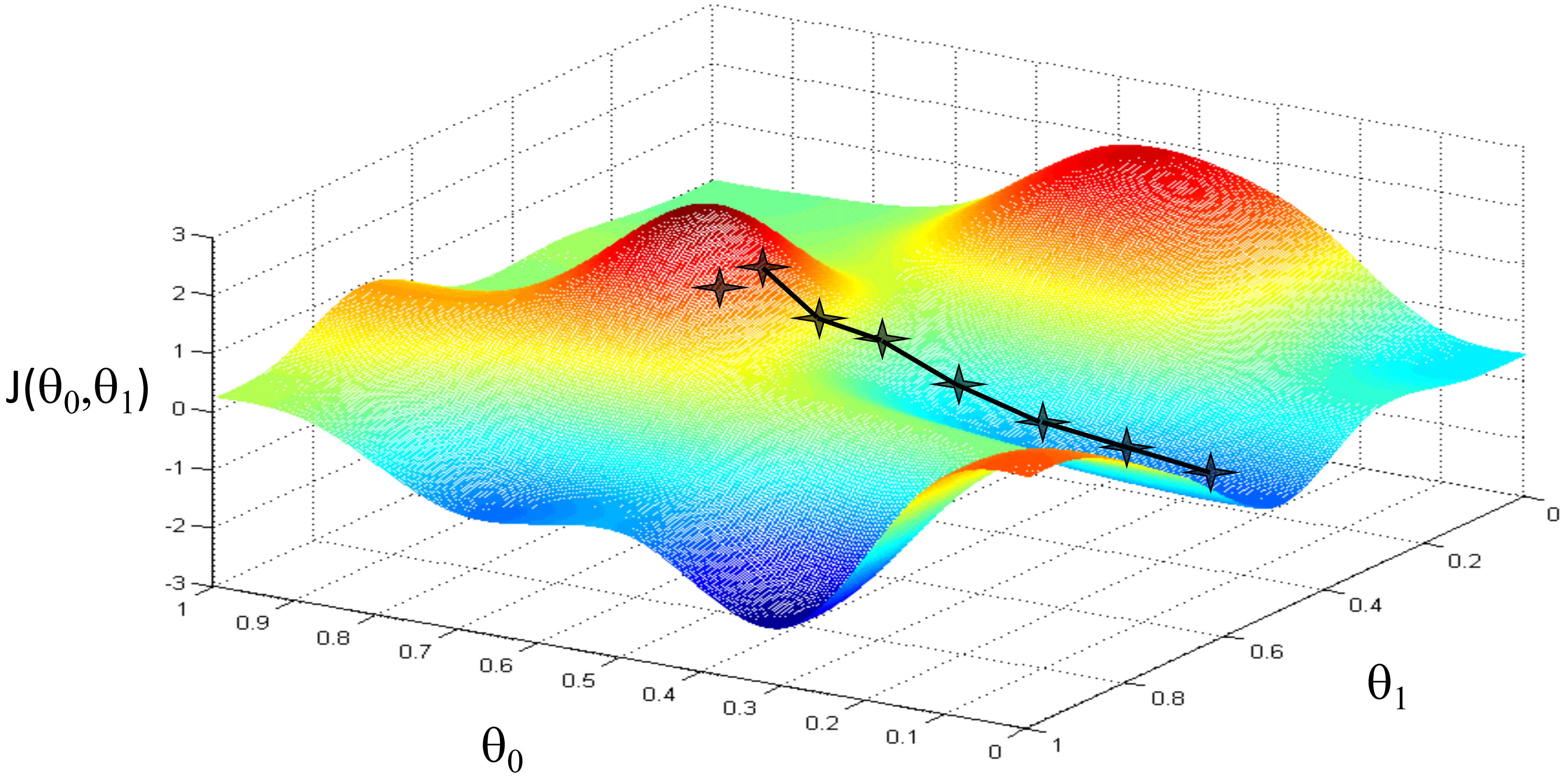
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \textstyle\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

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

}



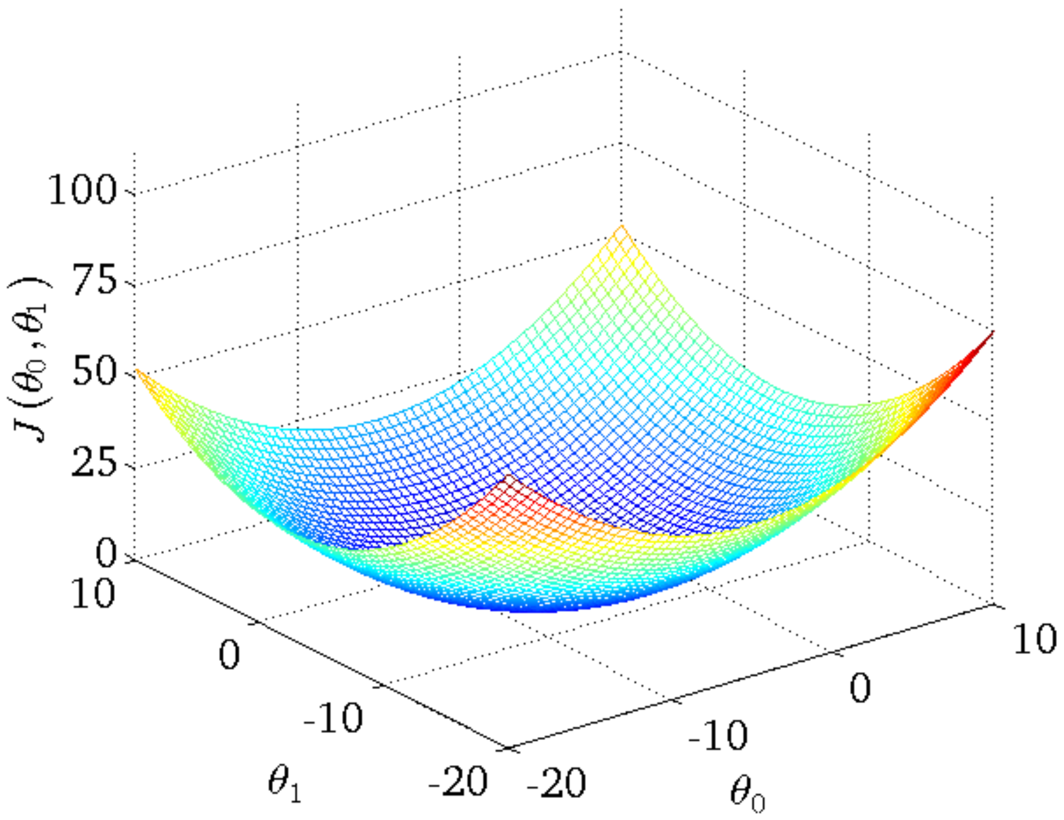
J(θ_0,θ_1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
0
0.2
0.4
0.6
0.8
1
θ_1



$J(\theta_0,\theta_1)$
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

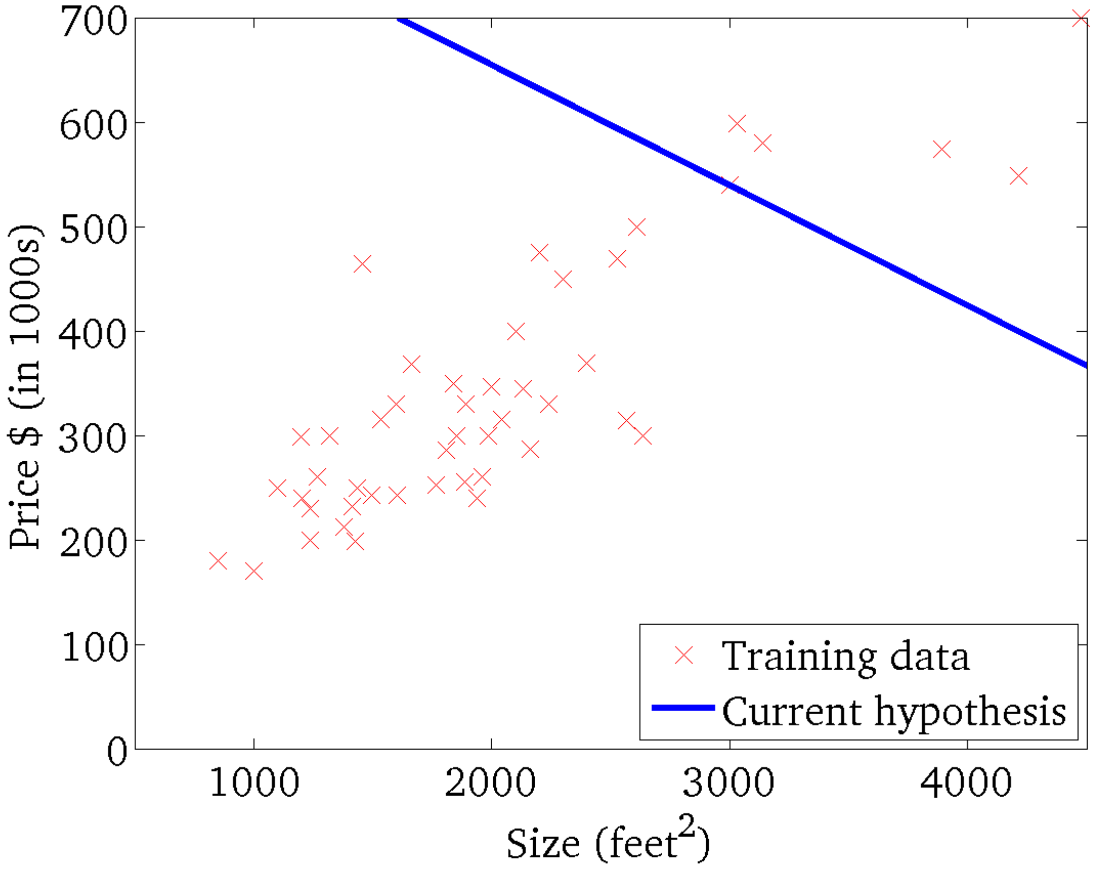
Gradient descent algorithm on Linear Regression

- Cost function for linear regression will always be a bowl shaped.
- Formally, these kind of functions are called as convex function.
- A convex function will always have only one local optima, which is also the global optima.



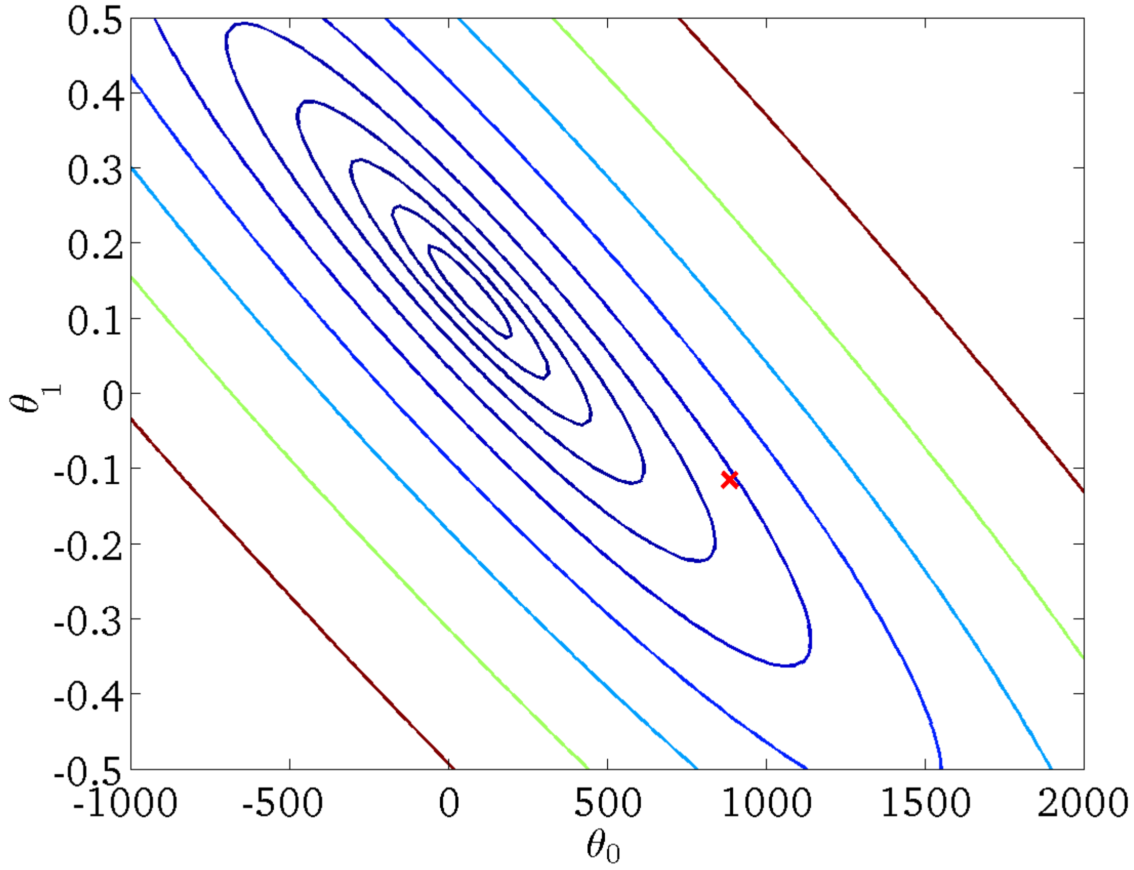
$h_\theta(x)$

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



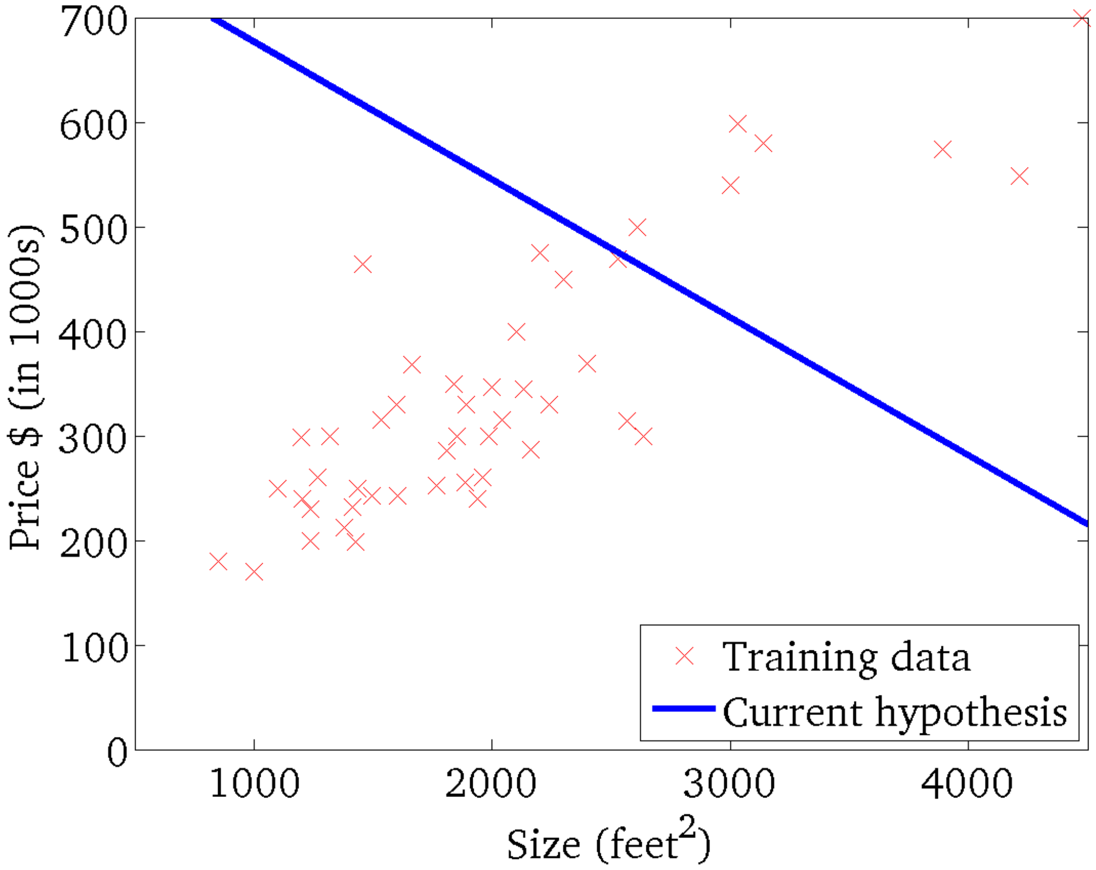
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



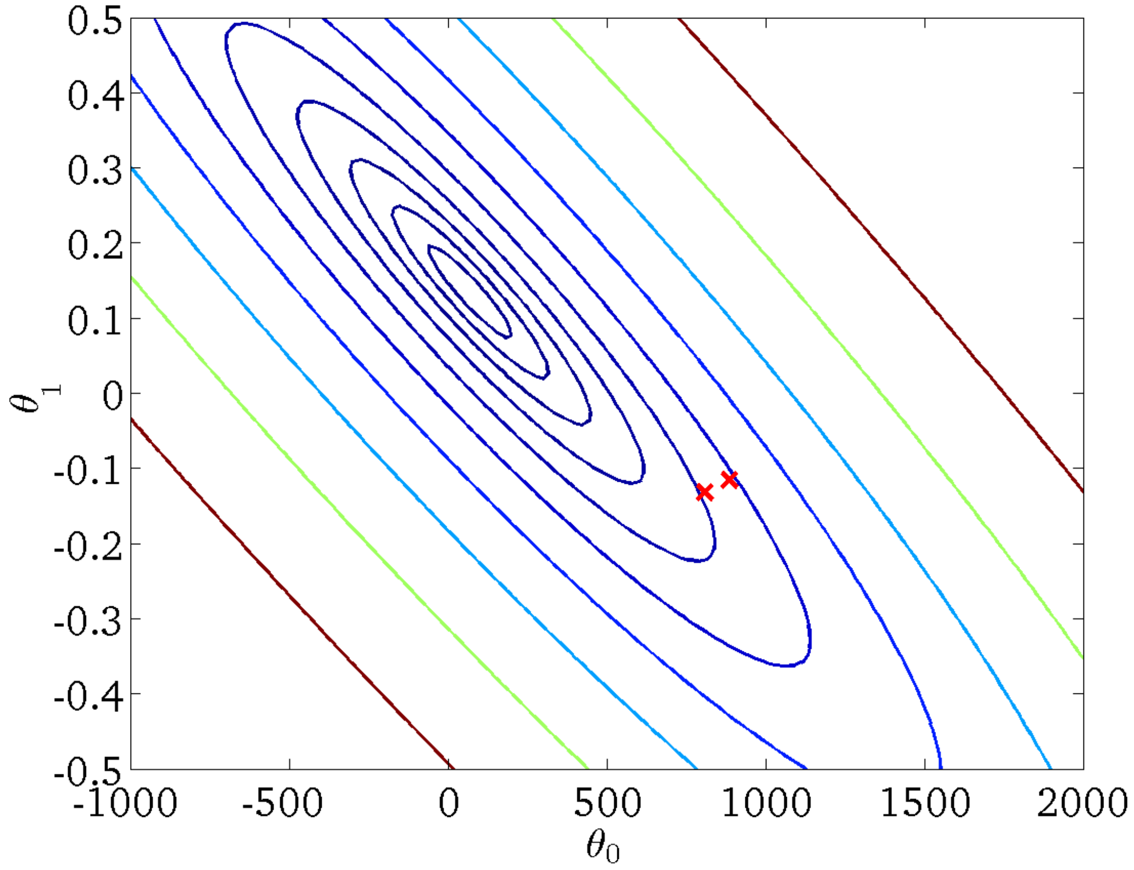
$$h_\theta(x)$$

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



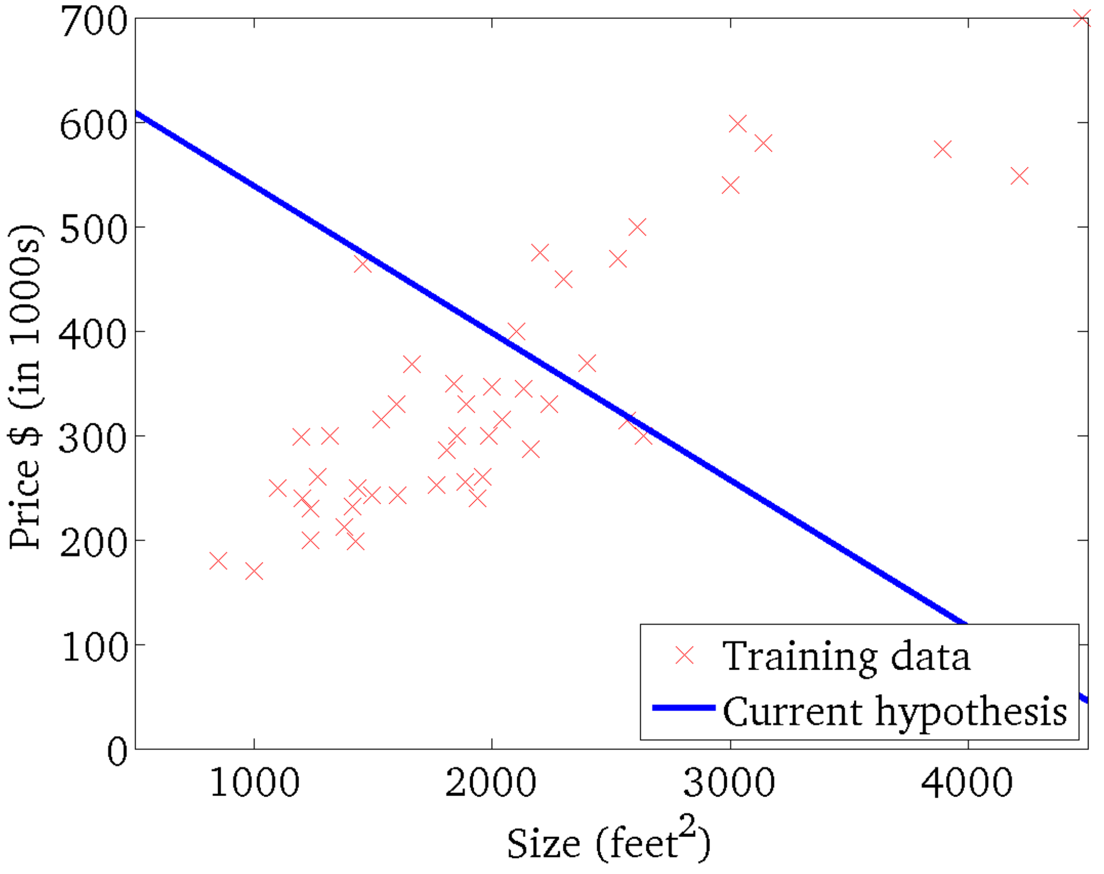
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



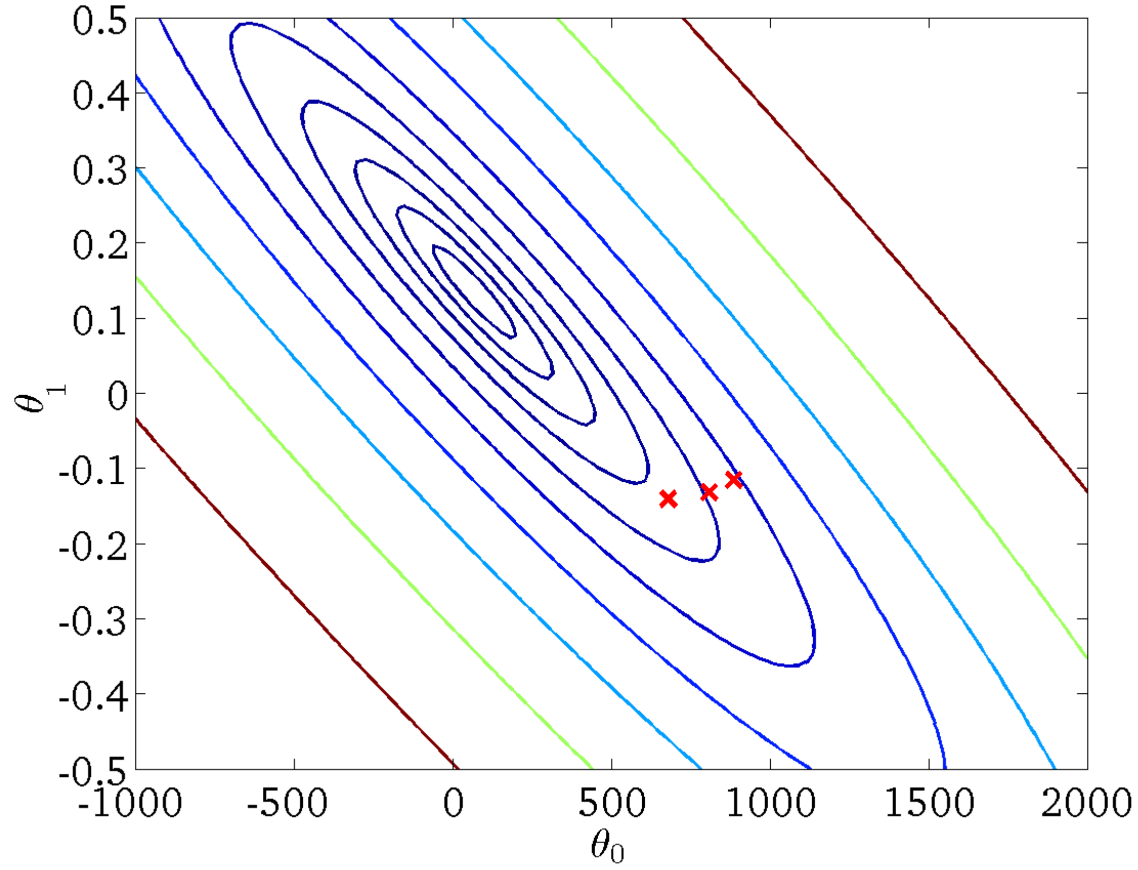
$$h_\theta(x)$$

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



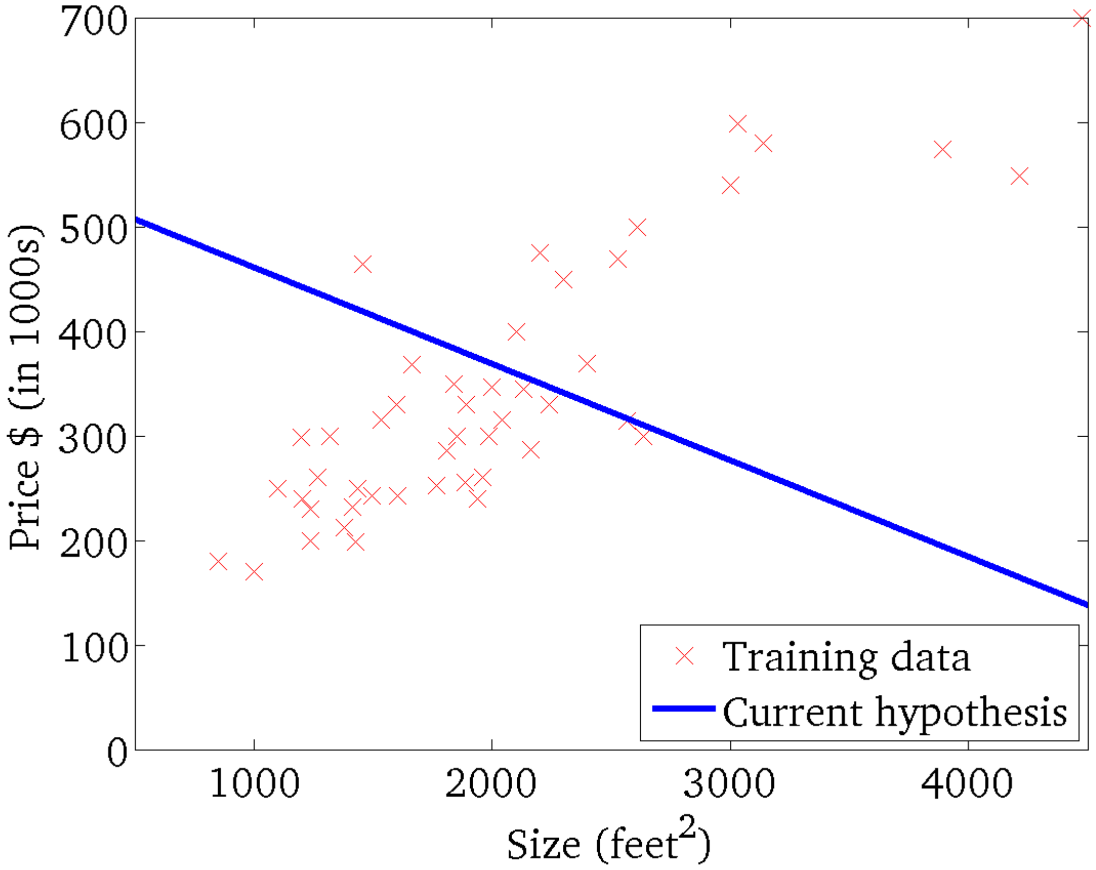
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



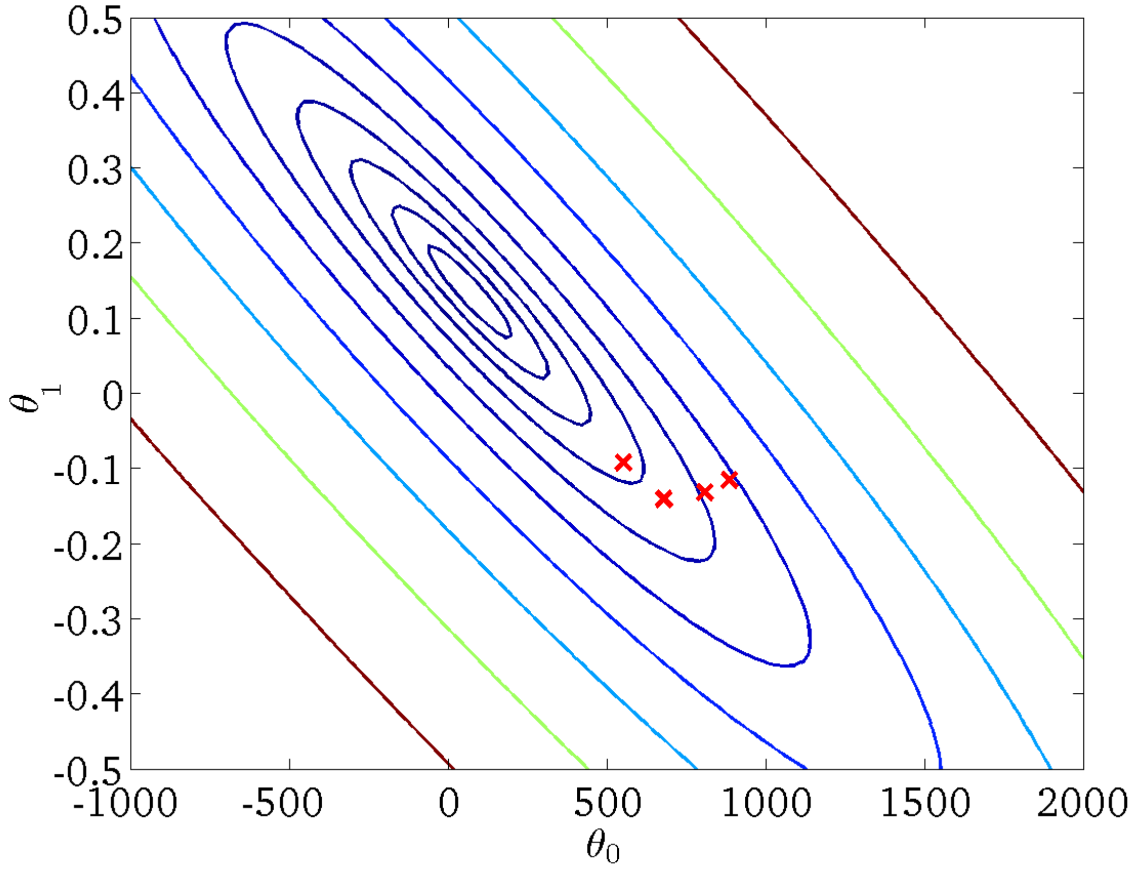
$$h_\theta(x)$$

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



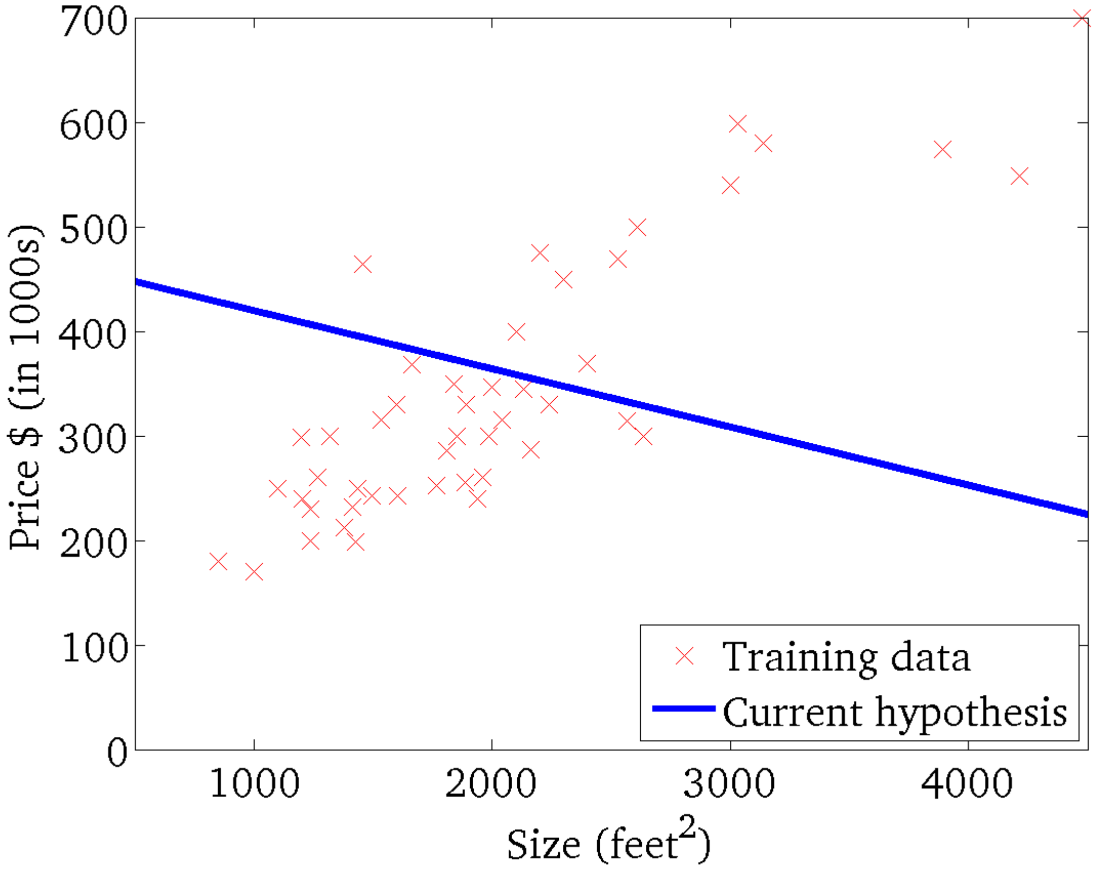
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



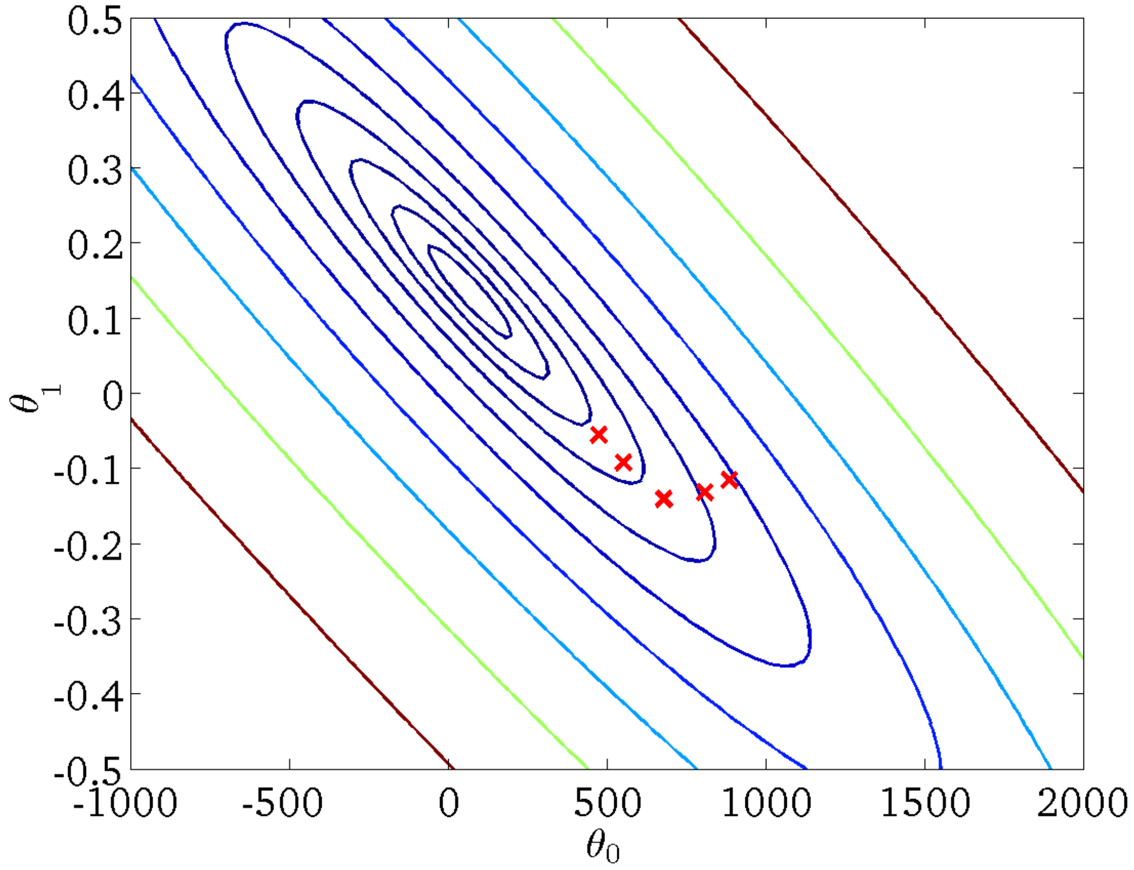
$$h_\theta(x)$$

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



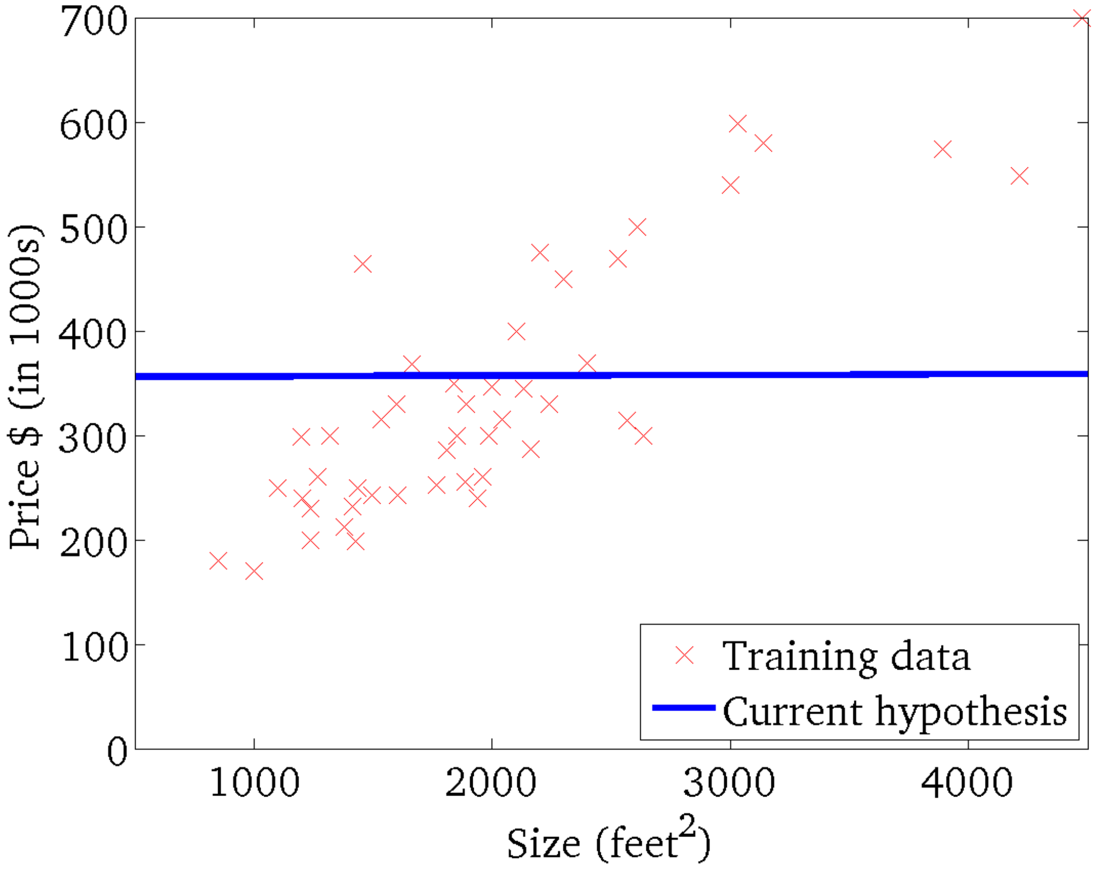
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



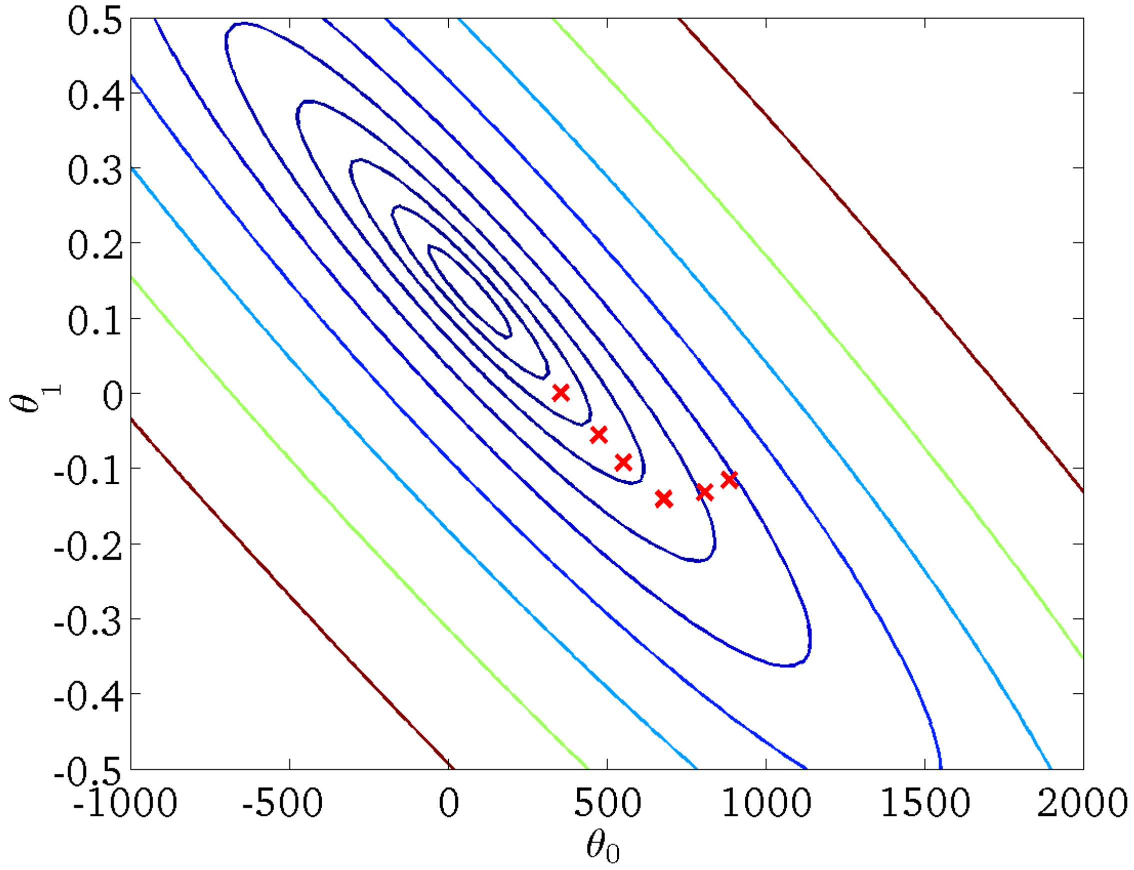
$h_\theta(x)$

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



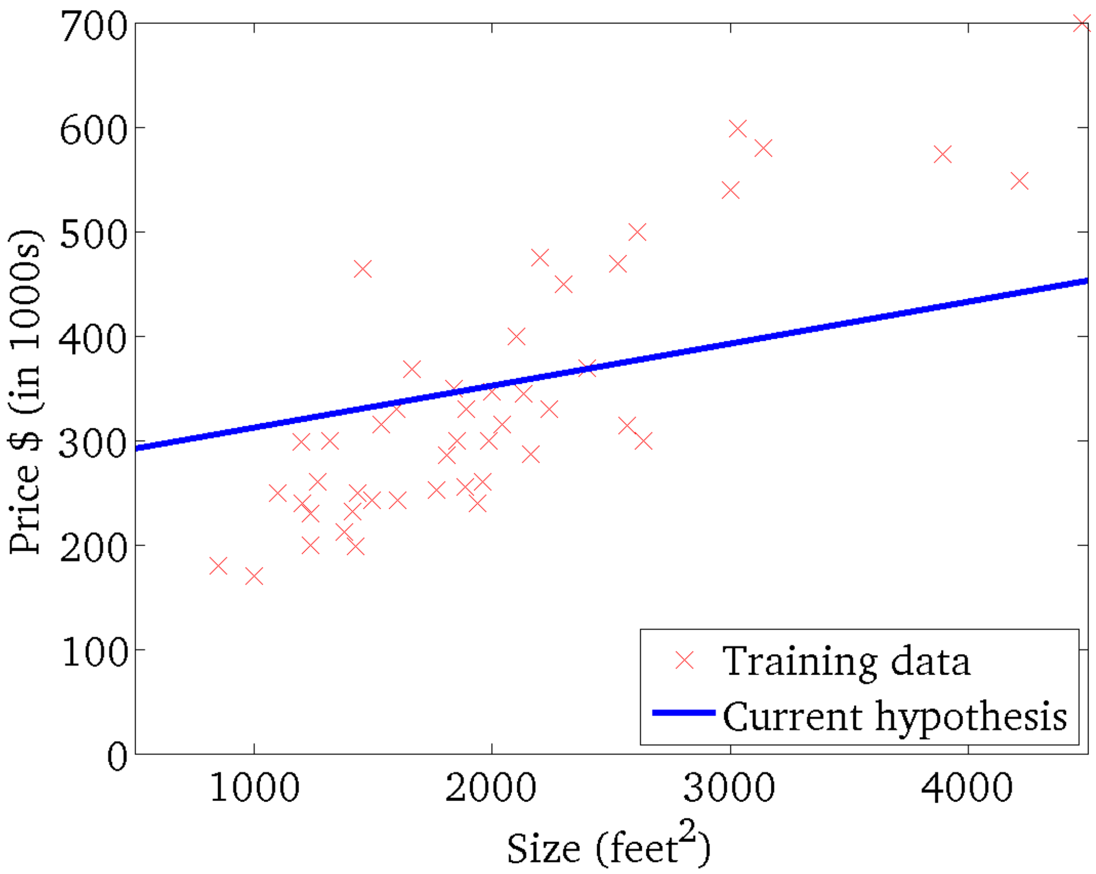
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



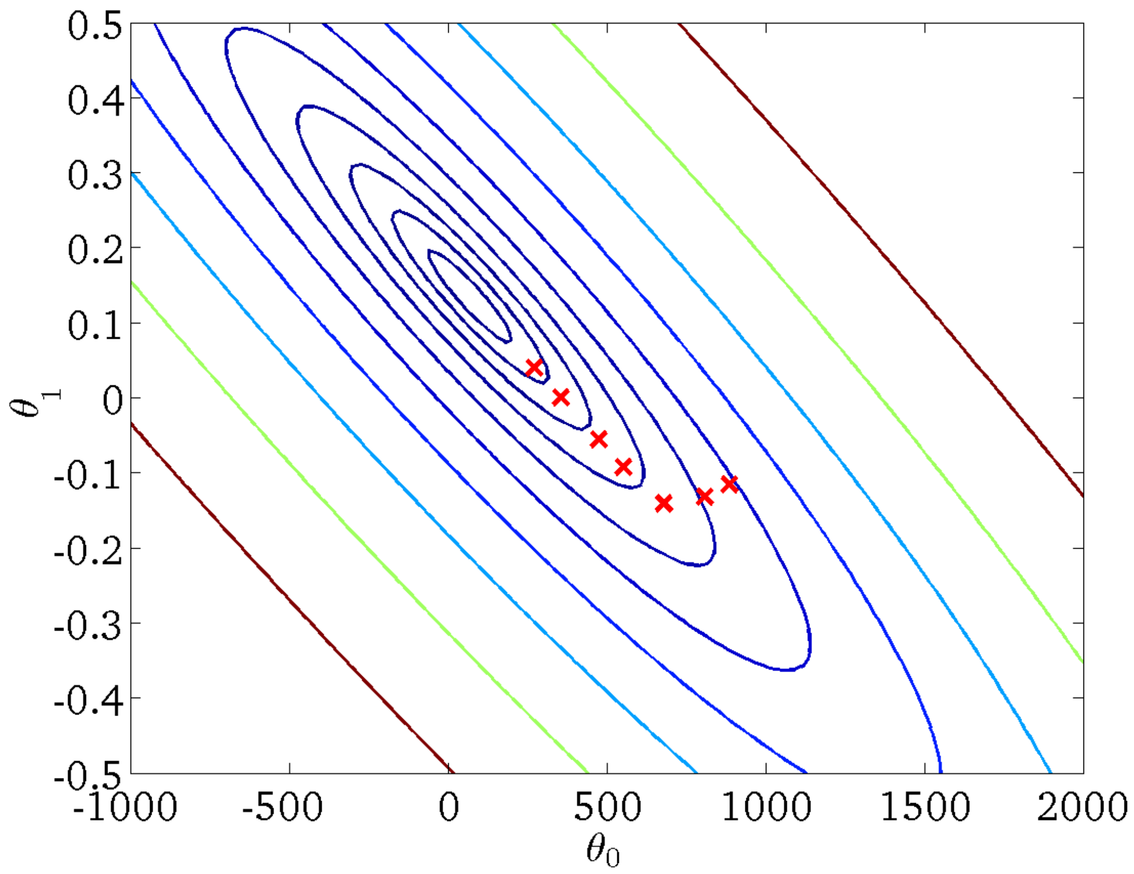
$h_\theta(x)$

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



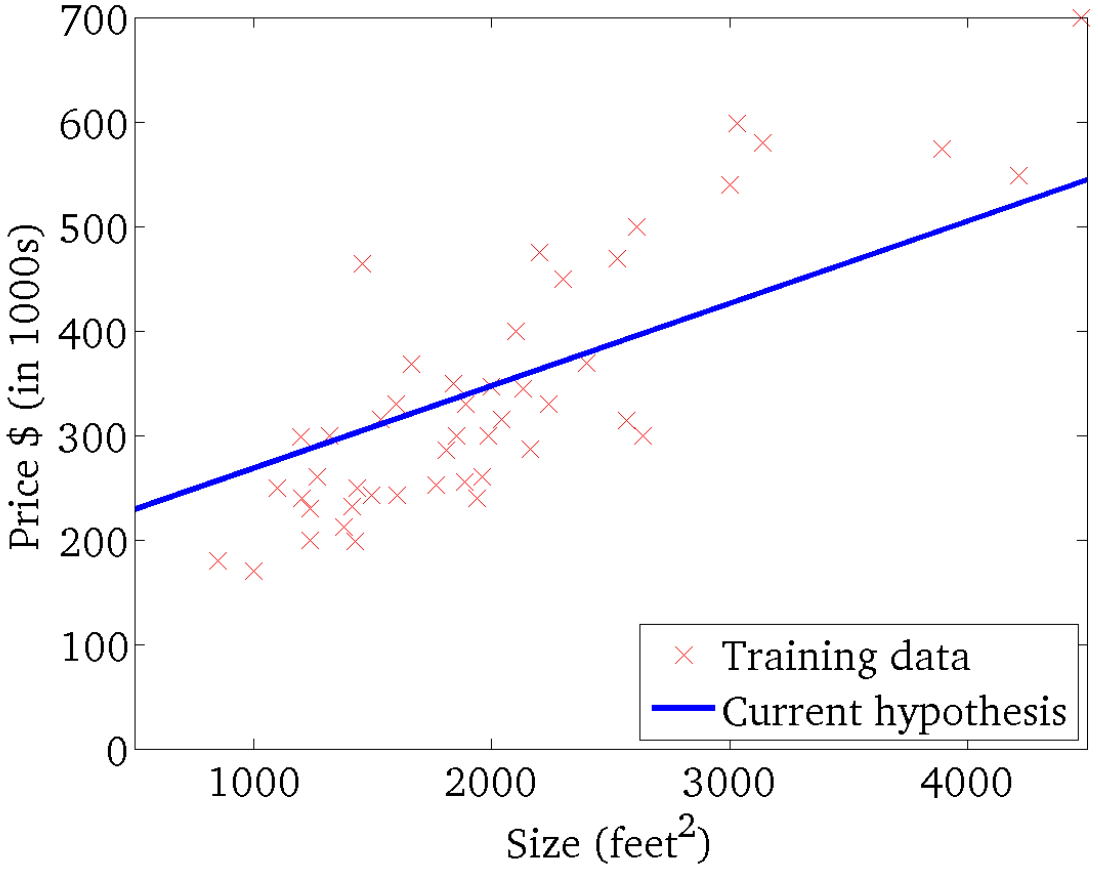
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



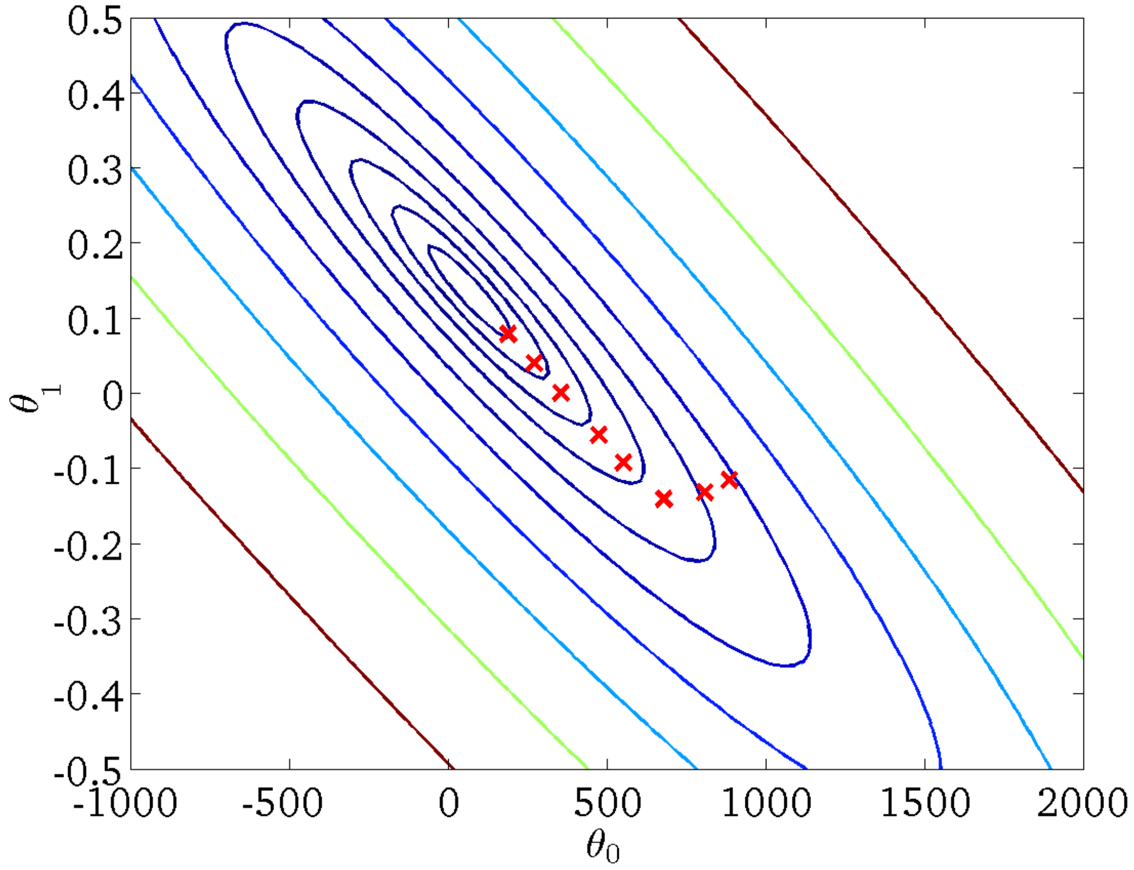
$$h_\theta(x)$$

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



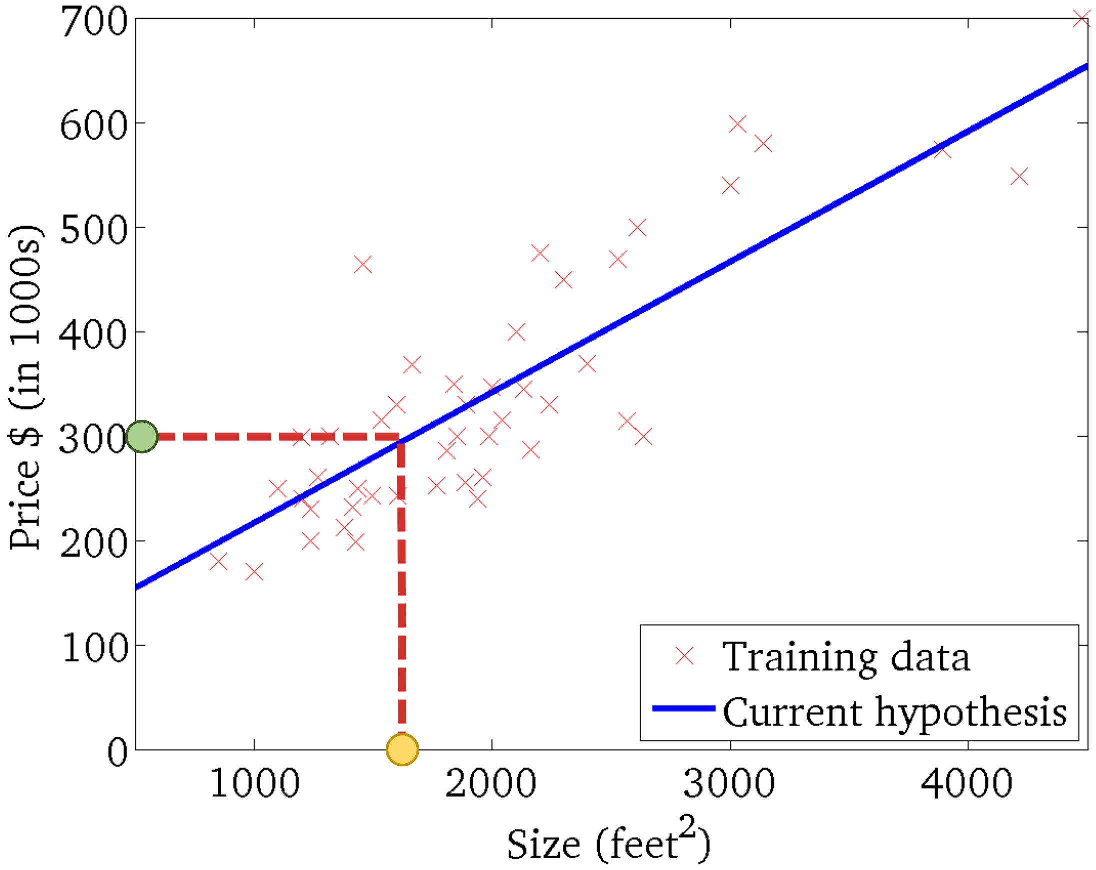
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



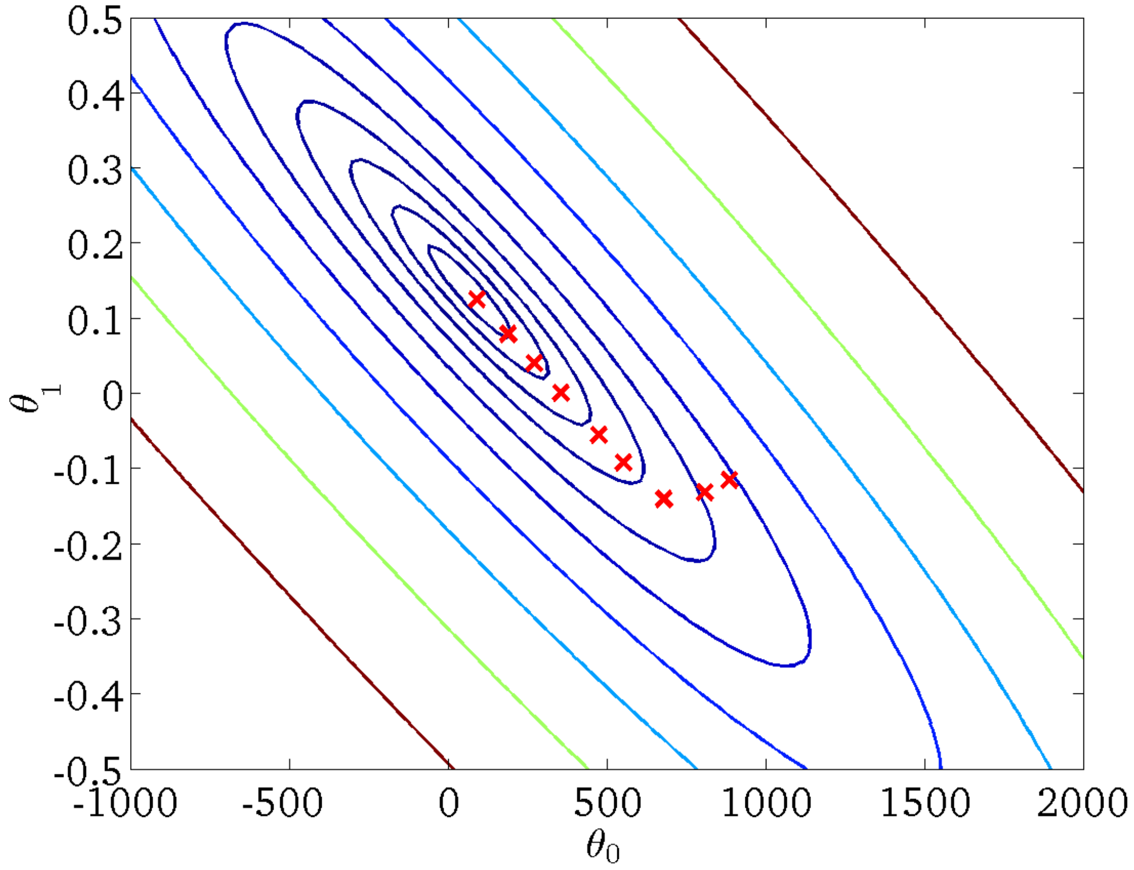
$$h_\theta(x)$$

(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)



Why “Linear Regression” and not “Affine Regression”?

Reference

- CS229: Machine Learning, https://cs229.stanford.edu/
- Machine Learning, https://www.coursera.org/learn/machine-learning