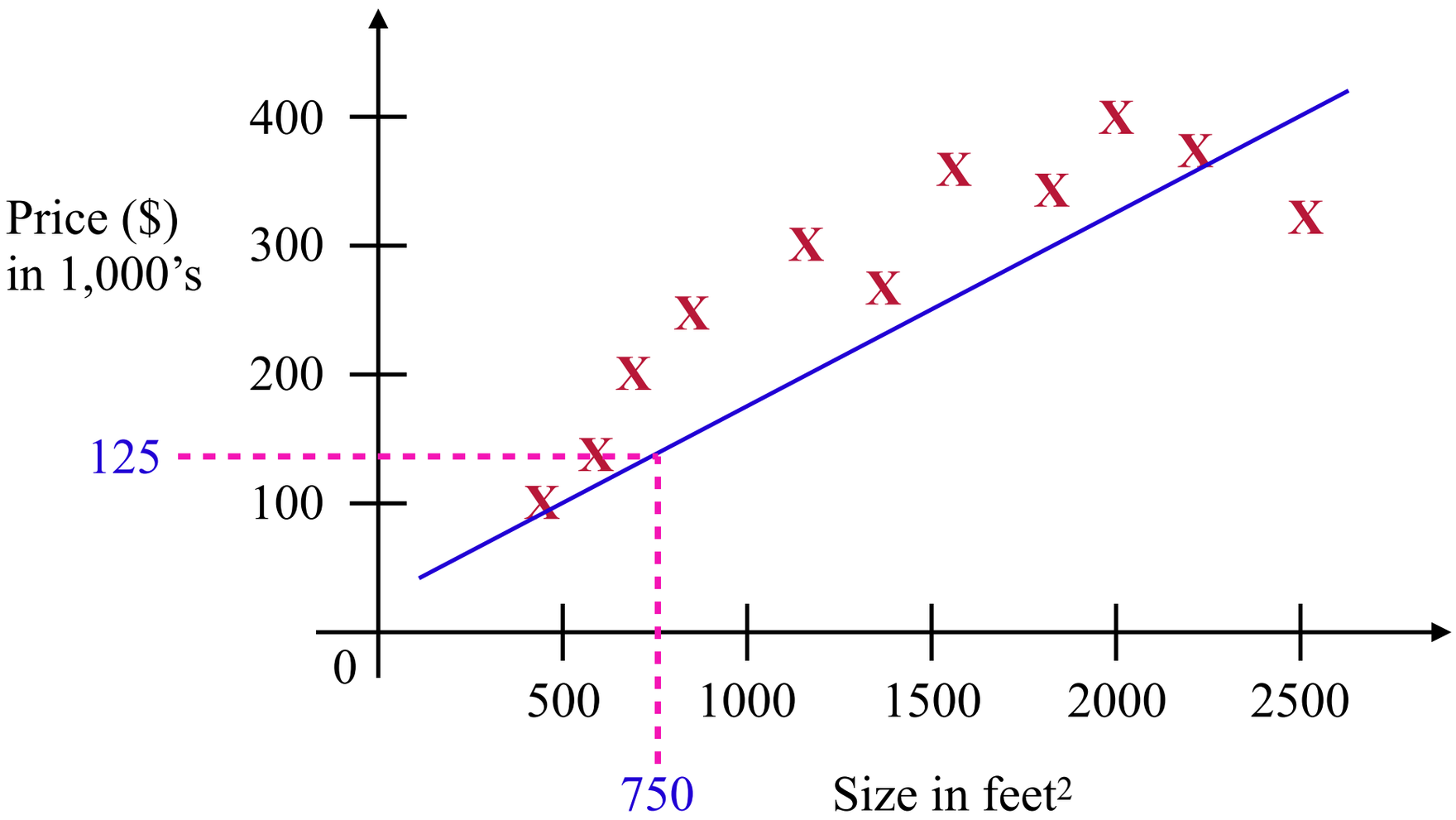
Machine Learning
1.2. Model and Cost Function

Teeradaj Racharak (เอ็กซ์)
r.teeradaj@gmail.com

Linear Regression with 1 variable

Intuition (Reviewed)

Housing price prediction



Supervised Learning

Given the 'right answer' for each example in the data

Regression Problem

Predict real-valued output

Intuition (Formally)

Training set of housing prices

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 / feature
- **y**'s : 'output' variable / 'target' variable
- (x, y) : one training example
- $(x^{(i)}, y^{(i)})$: ith training example

Intuition (Formally)

Training set of housing prices

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 / feature
- **y**'s : 'output' variable / 'target' variable
- (x, y) : one training example
- $(x^{(i)}, y^{(i)})$: ith training example

Example

$m = 4$ $\quad x^{(2)} = 1416$

$x^{(1)} = 2104$ $\quad y^{(1)} = 460$

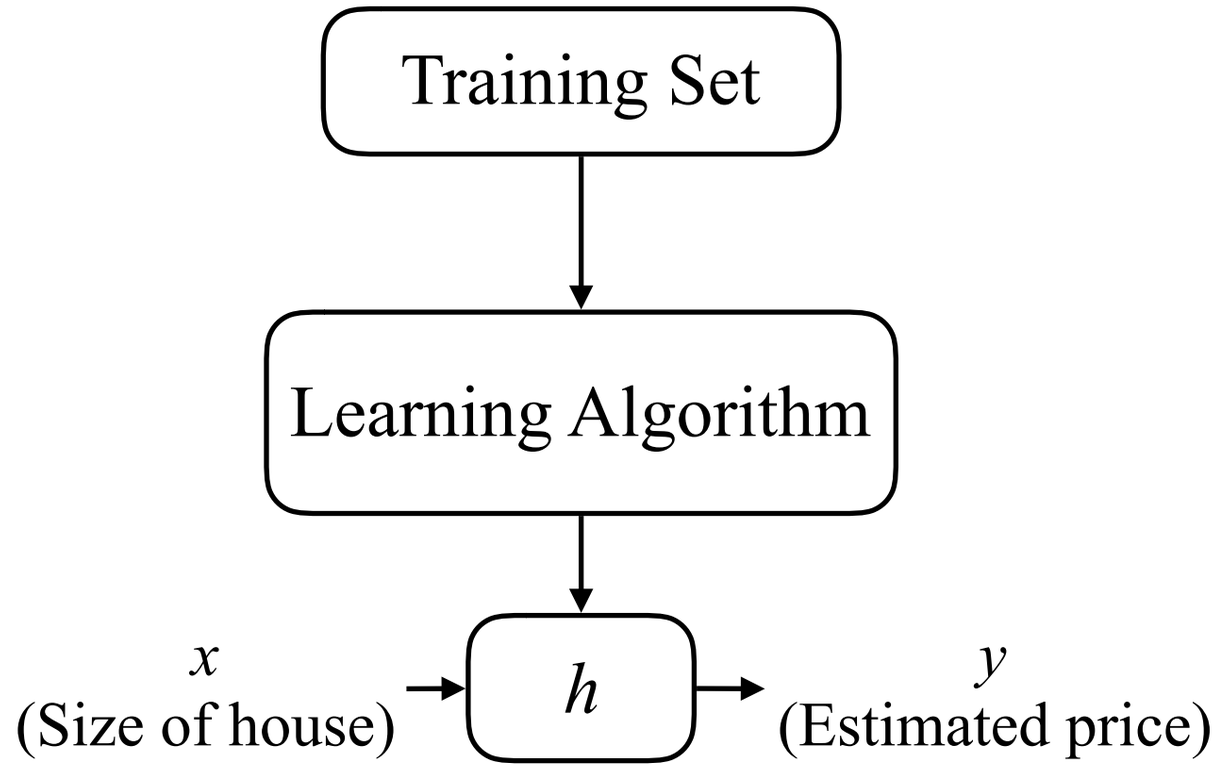
Question

- Consider the training set shown below. ($x^{(i)}$, $y^{(i)}$) is the ith training example. What is $y^{(3)}$?

Training set of housing prices

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

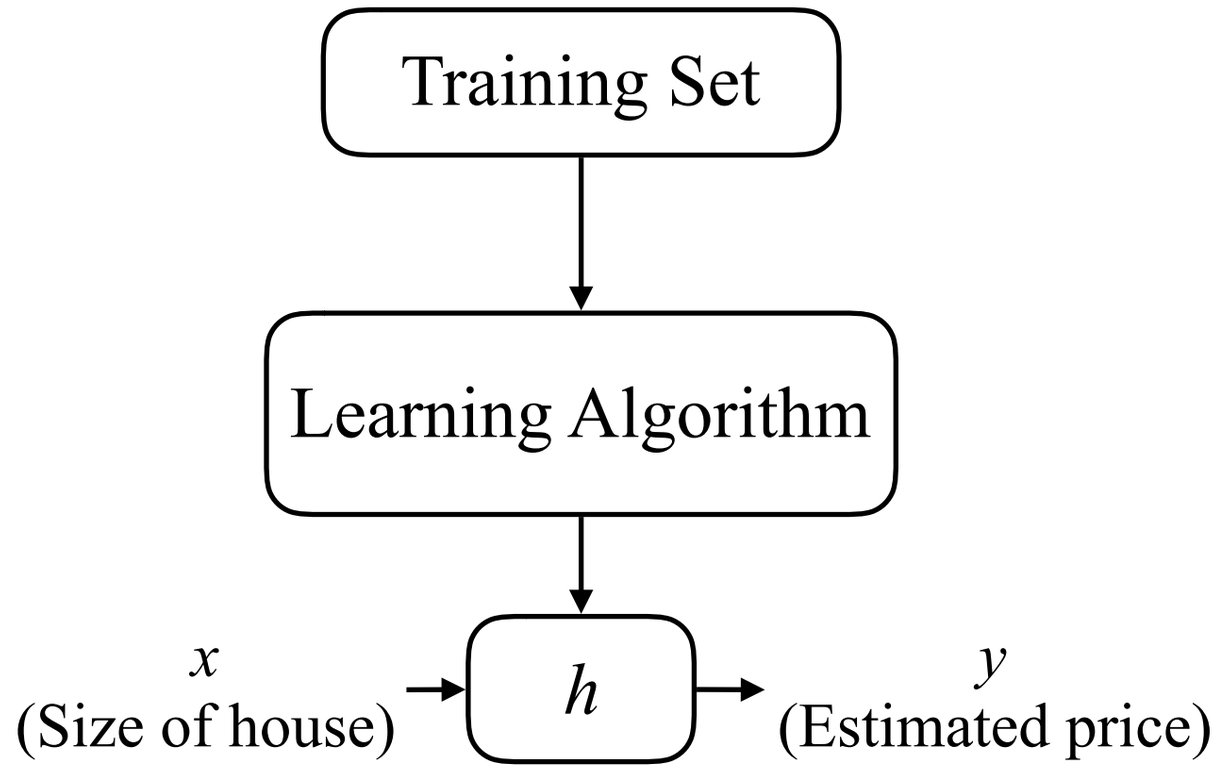
Intuition (Algorithm)



h (hypothesis function) maps from x's to y's

$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

Intuition (Algorithm)

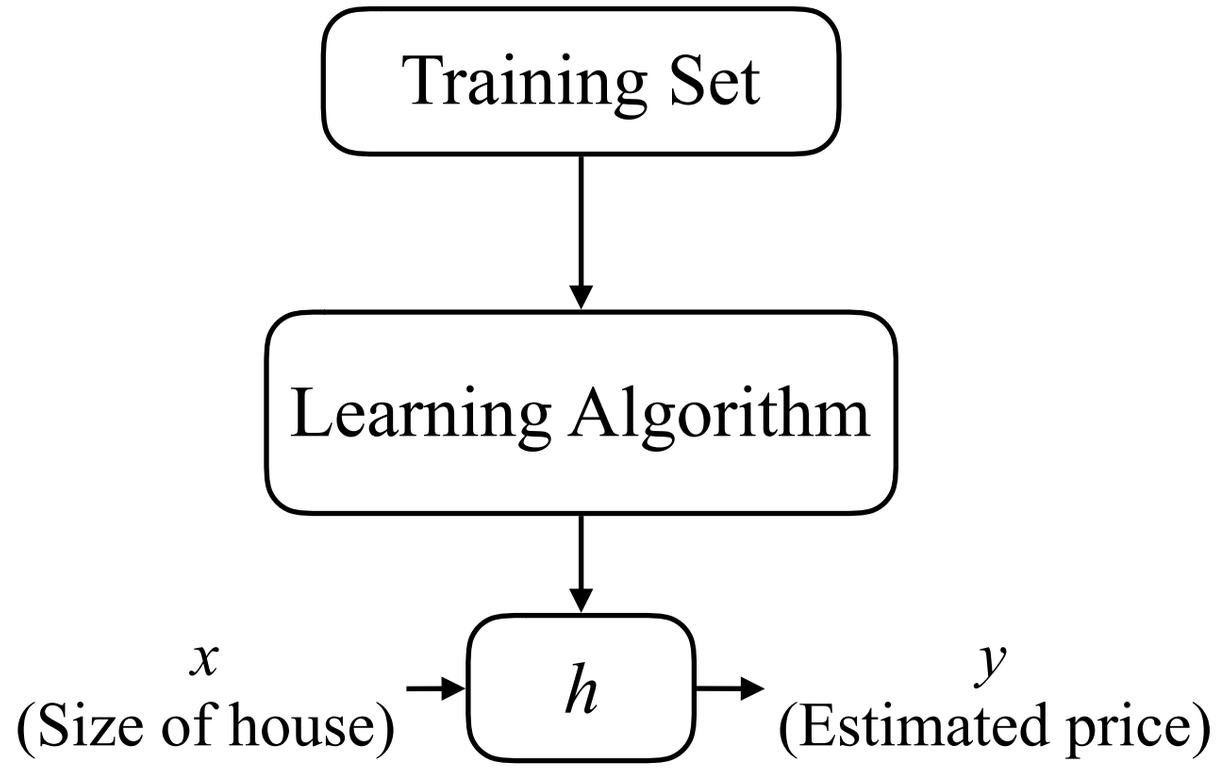


h (hypothesis function) maps from *x*'s to *y*'s

$$h : \mathcal{X} \to \mathcal{Y}$$

How do we represent *h*?

Intuition (Algorithm)



h (hypothesis function) maps from *x*'s to *y*'s

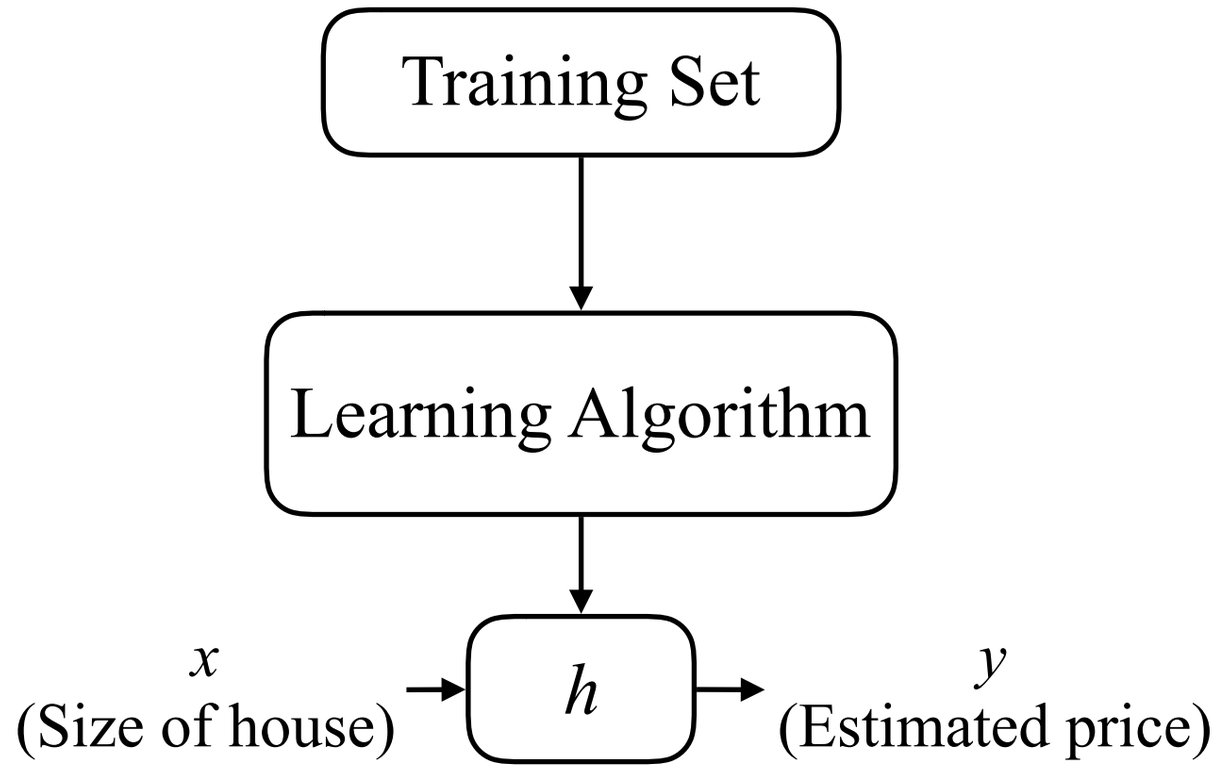
$$h : \mathcal{X} \rightarrow \mathcal{Y}$$

How do we represent *h*?

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

(shorthand: $h(x)$)

Intuition (Algorithm)



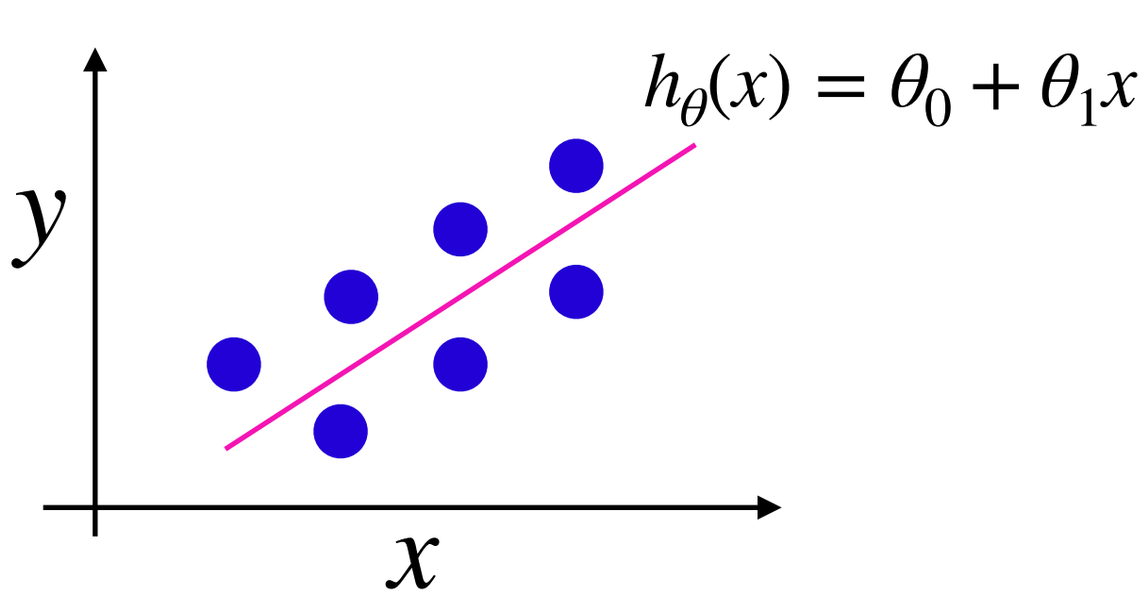
h (hypothesis function) maps from *x*'s to *y*'s

$$h : \mathcal{X} \to \mathcal{Y}$$

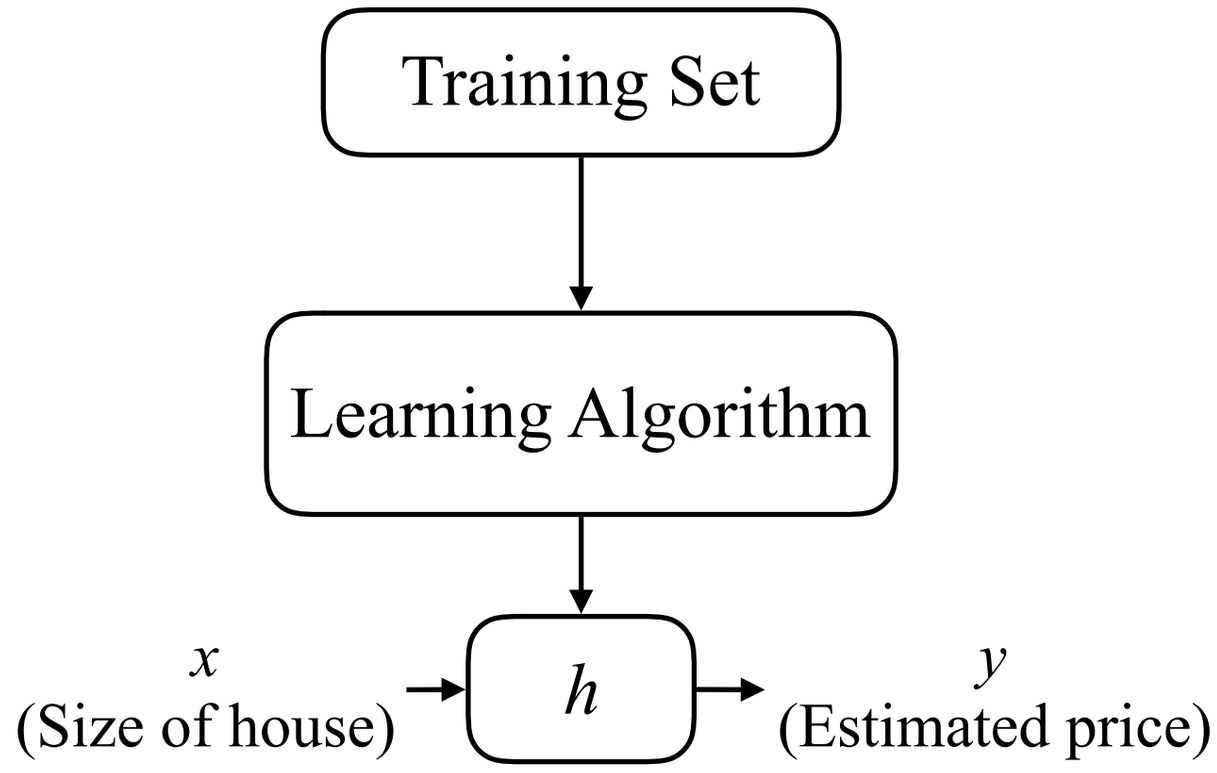
How do we represent *h*?

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

(shorthand: $h(x)$)



Intuition (Algorithm)



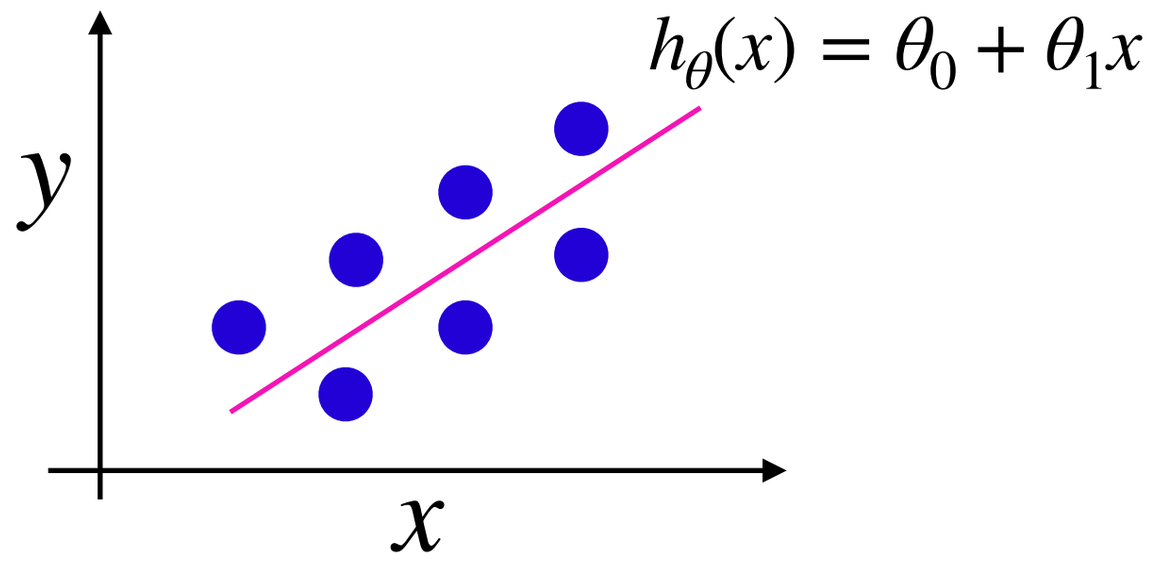
h (hypothesis function) maps from *x*'s to *y*'s

$$h : \mathcal{X} \to \mathcal{Y}$$

How do we represent *h*?

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

(shorthand: $h(x)$)



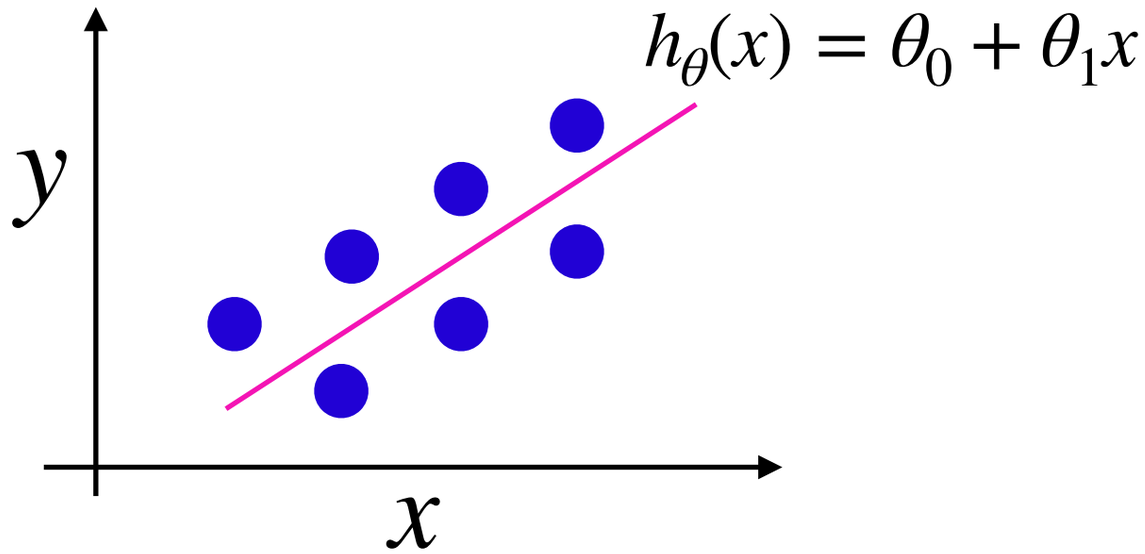
'Linear regression with one variable'

'Univariate linear regression'

Recap

- Given a training set, we want to find a function $h : \mathcal{X} \to \mathcal{Y}$ so that $h(x)$ is a 'good' predictor of the corresponding value of y.

- Question: how to decide if an upcoming h is a good predictor?



Cost Function

Univariate Linear Regression

Training set of housing prices

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

Let $m = 47$

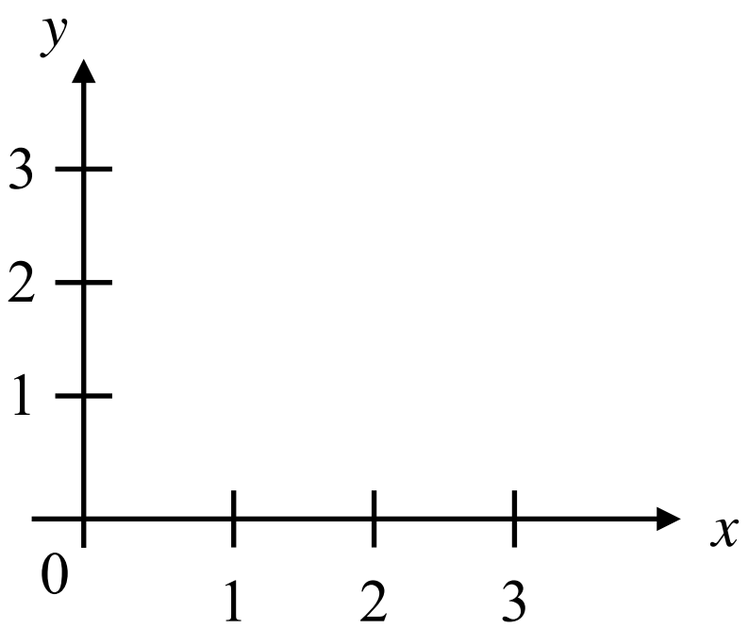
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_i's : Parameters or weights

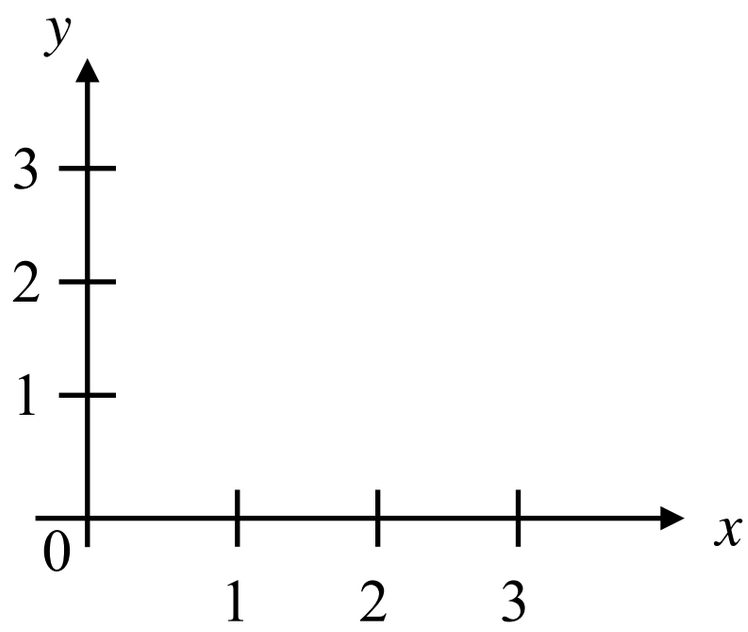
How to choose θ_i's ?

Univariate Linear Regression

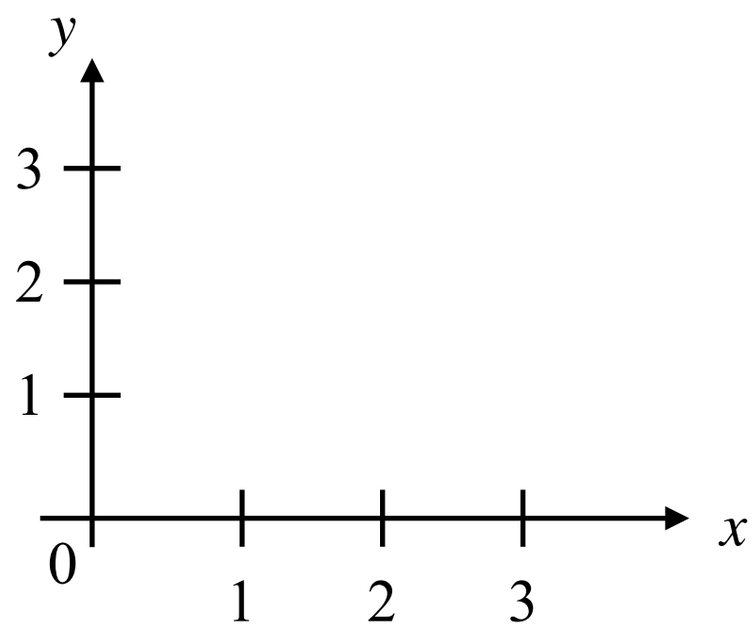
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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



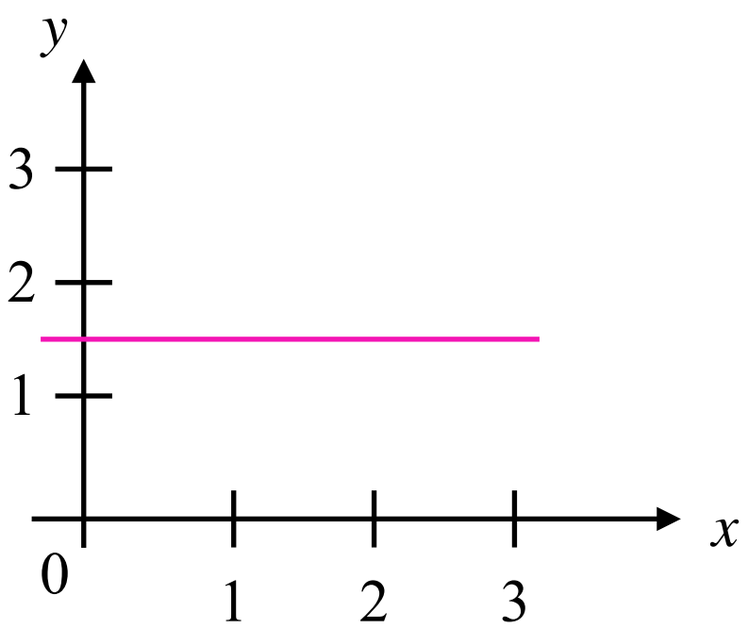
$\theta_0 = 0$
$\theta_1 = 0.5$



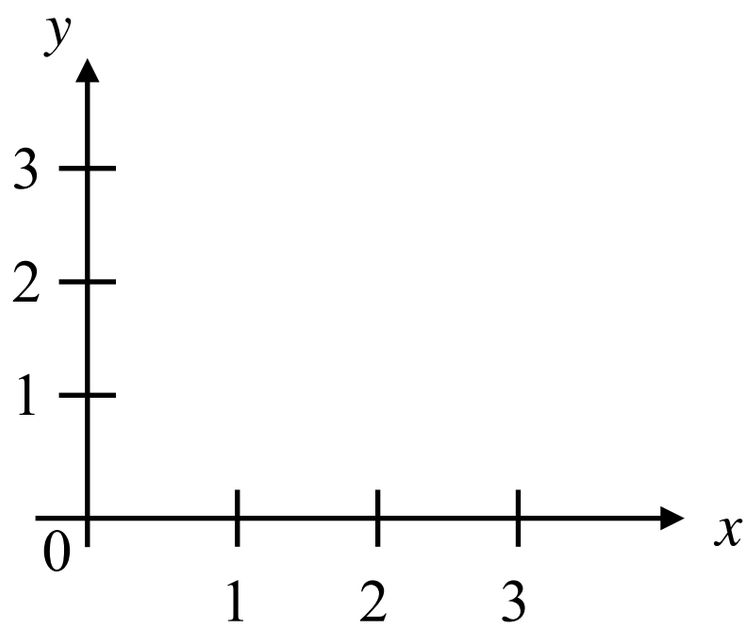
$\theta_0 = 1$
$\theta_1 = 0.5$

Univariate Linear Regression

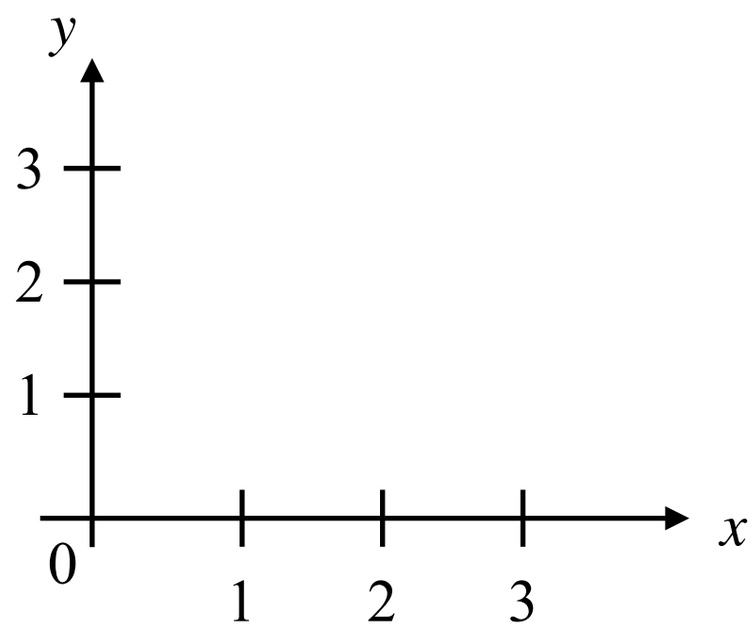
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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



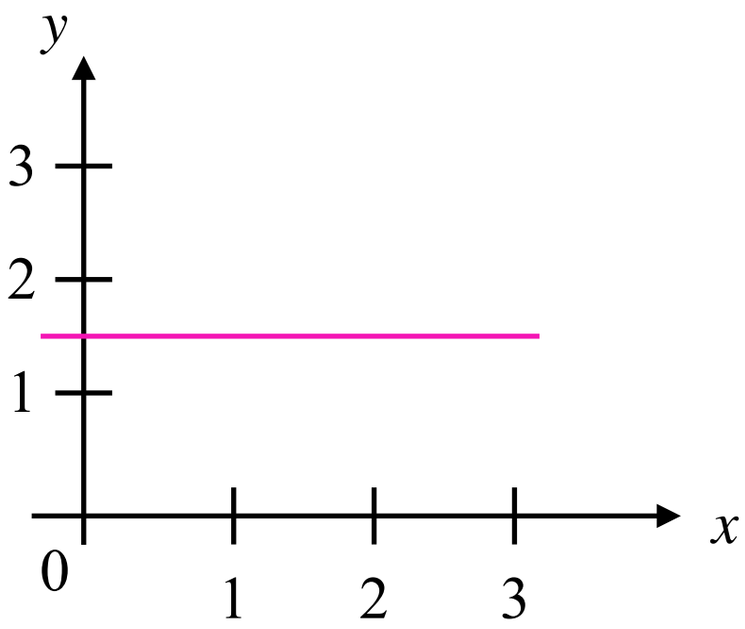
$\theta_0 = 0$
$\theta_1 = 0.5$



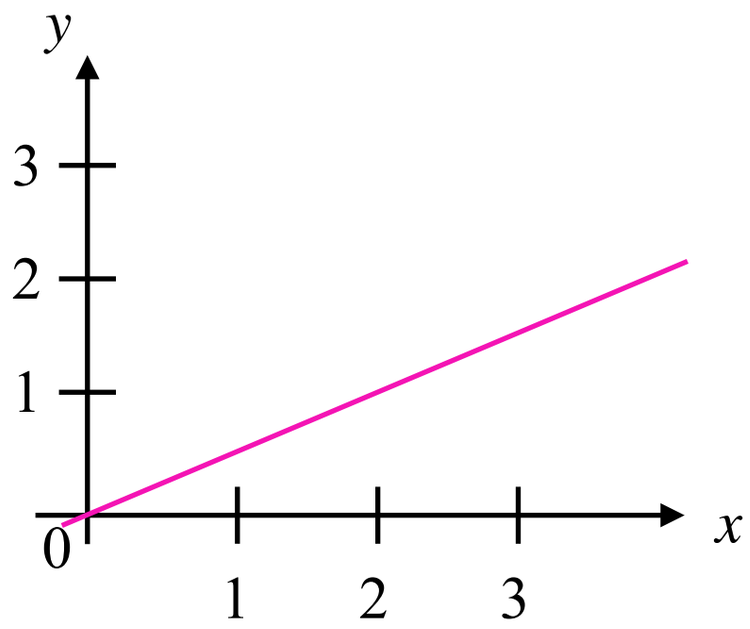
$\theta_0 = 1$
$\theta_1 = 0.5$

Univariate Linear Regression

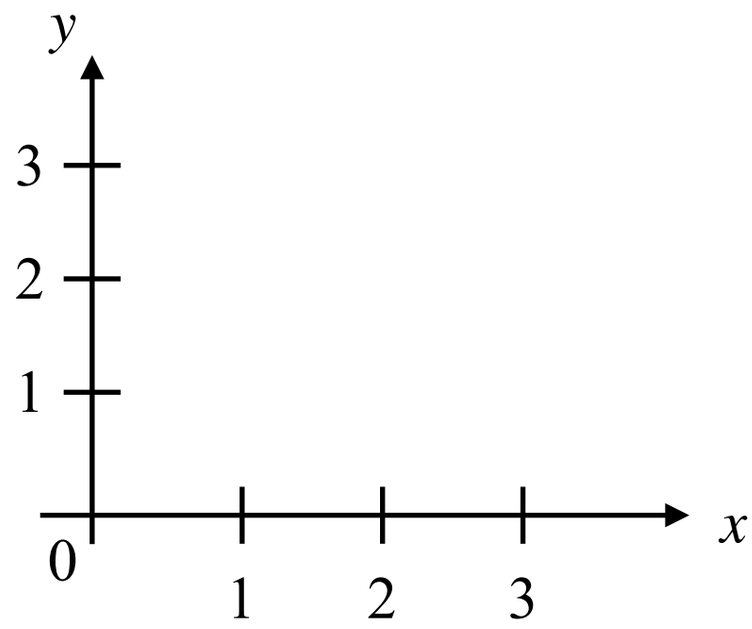
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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



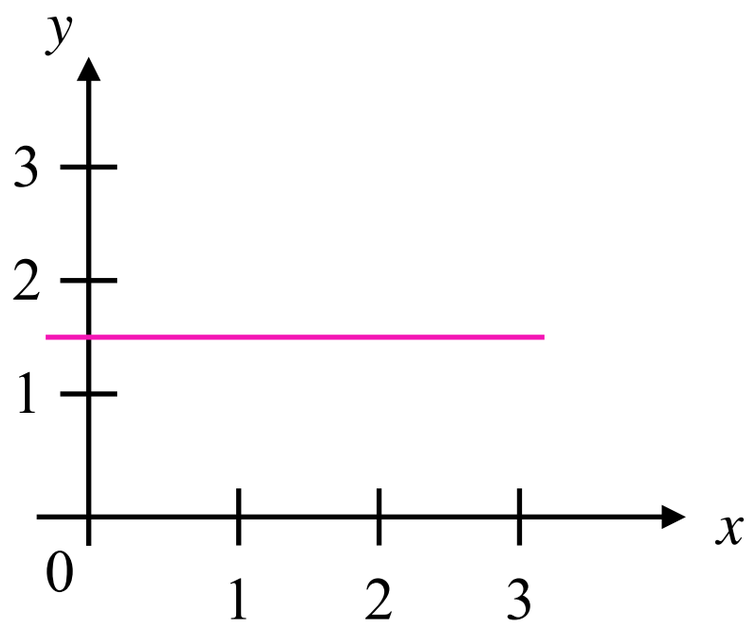
$\theta_0 = 0$
$\theta_1 = 0.5$



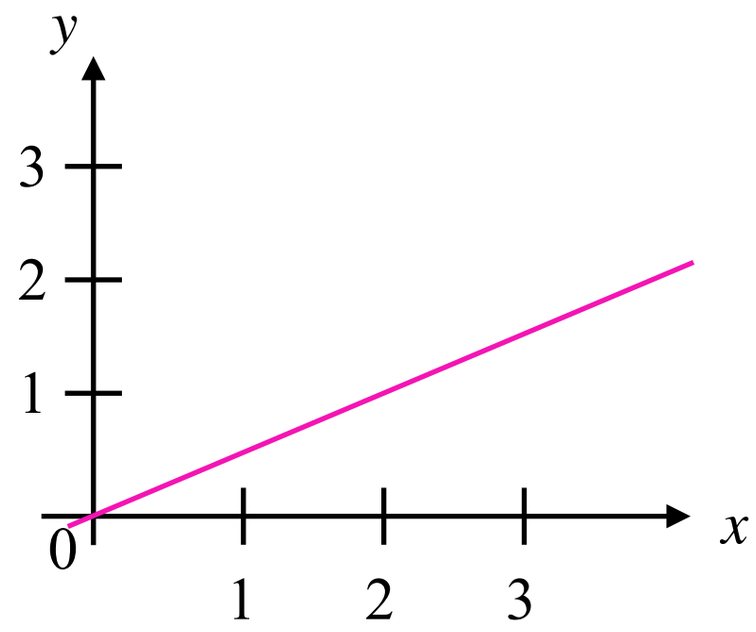
$\theta_0 = 1$
$\theta_1 = 0.5$

Univariate Linear Regression

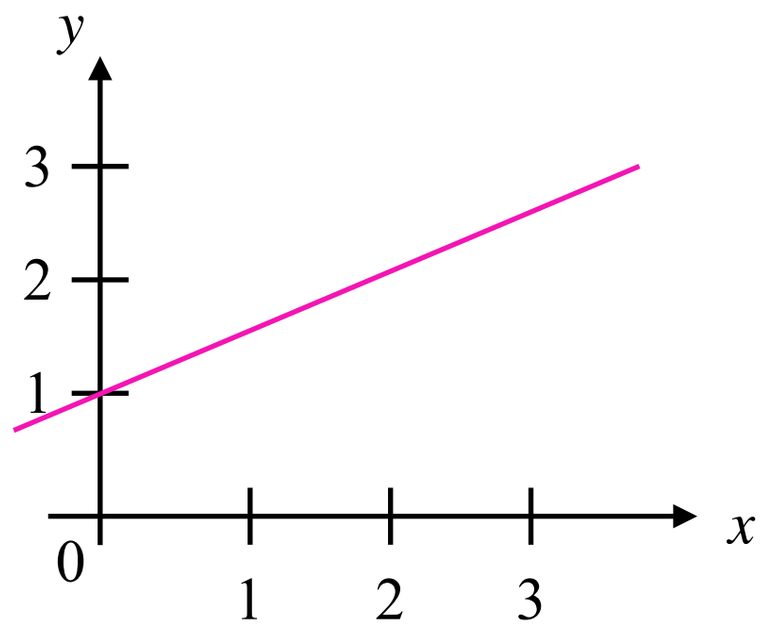
$$h_\theta(x) = \theta_0 + \theta_1 x$$



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



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



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

Question

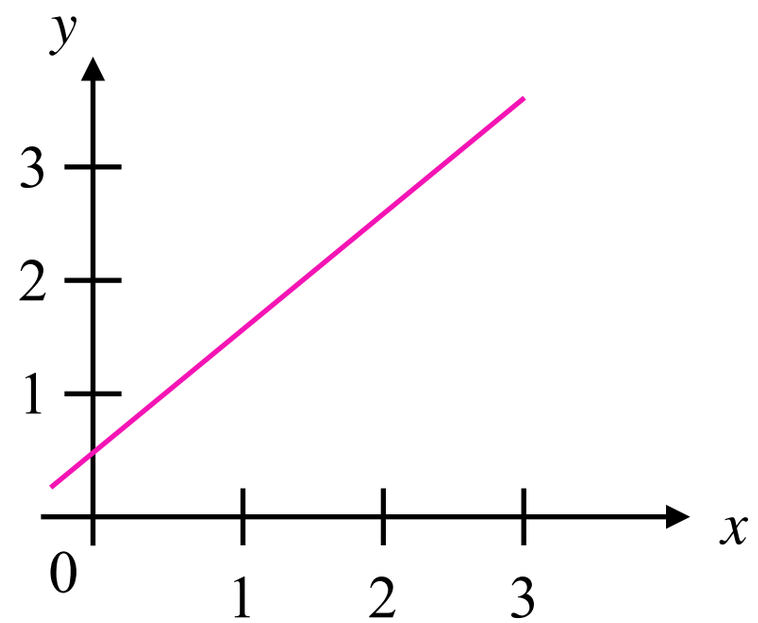
- Consider the plot below, what are θ_0 and θ_1 ?

(i) $\theta_0 = 0$, $\theta_1 = 1$

(ii) $\theta_0 = 0.5$, $\theta_1 = 1$

(iii) $\theta_0 = 1$, $\theta_1 = 0.5$

(iv) $\theta_0 = 1$, $\theta_1 = 1$



Question

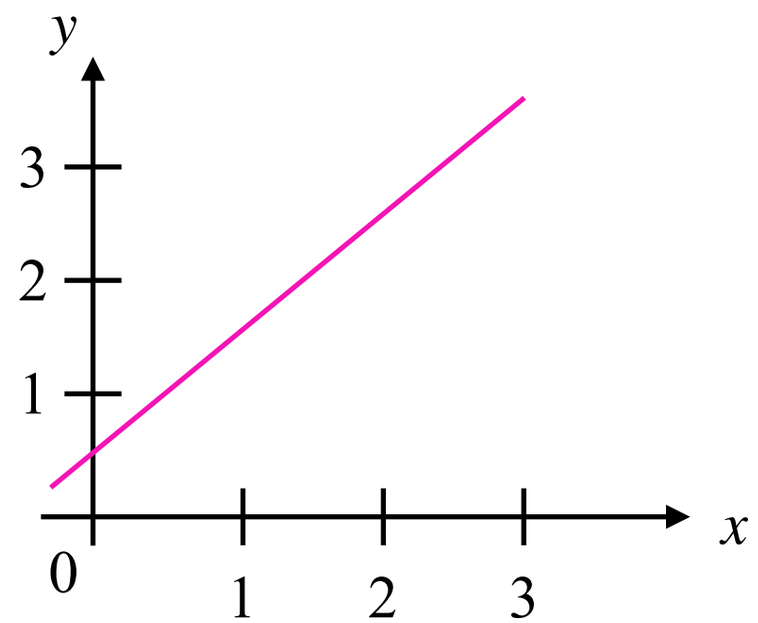
- Consider the plot below, what are θ_0 and θ_1 ?

(i) $\theta_0 = 0$, $\theta_1 = 1$

(ii) $\theta_0 = 0.5$, $\theta_1 = 1$

(iii) $\theta_0 = 1$, $\theta_1 = 0.5$

(iv) $\theta_0 = 1$, $\theta_1 = 1$

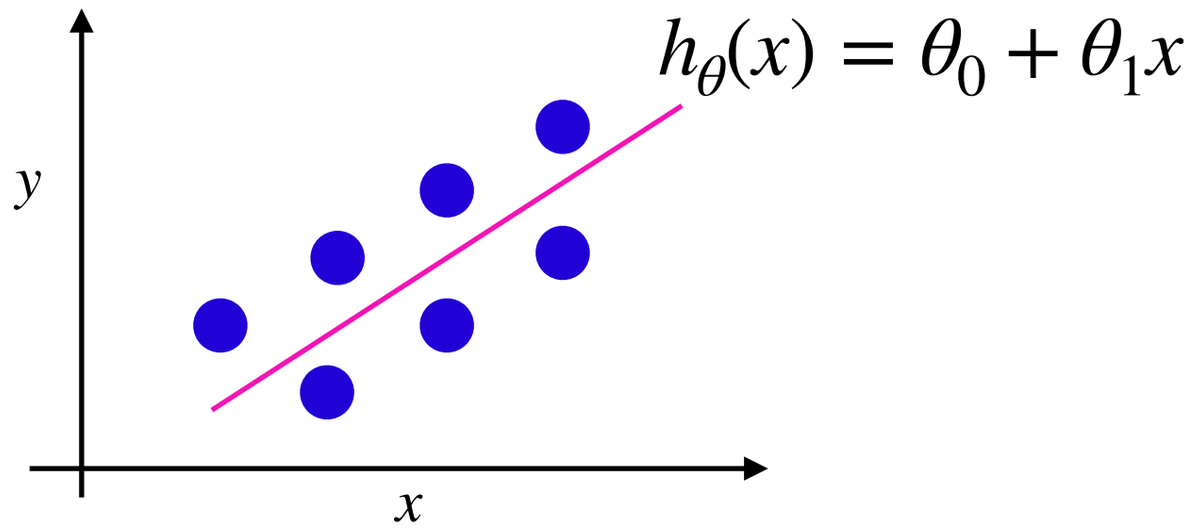


Cost Function

- Idea: To find a good h, we need to find good θ_i
- In linear regression, we require a cost function that assigns a large cost to 'bad' predictions $h(x)$ and a small cost to 'good' predictions $h(x)$

- Convince yourself that a choice presented next slide gives small value for good h and large values for bad h

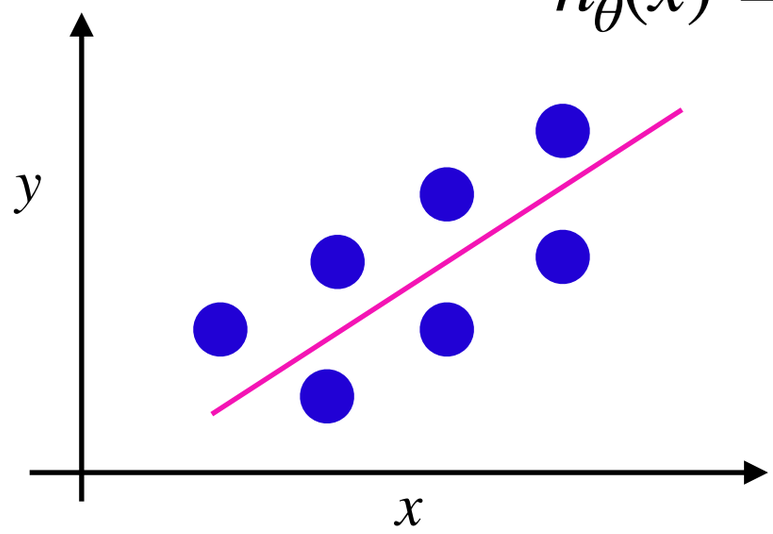
Cost Function (Intuition)



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

Cost Function (Intuition)

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

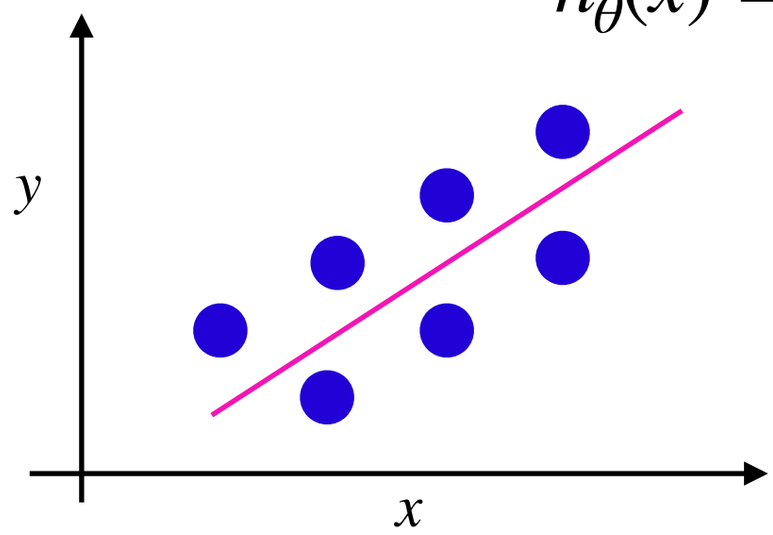


Idea: Choose θ_0 and θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

$$\min_{\theta_0,\theta_1} (h_\theta(x^{(i)}) - y^{(i)})$$

Cost Function (Intuition)

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



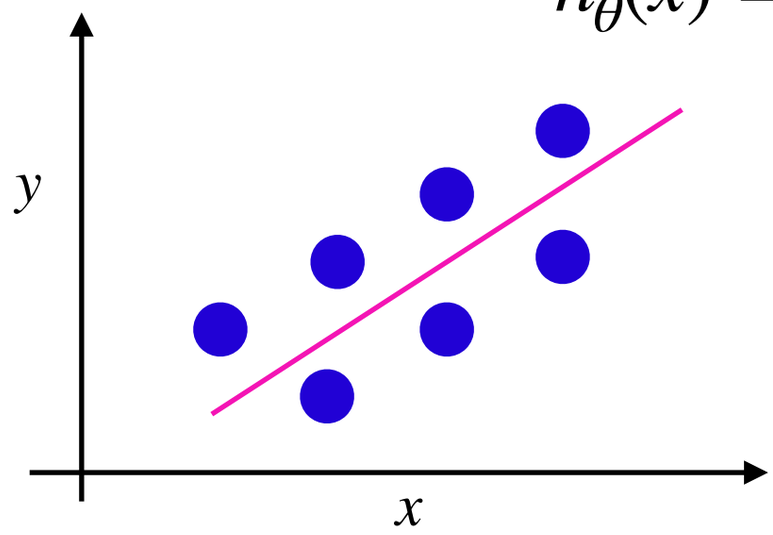
Idea: Choose θ_0 and θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

$$\min_{\theta_0,\theta_1} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\min_{\theta_0,\theta_1} (h_\theta(x^{(i)}) - y^{(i)})^2$$

Cost Function (Intuition)

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



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

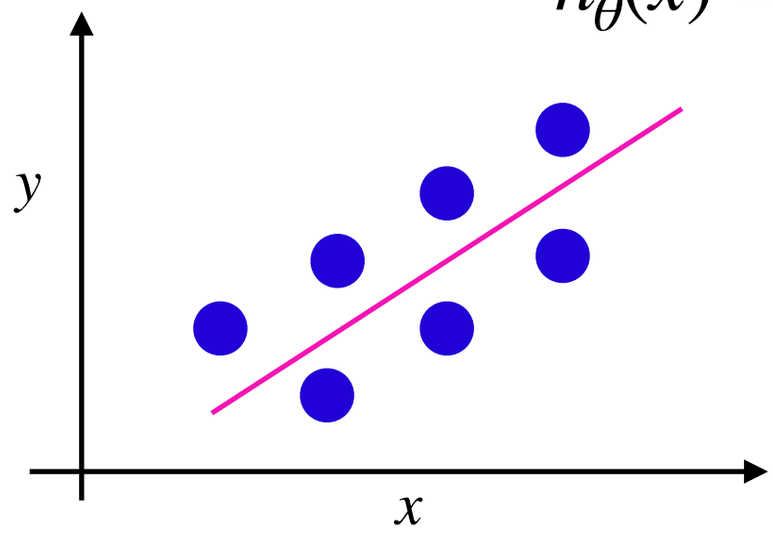
$$\min_{\theta_0,\theta_1} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\min_{\theta_0,\theta_1} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\min_{\theta_0,\theta_1} [\Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2]$$

Cost Function (Intuition)

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



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

$$\min_{\theta_0,\theta_1}(h_\theta(x^{(i)}) - y^{(i)})$$

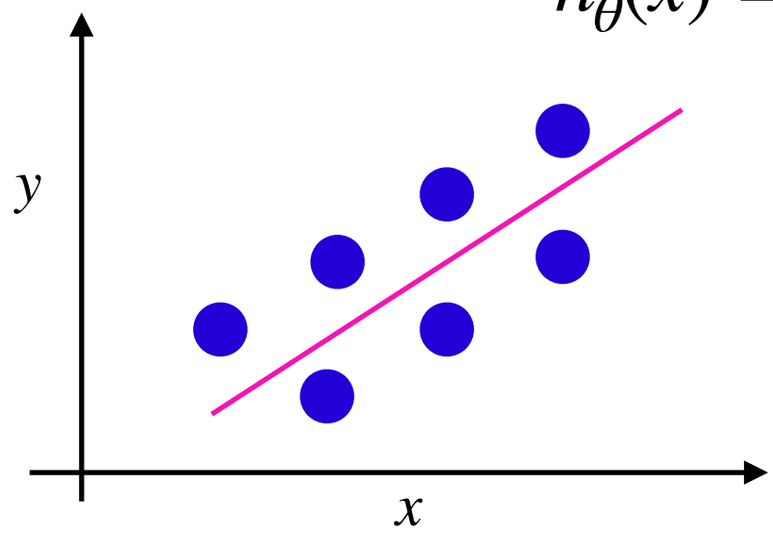
$$\min_{\theta_0,\theta_1}(h_\theta(x^{(i)}) - y^{(i)})^2$$

$$\min_{\theta_0,\theta_1}[\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2]$$

$$\min_{\theta_0,\theta_1}[\frac{1}{2m}\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2]$$

Cost Function (Intuition)

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



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

$$\min_{\theta_0,\theta_1}(h_\theta(x^{(i)}) - y^{(i)})$$

$$\min_{\theta_0,\theta_1}(h_\theta(x^{(i)}) - y^{(i)})^2$$

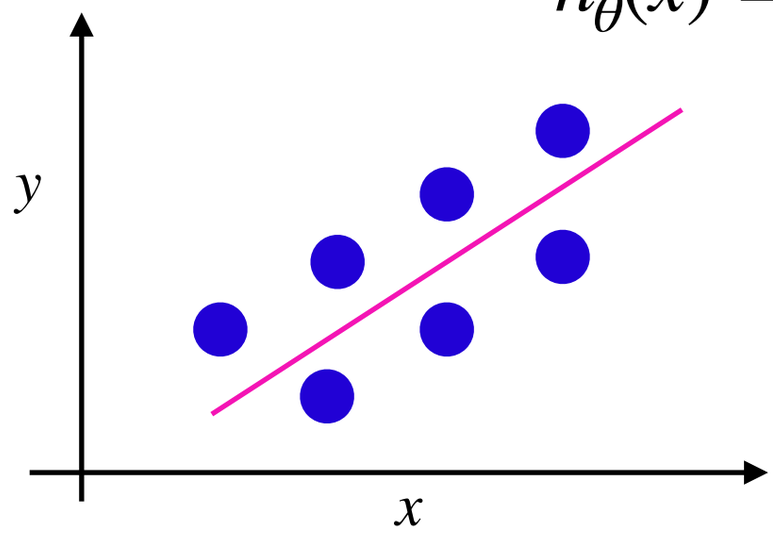
$$\min_{\theta_0,\theta_1}[\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2]$$

$$\min_{\theta_0,\theta_1}[\frac{1}{2m}\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2]$$

Objective function for univariate linear regression

Cost Function (Formally)

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



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

Cost Function :

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

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

Remark : $J(\theta_0, \theta_1)$ is sometimes called *squared error (cost) function*.

Recap

- Hypothesis:

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

- Parameters:

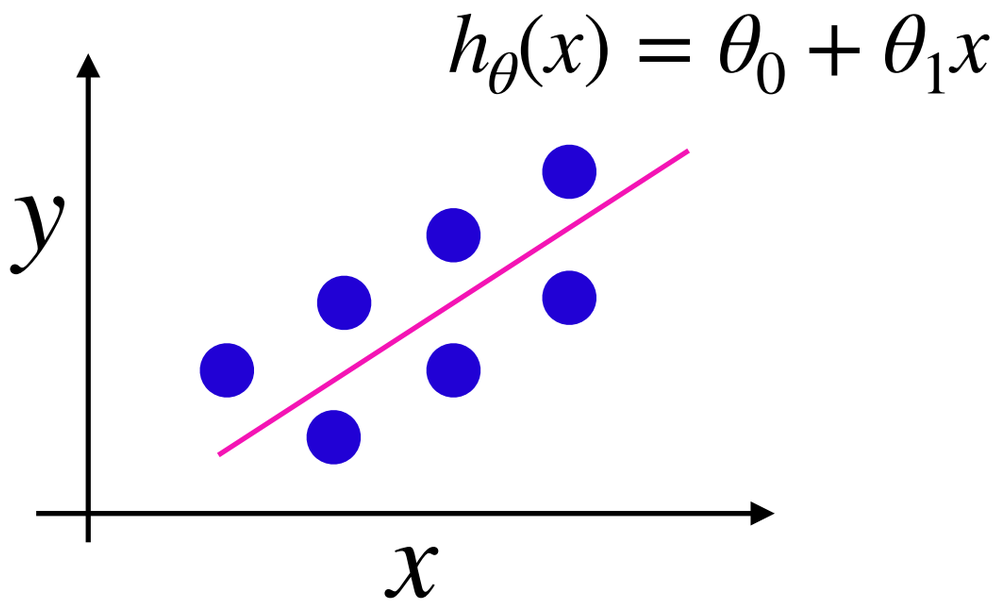
$$\theta_0, \theta_1$$

- Cost Function:

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

- Goal (*i.e.* 'Optimization Objective'):

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



How to minimize the cost function?

Let's simplify equations

- Hypothesis:

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

- Parameters:

$$\theta_0, \theta_1 \qquad \theta_1$$

- Cost Function:

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

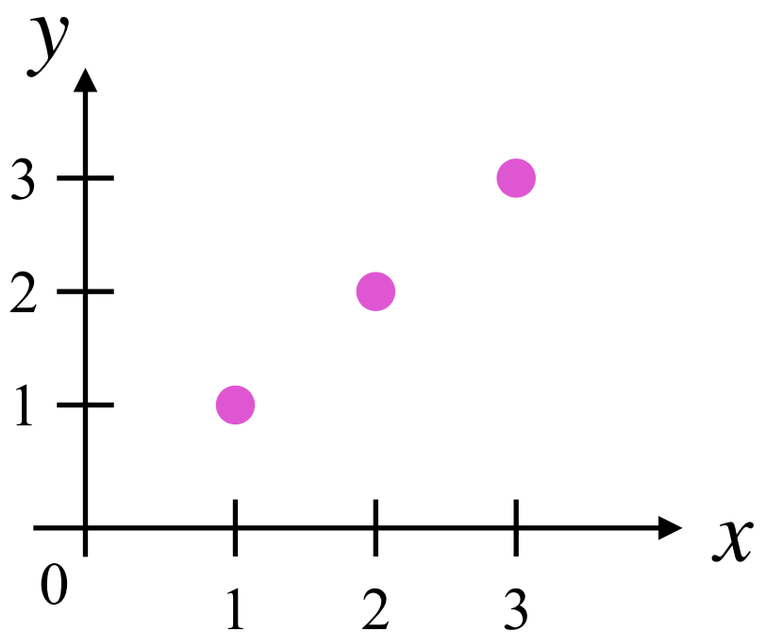
- Goal:

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

Hypothesis *vs.* Cost

$h_\theta(x)$

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



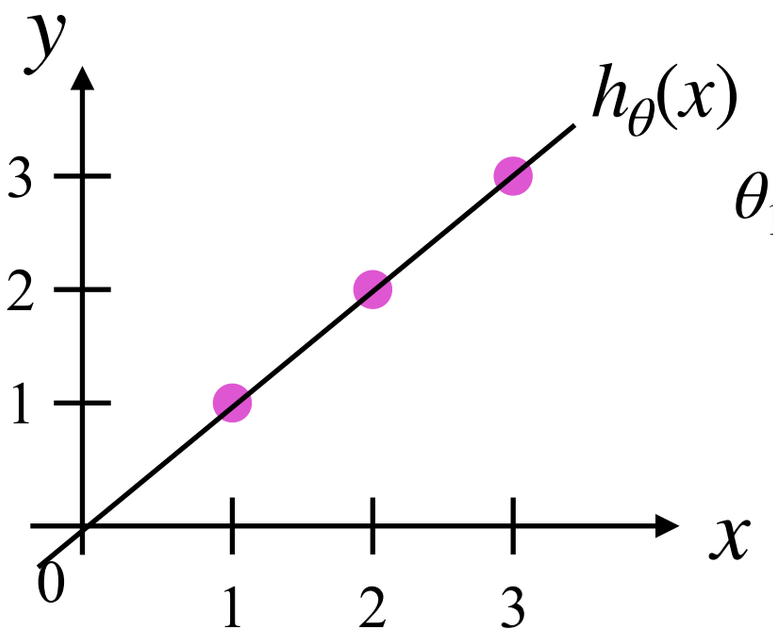
$J(\theta_1)$

(function of the parameter θ_1)

Hypothesis *vs.* Cost

$h_\theta(x)$

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



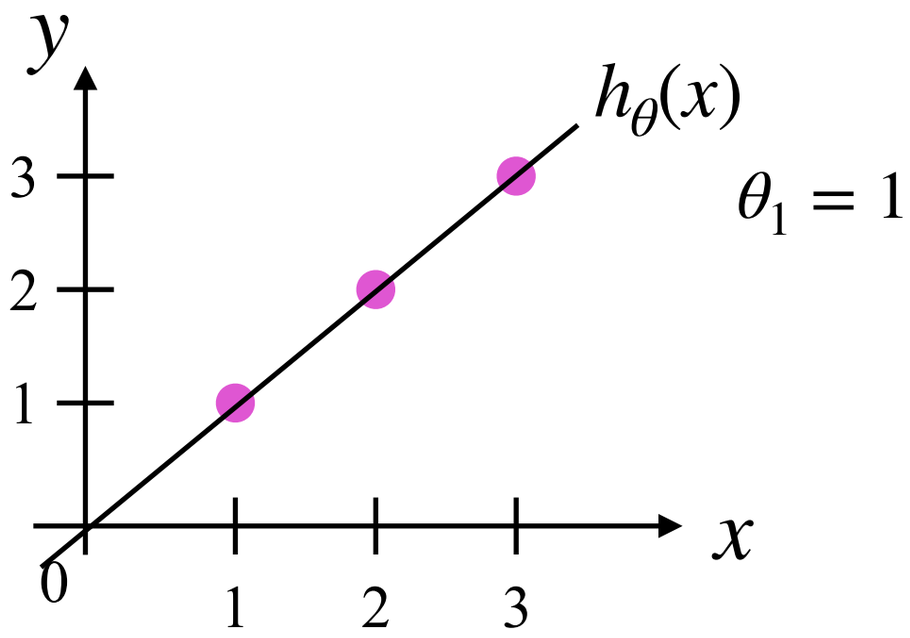
$J(\theta_1)$

(function of the parameter θ_1)

Hypothesis *vs.* Cost

$h_\theta(x)$

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



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

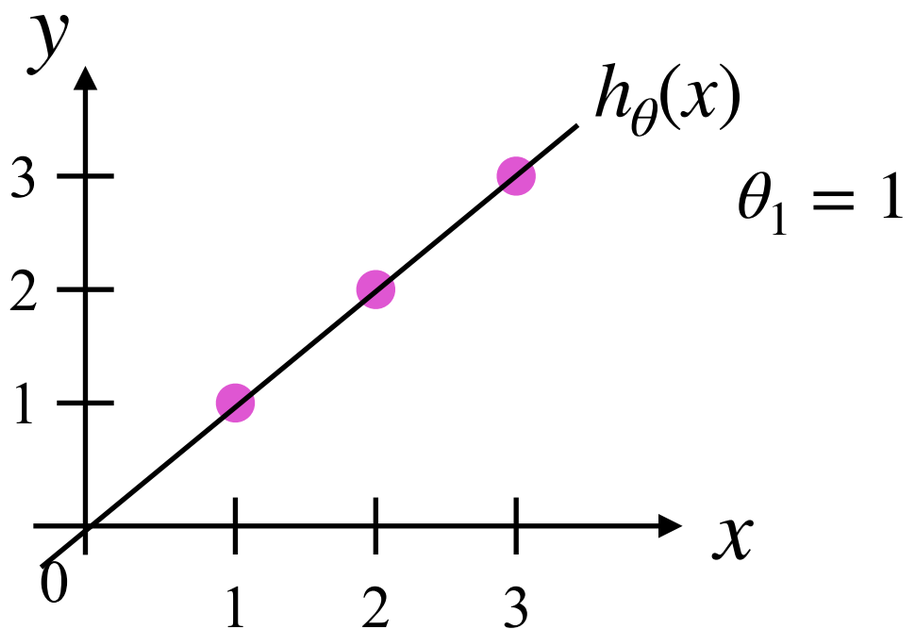
$J(\theta_1)$

(function of the parameter θ_1)

Hypothesis *vs.* Cost

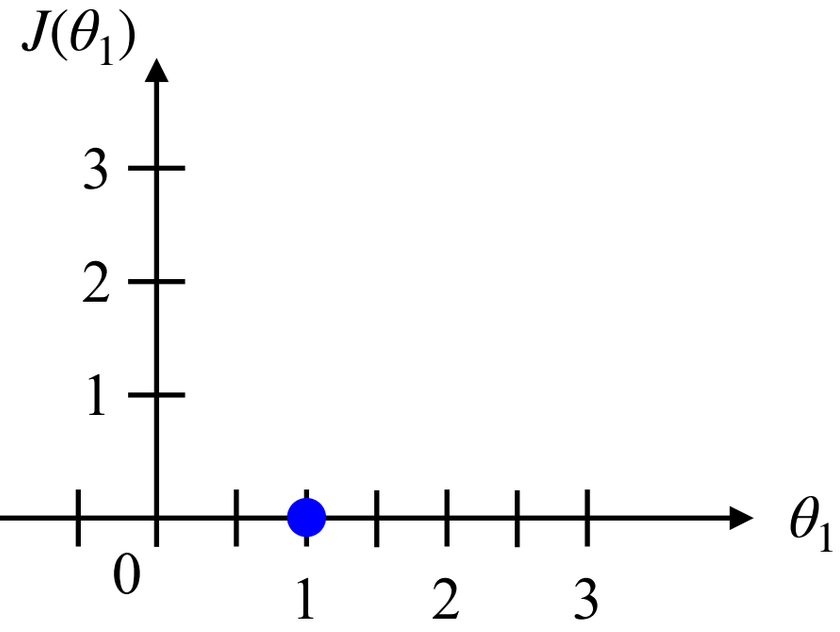
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



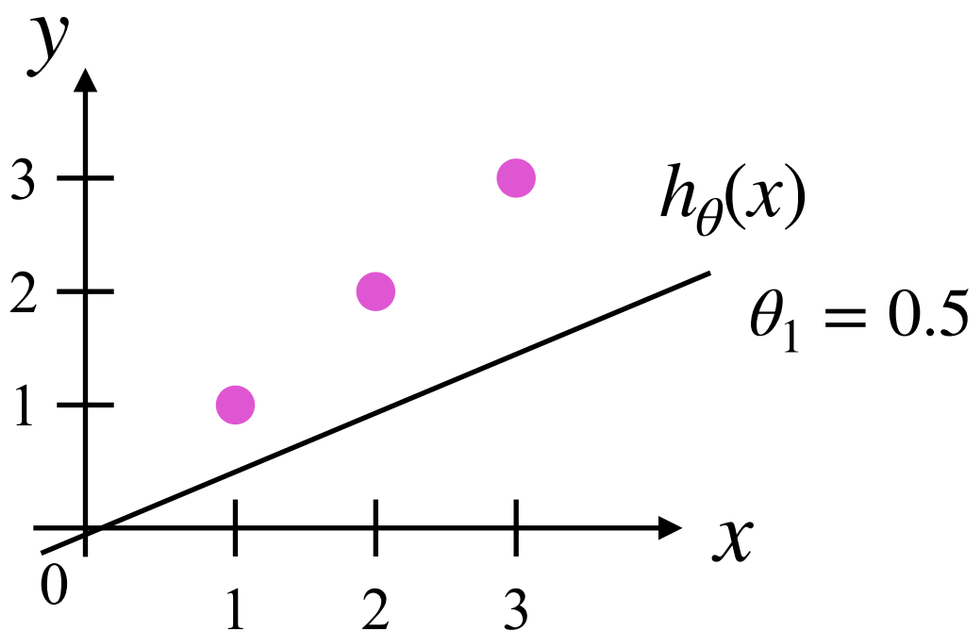
$$\because J(\theta_1) = \frac{1}{2m}[\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2]$$

$$\therefore J(1) = \frac{1}{2m}[\Sigma_{i=1}^{m}(\theta_1(x^{(i)}) - y^{(i)})^2] \quad = \frac{1}{2m}[0^2 + 0^2 + 0^2] = 0$$

Hypothesis *vs.* Cost

$h_\theta(x)$

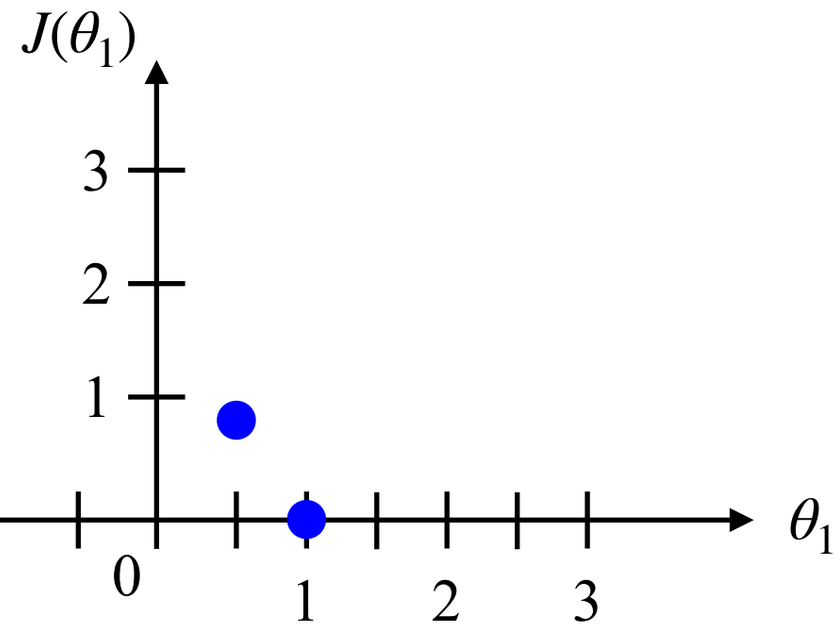
(for fixed θ_1, this is a function of x)



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

$J(\theta_1)$

(function of the parameter θ_1)



What if $J(0.5)$?

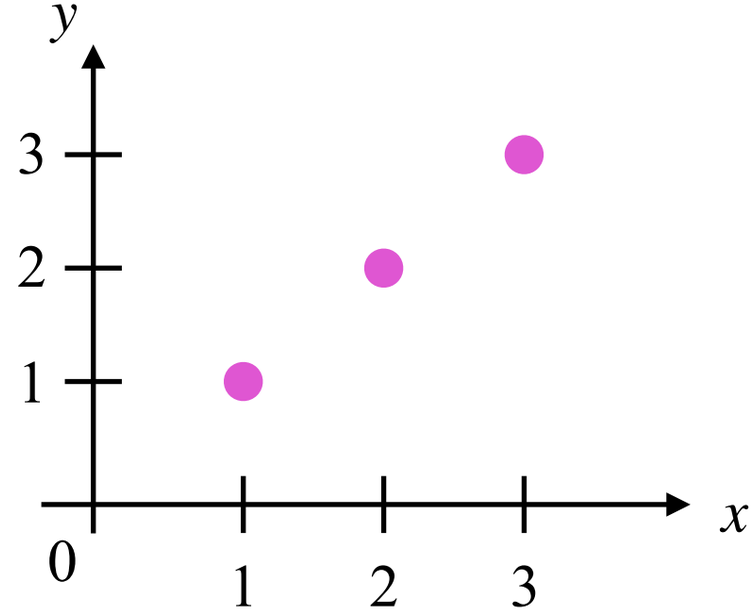
Question

- Suppose we have a training set with $m = 3$ examples, plotted below. Our hypothesis representation is $h_\theta(x) = \theta_1 x$, with parameter θ_1. The cost function $J(\theta_1)$ is:

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

What is $J(0)$?

(i) 0
(ii) 1/6
(iii) 1
(iv) 14/6



Question

- Suppose we have a training set with $m = 3$ examples, plotted below. Our hypothesis representation is $h_\theta(x) = \theta_1 x$, with parameter θ_1. The cost function $J(\theta_1)$ is:

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

What is $J(0)$? $\because \theta_1 = 0$

(i) 0

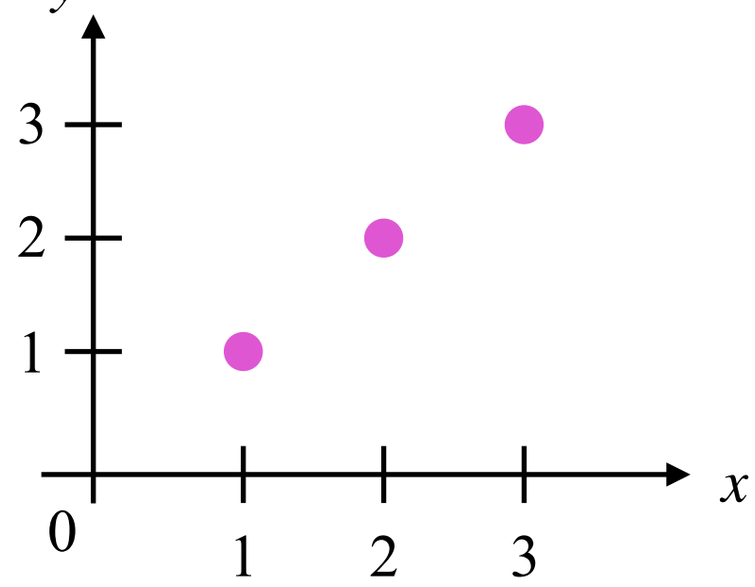
(ii) 1/6

(iii) 1

(iv) 14/6

$$\therefore J(\theta_1) = \frac{1}{2\times 3}[(0\times 1 - 1)^2 + (0\times 2 - 2)^2 + (0\times 3 - 3)^2]$$

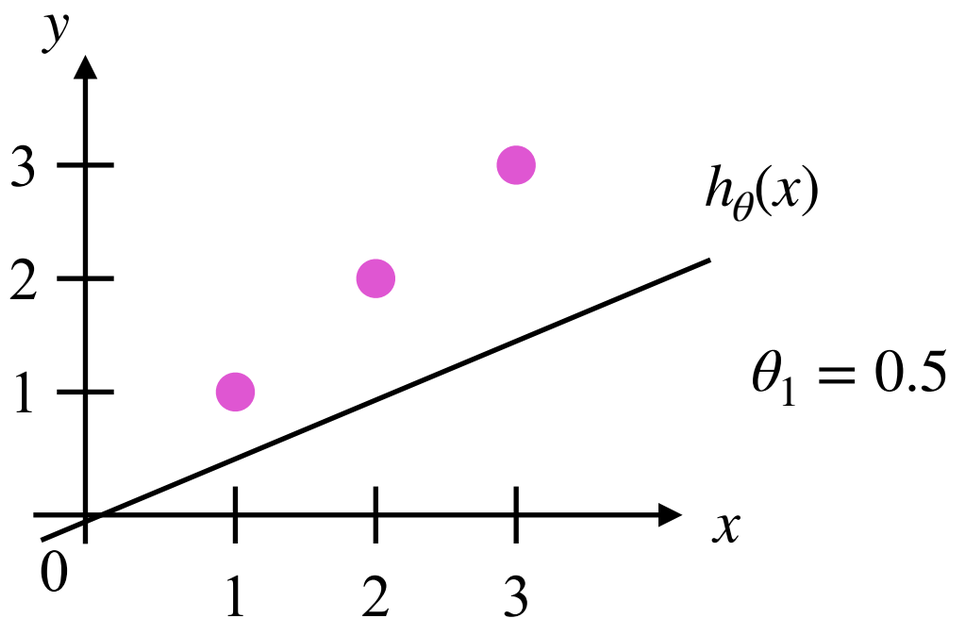
$$= \frac{1}{6}[1 + 4 + 9]$$



Hypothesis *vs.* Cost

$h_\theta(x)$

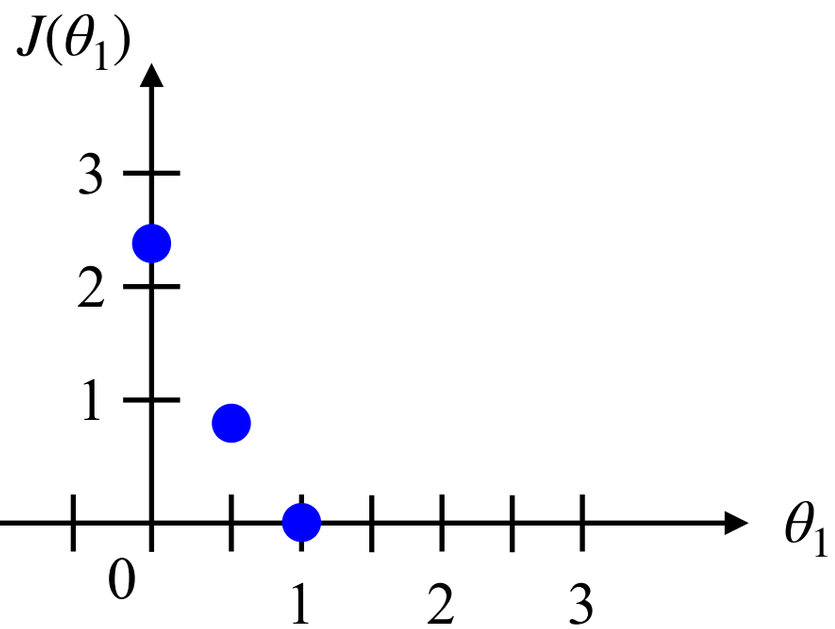
(for fixed θ_1, this is a function of x)



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

$J(\theta_1)$

(function of the parameter θ_1)



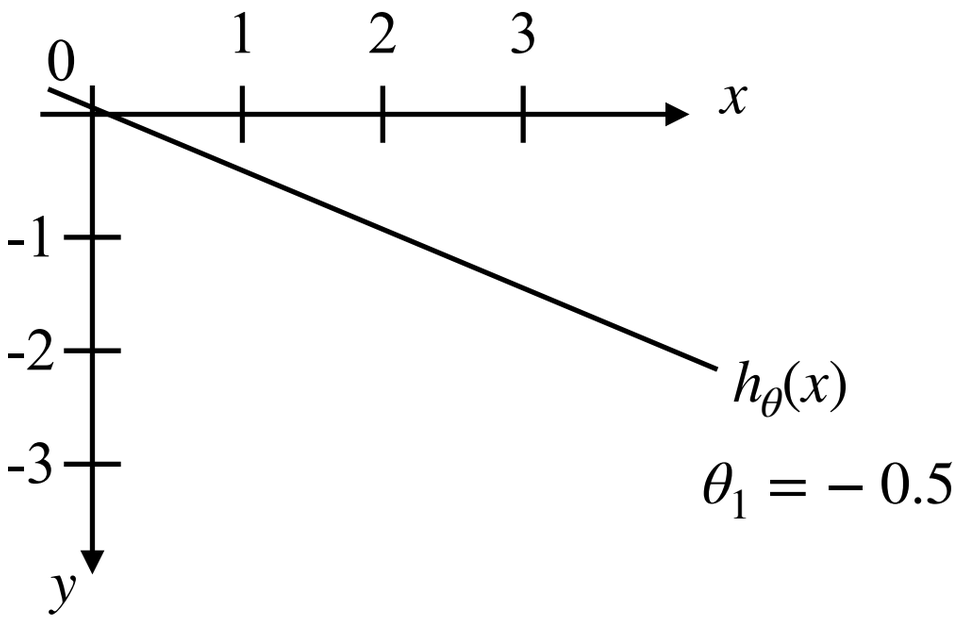
$J(0.5) \approx 0.58$

$J(0) = 14/6 \approx 2.33$

Hypothesis *vs.* Cost

$h_\theta(x)$

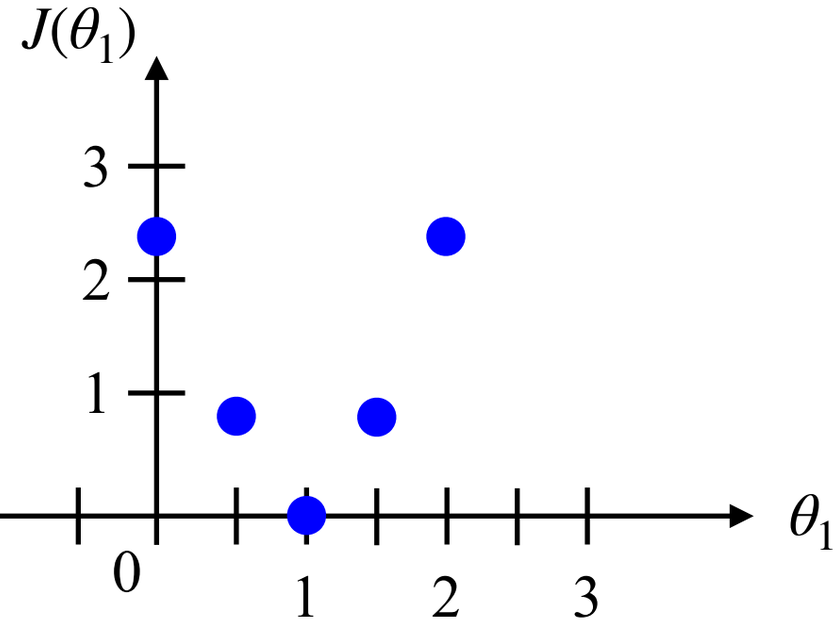
(for fixed θ_1, this is a function of x)



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

$J(\theta_1)$

(function of the parameter θ_1)



$J(0.5) \approx 0.58$

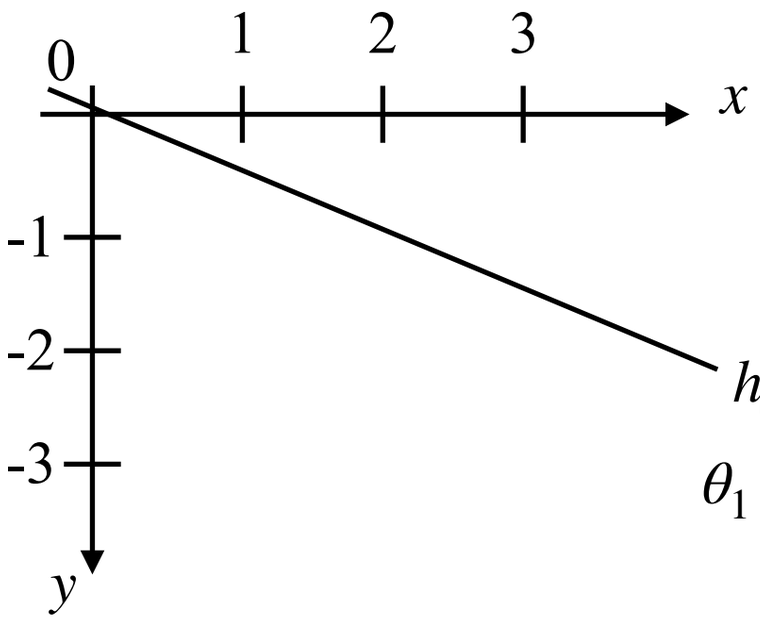
$J(0) = 14/6 \approx 2.33$

$J(-0.5) = 5.25$

Hypothesis *vs.* Cost

$h_\theta(x)$

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

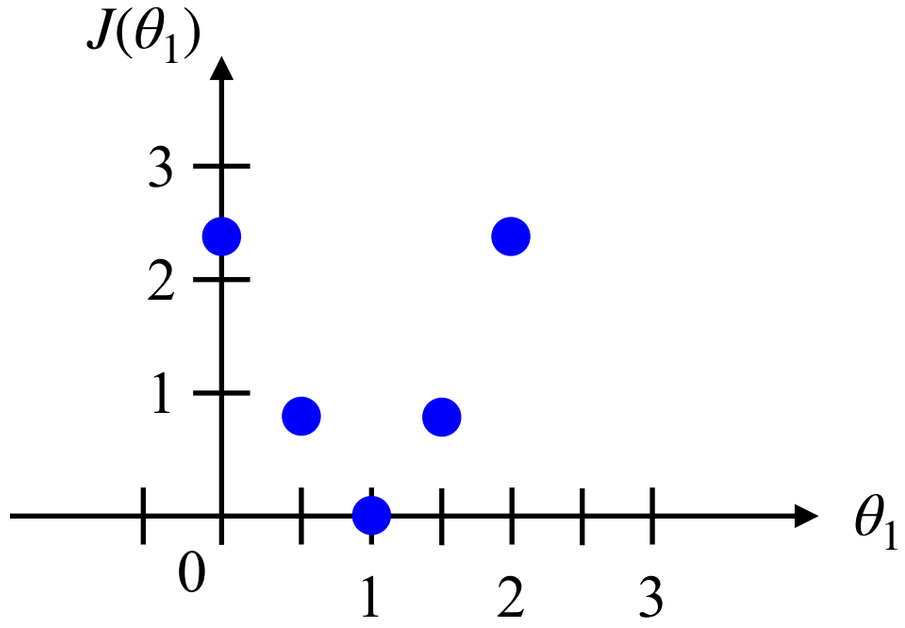


$\theta_1 = -0.5$

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

$J(\theta_1)$

(function of the parameter θ_1)



$J(0.5) \approx 0.58$

$J(0) = 14/6 \approx 2.33$

$J(-0.5) = 5.25$

<u>Goal</u>:

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

Let's get back to the original version

- Hypothesis:

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

- Parameters:

$$\theta_0, \theta_1$$

- Cost Function:

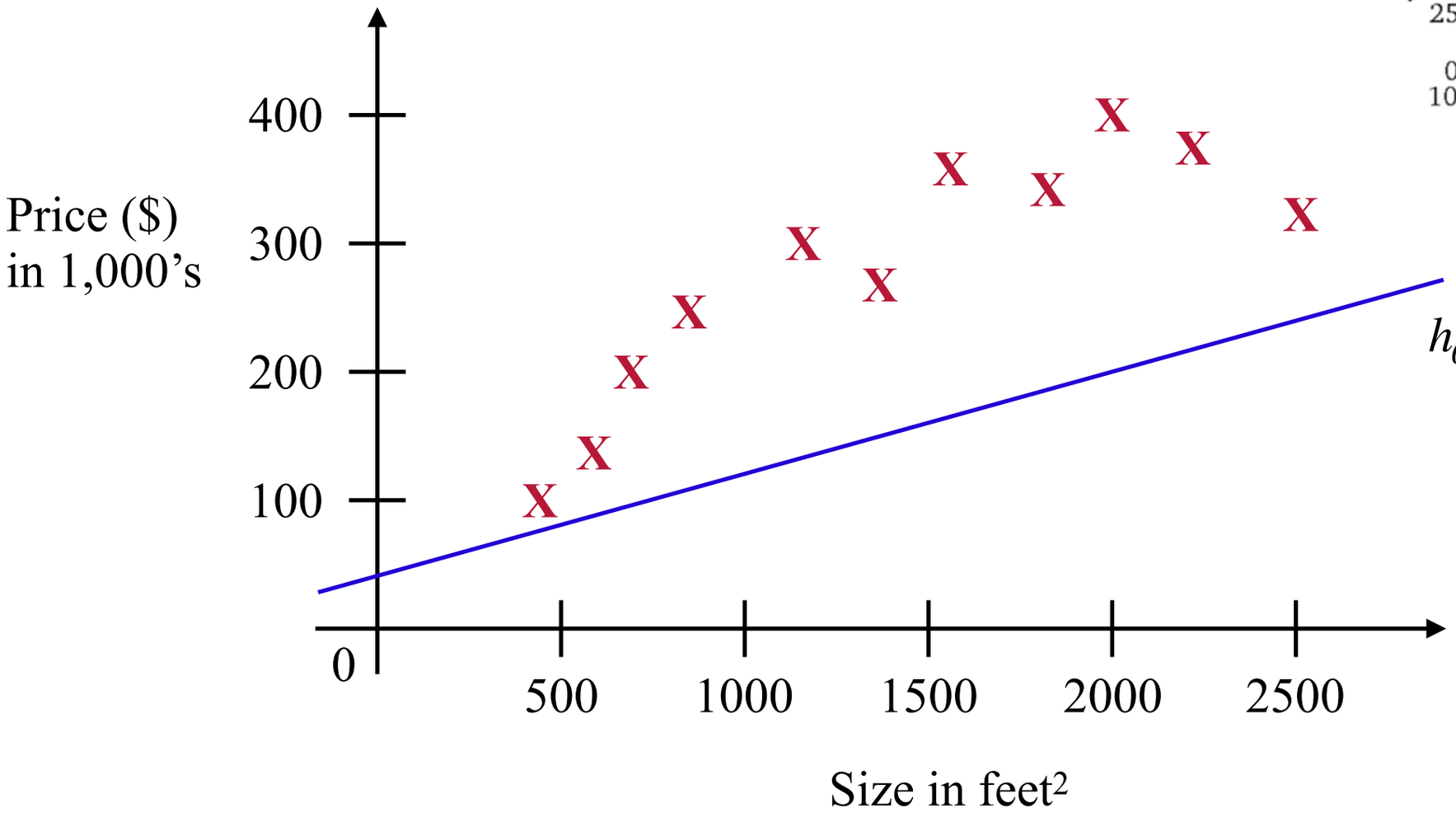
$$J(\theta_0, \theta_1) = \frac{1}{2m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

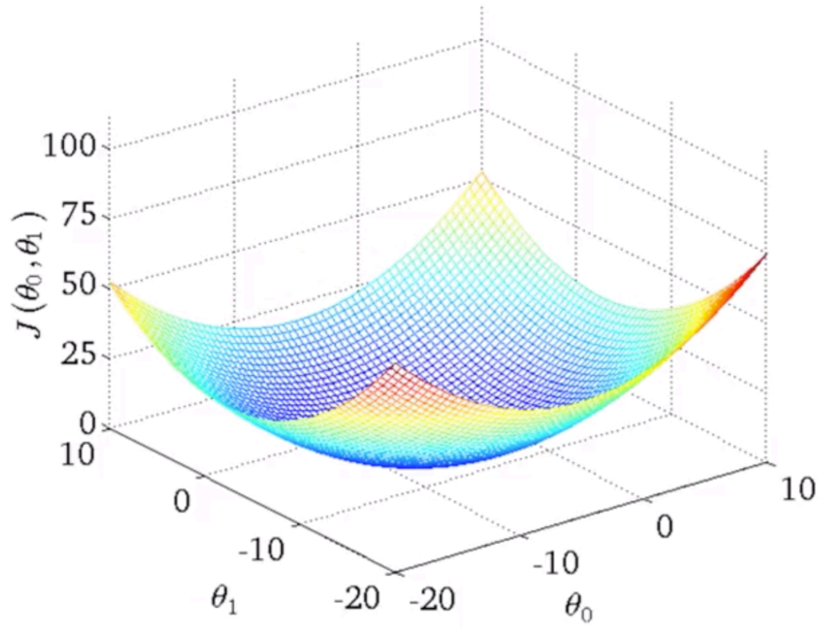
- Goal:

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

Hypothesis *vs.* Cost

Housing price prediction





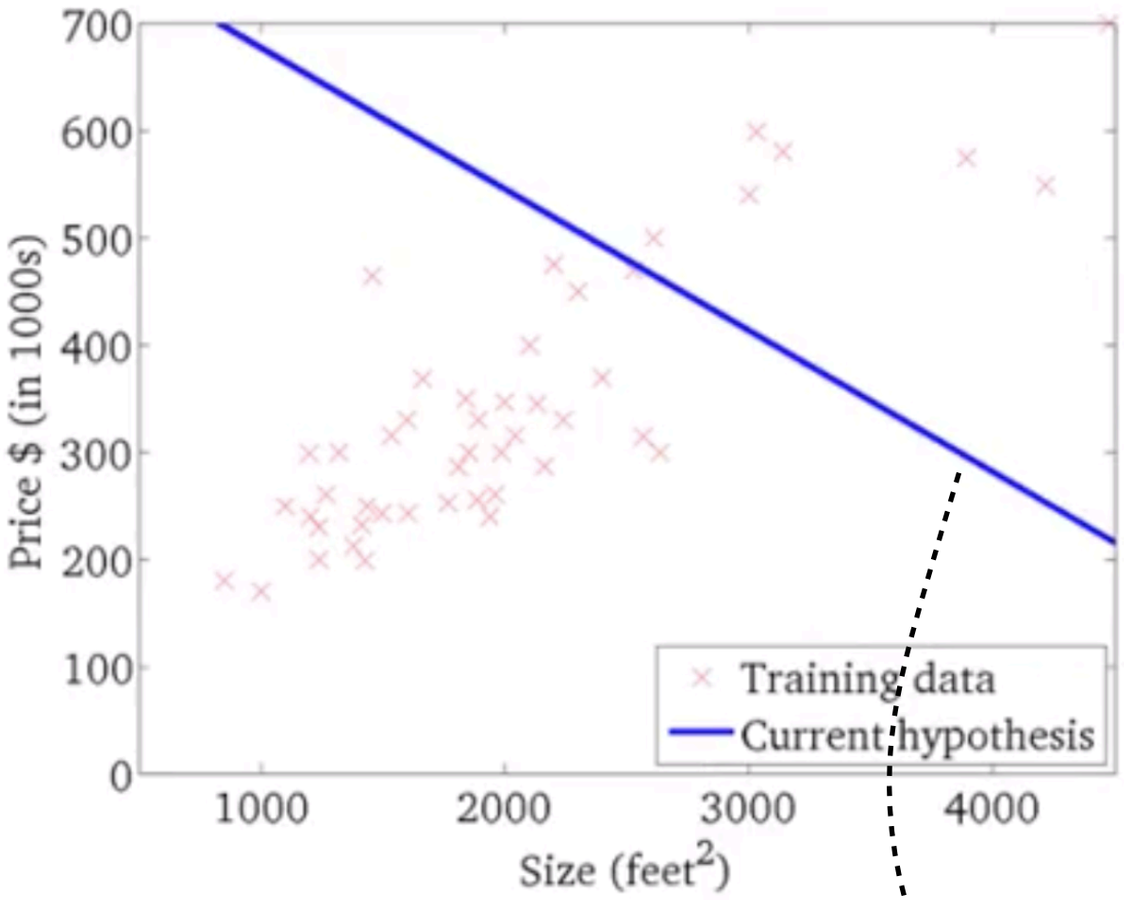
$h_\theta(x) = 50 + 0.06x$

(source: Andrew Ng, machine learning)

Contour Plot

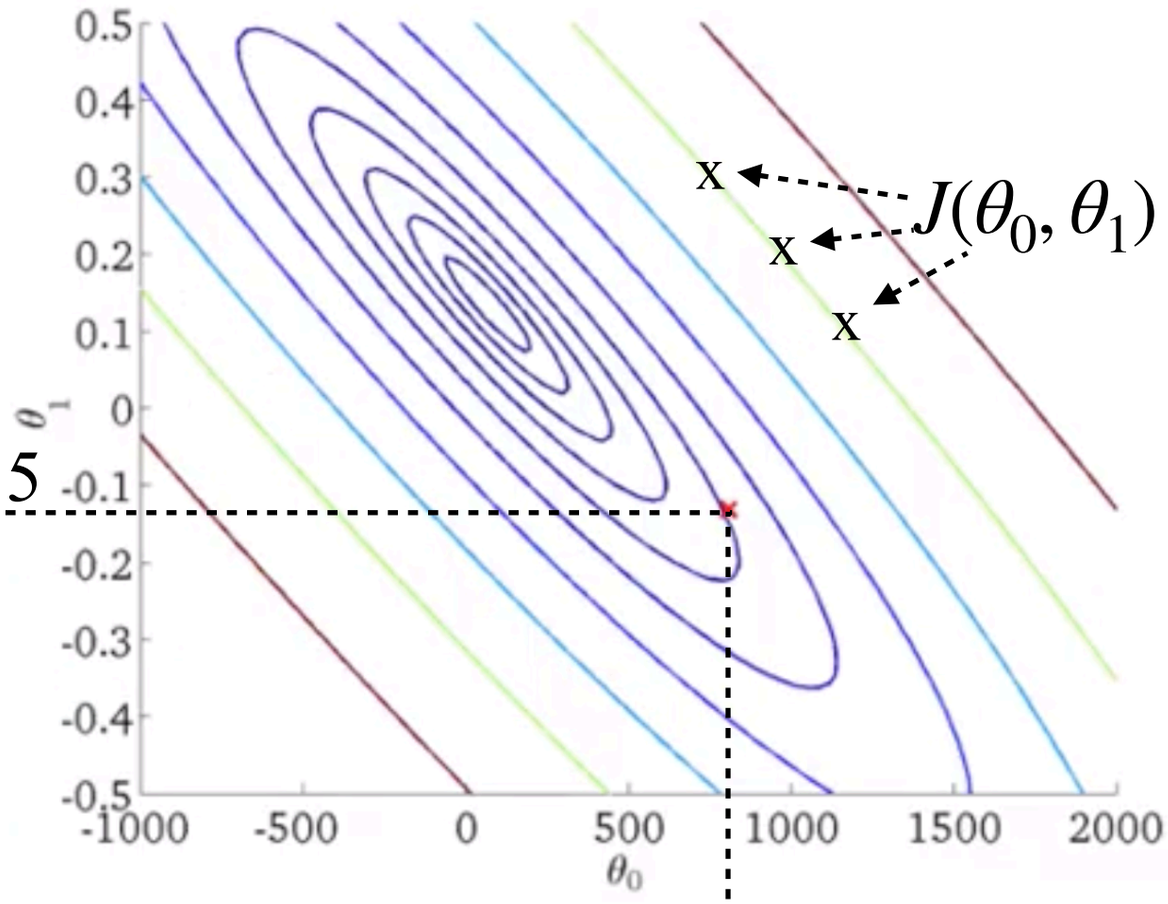
$h_\theta(x)$

(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)



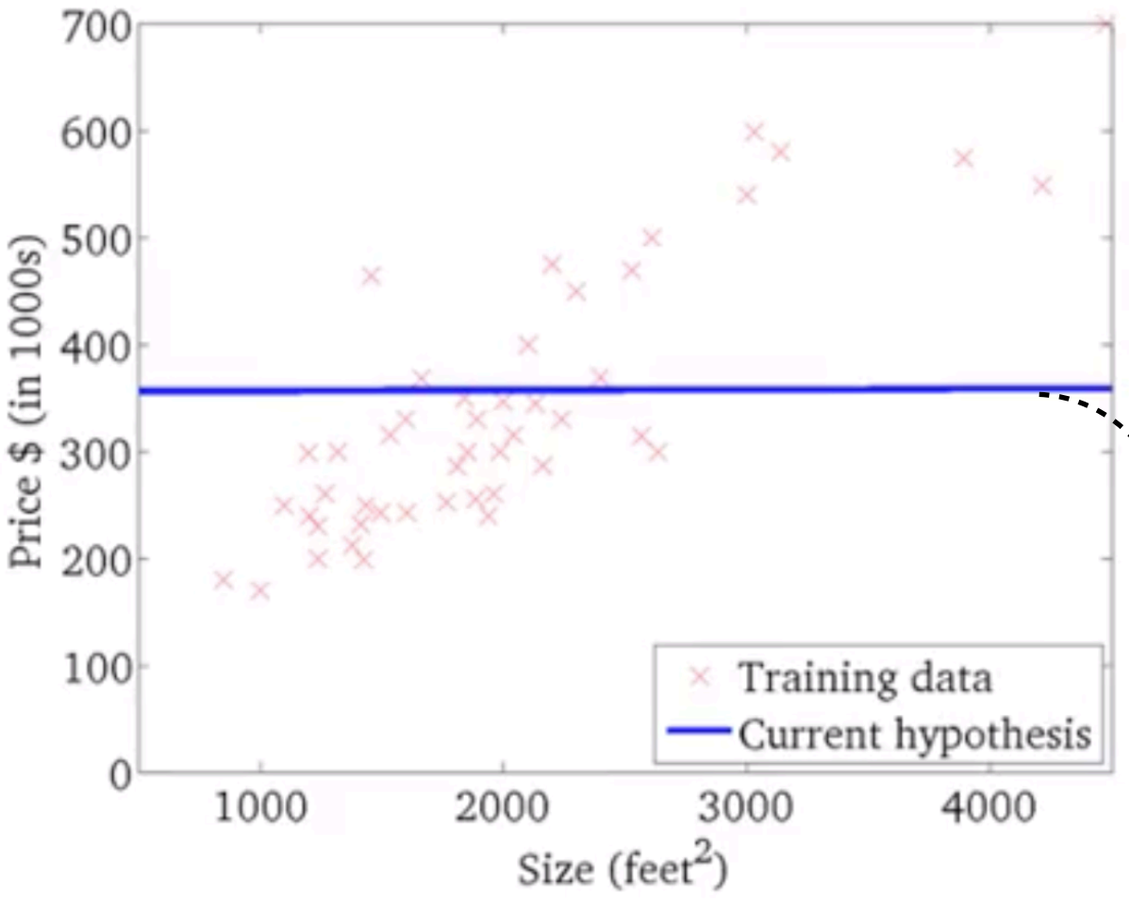
$\theta_0 = 800, \theta_1 = -0.15$

(source: Andrew Ng, machine learning)

Contour Plot

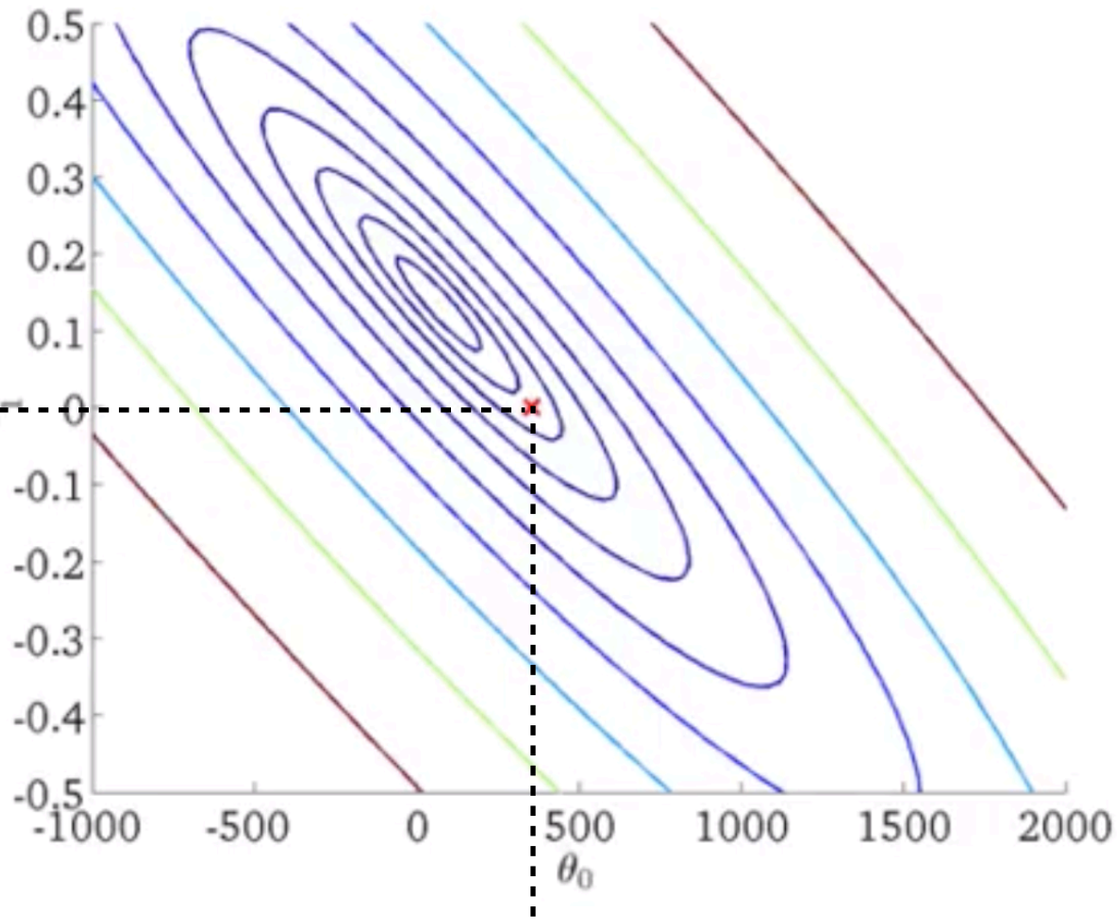
$h_\theta(x)$

(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)



$\theta_0 = 360, \theta_1 = 1$

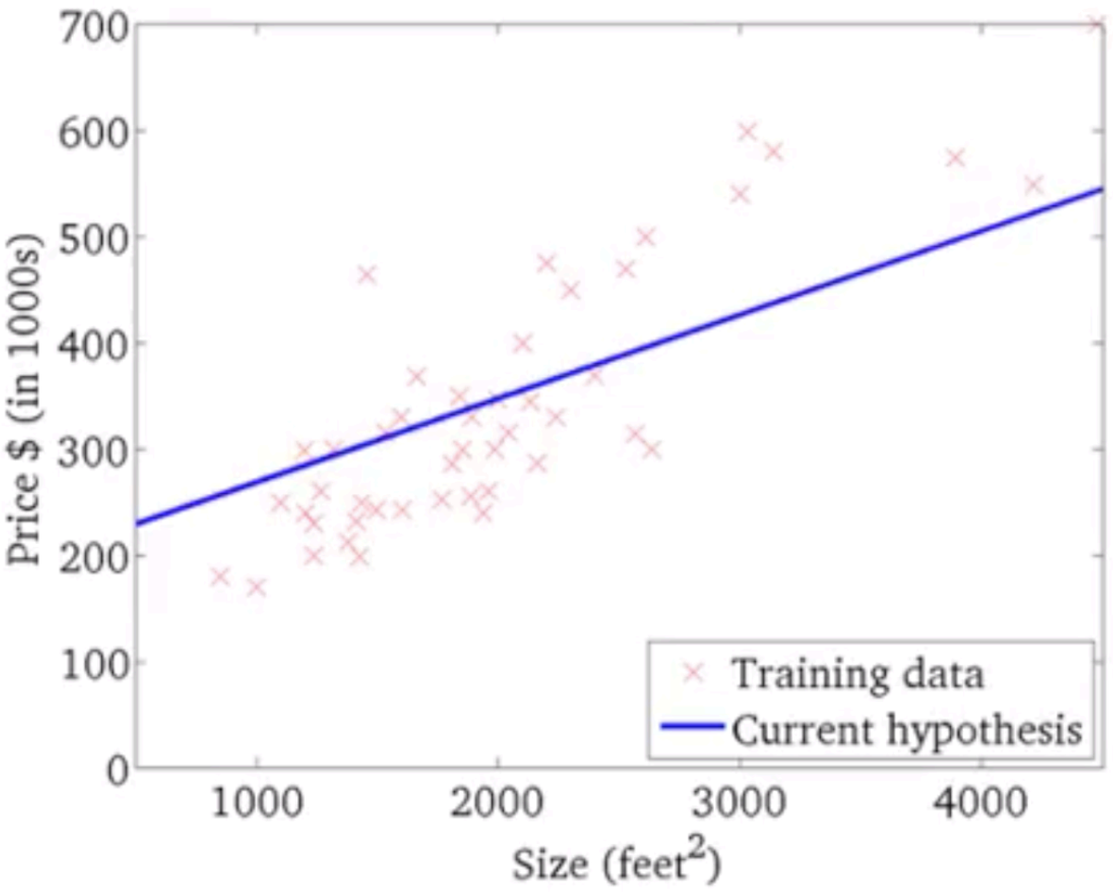
360

(source: Andrew Ng, machine learning)

Contour Plot

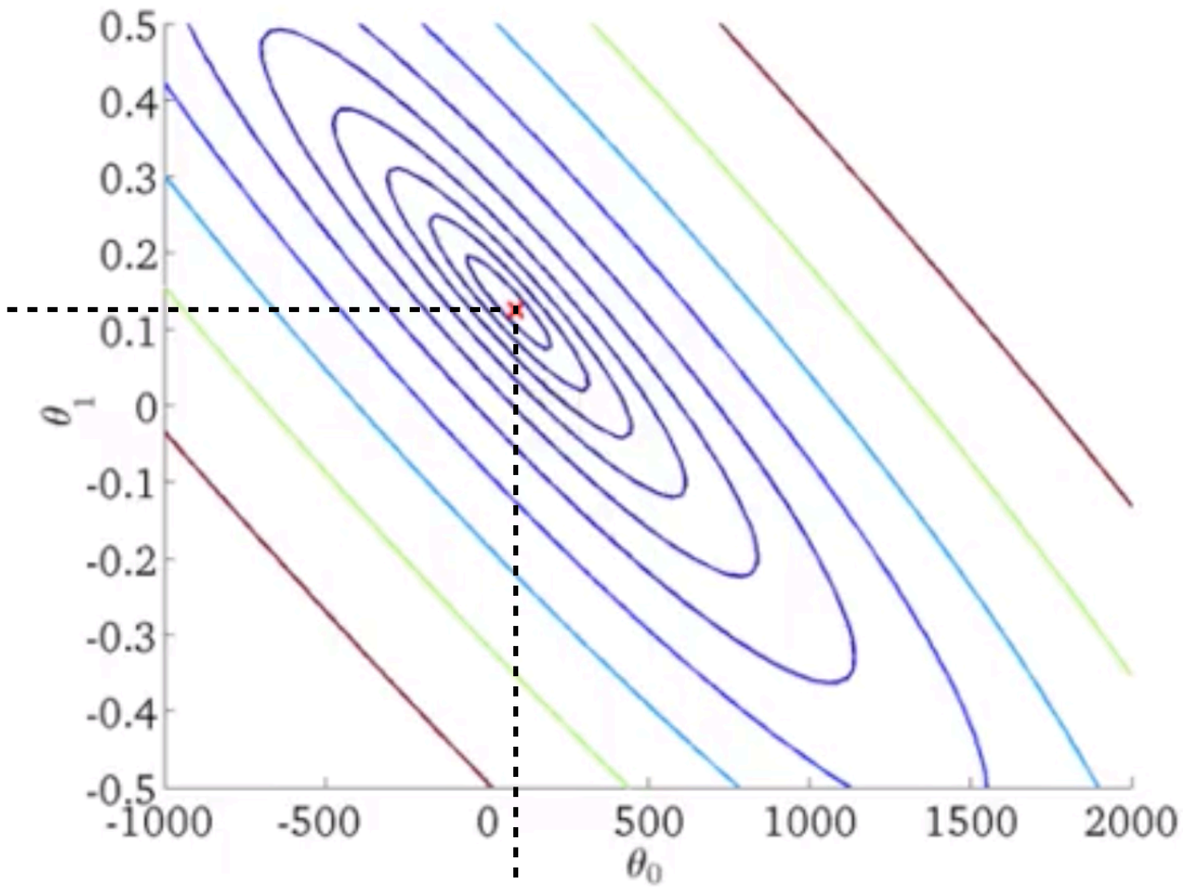
$h_\theta(x)$

(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)



(source: Andrew Ng, machine learning)

Gradient Descent - An Algorithm for Univariate Linear Regression

Things we have

- Have some function $J(\theta_0, \theta_1)$
- Goal: $\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$

- **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

- In fact, gradient descent is a 'general' algorithm

$$\min_{\theta_0,\ldots,\theta_n} J(\theta_0, \ldots, \theta_n)$$

Meaning of Gradient Descent

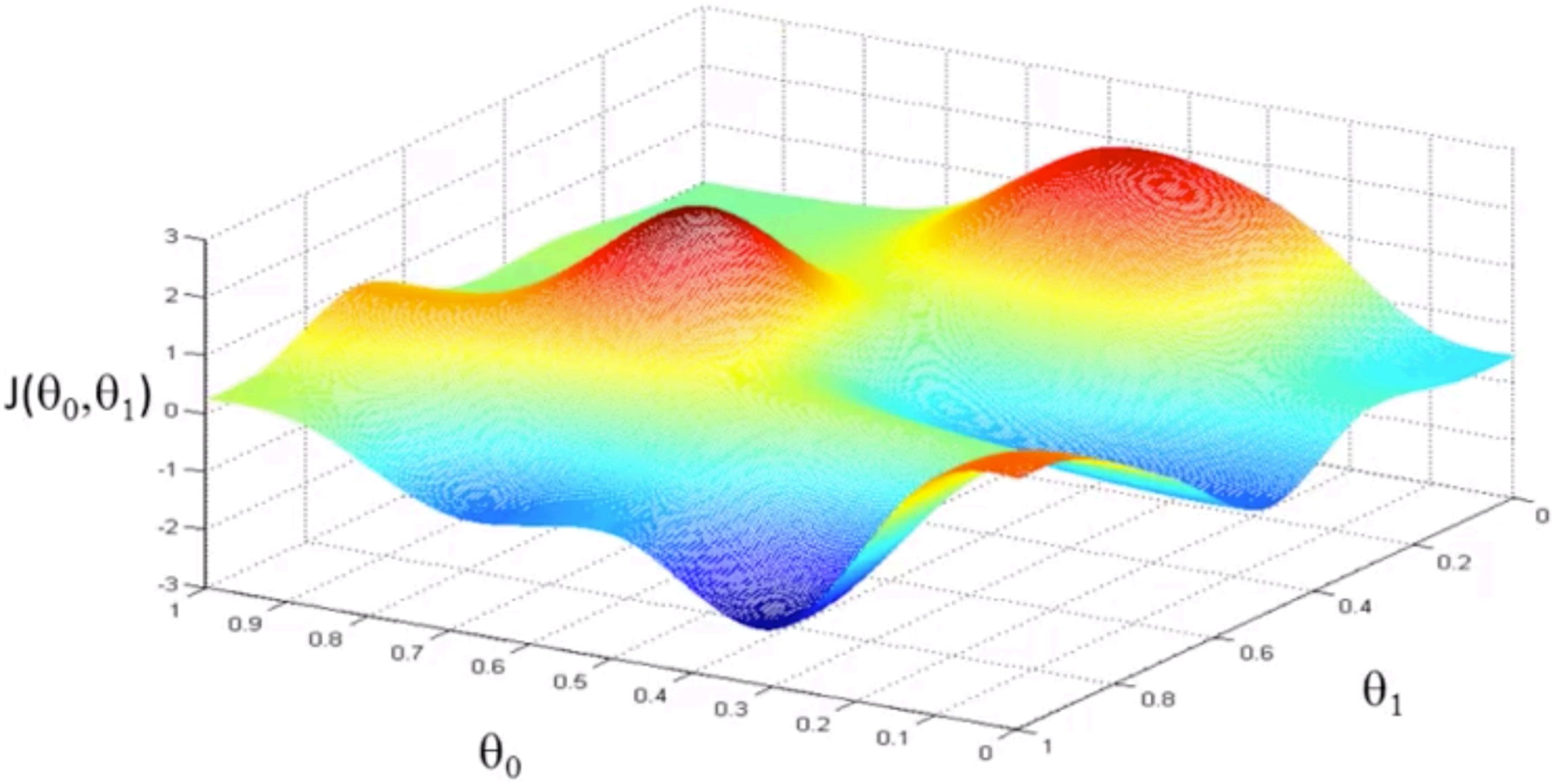
- Gradient: the degree of steepness of a graph at any point
- Descent: an action of moving downward, dropping, or falling

Formally,

The gradient of $f(x)$, *i.e.* $\nabla_f(x)$, evaluated at an arbitrary point x is a vector with two useful properties:

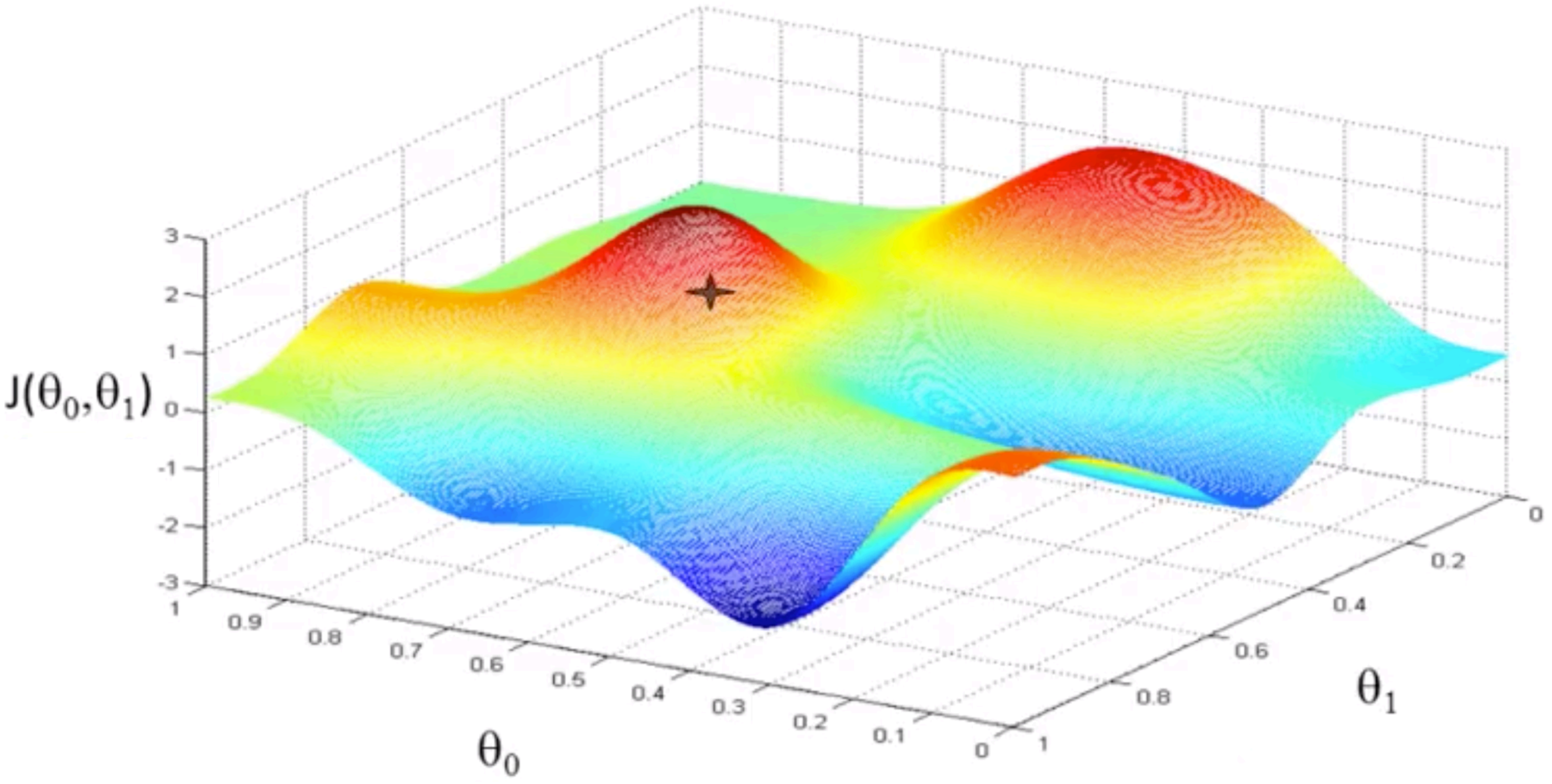
- The direction of $\nabla_f(x)$ is the direction in which $f(x)$ is increasing the fastest.
- The magnitude of $\nabla_f(x)$ is the slope of $f(x)$ in that direction of maximum increase.

Gradient Descent (Intuitively)



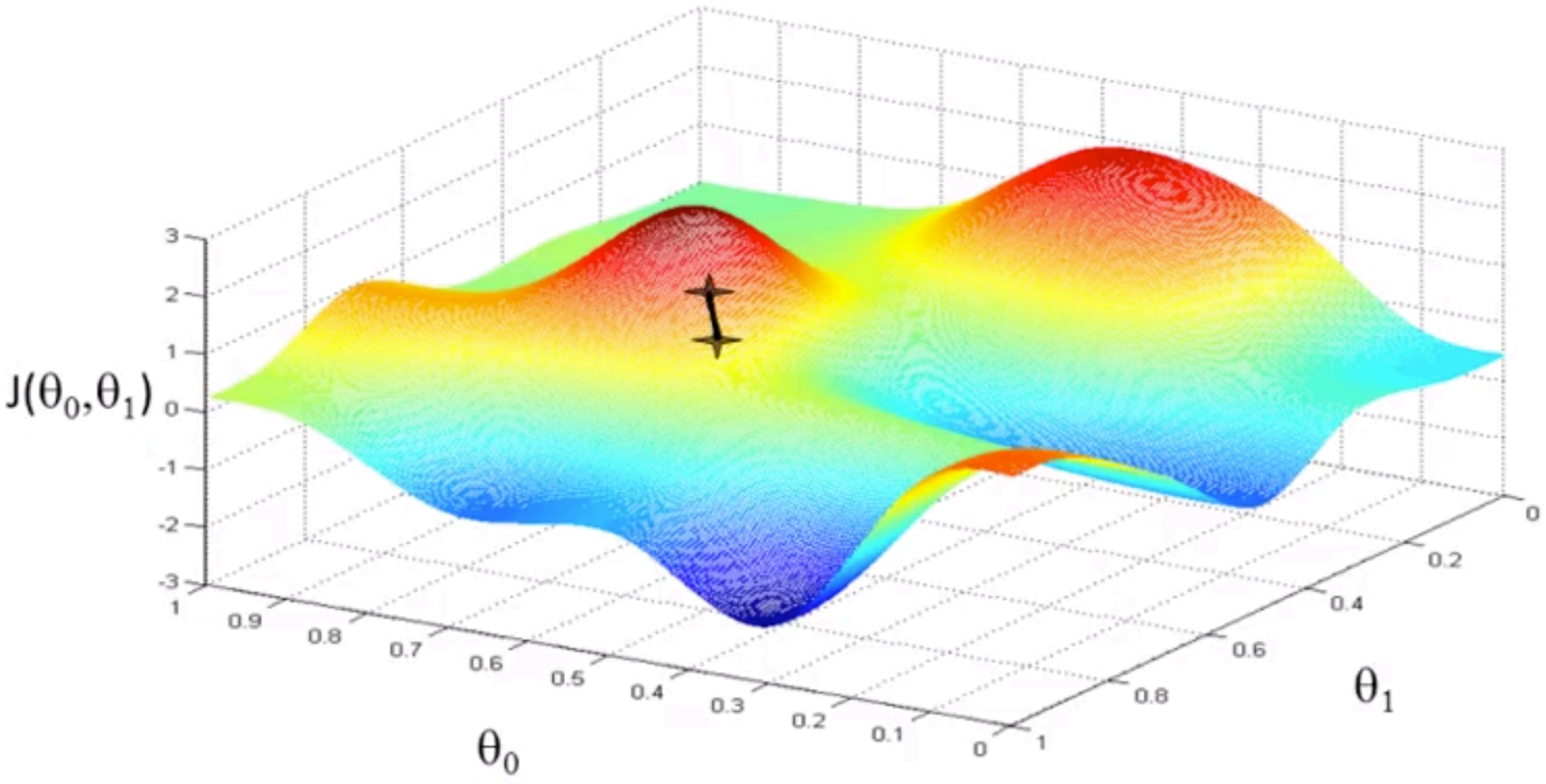
(source: Andrew Ng, machine learning)

Gradient Descent (Intuitively)



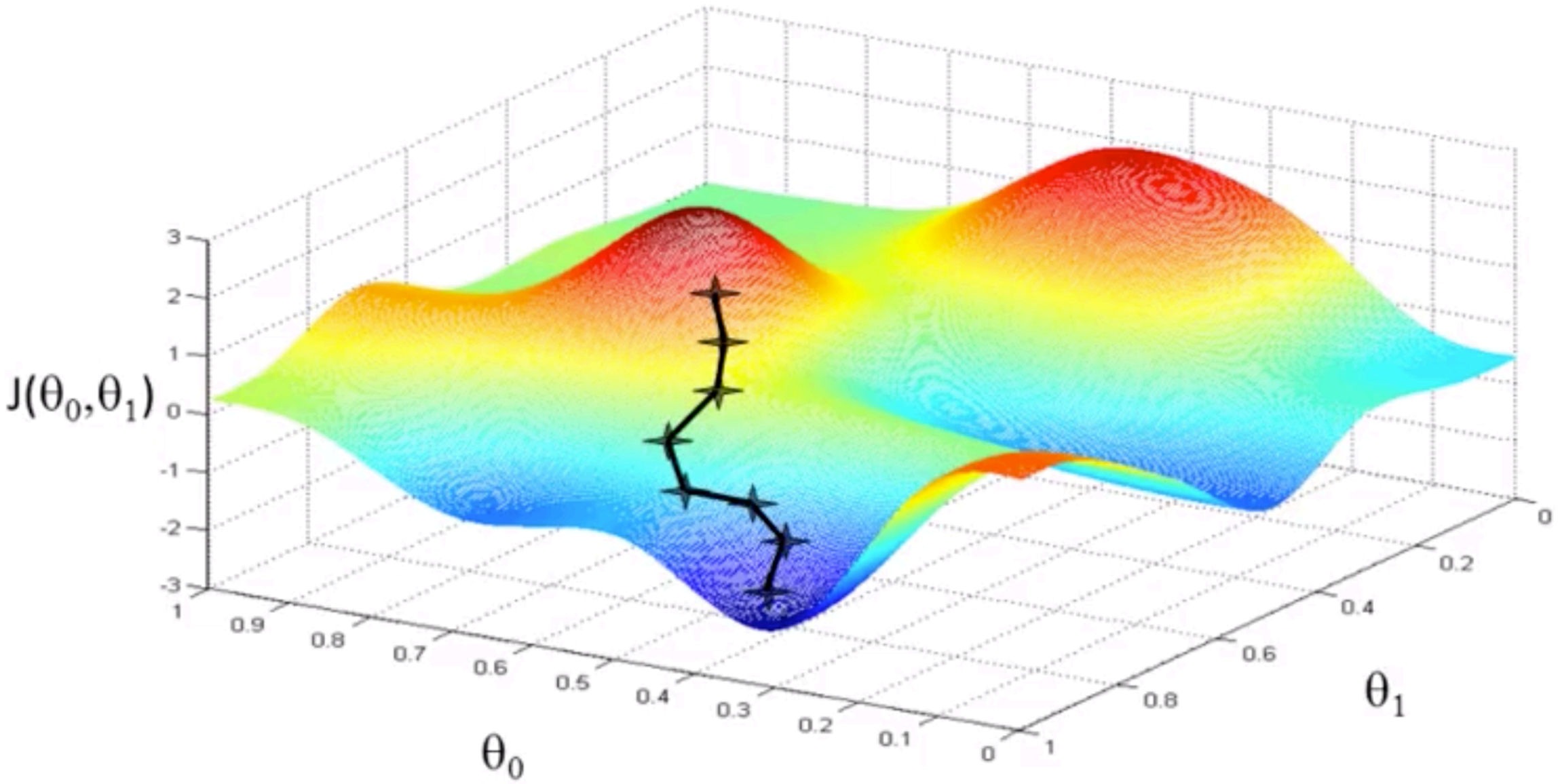
(source: Andrew Ng, machine learning)

Gradient Descent (Intuitively)



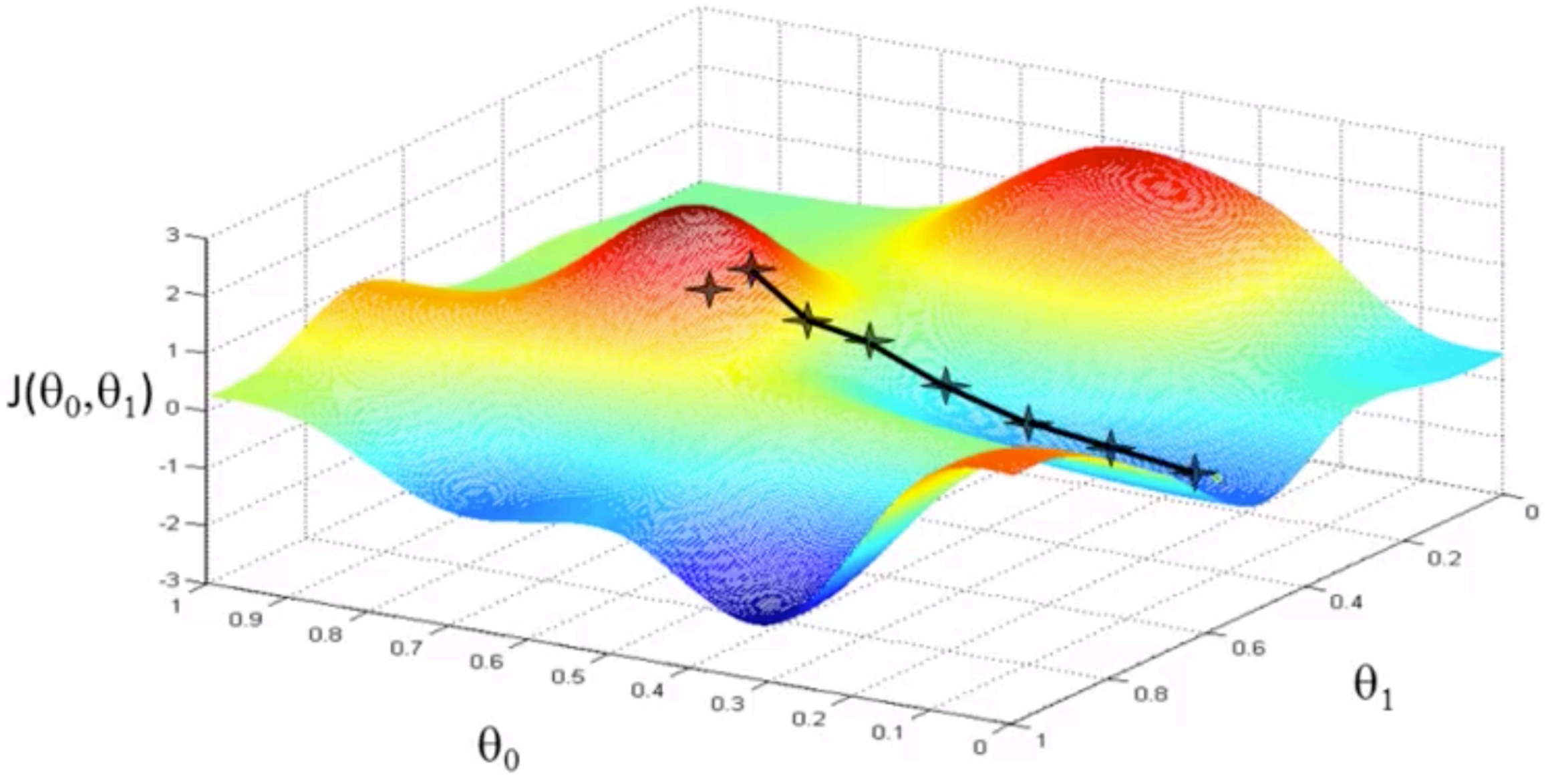
(source: Andrew Ng, machine learning)

Gradient Descent (Intuitively)



(source: Andrew Ng, machine learning)

Gradient Descent (Intuitively)



Gradient descent has an ‘interesting’ property !

(source: Andrew Ng, machine learning)

Gradient Descent Algorithm

repeat until convergence {

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

}

'learning rate' (always positive !)

if α is very large, then gradient descent is too aggressive

Correct : Simultaneous update

$$t_1 := \theta_0 - \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$t_2 := \theta_1 - \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := t_1$$

$$\theta_1 := t_2$$

Incorrect !

$$t_1 := \theta_0 - \frac{\partial J}{\partial \theta_0}(\theta_0, \theta_1)$$

$$\theta_0 := t_1$$

$$t_2 := \theta_1 - \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_1 := t_2$$

Question

- Suppose $\theta_0 = 1, \theta_1 = 2$, and we simultaneously update θ_0 and θ_1 using the rule: $\theta_j := \theta_j + \sqrt{\theta_0 \theta_1}$ (for $j = 0$ and $j = 1$). What are the resulting values of θ_0 and θ_1?

(i) $\theta_0 = 1$, $\theta_1 = 2$

(ii) $\theta_0 := 1 + \sqrt{2}$, $\theta_1 := 2 + \sqrt{2}$

(iii) $\theta_0 := 2 + \sqrt{2}$, $\theta_1 := 1 + \sqrt{2}$

(iv) $\theta_0 := 1 + \sqrt{2}$, $\theta_1 := 2 + \sqrt{(1 + \sqrt{2}) \cdot 2}$

Question

- Suppose $\theta_0 = 1, \theta_1 = 2$, and we simultaneously update θ_0 and θ_1 using the rule: $\theta_j := \theta_j + \sqrt{\theta_0 \theta_1}$ (for $j = 0$ and $j = 1$). What are the resulting values of θ_0 and θ_1?

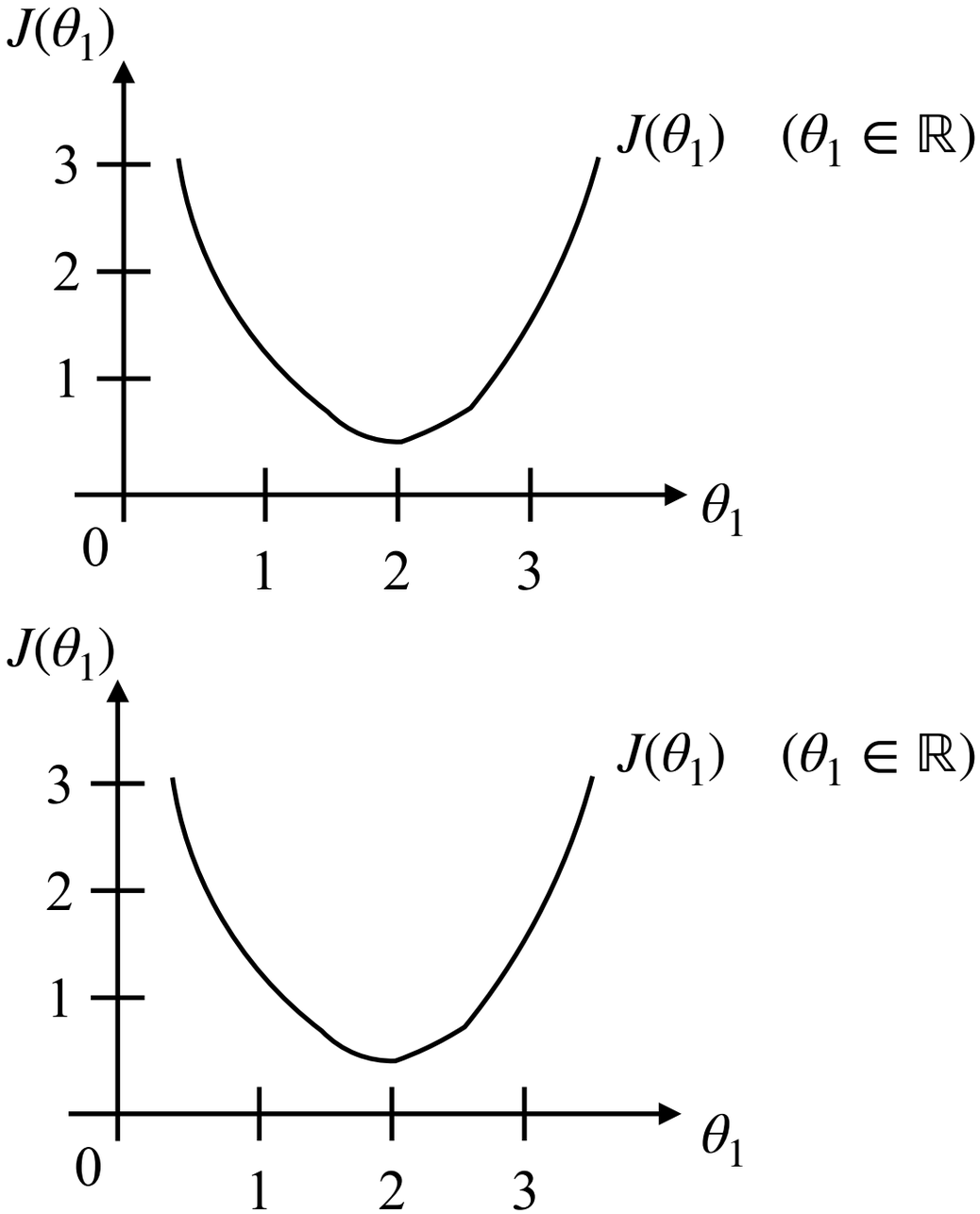
(i) $\theta_0 = 1$, $\theta_1 = 2$

(ii) $\theta_0 := 1 + \sqrt{2}$, $\theta_1 := 2 + \sqrt{2}$

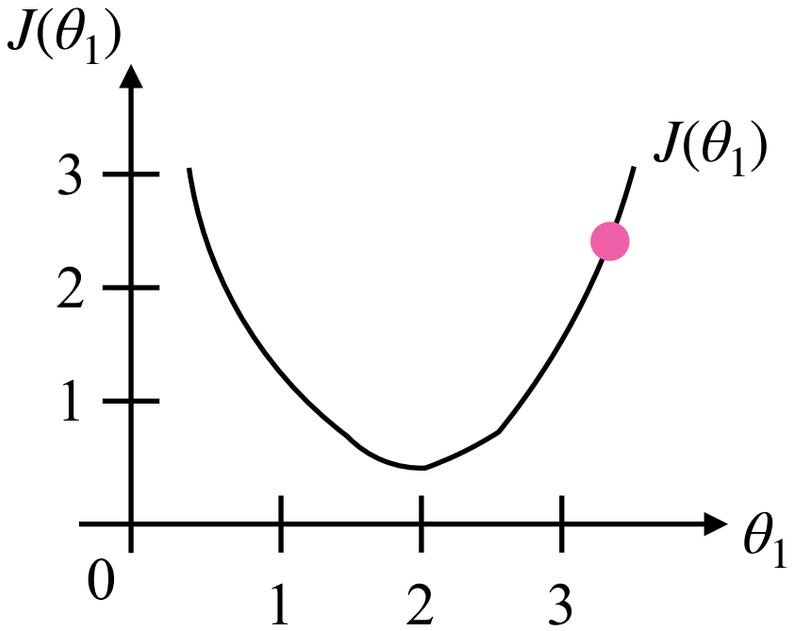
(iii) $\theta_0 := 2 + \sqrt{2}$, $\theta_1 := 1 + \sqrt{2}$

(iv) $\theta_0 := 1 + \sqrt{2}$, $\theta_1 := 2 + \sqrt{(1 + \sqrt{2}) \cdot 2}$

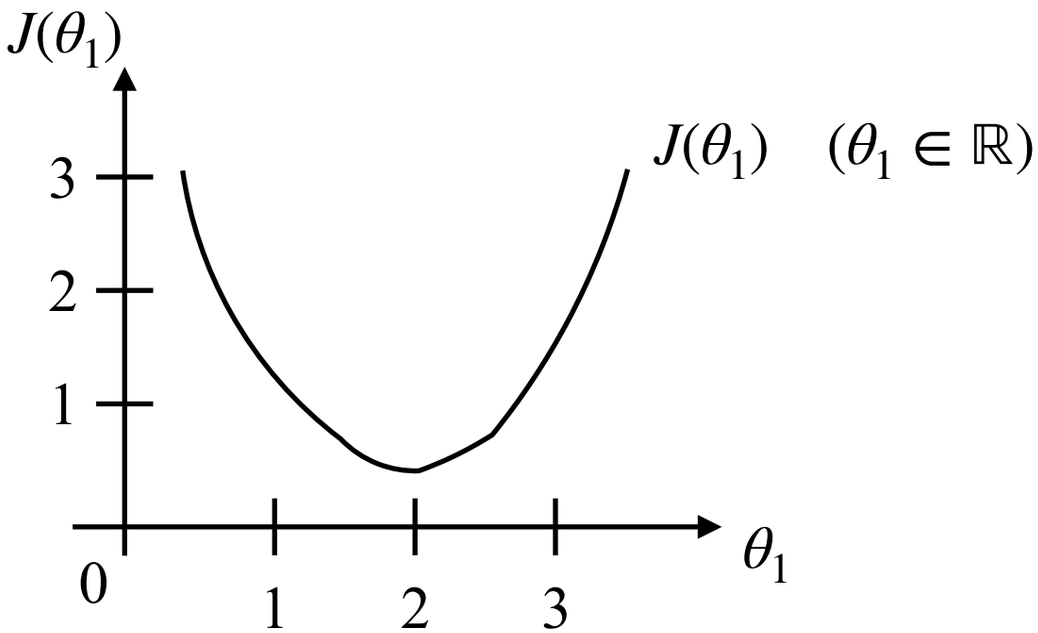
Gradient Descent Algorithm (Intuitively)



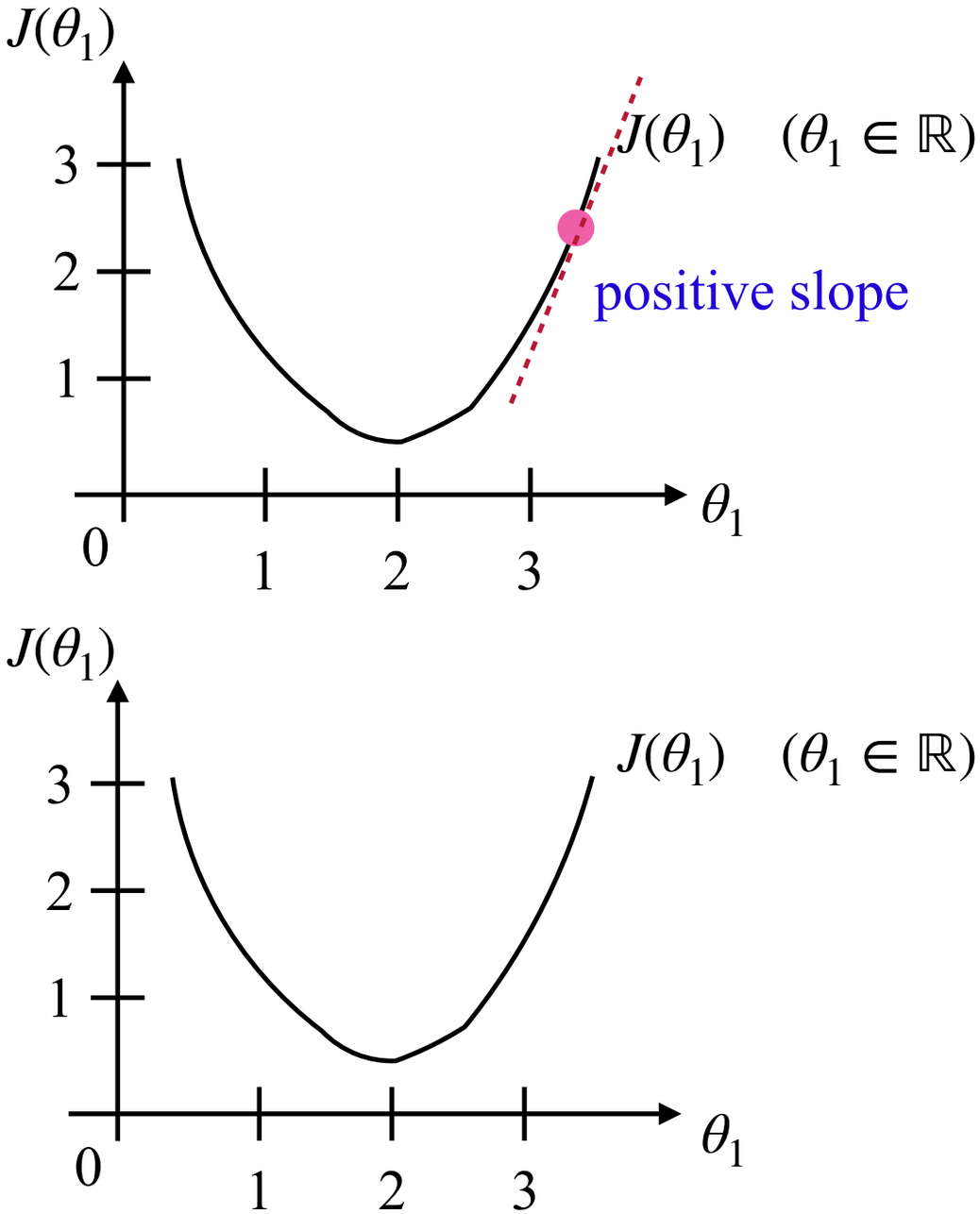
Gradient Descent Algorithm (Intuitively)



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

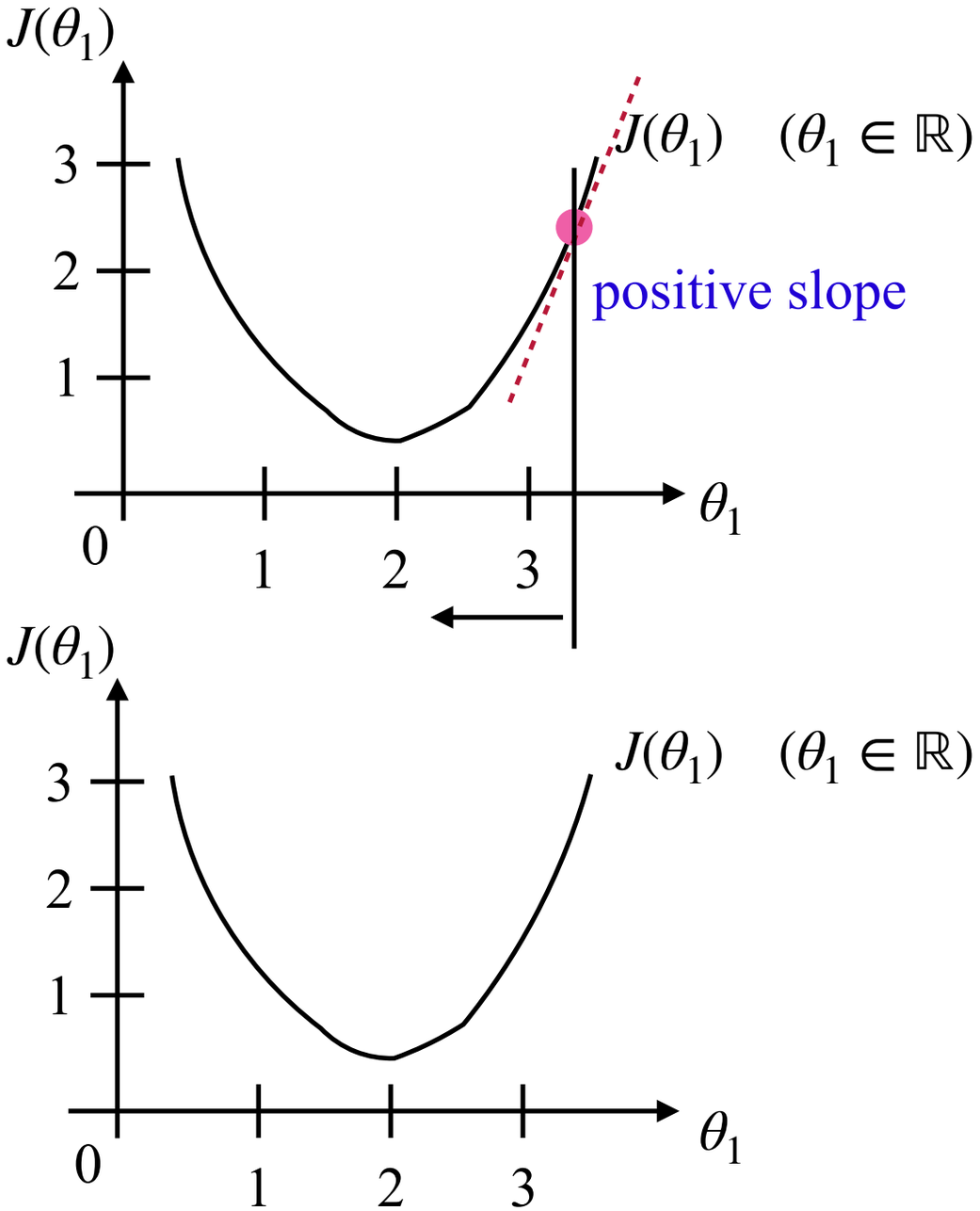


Gradient Descent Algorithm (Intuitively)



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

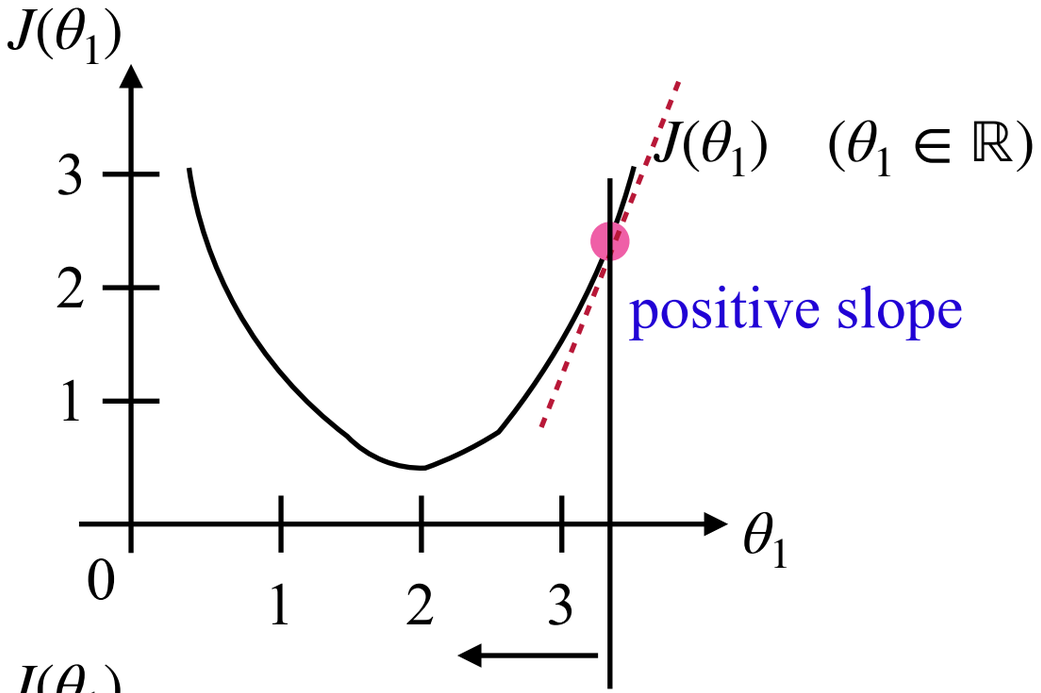
Gradient Descent Algorithm (Intuitively)



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

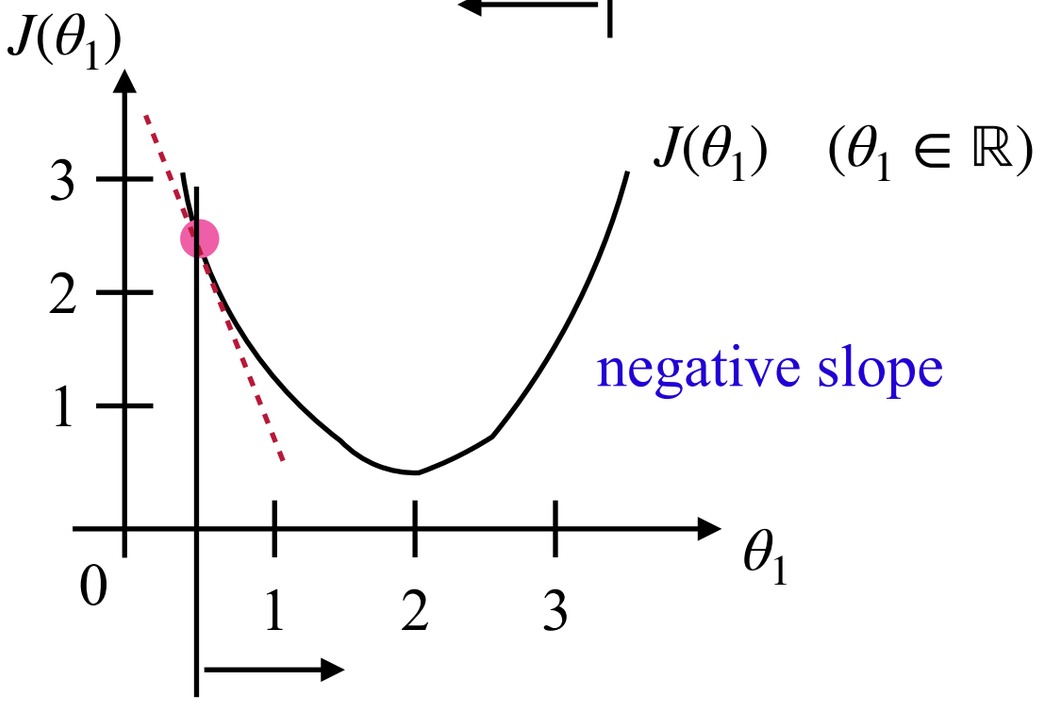
$$\theta_1 := \theta_1 - \alpha \cdot (+)$$

Gradient Descent Algorithm (Intuitively)



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

$$\theta_1 := \theta_1 - \alpha \cdot (+)$$



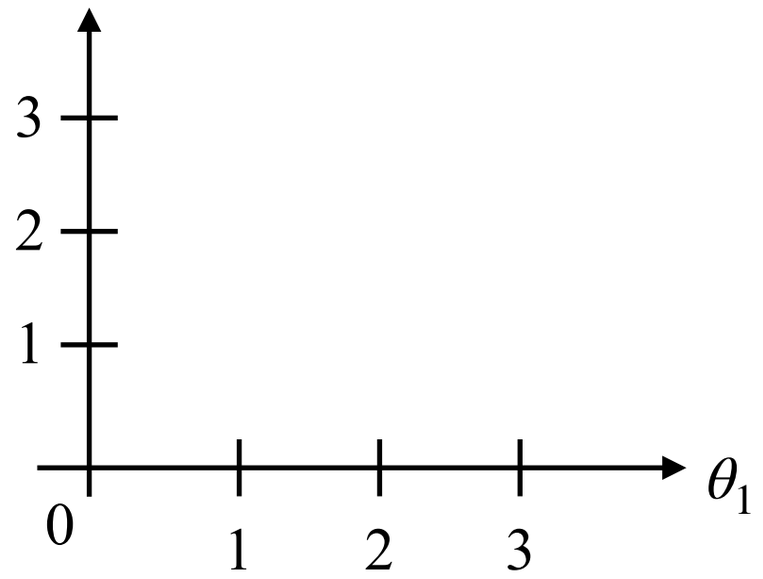
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

$$\theta_1 := \theta_1 - \alpha \cdot (-)$$

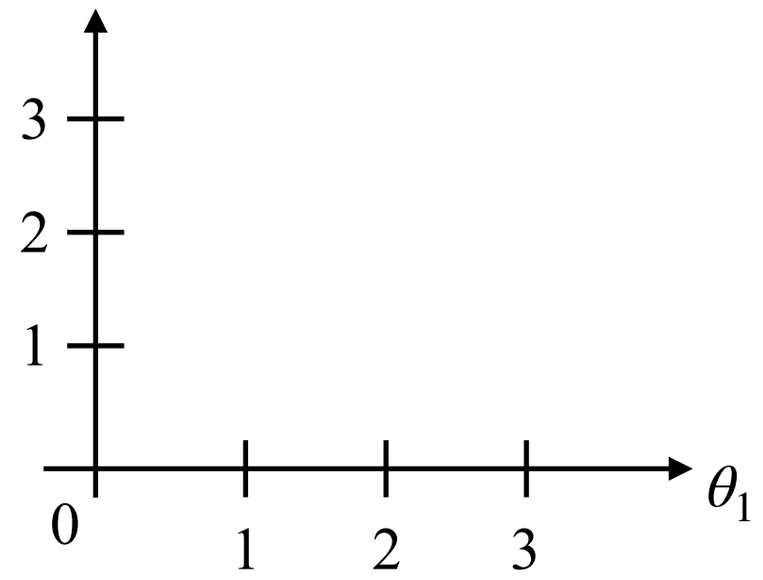
Gradient Descent Algorithm

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

If α is too small, gradient gradient can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Question

- Suppose θ_1 is at a local optimum of $J(\theta_1)$, such as shown in the figure. What will one step of gradient descent

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

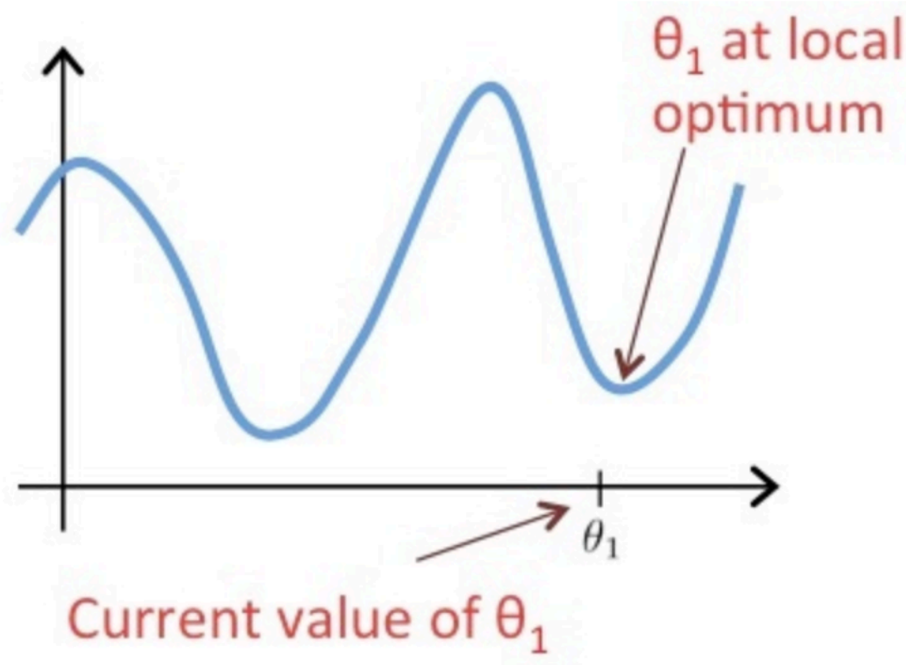
do?

(i) Leave θ_1 unchanged

(ii) Change θ_1 in a random direction

(iii) Move θ_1 in the direction of the global minimum

(iv) Decrease θ_1



Question

- Suppose θ_1 is at a local optimum of $J(\theta_1)$, such as shown in the figure. What will one step of gradient descent

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

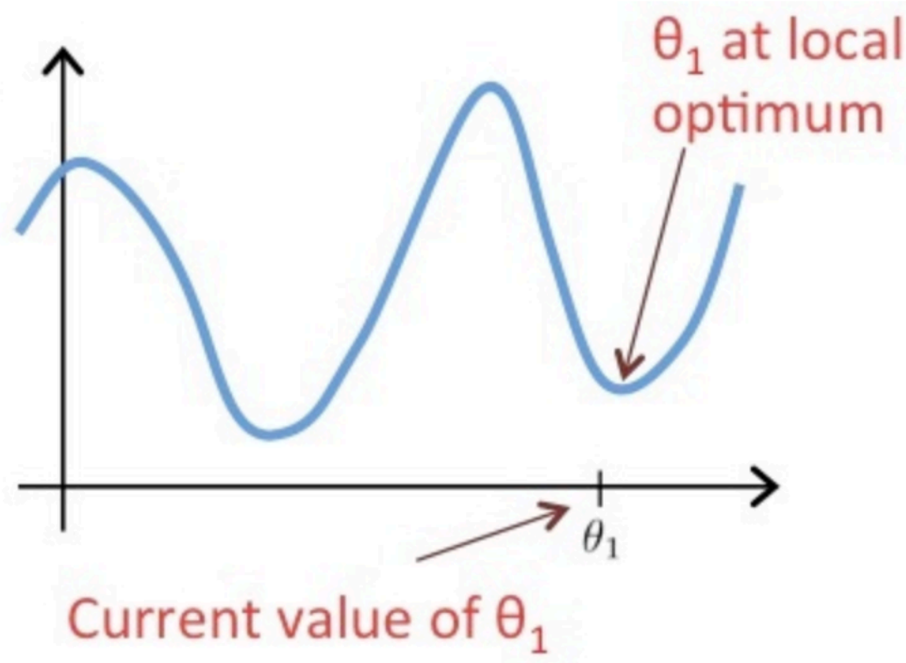
do?

(i) Leave θ_1 unchanged

(ii) Change θ_1 in a random direction

(iii) Move θ_1 in the direction of the global minimum

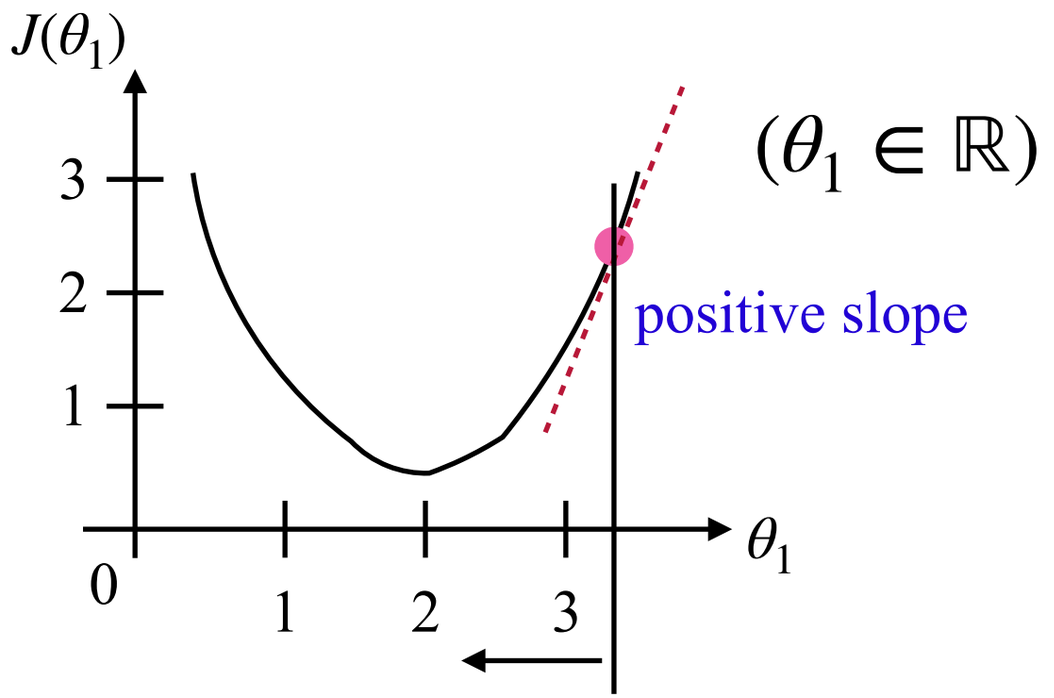
(iv) Decrease θ_1



Gradient Descent Algorithm (Intuitively)

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

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



As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.

Gradient Descent for Linear Regression

Gradient Descent Algorithm & Linear Regression Model

Gradient Descent Algorithm

repeat until convergence {

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

}

Linear Regression Model

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

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

Goal: Applying gradient descent to minimize the squared error function !

Gradient Descent Algorithm & Linear Regression Model

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

Gradient Descent Algorithm & Linear Regression Model

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

Gradient Descent Algorithm & Linear Regression Model

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

$$= \frac{\partial}{\partial \theta_j} [\frac{1}{2m} \Sigma_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2]$$

Gradient Descent Algorithm & Linear Regression Model

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

$$= \frac{\partial}{\partial\theta_j}[\frac{1}{2m}\Sigma_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2]$$

Using the chain rule

Case 1) when $j = 0$

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

Case 2) when $j = 1$

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

https://math.stackexchange.com/questions/70728/partial-derivative-in-gradient-descent-for-two-variables)

Gradient Descent Algorithm for Univariate Linear Regression

repeat until convergence {

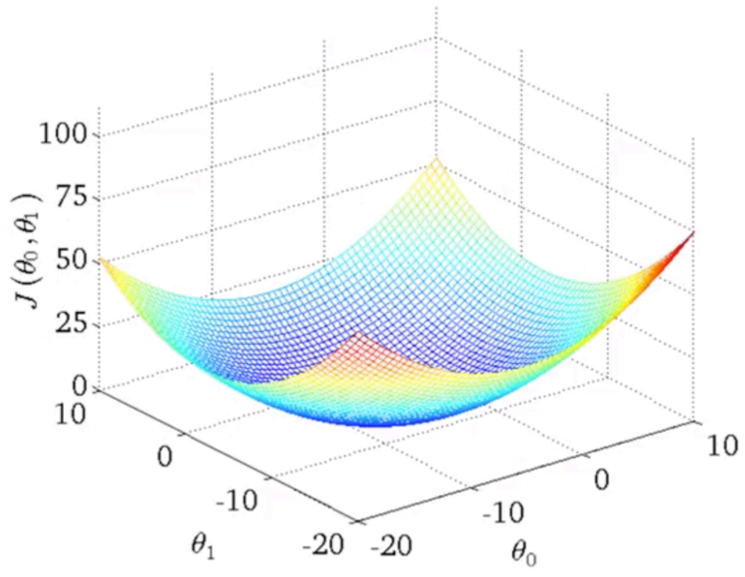
$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

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

}

Update θ_0 and θ_1 simultaneously

The cost function of linear regression has always the bowl shape like this.



Gradient Descent Algorithm for Univariate Linear Regression

repeat until convergence {

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

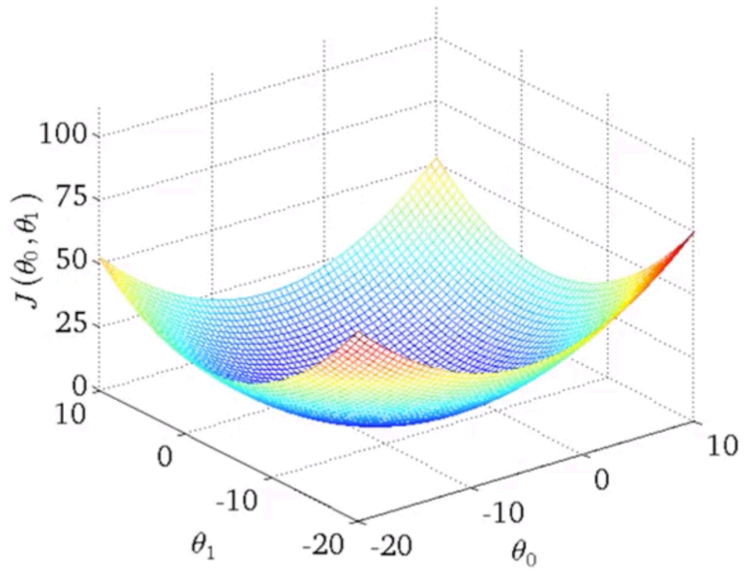
$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}

Update θ_0 and θ_1 simultaneously

The cost function of linear regression has always the bowl shape like this.

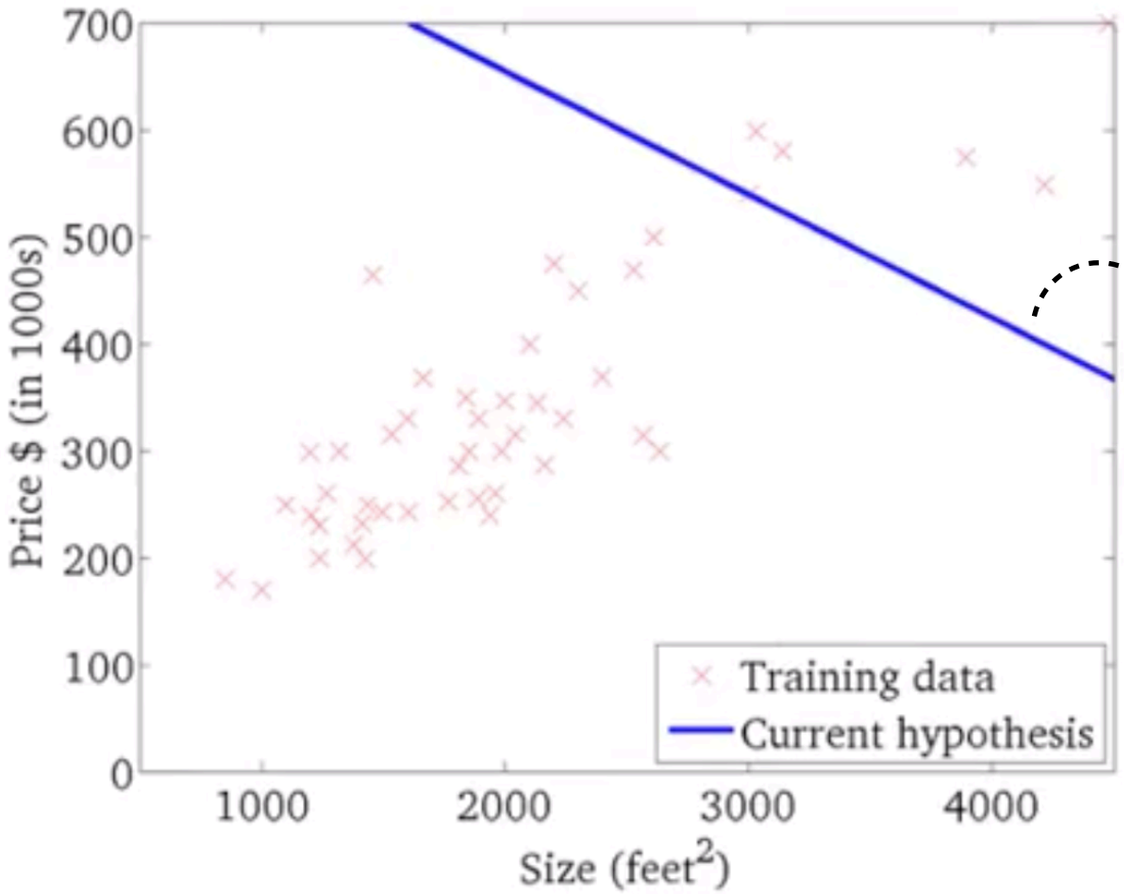
i.e. 'convex function'



Gradient Descent in Action

$h_\theta(x)$

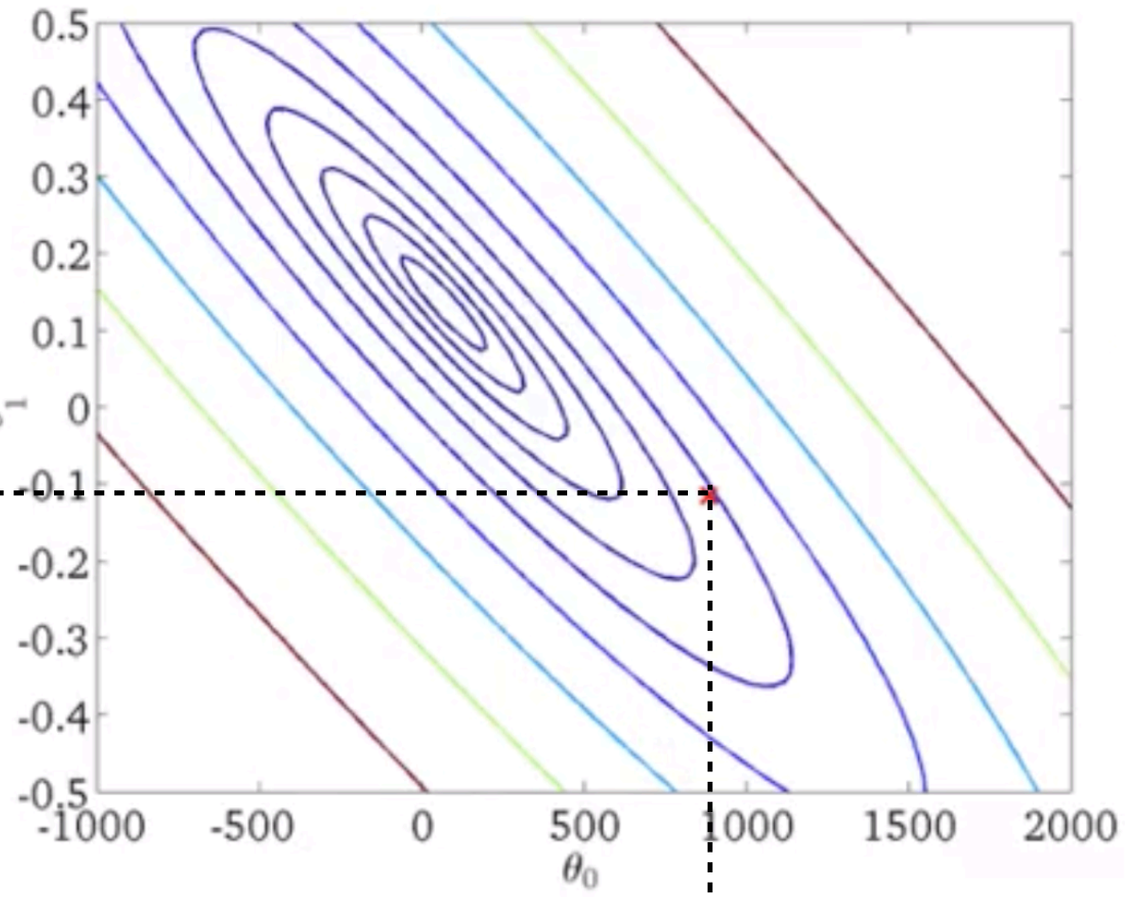
(for fixed θ_0, θ_1, this is a function of x)



$h_\theta(x) = 900 - 0.1x$

$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)

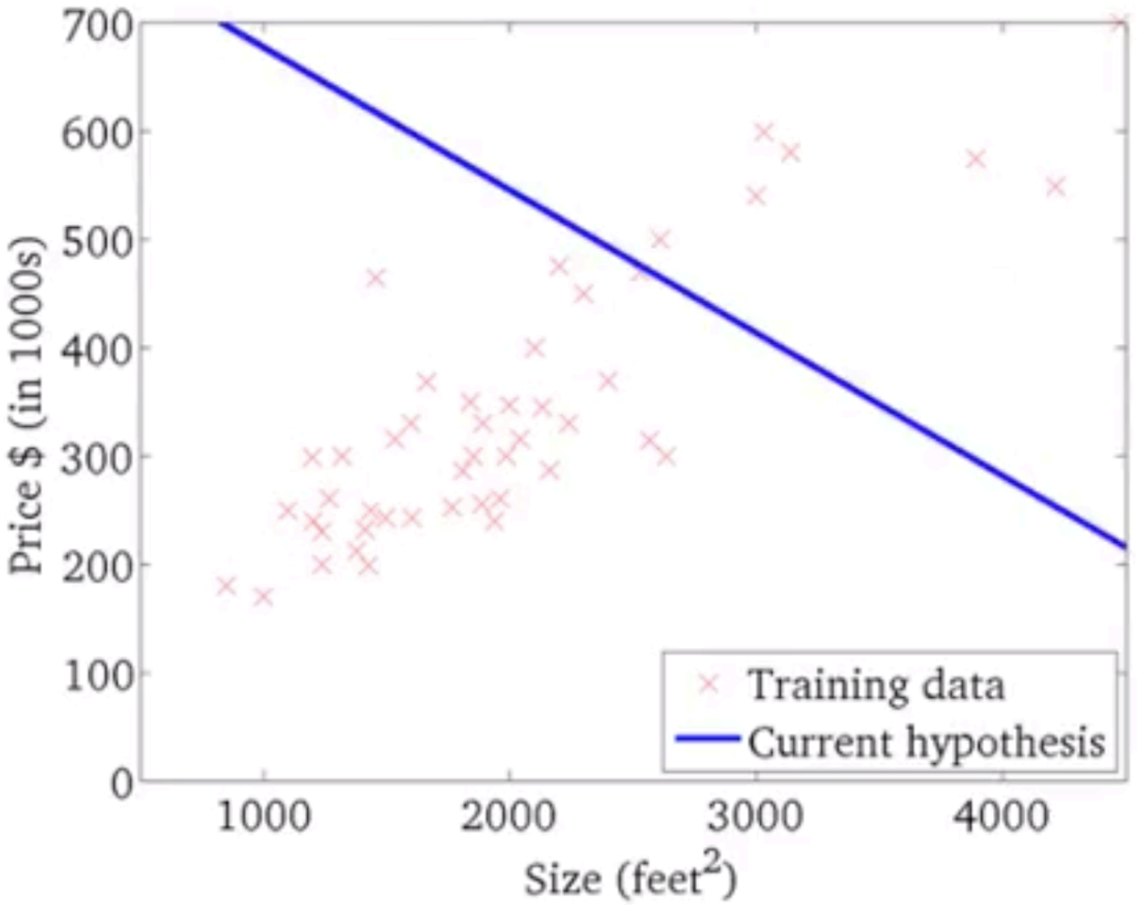


(source: Andrew Ng, machine learning)

Gradient Descent in Action

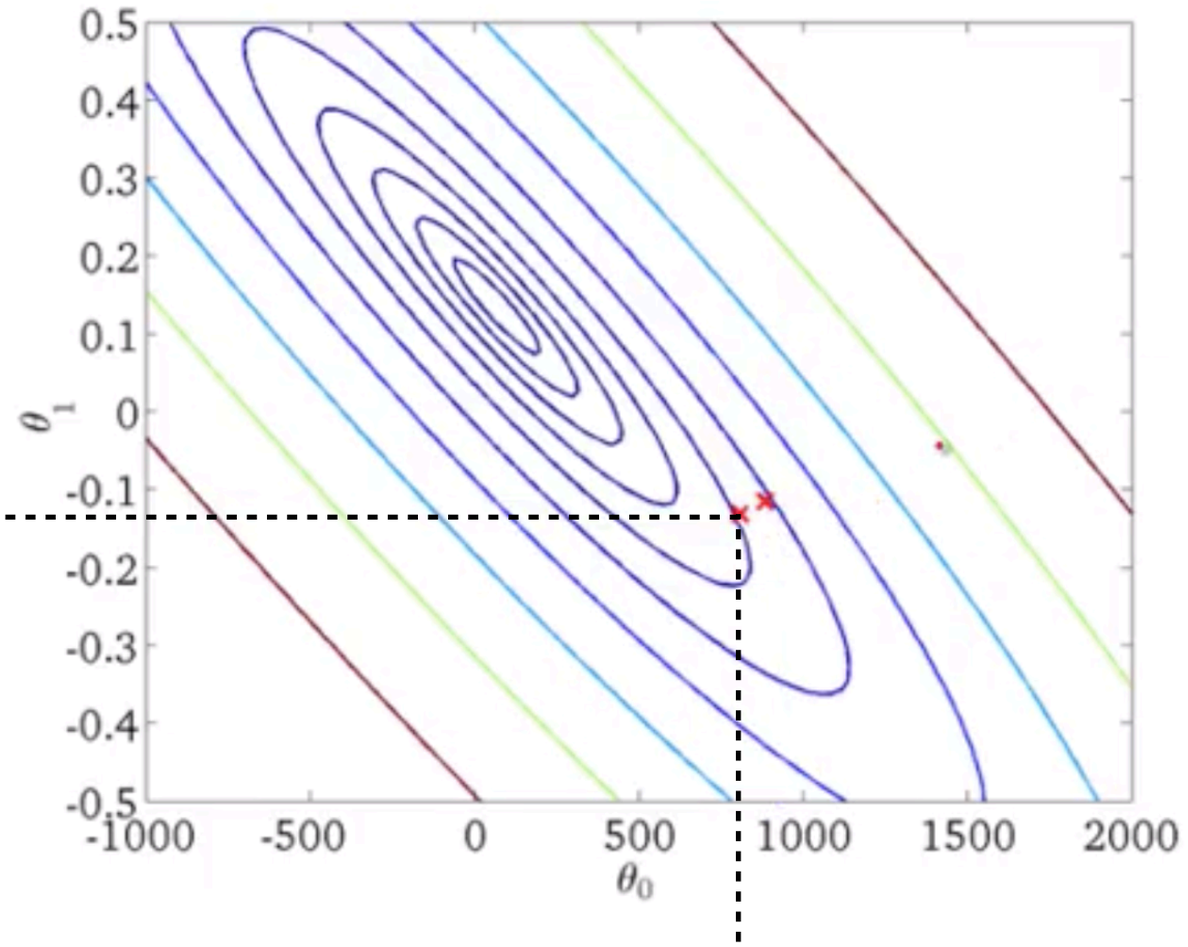
$h_\theta(x)$

(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)

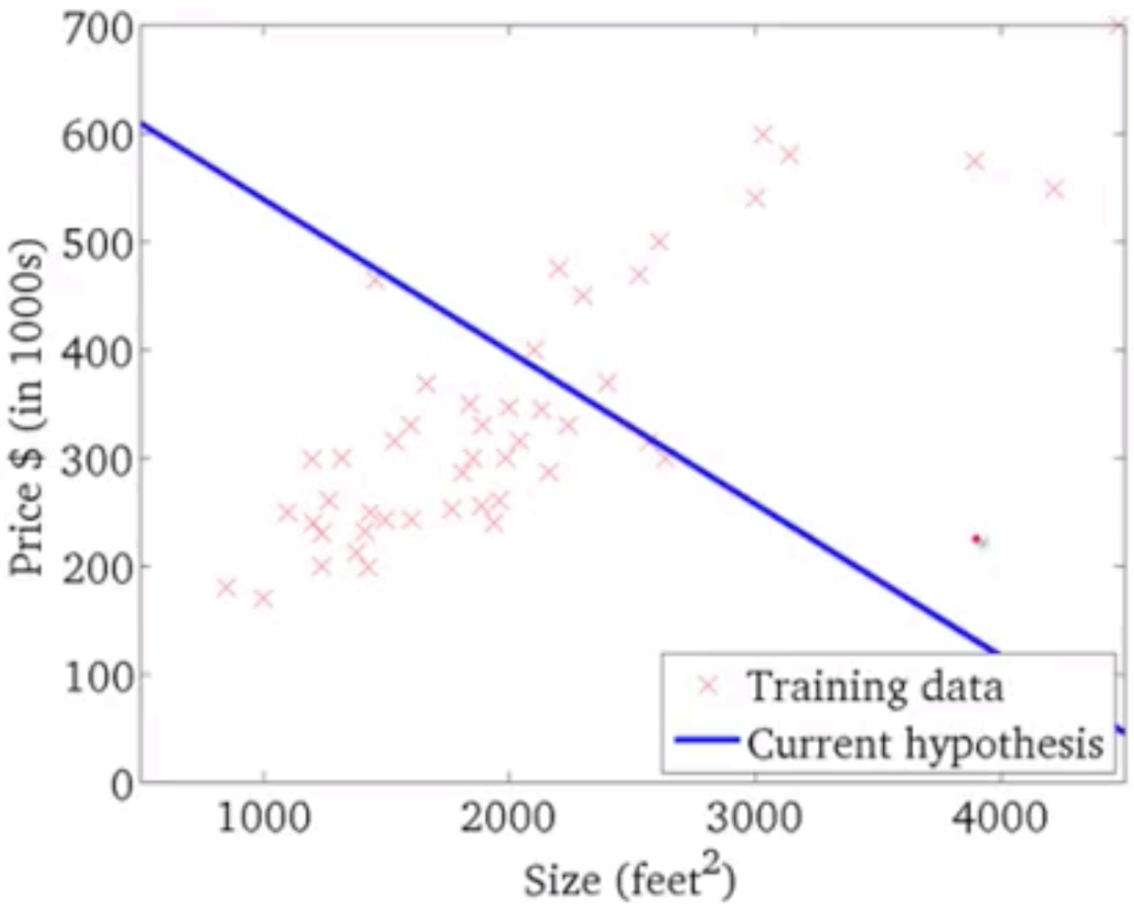


(source: Andrew Ng, machine learning)

Gradient Descent in Action

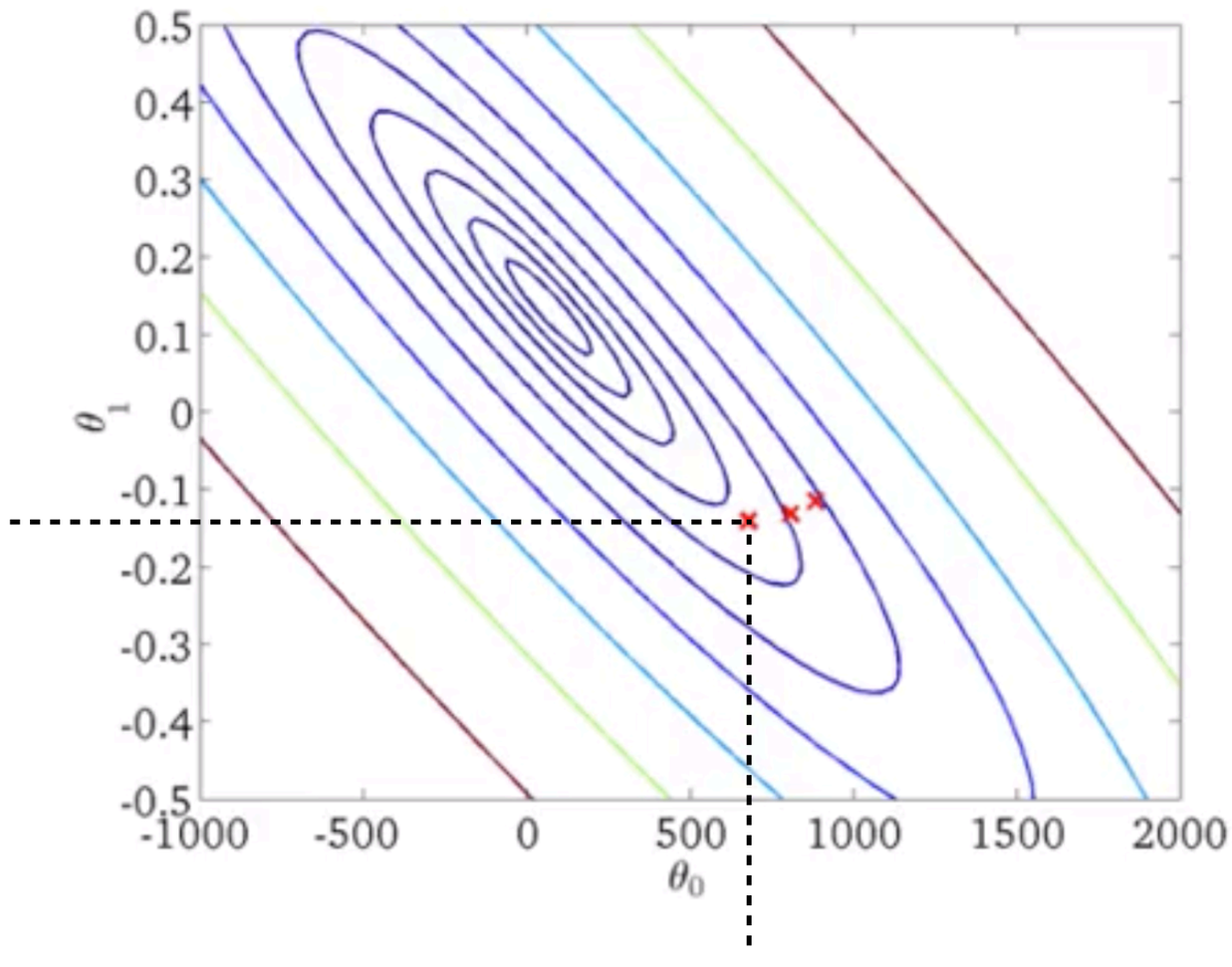
$h_\theta(x)$

(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)

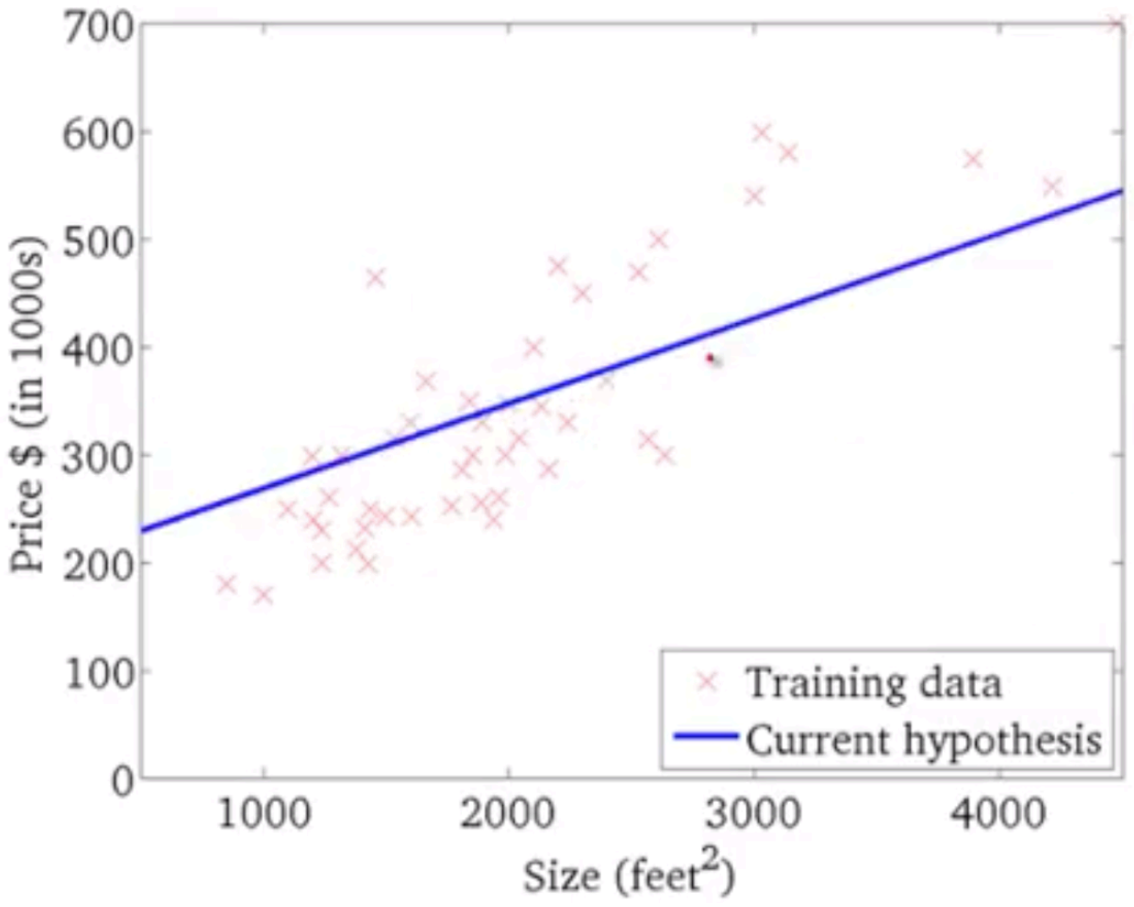


(source: Andrew Ng, machine learning)

Gradient Descent in Action

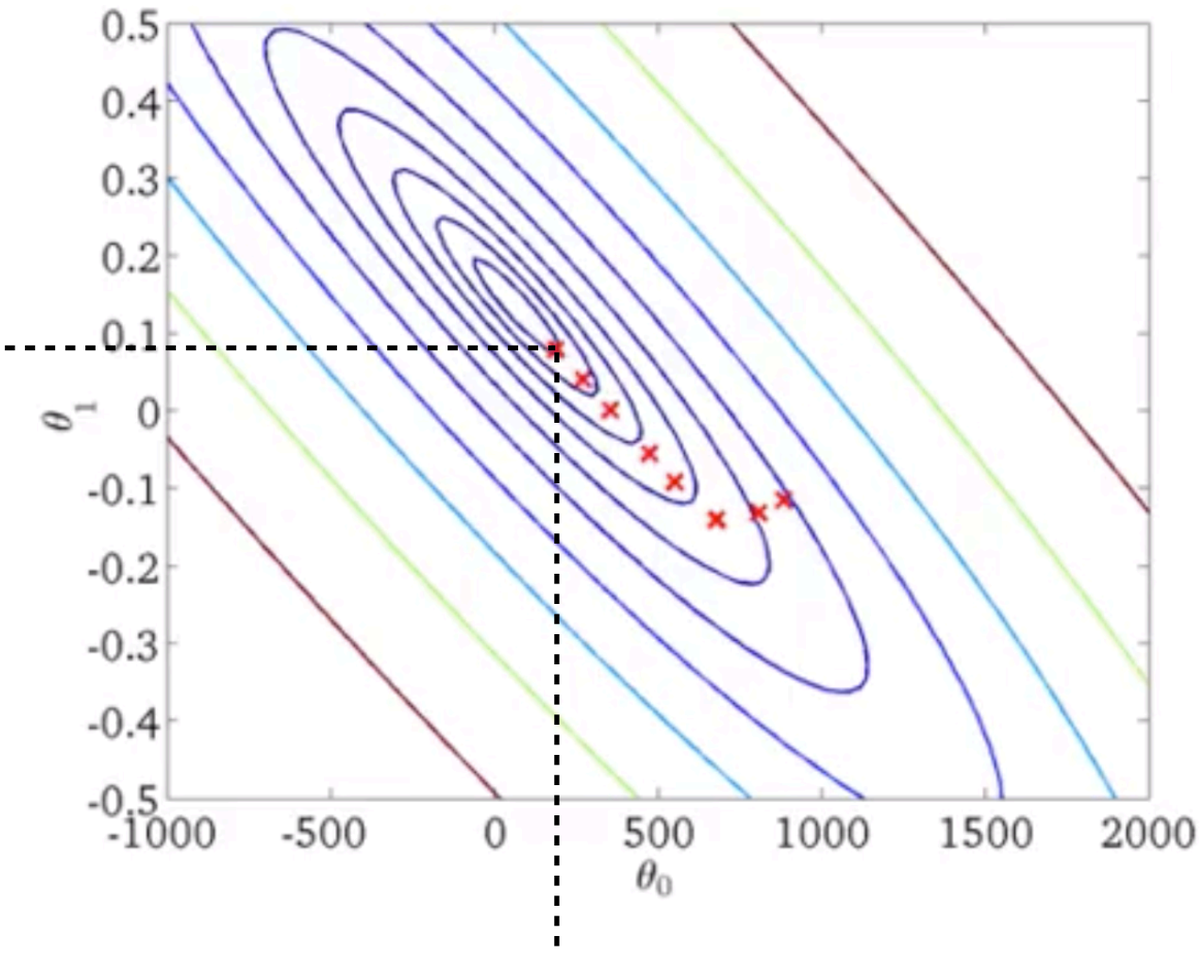
$h_\theta(x)$

(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)

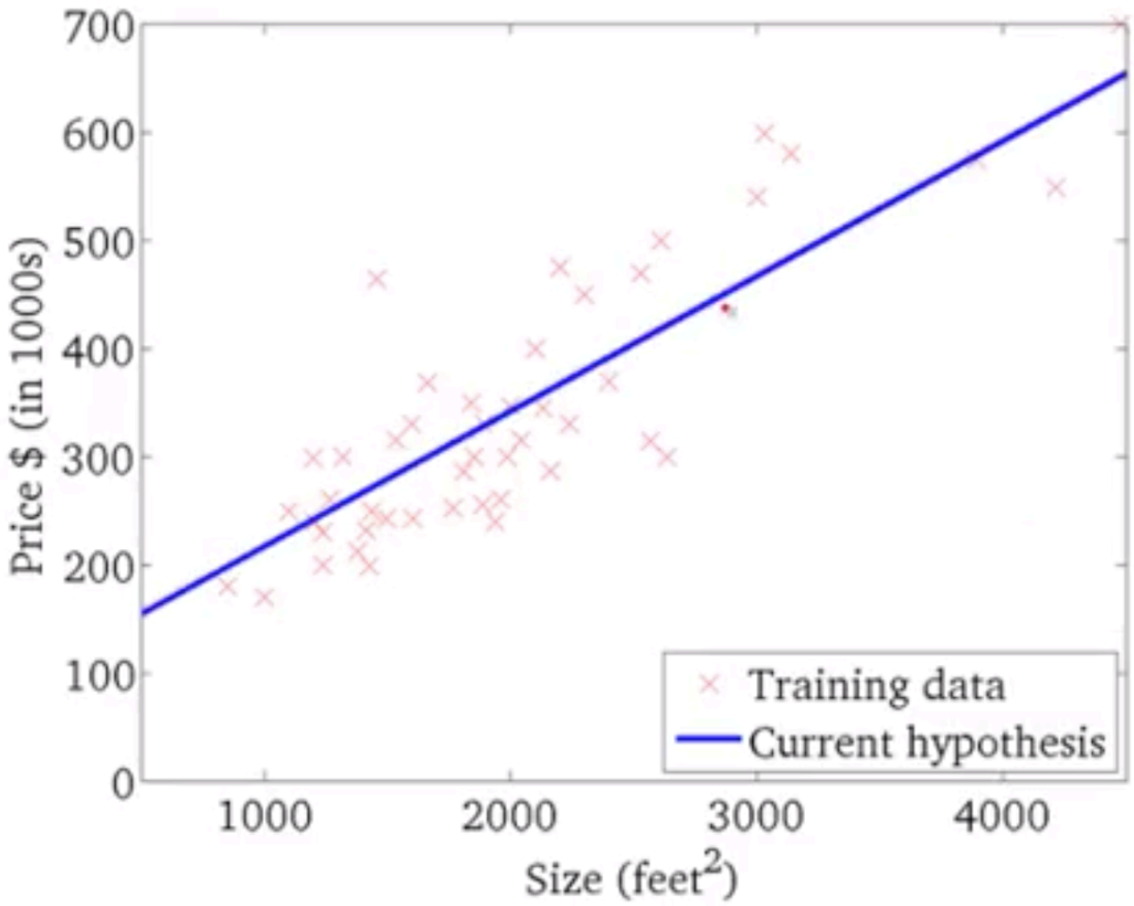


(source: Andrew Ng, machine learning)

Gradient Descent in Action

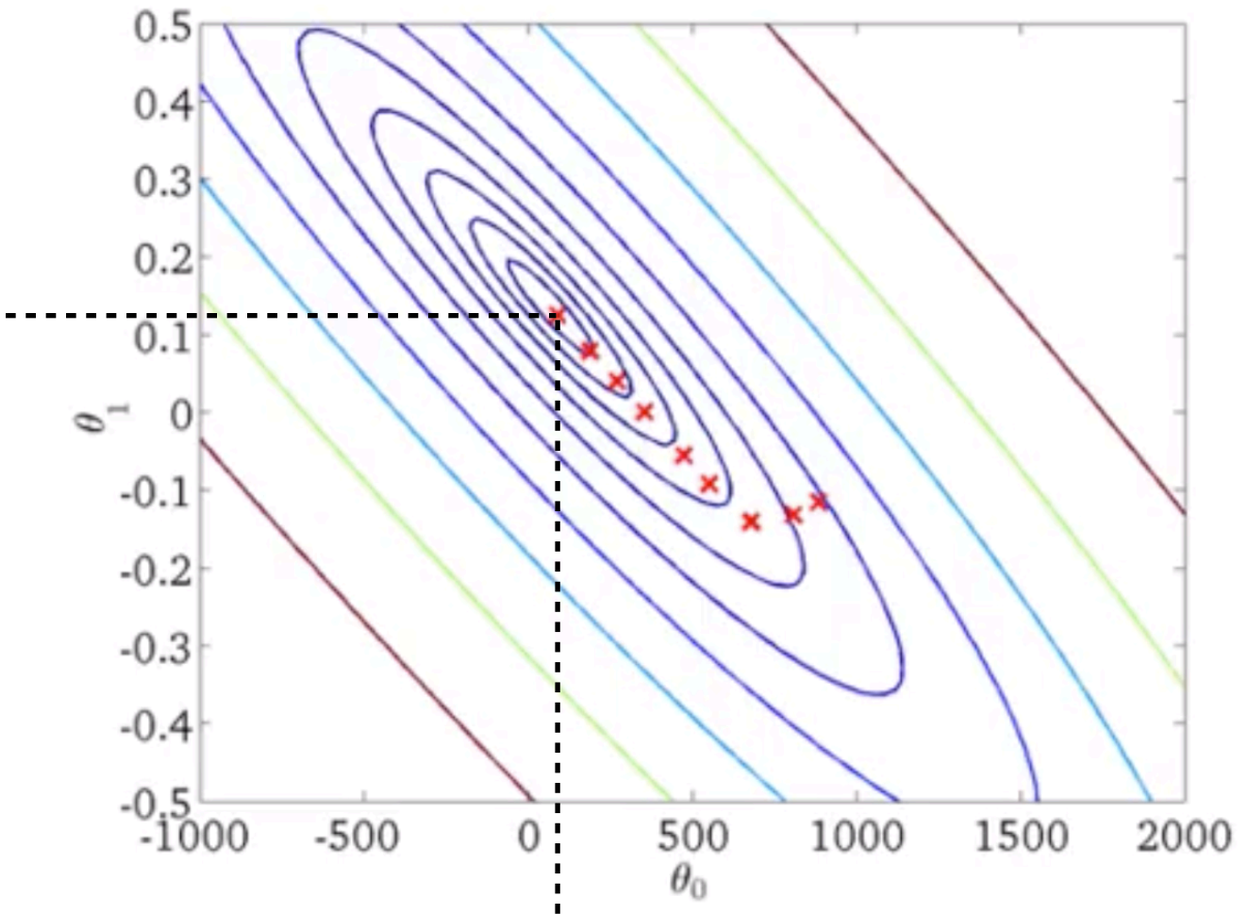
$h_\theta(x)$

(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)



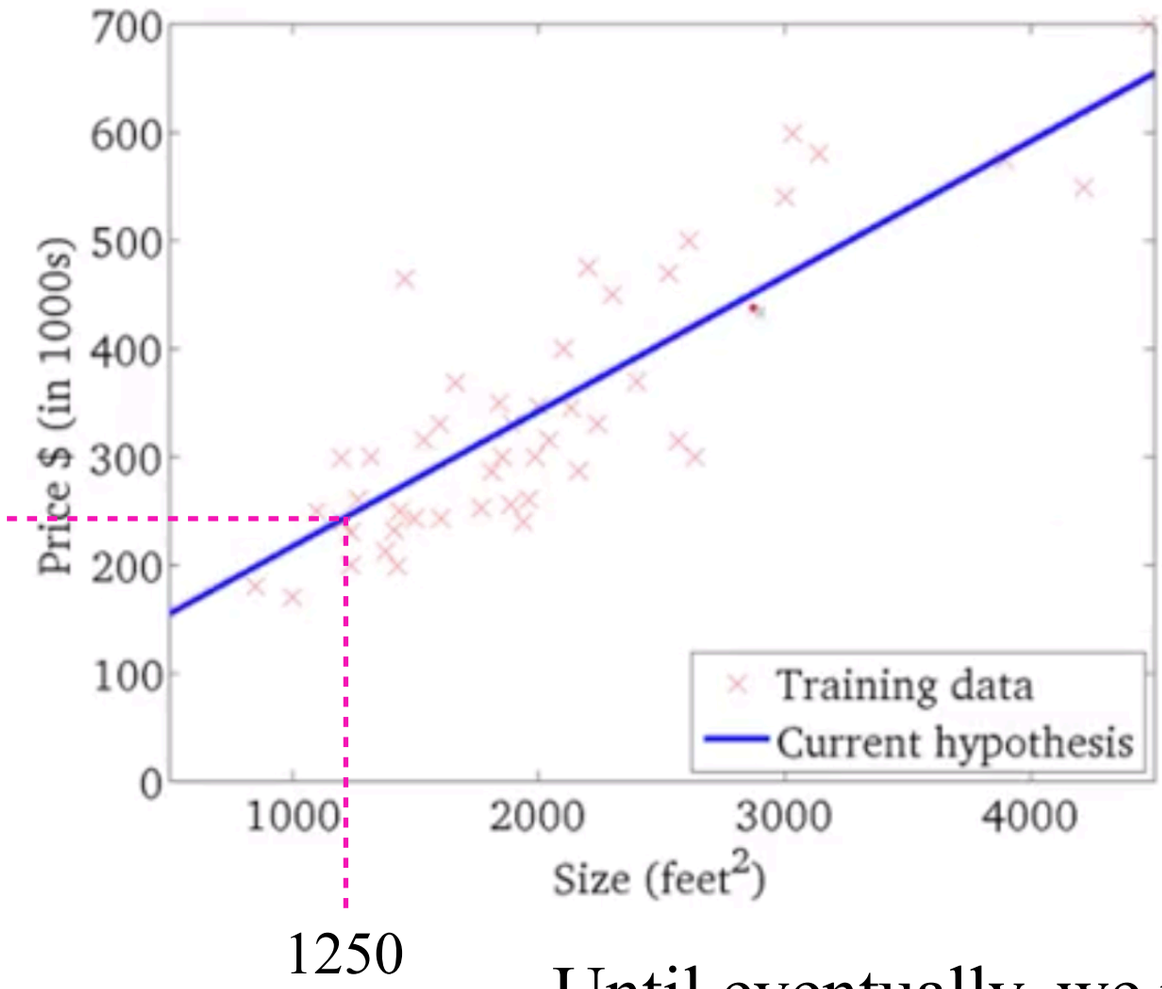
Until eventually, we reach the global minimum

(source: Andrew Ng, machine learning)

Gradient Descent in Action

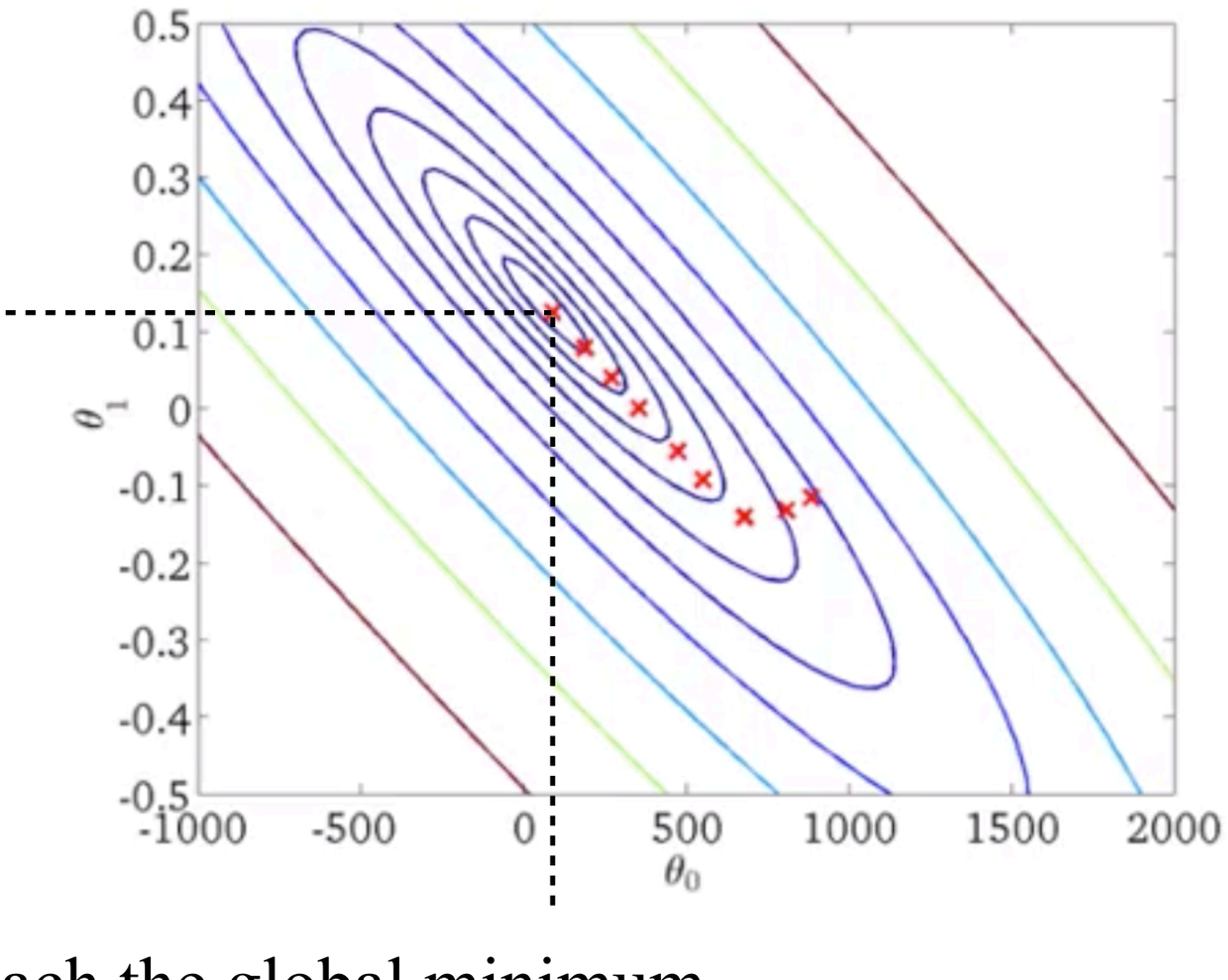
$h_\theta(x)$

(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)



Until eventually, we reach the global minimum

(source: Andrew Ng, machine learning)

Gradient Descent in Action

- ‘Batch’ gradient descent
- ‘Batch’ means “each step of gradient descent uses all the training examples” *i.e.*

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

- One alternative frequently used in machine learning is stochastic gradient descent, in which we repeatedly update using the gradient calculated for each training element.

Stochastic Gradient Descent

$\theta_0, \theta_1 :=$ some initial guess

repeat until convergence {

For $i \in \{1, \ldots, m\}$

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

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

}

Stochastic gradient descent tends to get close to the minimum faster than batch gradient descent, but may oscillate around the minimum.

Question

- Which of the following are true statements? Circle all that apply.

(i) To make gradient descent converge, we must slowly decrease α over time.

(ii) Gradient descent is guaranteed to find the global minimum for any function $J(\theta_0, \theta_1)$

(iii) Gradient descent can converge even if α is kept fixed (But α cannot be too large, or else it may fail to converge).

(iv) For the specific choice of cost function $J(\theta_0, \theta_1)$ used in linear regression, there are no local optima (other than the global optimum).

Question

- Which of the following are true statements? Circle all that apply.

(i) To make gradient descent converge, we must slowly decrease α over time.

(ii) Gradient descent is guaranteed to find the global minimum for any function $J(\theta_0, \theta_1)$

(iii) Gradient descent can converge even if α is kept fixed (But α cannot be too large, or else it may fail to converge).

(iv) For the specific choice of cost function $J(\theta_0, \theta_1)$ used in linear regression, there are no local optima (other than the global optimum).

Summary

OK! Now, we have 2 methods for minimizing $J(\theta)$:

1. Batch gradient descent
2. Stochastic gradient descent

Are there non-iterative methods? what's about normal equations?

Let's discuss more about normal equations in the next chapter.

Why mean squared error?

- Why not some other cost function?

For linear regression, least squares and maximum likelihood are equivalent.

The equivalence comes from the assumption of independently and identically distributed Gaussian errors in our samples $y^{(i)}$.

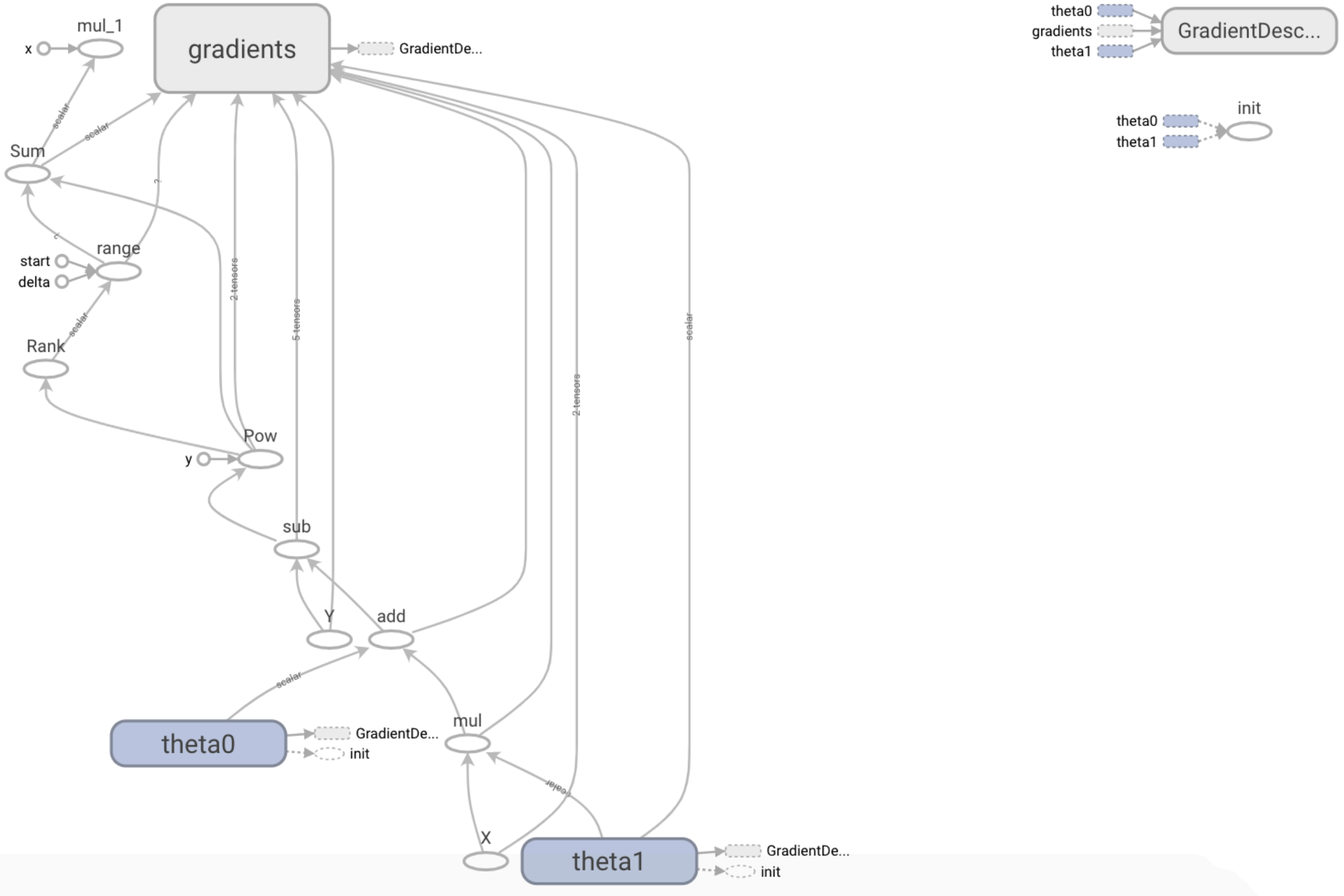
Let's discuss more about this in the next chapter, too.

First TensorFlow Tutorial

Now that we understand the math.

Let’s see what TensorFlow can do for us !

What variables to update?



Optimizer

- GradientDescentOptimizer means that we're running gradient descent !
- This means TensorFlow:
  - does auto-differentiation for us,
  - updates the value of each parameter.
- This is not the only optimizer provided in TensorFlow.
  - see its official website for more details
  - see this blog for comparisons among these optimizers.