parameter learning $\boldsymbol{\theta}$ gradient descent basics

gradient descent

the use of gradient descent as a mechanism for minimizing the function of $J(\theta_1, \theta_0)$

note: for brevity, the functions $J(\theta_1, \theta_0)$ are shortened throughout the document

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

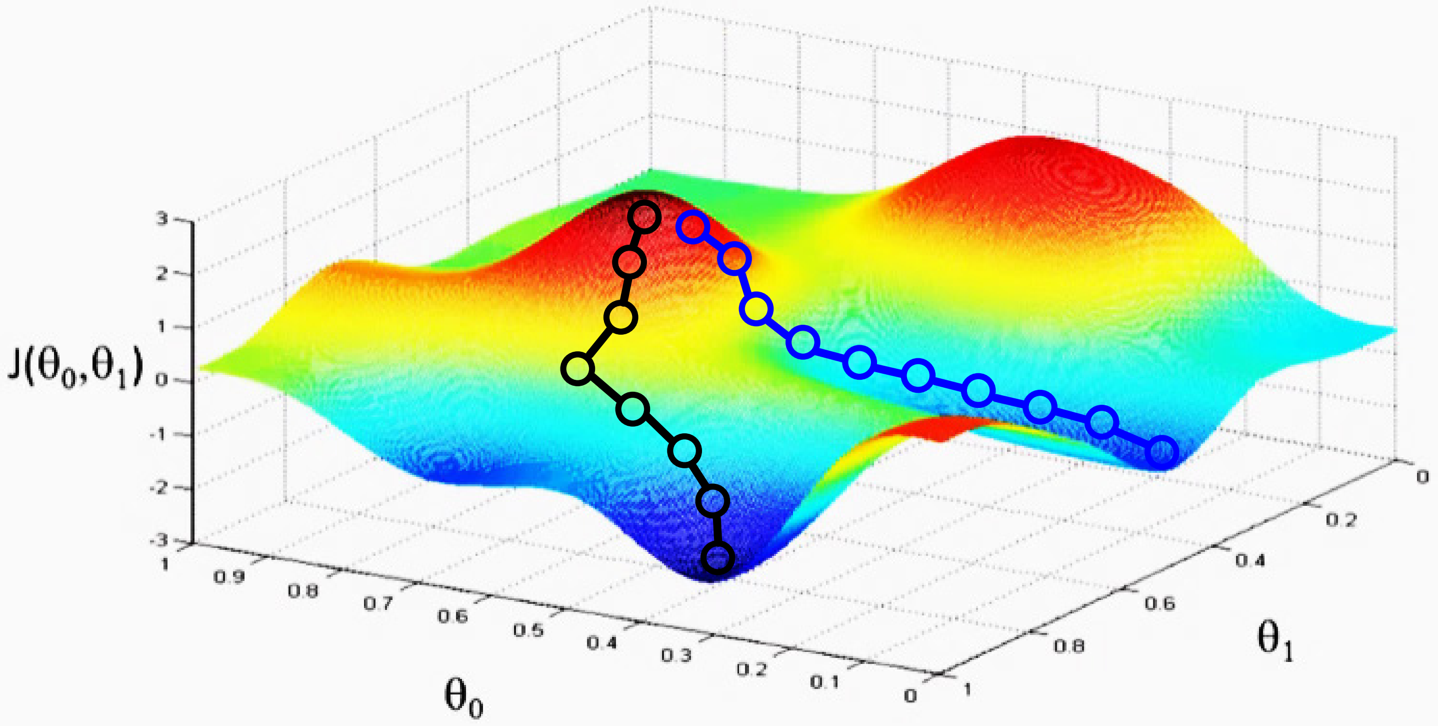
$$\min_{\theta_0, \theta_1, \theta_2, \ldots, \theta_n} J(\theta_0, \theta_1, \theta_2, \ldots, \theta_n) \to \min_{\theta_0, \theta_1} J(\theta_1, \theta_0)$$

the process begins by initializing some value of parameters θ_0, θ_1 (e.g. $\theta_0 = 0, \theta_1 = 0$)

the values θ_0, θ_1 will then change through reiterations to reduce the value of $J(\theta_0, \theta_1)$

the reiterations will run continuously with the intended arrival at a minimum

illustrated below, the initialized values will be plotted against the function:



in narrative explanation, gradient descent begins at the points initialized and scans the contour in a 360°-like fashion and chose a direction the descends the counter in the most rapid route. the initialized values will update to the new point on the contour and repeat the process until the functions determines it has reached a **local minimum**

a unique property of gradient descent occurs dependent upon the initialized values of parameters . different initialization values can result in a disjoint **local minimum** the formal gradient descent algorithm is as follows:

repeat until convergence {

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

}

the a term determines the magnitude of each step in gradient descent

the term $\frac{\partial}{\partial\theta_j}$ is the partial derivative active in gradient descent

the gradient descent algorithm functions as a simultaneous update of θ_0, θ_1:

<u>correct simultaneous update:</u>

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

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

$$\theta_0 := \text{temp0}$$

$$\theta_1 := \text{temp1}$$

<u>incorrect simultaneous update:</u>

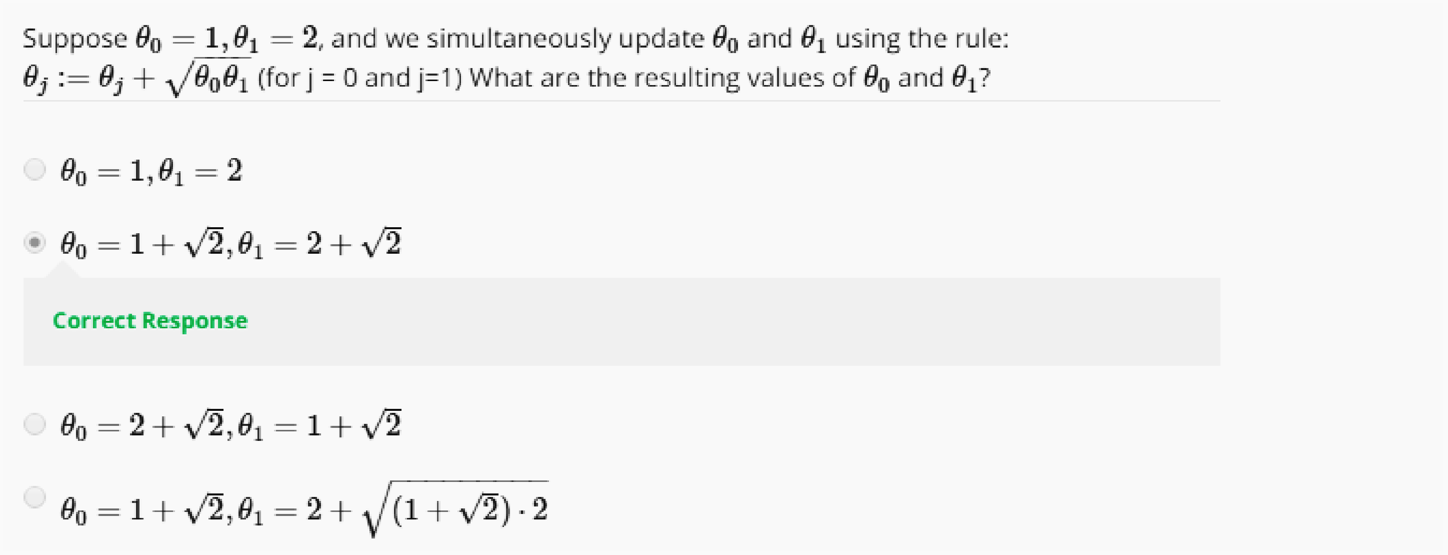
$$\text{temp0} := \theta_0 - \alpha\frac{\partial}{\partial\theta_0}J(\theta_0,\theta_1)$$

$$\theta_0 := \text{temp0}$$

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

$$\theta_1 := \text{temp1}$$

the incorrect example of simultaneous update displays the use of the updated value of θ_0 in the computation of updating θ_1



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?

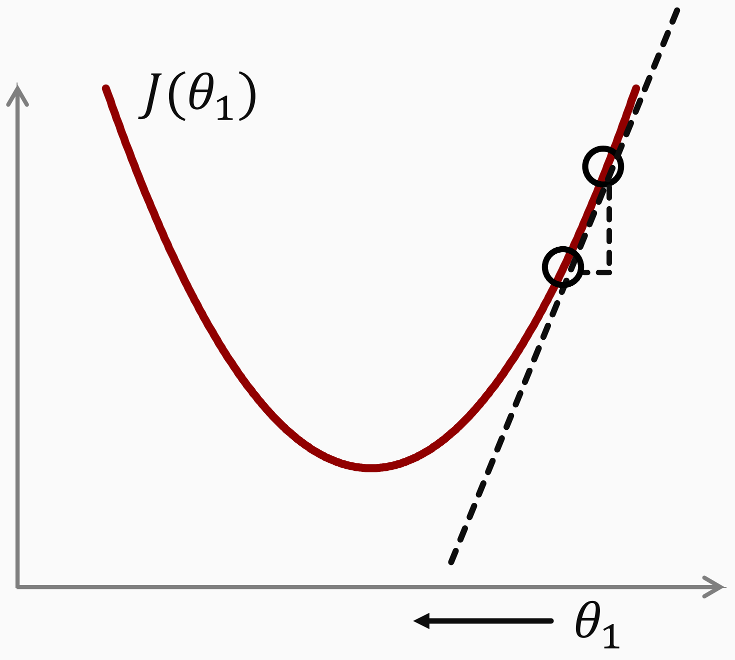
- $\theta_0 = 1, \theta_1 = 2$
- [✓ $\theta_0 = 1 + \sqrt{2}, \theta_1 = 2 + \sqrt{2}$]

Correct Response

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

gradient descent intuition

the function of the derivative in the gradient descent function takes the tangent slope of the line plotted against θ_1 and multiplies it against learning rate alpha α and cost function $J(\theta_1)$; the functions then apples the direction and magnitude into a step

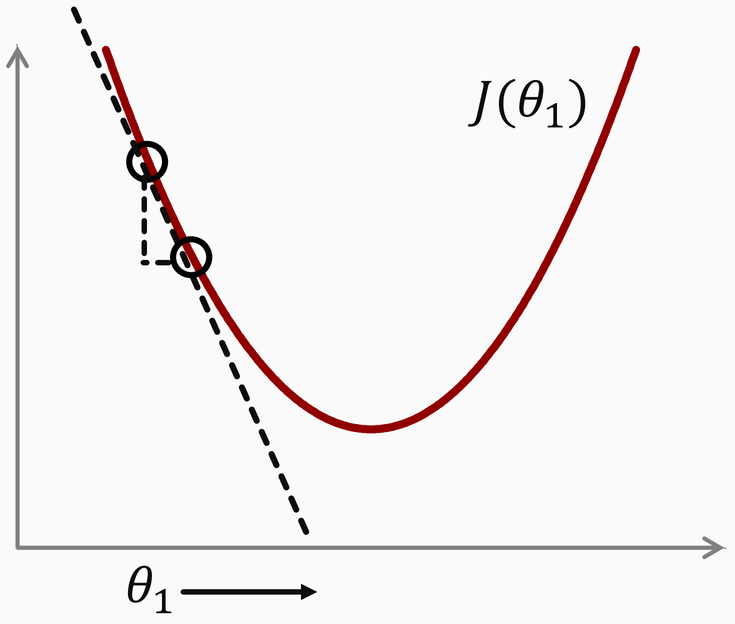


$(\theta_1 \in \mathbb{R})$

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

$$\geq 0$$

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



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

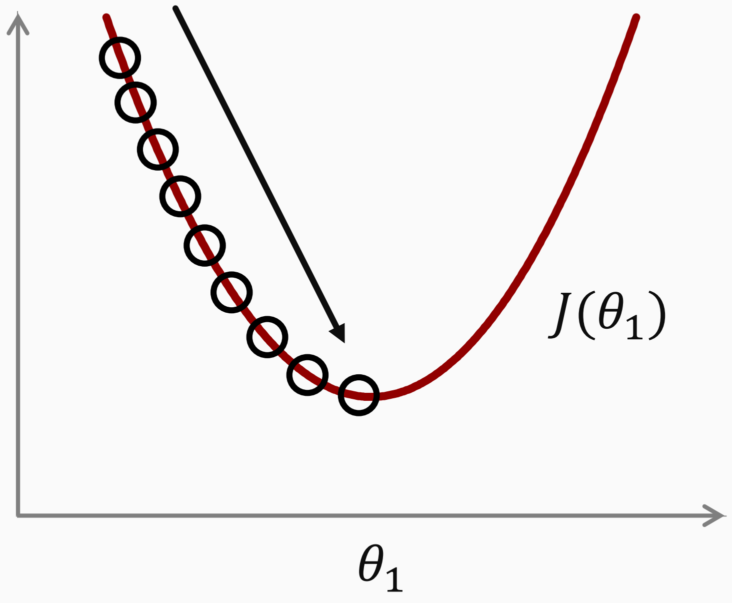
$$\leq 0$$

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

the initialized values in the gradient descent function will determine whether the tangent slope of the derivative will require a **positive** or **negative** number to make a step towards finding a local minimum with reference to the behavior of the assigned learning rate α:

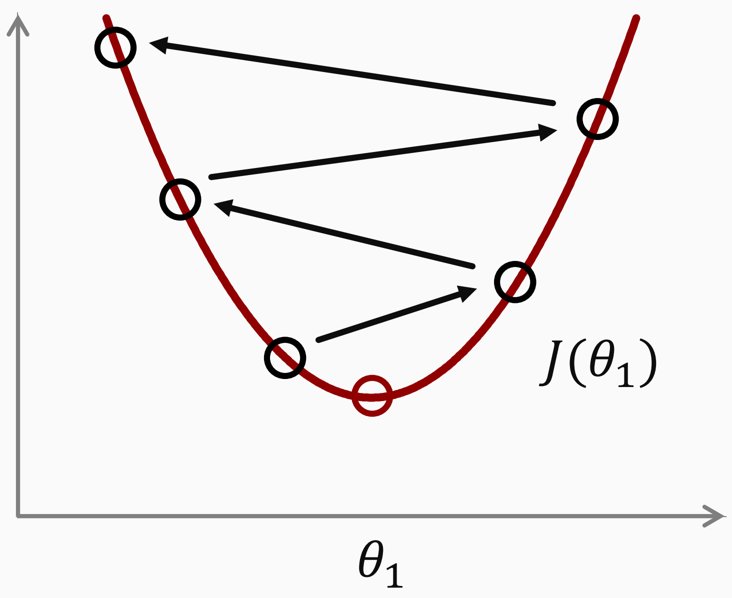
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

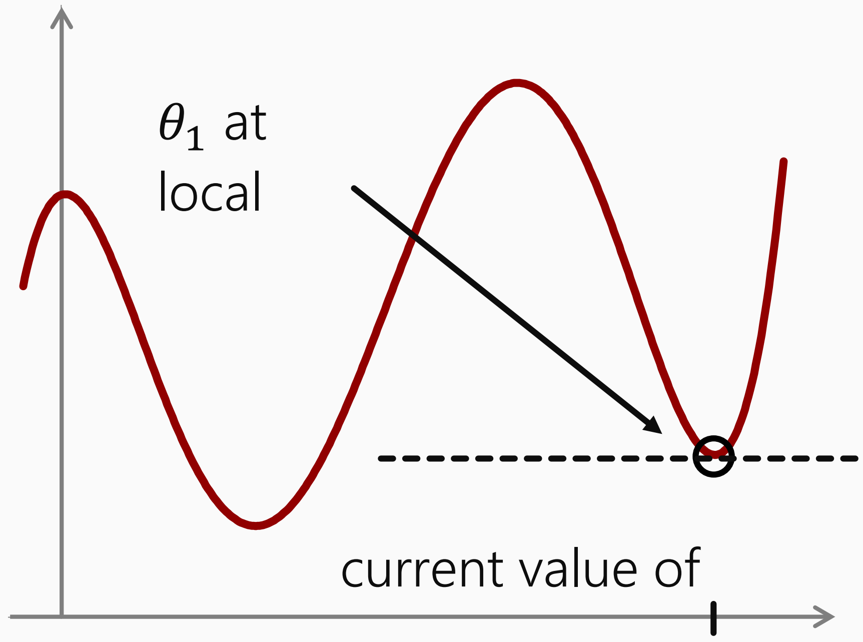
if the learning rate α is too **small**, gradient descent can processes slowly and take substantial time and resources converging to a local optima (minimum)



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

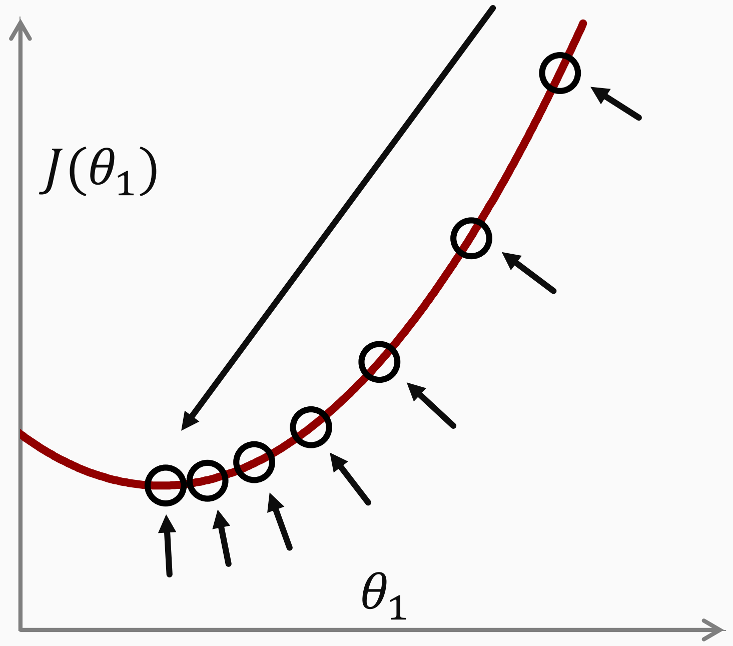
if the learning rate α is too **large**, gradient descent can overshoot the local optima (minimum); it might equally fail to converge, or diverge entirely from obtaining a local optima (minimum)





if θ_1 is already at a local optimum of $J(\theta_1)$, a single additional step gradient descent $\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$ will leave θ_1 unchanged

at local minimum, the tangent slope of the derivative = 0 and thus will not affect θ_1



if the learning rate alpha α is fixed, gradient descent can still converge to a local minimum

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

after each simultaneous update of gradient descent, the algorithm autonomously takes smaller steps due to a decreasing derivative computation with each iteration

gradient descent for linear regression

applying gradient descent to minimize the linear regression cost function:

gradient descent algorithm

repeat until convergence {

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

$$(\text{for } j = 0 \text{ and } j = 1)$$

}

linear regression model

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

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

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

determining the function of the derivative term $\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)$:

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

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

determining the partial derivatives in both cases of θ_0, θ_1

$$\theta_0 \rightarrow j = 0: \frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

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

the partial derivatives are simply the slopes of the cost function $J(\theta_0,\theta_1)$

gradient descent algorithm for linear regression:

repeat until convergence {

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

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

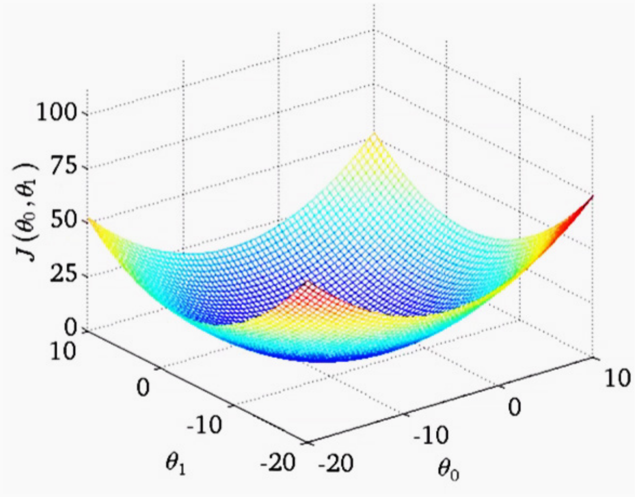
update θ_0 and θ_1 simultaneously

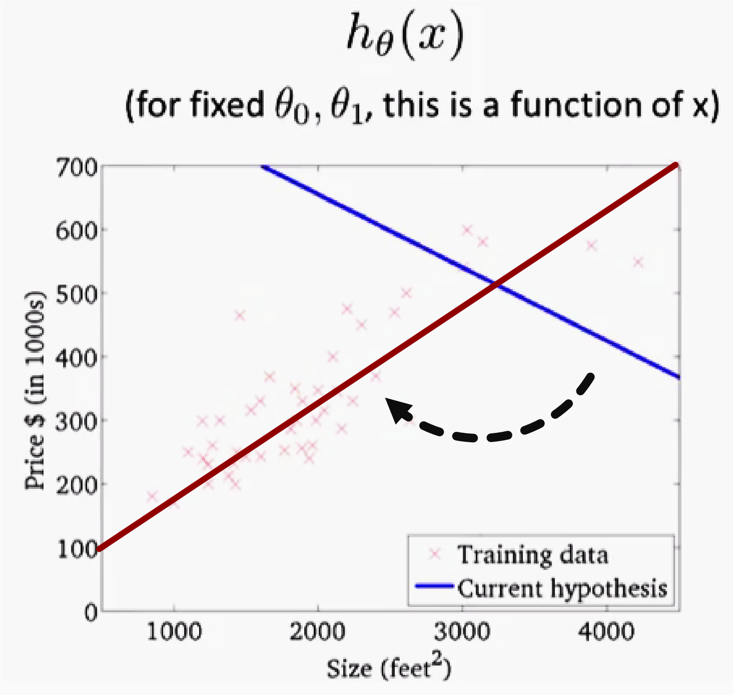
}

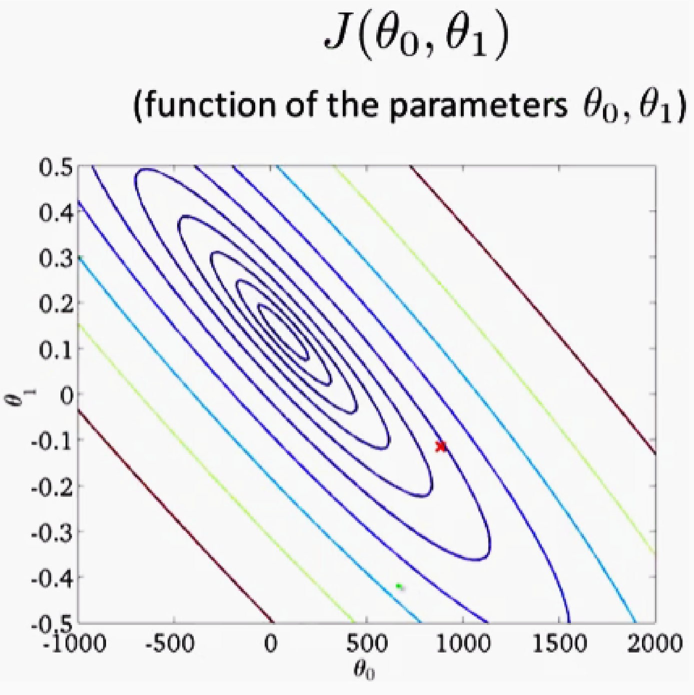
the cost function for linear regression will always be represented as a **convex function**

"**batch**" gradient descent is a common term in machine learning to refer to each step of gradient descent using all of the training examples in the dataset

convex functions do not consist of any local optima, there is now only an available **global optima** (global minimum) of the convex function


$J(\theta_0,\theta_1)$
θ_1
θ_0


$h_\theta(x)$
(for fixed θ_0, θ_1, this is a function of x)
Price $ (in 1000s)
Size (feet^2)
Training data
Current hypothesis


$J(\theta_0,\theta_1)$
(function of the parameters θ_0, θ_1)
θ_1
θ_0