Gradient Descent

θ_0, θ_1

- Now we know how to evaluate a model, using a cost function, how do we make the model ***learn*** the optimal parameters ?
- In other words, how do we **minimize** the cost function **without testing all** the different possible models ?

- The algorithm used to do this is called ***Gradient Descent***, and is **essential** to most machine learning algorithms, not just linear regression !
- In DL libraries this type of algorithm is called an **'optimizer'** and other variants exist.

Gradient Descent

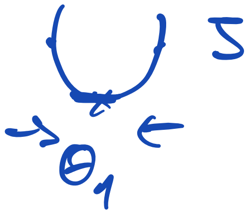
- We have some function $J(\theta_1, \theta_2)$ which we want to minimize…

$\theta_1 \to +$, $\theta_1 \to -$

$\theta_2 \to +$, $\theta_2 \to -$

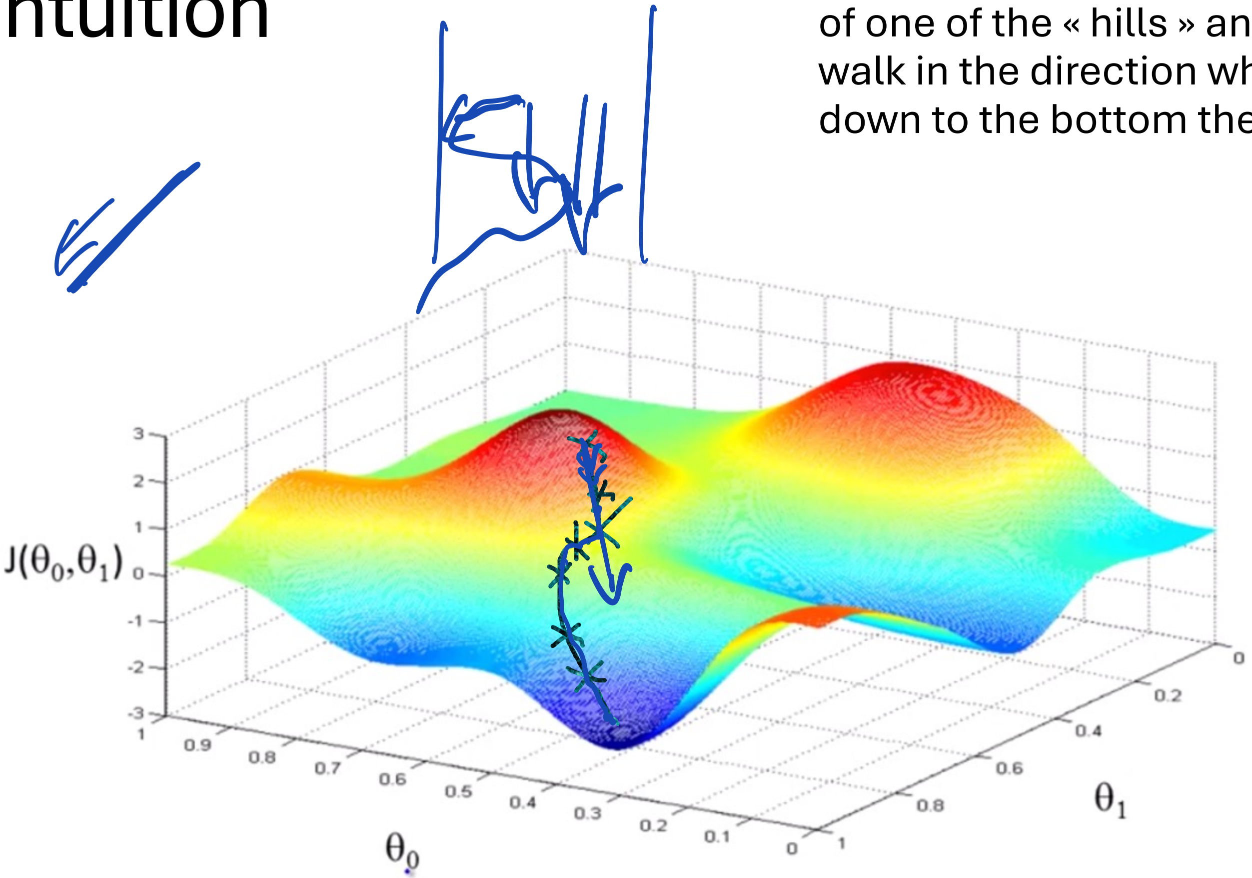
Outline :

- Start with some inital guess, some random values for θ_1, θ_2
- Keep **updating** θ_1, θ_2 a little bit to reduce $J(\theta_1, \theta_2)$ until we end up at a **minimum** (global or local)



GD intuition

- This is your cost function in 3D
- Imagine you start somewhere near the top of one of the « hills » and your goal is to walk in the direction which will take you down to the bottom the fastest.



GD formula

repeat until convergence {

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

}

$\theta_{j\_new} : \theta_j - \boxed{10}$

- This is the update formula for each of the parameters
- := signifies assignment
- α is a number called the ***learning rate***. If α is very **large**, then it corresponds to an **aggressive** learning procedure and big steps being taken « downhill » and vice versa.
- $\boxed{\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)}$ is a derivative term, which requires a bit of calculus

GD Intuition

- Why does this update make sense ?
- Why are we putting those 2 terms together ?

- Let’s try and get a basic understanding of derivatives before we go any further.

Derivatives

- The derivative describes **how the output of a function varies** with regard to a very very very **tiny positive nudge** to the **input**, to the point where we consider *almost* no variation in input….

- Informally, the deriviative tells you how a function behaves at a particular « instant », i.e. for a given input value.

- The derivative is commonly refered to as « **instantaneous rate of change** »

Derivatives

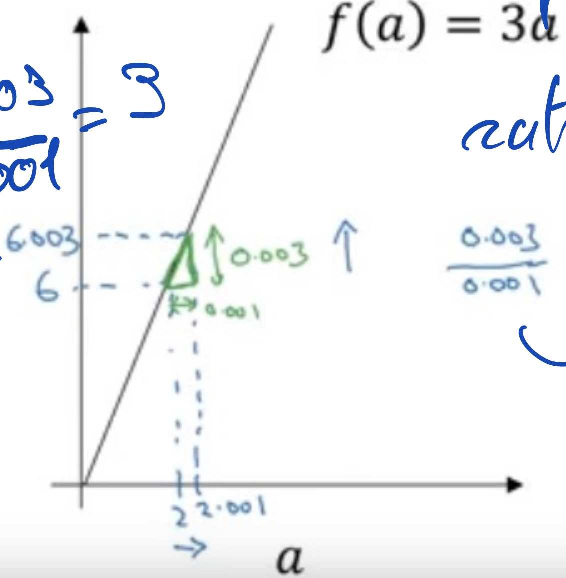
- Here is a linear function as an example. What happens when we shift the input by a ‘small’ value like 0.001
- when a=2 ?
- when $\boxed{a=5}$?

$a = 5$

$f(a) = 15$

$$\frac{15{,}003 - 15}{5{,}001 - 5} = \frac{0{,}003}{0{,}001} = 3$$

(change in a)



$a = 2 \quad f(a) = 6$

$a = 2.001 \quad f(a) = 6.003$

ratio :

$\frac{0.003}{0.001}$ $\boxed{\frac{\text{height}}{\text{width}}}$ → slope (derivative) of f(a) at a=2 is 3

$$\frac{0{,}003}{0{,}001} = 3$$

[cf. Andrew Ng’s course on DL]

Derivatives

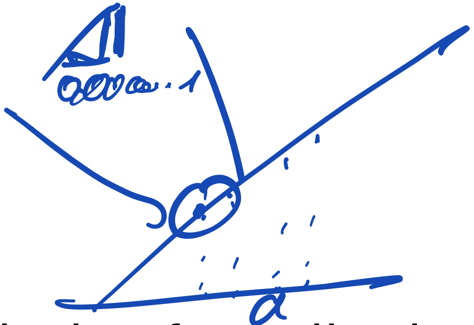
- With this function, we expect a small positive nudge in the input to make the ouput increase by 3 times the value of that nudge.

$$f(5.001) = 15.003$$

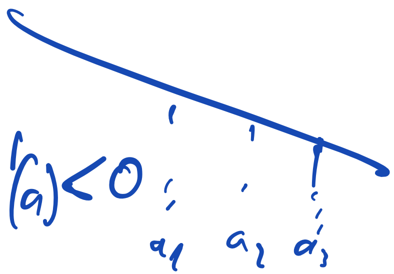
- In other words the **ratio** between the change in output and the change in input is 3 :

$$\frac{change\ in\ f(a)}{change\ in\ a} = \frac{df(a)}{da} = \frac{0.003}{0.001} = 3$$

Derivatives



$f'(a) > 0$



$f'(a) < 0$

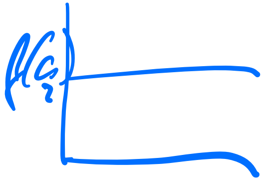
- This is just an example, but formally, the derivative considers this ratio when the input is increased by a **much tinier** amount !
- Nudge < 0.00000000000…..1 => nudge gets **as close to 0 as possible**

$f'(a)$ $f'(a) = \boxed{3}$

- In this previous example, whatever input value we pick, the derivative will be the same.
- This makes sense since the **function is a line** and the output increases at a constant rate
- Question : What if the derivative was negative everywhere ? What would the function look like ?

$f'(a) = 0$ $\frac{dy}{dx} > 0$ $\frac{dy}{dx}$ $f'(a) = -3$

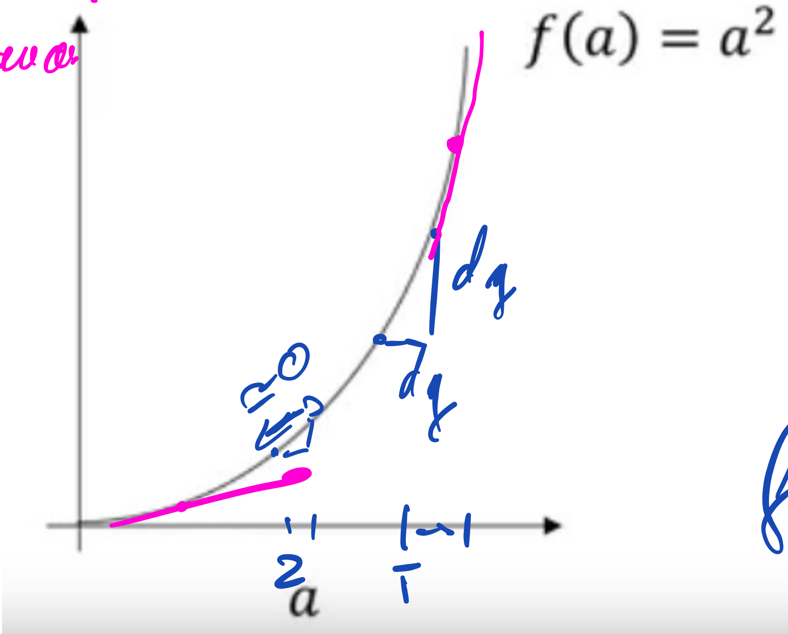
$f(a) = 2a+2$



Derivatives

- What if our function isn't a line ?

derivative: instantaneous rate of change



$a = 2$ $\quad f(a) = 4$

$a = 2.001$ $\quad f(a) \approx 4.004$

(4.004001)

$$f'(a=2) = \frac{0{,}004}{0{,}001} = 4$$

$$f'(a=5) = \frac{f(a=5{,}001) - f(a=5)}{5{,}001 - 5}$$

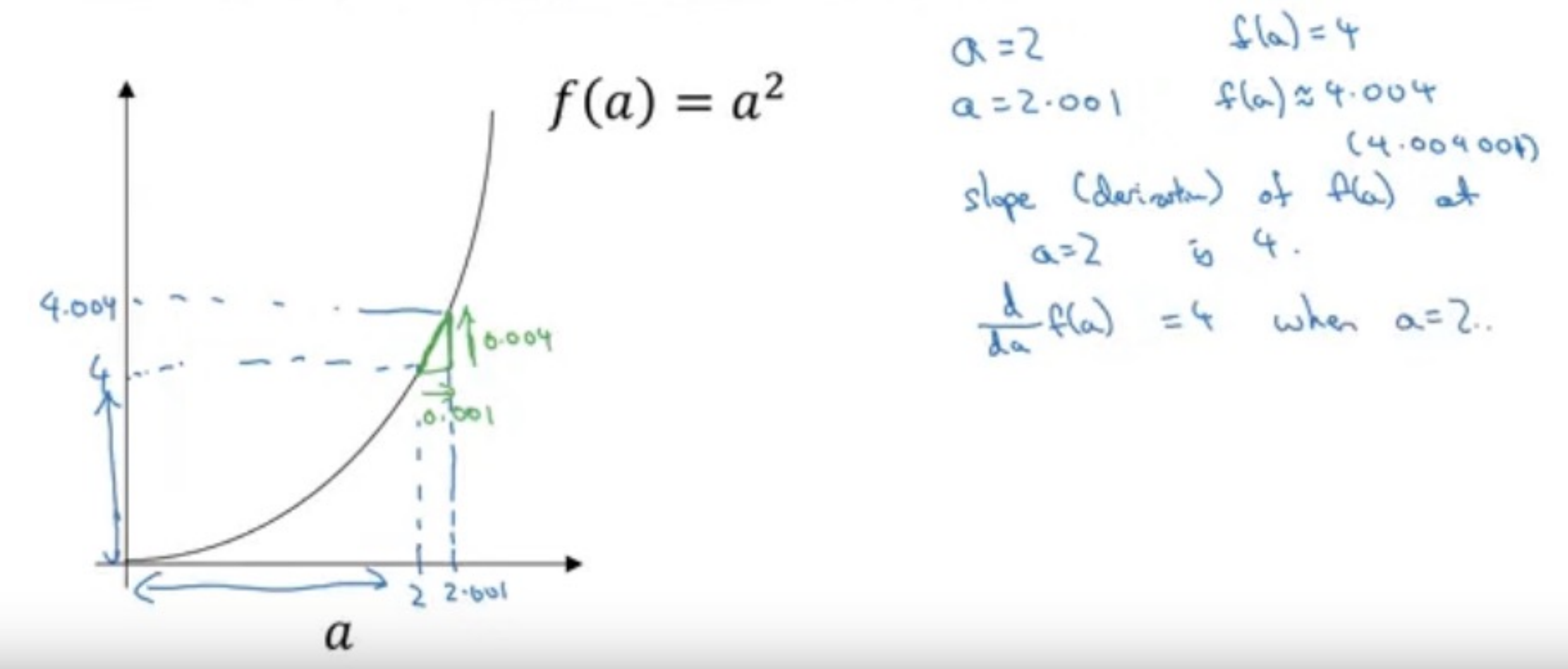
$$= 25{,}01 - 25$$

$f'(2) = 4$

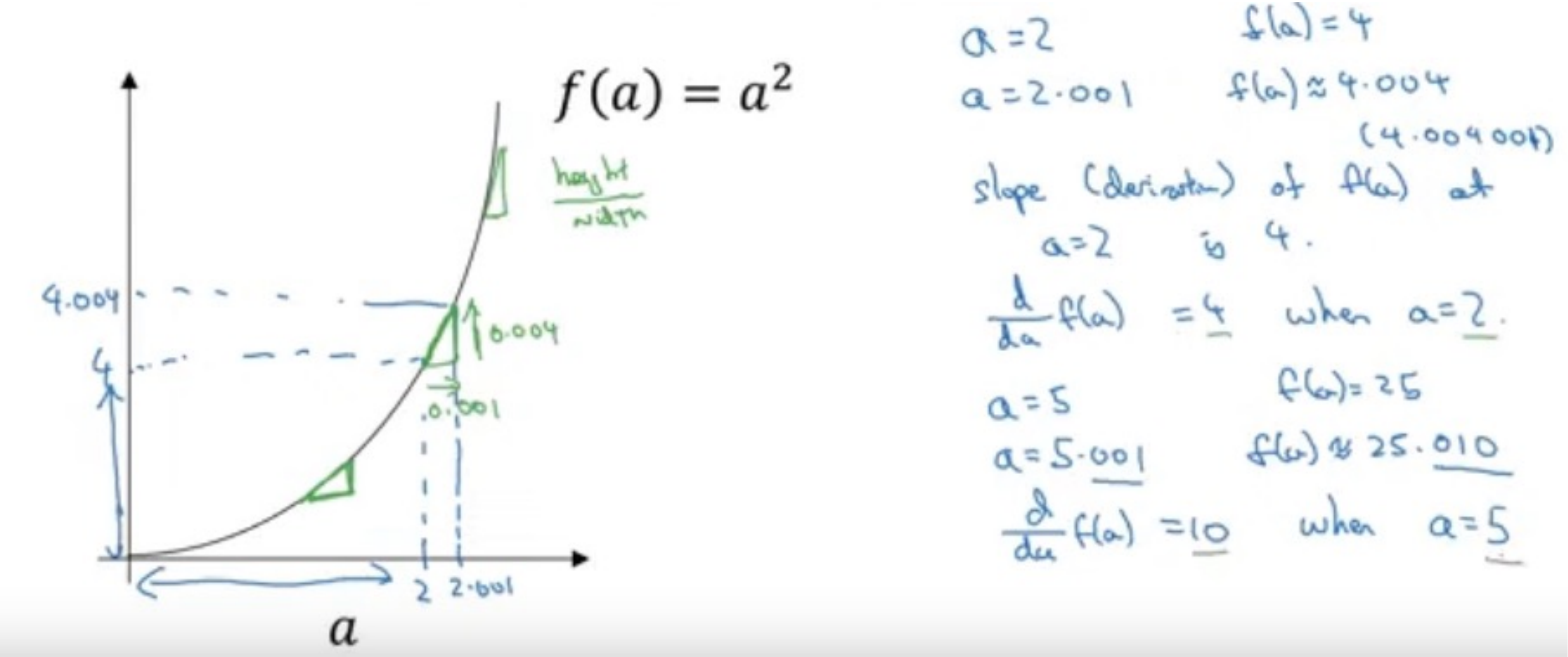
$f'(5) = 10$

$$= \frac{0{,}01}{0{,}001} = \boxed{10}$$

- The derivative at a=2 is ...



- The derivative at a=5 is ...



- Rules exist to compute derivatives
- For example, the function

$$f(a) = a^2$$

a^n

$\boxed{n a^{n-1}}$ $f(a^2) = 2a^1$

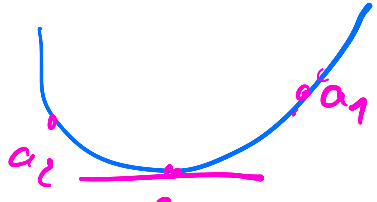
$= 2a$

$$f'(a) = \frac{d}{da} f(a) = \boxed{2a}$$

(The notations are called Lagrange and Leibniz notations and are both common)

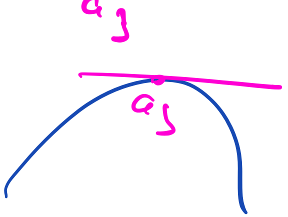
- If we look at the derivatives/slopes/ratios we calculated previously, this does indeed seem to work !
- Note: the derivative is equal to the **slope of the tangent line** on the graph **at our input value**.

$f'(a_1) > 0$: fct is increasing at that point a_1



$f'(a_2) < 0$: " decreasing " a_2

$f'(a_3) = 0$: " constant " a_3



$f'(a_3) = 0$

Derivatives: (optional)

$$f'(x) = \frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

diff. x

Example 2: $f(x) = x^2$ $\quad f'(x) = 2x$

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{\cancel{x^2} + (2x\Delta x + (\Delta x)^2) - \cancel{x^2}}{\Delta x}$$

$$= \lim_{\Delta x \to 0} \frac{2x\cancel{\Delta x} + (\cancel{\Delta x})^2}{\cancel{\Delta x}} \quad \Delta x$$

$$= \lim_{\Delta x \to 0} \boxed{2x + \Delta x.}$$

- As Δx approaches 0, the derivative
- Approaches 2x.

GD Intuition

- Now we have a basic understanding of derivatives, let's apply this understanding to the gradient descent algorithm by using **a simpler example**, with a **cost function of only 1 single parameter**.

- We use $\boldsymbol{J(\theta_1)}$ instead of $J(\theta_0, \theta_1)$

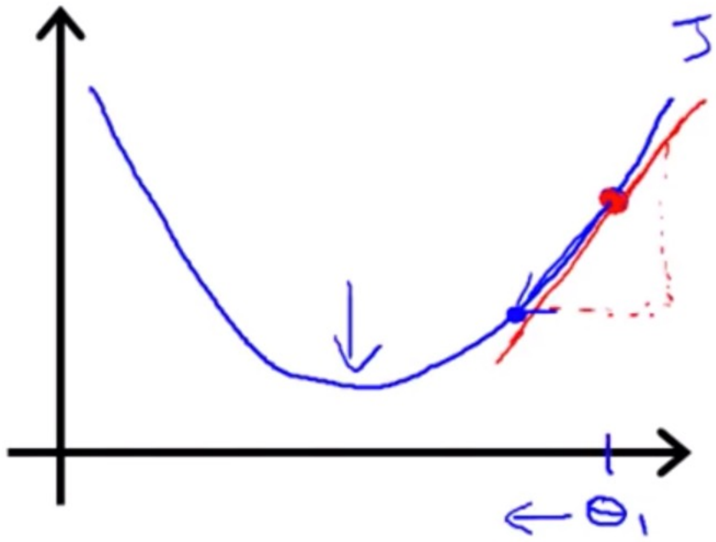
- Let's look at a couple scenarios to see how Gradient Descent updates our parameter θ_1.

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

When the derivative is positive...

- Remember, our cost function looks like a parabola.
- When θ_1 is too high, we want our optimizer to **reduce** this parameter and bring it closer to the « **sweet spot** », where the cost is minimized.
- Let’s see if it does the right thing :

minimize $J(\theta_1)$



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

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

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

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

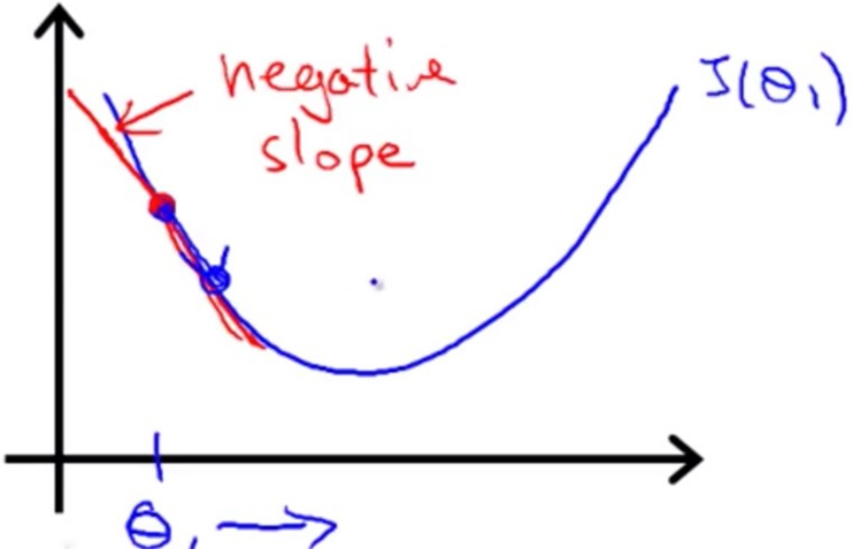
$\theta_1 = 3$

$\delta - 1.2$

See [Andrew Ng’s course]()

When the derivative is negative…

- When θ_1 is too low, let’s see if Gradient Descent **increases** it and brings it closer to the « sweet spot », where tht cost is minimized :



$$\frac{d}{d\theta_1} J(\theta_1)$$

$$\leq 0$$

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

$$- \alpha \cdot -2$$

$$+ \alpha 2$$

Recap

$x \rightarrow$ $\frac{dy}{dx} > 0$

Cost is going up

- When the the parameter value is too high, the derivative is positive and the update rule decreases the value for the parameter.

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

- Conversely, when the parameter value is too low, the parameter value will be increased by the update rule.

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

Okay so now what about α ?

0,01

< 0

- Remember the update rule : $$\theta_1 := \theta_1 - \boxed{\alpha}\left|\frac{d}{d\theta_1}J(\theta_1)\right|$$

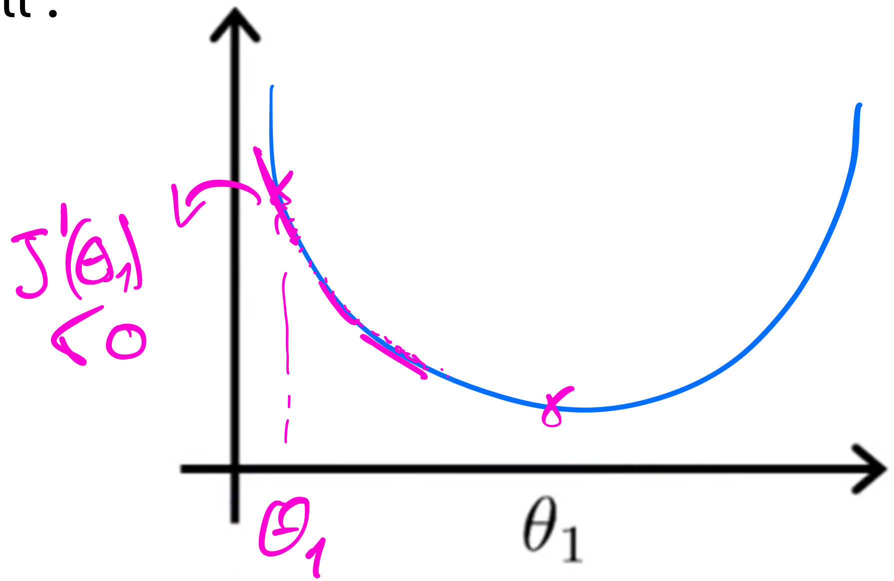
10

1000

- How does α influence the update of our parameter θ_1 ?

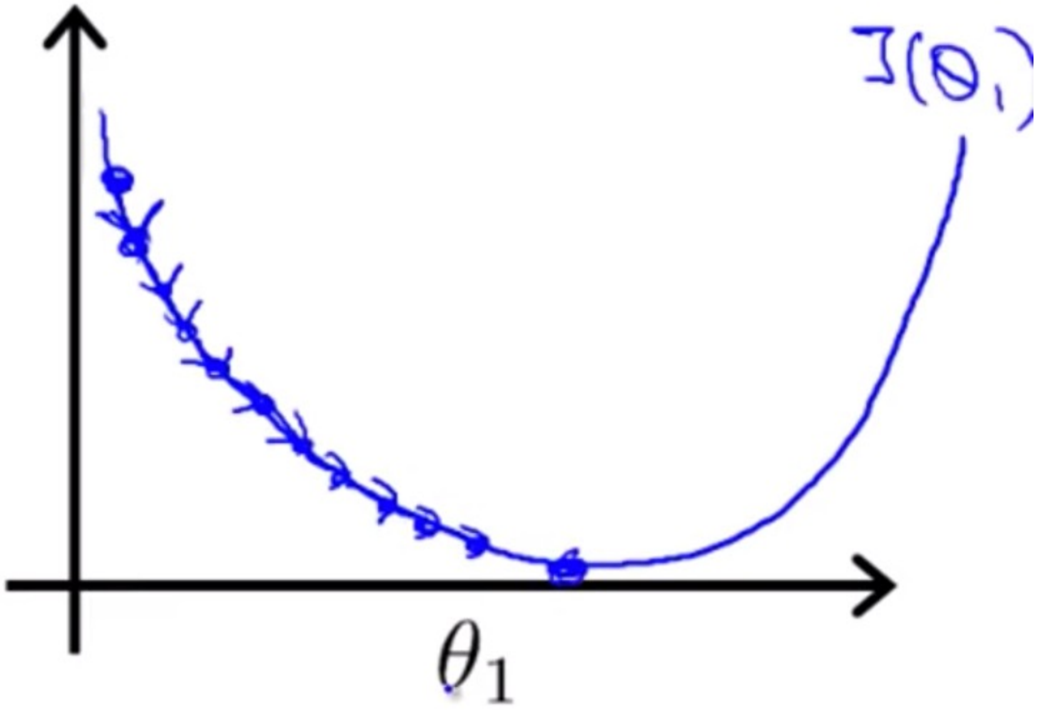
0,5

- If α is too small :



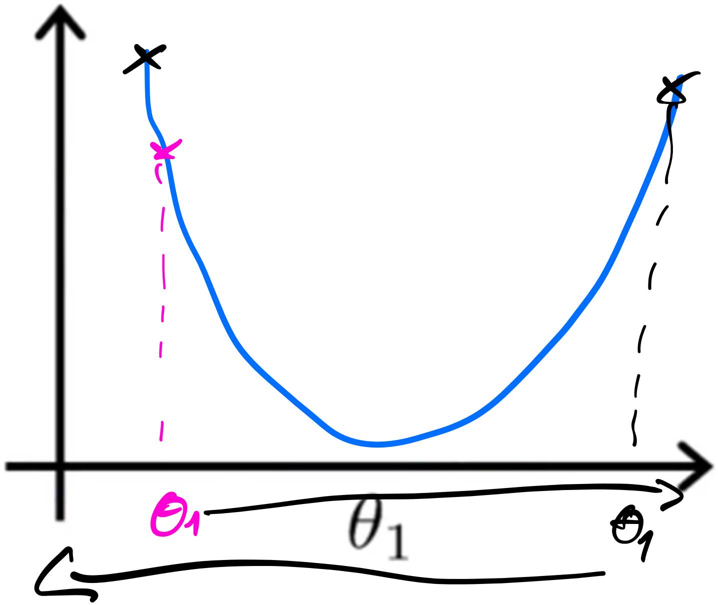
If α is too small

- Many small steps will be taken , which makes Gradient Descent very slow



If α is too large…

- Gradient descent may « overshoot », go past the minimum. It may even never converge (never find the minimum) and keep jumping around.

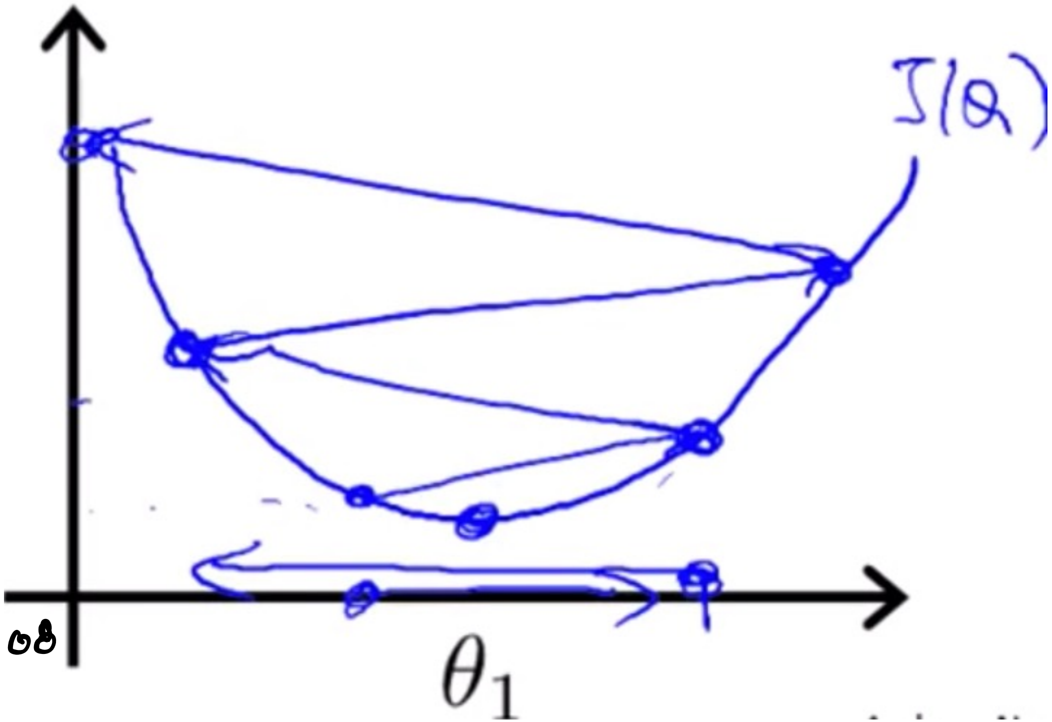


If alpha too large

α : hyperparameter

$0{,}1 \xrightarrow{\times 10^{-1}} 0{,}01 \xrightarrow{\times 10^{-1}} \boxed{0{,}001}$

$0{,}002 \quad 0{,}0008$



Scheduler : linear 0,9

cosine

GD - momentum
adam

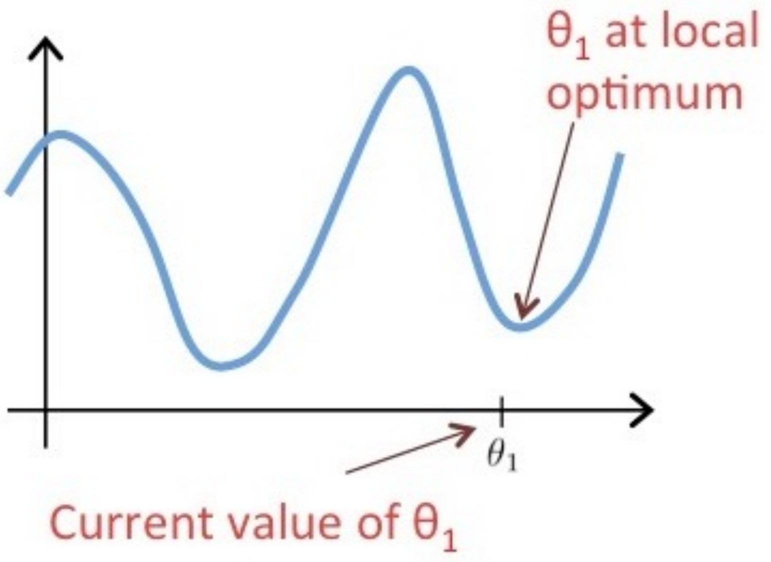
Question

1. Change θ_1 in a random direction ?
2. Move θ_1 in the direction of the global minimum of $J(\theta_1)$?
3. Leave θ_1 unchanged ?
4. Decrease θ_1 ?

global
vs

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



Recap

- To update our parameter with the Gradient Descent algorithm, we perform 2 essential steps :

1. Compute the derivative of the parameter with respect to the value we want to minimize (ie. our cost: a score to express how good our model is doing)

2. Take an optimization step/update the parameter. This update will be proportional to the derivative and the learning rate.

Large derivative (steep tangent line) + large learning rate = big update

Piecing everything together

- This is all we need :
  - A **hypothesis** function (our model)
  - A **cost function** (to tell us how well/bad our model is doing)
  - **Gradient Descent or variant** (to update our parameters and get closer to a better model)

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 $j = 0$)

}

$|\theta_{j_t} - \theta_{j_{t+1}}| < 0,0001$ then Stop.

Linear Regression Model

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

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

Derivatives vs. Partial derivatives

- Except, instead of having a cost function with a single input, we are back to 2 inputs, our 2 parameters $\theta_0$and θ_1.

- When we have functions with multiple inputs (known as multivariate functions), computing 1 single derivative is no longer enough!

- The function's « **instantaneous rate of change** » for a given combination of parameters is now determined by 2 values :

  - How does a tiny change in $\theta_0$change $J(\theta_0, \theta_1)$?

  - How does a tiny change in $\theta_1$change $J(\theta_0, \theta_1)$?

=> Packed together into a vector, these 2 derivatives make up what is referred to as the **gradient**

- Each derivative is a **partial derivative**. (you need both together to get the whole picture !)

Derivatives vs. Partial derivatives

- *Partial Derivative :*

This comes down to calculating the derivative at each input value, treating the other input as a constant

- We pretend for a second that the other input value has basically no effect on the function

  - when looking at θ_0 , we treat θ_1 as a constant

  - when looking at θ_1 , we treat θ_0 as a constant

Partial derivatives visually

- To help illustrate things and relate them to our simple Gradient Descent intuition:

https://www.youtube.com/watch?v=dfvnCHqzK54&t=1s

Gradient Descent

- Each partial derivative tells us how the function behaves (increases/decreases, quickly/slowly, stays constant…) with respect to a single input

- We can then use this information to know if we should increase or decrease each input to get closer to our minimum cost value !

- **Gradient** : the partial derivatives packed together in a vector
- **Descent** : we want to find the cost function’s minimum, using the gradient as a source of information to tell us if the cost is increasing/decreasing with respect to each input.

Update rule

- So we need to figure out the partial derivatives for each parameter !

- the partial derivative of $J(\theta_0, \theta_1)$ with respect to θ_1
- the partial derivative of $J(\theta_0, \theta_1)$ with respect to θ_2

Partial derivatives of $J(\theta_1, \theta_2)$

- You can treat these results as being **given**, in order not to go into the details of the derivation.

- General formula

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

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

See here for the MSE derivative

Partial derivatives of $J(\theta_1, \theta_2)$

- Here are the partial derivatives obtained (take these at face value for now):

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

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

- These formulas allow us to compute the partial derivatives for each of the parameters, which we can then plug into our Gradient Descent algorithm.

Gradient Descent

- We now have formulas to update our parameters !

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

Quick recap to put things into perspective

- We have :
- a **model**, which is a line :

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

- a **cost function,** to tell us how good/bad our model fits the data:

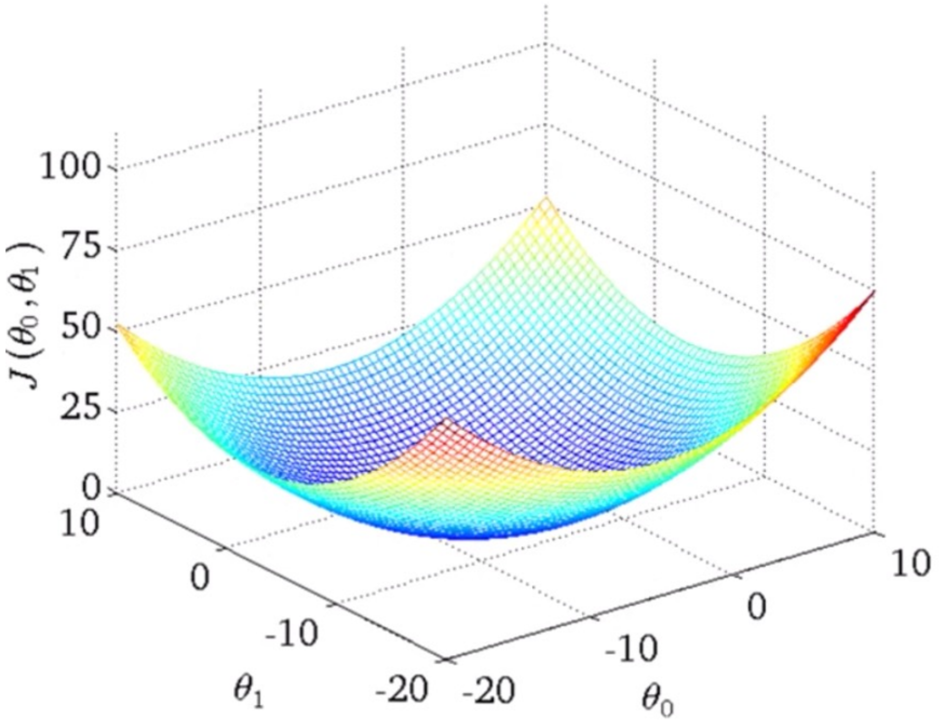
$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h\left(x^{(i)}\right) - y^{(i)})^2$$

- **Gradient Descent,** a method to update our parameters so as to minimize the cost function:

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

Update examples

- For linear regression, the cost function will always be bowl-shaped

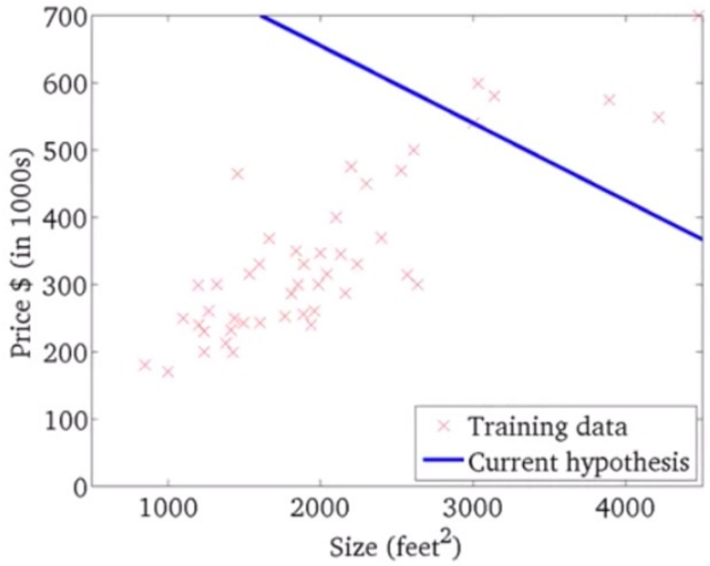


Update examples

Say we initialize our parameters randomly, this is the model and cost :

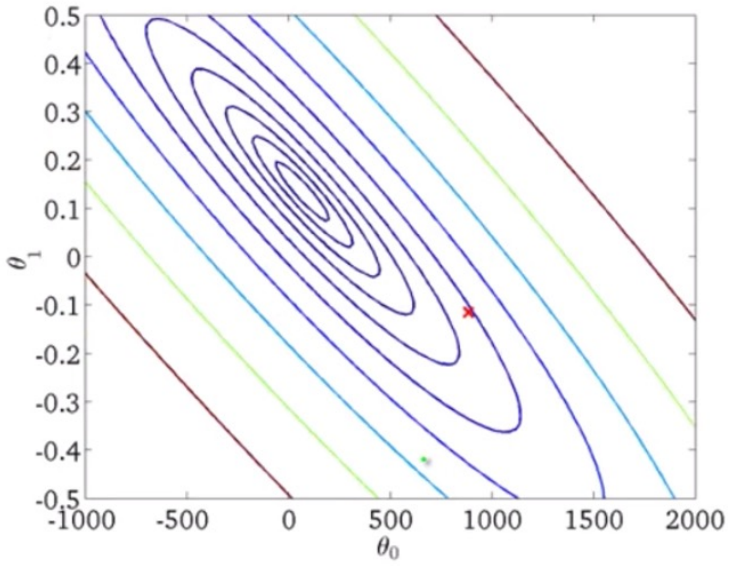
$$h_\theta(x)$$

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



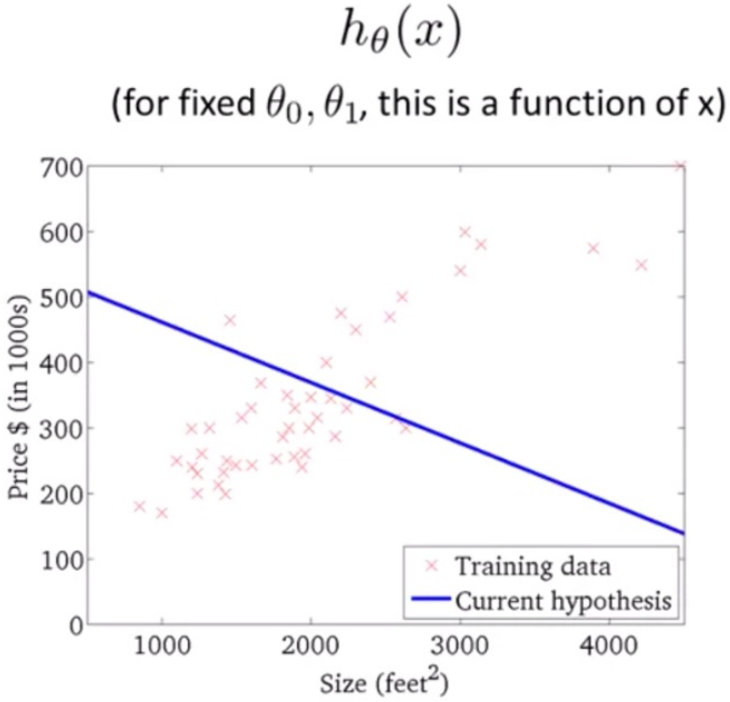
$$J(\theta_0, \theta_1)$$

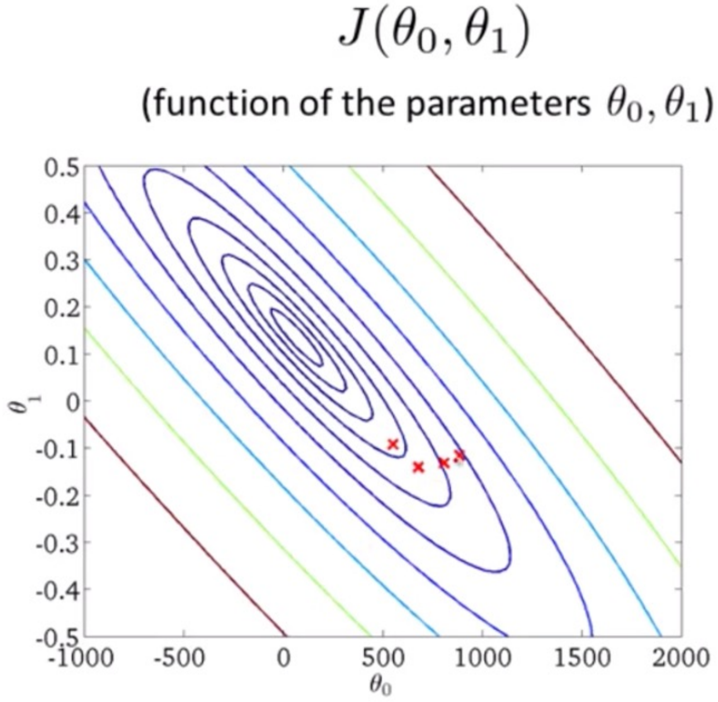
(function of the parameters θ_0, θ_1)



Update examples

As we take Gradient Descent steps, the model (line) seems to be fitting the data better





Update examples

Until we reach the global minimum