Parameter Learning

29 November 2020 05:09 PM

tdsp

GRADIENT DESCENT

→ This is an algorithm which is used to minimize the cost function (J)
→ We'll start with a simple example first - Linear Regression with one variable - and then we'll go on the more complex ones

→ Here is the idea for Gradient Descent

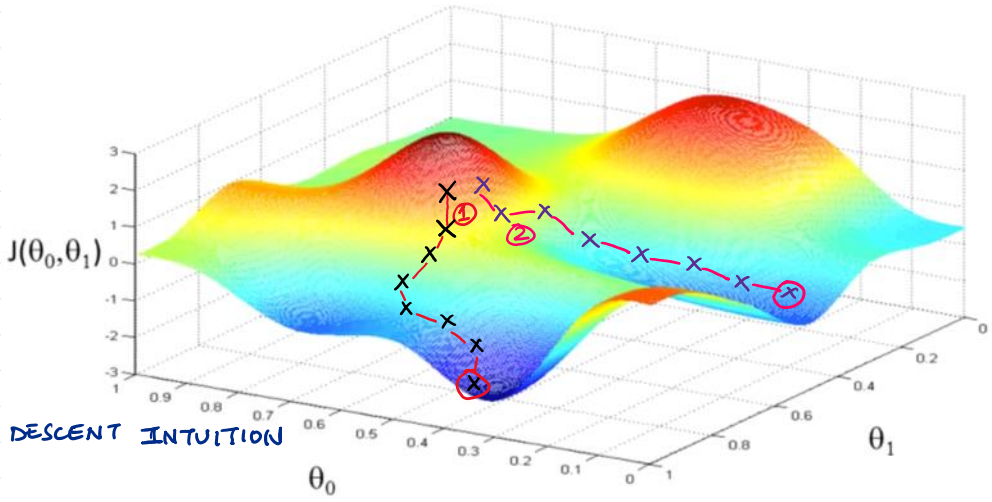
- Have some function $J(\theta_0, \theta_1)$
- Want to minimize $J(\theta_0, \theta_1)$ θ_0, θ_1

Could be $J(\theta_0, \theta_1, \ldots \theta_n)$
Could be $\min_{\theta_0 \ldots \theta_n} J(\theta_0 \ldots \theta_n)$

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

{ Same steps to be followed for $J(\theta_0 \ldots \theta_n)$ }

→ Let's see how it works on this graph



- θ_0 and θ_1 are the axis parameters
- $J(\theta_0, \theta_1)$ is found by calculating the height from the $(\theta_0 - \theta_1)$ surface

GRADIENT DESCENT INTUITION

1. Let's start with some random values for θ_0 and θ_1 and start with some random point on the graph.
   X marks this starting point.
2. Now imagine this graph to be a part of some garden and X marks the spot where you are physically standing.
3. In Gradient Descent, what we are going to do now is to take a 360° spin just to look all around my spot, and ask, "If I were to take a little baby step in some direction, and I want to go downhill as quickly as possible, which direction do I take that little baby step in?".
4. Suppose we choose to go down path (1). X marks our new position after traversing down (1).
5. Now again we repeat Step 3 like we did initally. We look around to choose which direction takes us downhill and we travel that way.
6. We continue with this exercise in the exact same way until we converge to a local minima.

→ Gradient Descent has an interesting property
↳ If we had started with the whole process from a position just a little towards the right, marked by X, Gradient Descent would have taken us down a path to some other local minima

GRADIENT DESCENT ALGORITHM

a path to some other local minima

- GRADIENT DESCENT ALGORITHM

→ Assignment Operator → $:=$
↳ If we have,

$a := b$
Take the value of b and assign it to a

$a := a + 1$
Take the value of $(a+1)$ and assign it to a
The value of a, hence, gets updated

→ Truth Assertion → $=$
↳ If we have,

$a = b$
This implies a and b are equal

$a = a + 1$ ✗
Cannot write this as this is mathematically incorrect

→ Algorithm :

repeat until convergence {

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

}

↳ $:=$ — Assignment operator signify value of θ_j is changing

↳ α — Learning Rate - Controls how big/small we take a step downhill during gradient descent
$\alpha \uparrow\uparrow$: High value => Bigger steps
$\alpha \downarrow\downarrow$: Low value => Smaller steps
} ALWAYS A POSITIVE NUMBER

↳ ∂ — Partial Derivative

↳ (for j=0 and j=1) —
Step 1 : Calculate updated θ_0 in $temp_0$
Step 2 : Calculate upated θ_1 in $temp_1$
Step 3 : Assign $temp_0$ value to θ_0
Step 4 : Assign $temp_1$ value to θ_1

"Simultaneous Update"

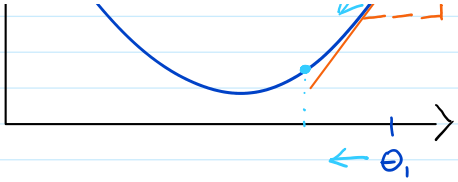
- GRADIENT DESCENT INTUITION : DERIVATIVE PERSPECTIVE

→ Let's go back to one of the cost functions we have used before for the sake of simplicity : Cost Function with one parameter : $J(\theta_1)$

→ Let θ_1 be a real number => $\theta_1 \in \mathbb{R}$

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

After first updation, θ_1 decreases



After first updation, θ_1 decreases

- The difference between "∂" and "d" when it comes to derivatives is that "∂" is used for partial derivative and "d" symbolizes normal derivative
  - Partial derivative → Derivation with respect to some particular entity only
  - Normal derivative → The normal derivation

→ A derivative is nothing but the angle made by a tangent at that particular point
  ↳ so if we find the derivative of a function at a point, we will get the angle the tangent at that point makes.
  ↳ In other words the slope made by the tangent at that particular point

→ In the figure above, the tangent to the function J at the point θ_1 gives the derivative at that point

→ As we can see, the slope of the tangent is POSITIVE, which basically means that the derivative is POSITIVE

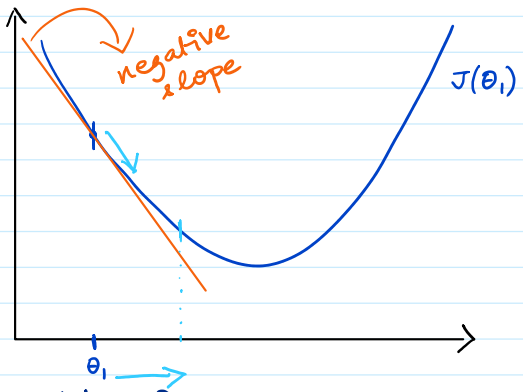
$$\Rightarrow \boxed{\frac{\partial}{\partial \theta_1} J(\theta_1) \geq 0}$$

→ Now if we update the value of θ_1 –

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

→ This will decrease the value of θ_1 and move θ_1 towards the left on the graph, in turn taking it closer to the minima (the lowest point of the curve)

→



As we have a negative slope in this case, this implies that the derivative is also NEGATIVE

$$\Rightarrow \boxed{\frac{\partial}{\partial \theta_1} J(\theta_1) \leq 0}$$

→ If we update θ_1 now –

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

→ This will actually increase the value of θ_1 and move θ_1 towards the right on the graph, in turn taking it closer to the minima

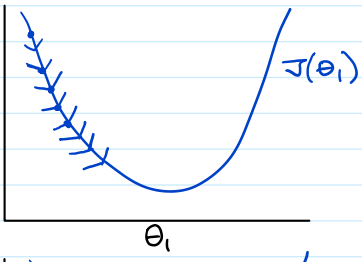
Effect of Learning Rate : α $\gg$ $\boxed{\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)}$

Update value

Q How does the value of α affect Gradient Descent?

→ Gradient ↓ $J(\theta_1)$ If α is too small, then the update value which is added / subtracted will be small

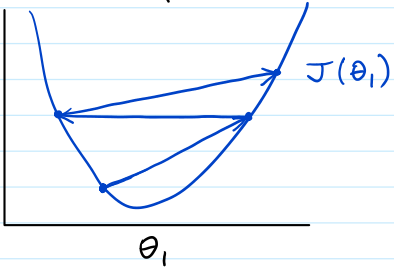
→ Gradient Descent is SLOW!



If α is too small, then the update value which is added / subtracted will be small

Hence, gradient descent will take little baby steps as it approaches the minima

→ Gradient Descent can overshoot the global minimum!

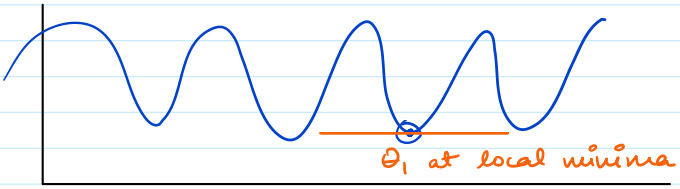


If α is too large, then the update value which is added / substracted will be large

Hence, gradient descent will take huge steps which may even result in a situation worse than the current situation

So gradient descent might just overshoot again and again, possibly never really finding the correct minima, that is, gradient descent may even fail to converge or even diverge

Q What happens if θ_1 is already at some local minima?



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

Derivative at this point is ZERO as the slope is ZERO

→ Let's suppose you initialize θ_1 at a local minima
→ The derivative term will be ZERO
⇒ There will be no updates to the value of θ_1

With a fixed α, as we approach a local minimum, gradient descent will automatically take smaller steps.

⊡ GRADIENT DESCENT FOR LINEAR REGRESSION

→ Gradient Descent Algorithm

repeat until convergence {

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

(for $j=1$ and $j=0$)

}

Linear Regression Model

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

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

→ Now we will apply Gradient Descent to minimize the cost function mentioned above
→ To do that, first we need to find out the derivative term

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

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

For θ_0 : $j = 0 \Rightarrow \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) \Rightarrow \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$

For θ_1 : $j = 1 \Rightarrow \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) \Rightarrow \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) . x^{(i)}$

→ Derivations using Multivariate calculus :

- For θ_0 : $j = 0$

$$\frac{1}{2m} \cdot 2 \cdot \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right) . (1 + 0 - 0)$$

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

- For θ_1 : $j = 1$

$$\frac{1}{2m} \cdot 2 \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right) . \left(0 + x^{(i)} - 0 \right)$$

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

→ Now we just need to put these equations back into the Gradient Descent algorithm

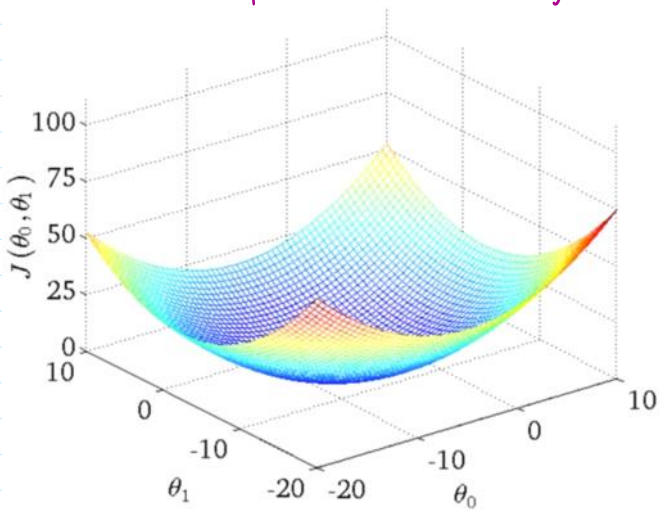
repeat until convergence {

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

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

} Update θ_0 and θ_1 simultaneously

→ For linear regression the cost function will always be a bowl-shaped convex function



→ There is no local optimal minima except for the global optima