Gradient Descent (A Beginners Overview with Simple Python Code)

https://blog.csdn.net/Nishith/article/details/108473749

Hello mates, today I am going to discuss another fundamental algorithm of machine learning.

In machine learning, optimization is a big part. In almost all machine learning system has an optimization algorithm at its core. In this post, I'll try to introduce such a simple optimization algorithm named Gradient Descent shortly GD.

So, What is optimization algorithm?
An optimization algorithm is a procedure which is executed iteratively by comparing various solutions until an optimum or a satisfactory solution is found.
If you want to learn more about optimization method you can read this PDF
Now, lets talk about Gradient Descent.
Some researchers say used to say that Gradient Descent is the backbone of machine learning.
What is gradient descent (GD)?
Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. It is simply used to find the values of a function's parameters (coefficient) that minimize a cost function as far as possible.

To understand the concept of Gradient Descent it’ s important to know about Gradients.

What is Gradient?

> “A gradient measures how much the output of a function changes if you change the input’ s a little bit .” - Lex Fridman (MIT)

A gradient normally measures the change in all weights with regard to the change error.

We can assume it as a slope of a function. The higher the gradient, the steeper the slope, and the faster a model can learn. But if the slope is zero, the model stops learning. In mathematically, A gradient is a partial derivative with respect to its input.

To explain the **Gradient Descent** in a practical way, suppose you are in a peak of a mountain that is covered with snow. Suddenly a snowstorm started and you are not able to see anything in front of you through the snow. It’ s not possible for you to survive there for a long time. And there is no one to help you but you have a gadget which tells you the ‘ current height from the sea level.

So, at that moment you decided to come down from the peak of the mountain to flat land. As you can' t see anything you started walking in a random direction. Then you asked the gadget what is the height now? If the gadget tells you the height and it' s more than the initial height, then you know you started in the wrong direction. Then you will change the direction and repeat the process. This way in many iterations finally you will be able to descend down successfully.

Here,

- You took the step size in any direction represents **Learning Rate.**
- The gadget tells you the height is actually **Cost Function.**
- And the direction of your steps representent the **Gradients.**

The mathematical representation is given below :

$$\text{Cost function,} \quad J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (\mathrm{h}(\theta)^{(i)} - y^{(i)})^2$$

$$\text{Gradient,} \quad \frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (\mathrm{h}(\theta)^{(i)} - y^{(i)}) x_j{}^{(i)}$$

Where, **m** is the number of observation.

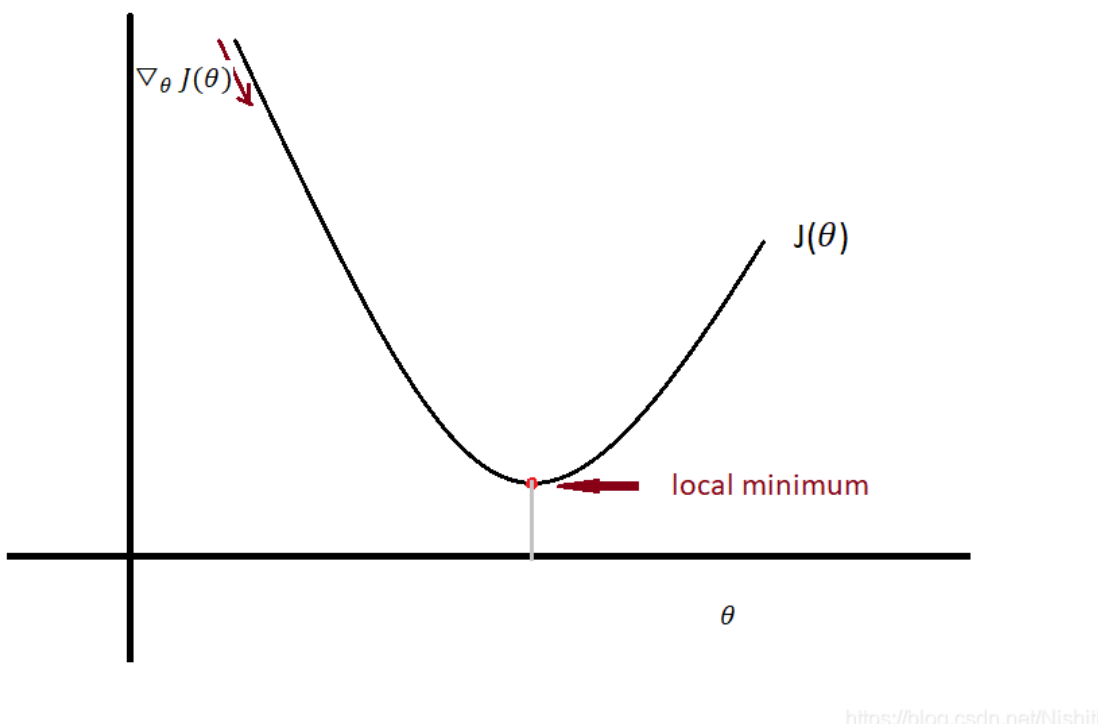
$$\text{Update equation,} \ \theta = \theta - \eta * \nabla_{\theta} J(\theta)$$

Where J(θ) is a objective function.

$$\theta \in R^{d} = Model's\ Parameters$$

$$\eta = Learning\ Rate\ ,$$

Learning rate: Which determines the size of steps we take to reach a local minimum.



Here is the generalize form to calculate θ:

$$\theta_0 := \theta_0 - \eta \left(\frac{1}{m}\sum_{i=1}^{m}(\mathrm{h}(\theta)^{(i)} - y^{(i)})x_0{}^{(i)}\right)$$

$$\theta_1 := \theta_1 - \eta \left(\frac{1}{m}\sum_{i=1}^{m}(\mathrm{h}(\theta)^{(i)} - y^{(i)})x_1{}^{(i)}\right)$$

$$\theta_2 := \theta_2 - \eta \left(\frac{1}{m}\sum_{i=1}^{m}(\mathrm{h}(\theta)^{(i)} - y^{(i)})x_2{}^{(i)}\right)$$

$$\theta_j := \theta_j - \eta \left(\frac{1}{m}\sum_{i=1}^{m}(\mathrm{h}(\theta)^{(i)} - y^{(i)})x_j{}^{(i)}\right)$$

Gradient Descent Method:

Suppose the function y = f(x1,x2,x3,...,xn) has only one minimum ,the initial position is given x0

To solve a problem using gradient descent you can follow the methods given below.

1. Set a smaller positive value for **η and ϵ** .
2. Obtain each partial derivative at the current position.
3. Update the value of current function. You have to use the update equation for updating value.

$$\text{Update equation, } \theta = \theta - \eta * \nabla_\theta J(\theta)$$

4. If the parameter change is less than the **ϵ** : **exit**;
   Otherwise, return to **point No 2**

Here,

η = Learning rate

ϵ = Accuracy

Lets see an example:

For any initial starting point, let’ s set x0 = - 4 and use the gradient descent method to find the minimum value of the following function:

$$y = \frac{x^2}{2} - 2x$$

Solution:

1. First, two parameters are given:
   η = 0.9 and ϵ= 0.01
2. Find the derivative of the given function:

$$y' = \frac{dy}{dx} = x - 2$$

3. Calculate the current derivative value:
   y’ = (-4) - 2 = - 6
4. Modify the current values/Update the values:

$$x' = x - \eta * y' = (-4) - 0.9 * (-6) = 1.4$$

$$\Delta x = x' - x = 1.4 - (-4) = 5.4$$

5. As Δx > ϵ so, we need to calculate the current derivative value.

   y’ = 1.4 - 2 = - 0. 6

6. Modify the current values/Update the values:

$$x' = x - \eta * y' = (1.4) - 0.9 * (-0.6) = 1.94$$

$$\Delta x = x' - x = 1.94 - 1.4 = 0.6$$

7. As $\Delta x > \epsilon$ so, we need to calculate the current derivative value.

y' = 1.94 - 2 = - 0.06

8. Modify the current values/Update the values:

$$x' = x - \eta * y' = (1.94) - 0.9 * (-0.06) = 1.994$$

$$\Delta x = x' - x = 1.994 - 1.94 = 0.054$$

9. As $\Delta x > \epsilon$ so, we need to calculate the current derivative value.

y' = 1.994 - 2 = - 0.006

10. Modify the current values/Update the values:

$$x' = x - \eta * y' = (1.994) - 0.9 * (-0.006) = 1.9994$$

$$\Delta x = x' - x = 1.9994 - 1.994 = 0.0054$$

11. As $\Delta x=0.0054 < \epsilon$ so, We will stop here and finish our process because we have already found the minimum value.

Finally we found the solution. But it was really a simple function thats why we found the minimum of the function after only 10 iteration. But if we change the function like,

$$y = x^2 + 1$$

it will need almost 31 iteration to find the local minima. So we aill use programming to solve this problem. And this is why we are learning machine learning !!

Ok now let’ s jump into python code for the same problem we have done with pen and paper above.

```
1  cur_x = -4                          #The_algorithm_starts_at x=-4
2  rate = 0.9                         #Learning_rate
3  precision = 0.001                 #This_tells_us_when_to_stop_the_algorithm
4  step_size = 1
5  max_iters = 1000                  #maximum_number_of_iterations
6  iters = 0                     #iteration_counter
7  df = lambda x:x-2                  #Gradient_of_our_functio
8  while step_size > precision and iters < max_iters:
9      prev_x = cur_x                 #Store current_x_value_in prev_x
10     cur_x = cur_x - rate * df(prev_x)     #Grad_descent
11     step_size = abs(cur_x - prev_x)    #Change in x
12     iters = iters+1     #iteration_count
13     print("Delta_x",prev_x-cur_x,"Iteration",iters,"\nX value is",cur_x)     #Print
14
15 print("The local minimum occurs at", cur_x)
```

you can use any function here in this code. And I used the derivative form of the function in line 7.So don’ t be confused.

And remember 1 thing try to set the precision/accuracy as small as possible.

Now try to solve this problem using the given code .

For any initial starting point, let' s set x0 = - 5 and use the gradient descent method to find the minimum value of the following function:

$$y = \frac{1}{2}(x - \frac{1}{4})^2 + 1$$

here, assume η = 0.6 and ϵ= 0.000001

We will finish here today. And like you I' m also learning machine learning algorithms. So, if there is anything wrong please inform me. And if you have questions feel free to ask. I believe, **knowledge increases by sharing not by saving.**

Thank you!!

----------------------------> Happy Coding <---------------------------

****

This content is created by Nishith Ranjan Biswas

Major : CST

Batch: 2018

Email: nishith170217@gmail.com

Date : 08.09.2020

To see Original blog post visit : https://blog.csdn.net/Nishith/article/details/108473749

****