ML | Stochastic Gradient Descent (SGD)

What is Gradient Descent?

Before talking about Stochastic Gradient Descent (SGD), let's first understand what is Gradient Descent? Gradient Descent is a very popular optimization technique in Machine Learning and Deep Learning and it can be used with most, if not all, of the learning algorithms. A gradient is basically the slope of a function; the degree of change of a parameter with the amount of change in another parameter. Mathematically, it can be described as the partial derivatives of a set of parameters with respect to its inputs. The more the gradient, the steeper the slope. Gradient Descent is a convex function.

Gradient Descent can be described as an iterative method which is used to find the values of the parameters of a function that minimizes the cost function as much as possible. The parameters are initially defined a particular value and from that, Gradient Descent is run in an iterative fashion to find the optimal values of the parameters, using calculus, to find the minimum possible value of the given cost function.

Types of Gradient Descent:

Typically, there are three types of Gradient Descent:

1. Batch Gradient Descent
2. Stochastic Gradient Descent
3. Mini-batch Gradient Descent

In this article, we will be discussing **Stochastic Gradient Descent** or SGD.

Stochastic Gradient Descent (SGD):

The word '*stochastic*' means a system or a process that is linked with a random probability. Hence, in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration. In Gradient Descent, there is a term called "batch" which denotes the total number of samples from a dataset that is used for

calculating the gradient for each iteration. In typical Gradient Descent optimization, like Batch Gradient Descent, the batch is taken to be the whole dataset. Although, using the whole dataset is really useful for getting to the minima in a less noisy or less random manner, but the problem arises when our datasets get really huge.
Suppose, you have a million samples in your dataset, so if you use a typical Gradient Descent optimization technique, you will have to use all of the one million samples for completing one iteration while performing the Gradient Descent, and it has to be done for every iteration until the minima is reached. Hence, it becomes computationally very expensive to perform.

This problem is solved by Stochastic Gradient Descent. In SGD, it uses only a single sample, i.e., a batch size of one, to perform each iteration. The sample is randomly shuffled and selected for performing the iteration.

SGD algorithm:

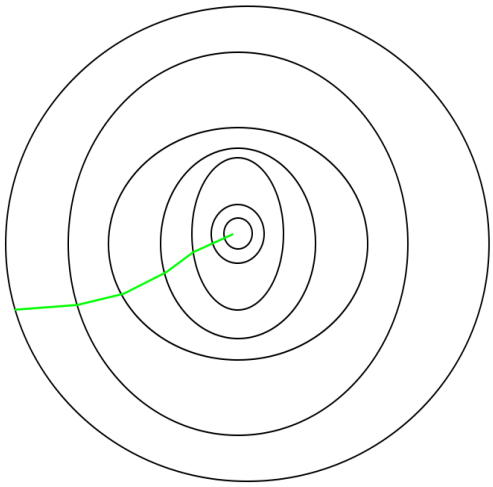
$$for\ i\ in\ range\ (m):$$

$$\theta_j = \theta_j - \alpha\,(\hat{y}^i - y^i)\,x_j^i$$

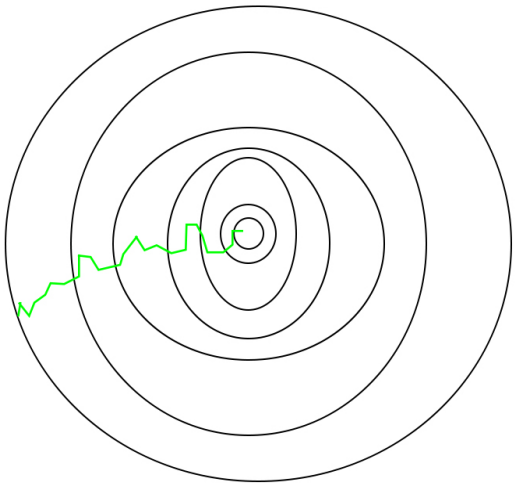
So, in SGD, we find out the gradient of the cost function of a single example at each iteration instead of the sum of the gradient of the cost function of all the examples.

In SGD, since only one sample from the dataset is chosen at random for each iteration, the path taken by the algorithm to reach the minima is usually noisier than your typical Gradient Descent algorithm. But that doesn't matter all that much because the path taken by the algorithm does not matter, as long as we reach the minima and with significantly shorter training time.

Path taken by Batch Gradient Descent –



Path taken by Stochastic Gradient Descent –



One thing to be noted is that, as SGD is generally noisier than typical Gradient Descent, it usually took a higher number of iterations to reach the minima, because of its randomness in its descent. Even though it requires a higher number of iterations to reach the minima than typical Gradient Descent, it is still computationally much less expensive than typical Gradient Descent. Hence, in most scenarios, SGD is preferred over Batch Gradient Descent for optimizing a learning algorithm.

Pseudo code for SGD in Python:

```
def SGD(f, theta0, alpha, num_iters):
    """
       Arguments:
       f -- the function to optimize, it takes a single argument
            and yield two outputs, a cost and the gradient
            with respect to the arguments
       theta0 -- the initial point to start SGD from
       num_iters -- total iterations to run SGD for
       Return:
       theta -- the parameter value after SGD finishes
    """
    start_iter = 0
    theta = theta0
    for iter in xrange(start_iter + 1, num_iters + 1):
        _, grad = f(theta)

        # there is NO dot product ! return theta
        theta = theta - (alpha * grad)
```

This cycle of taking the values and adjusting them based on different parameters in order to reduce the loss function is called **back-propagation**.

Recommended Posts:

Gradient Descent algorithm and its variants

Optimization techniques for Gradient Descent

Gradient Descent in Linear Regression

ML | Mini-Batch Gradient Descent with Python

ML | T-distributed Stochastic Neighbor Embedding (t-SNE) Algorithm

ML | XGBoost (eXtreme Gradient Boosting)

ML | Momentum-based Gradient Optimizer introduction

Demystify Autorun and Malwares

Applications of Pattern Recognition

How Can Machine Learning Save the Environment From Disaster?

Best Books To Learn Machine Learning For Beginners And Experts

ML | Credit Card Fraud Detection

ML | AutoEncoder with TensorFlow 2.0

Python | Haar Cascades for Object Detection

Rahul_Roy
An ordinary kid with a passion for coding

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags : Advanced Computer Subject Machine Learning

Practice Tags : Machine Learning

3

3

To-do Done

Based on **1** vote(s)

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments