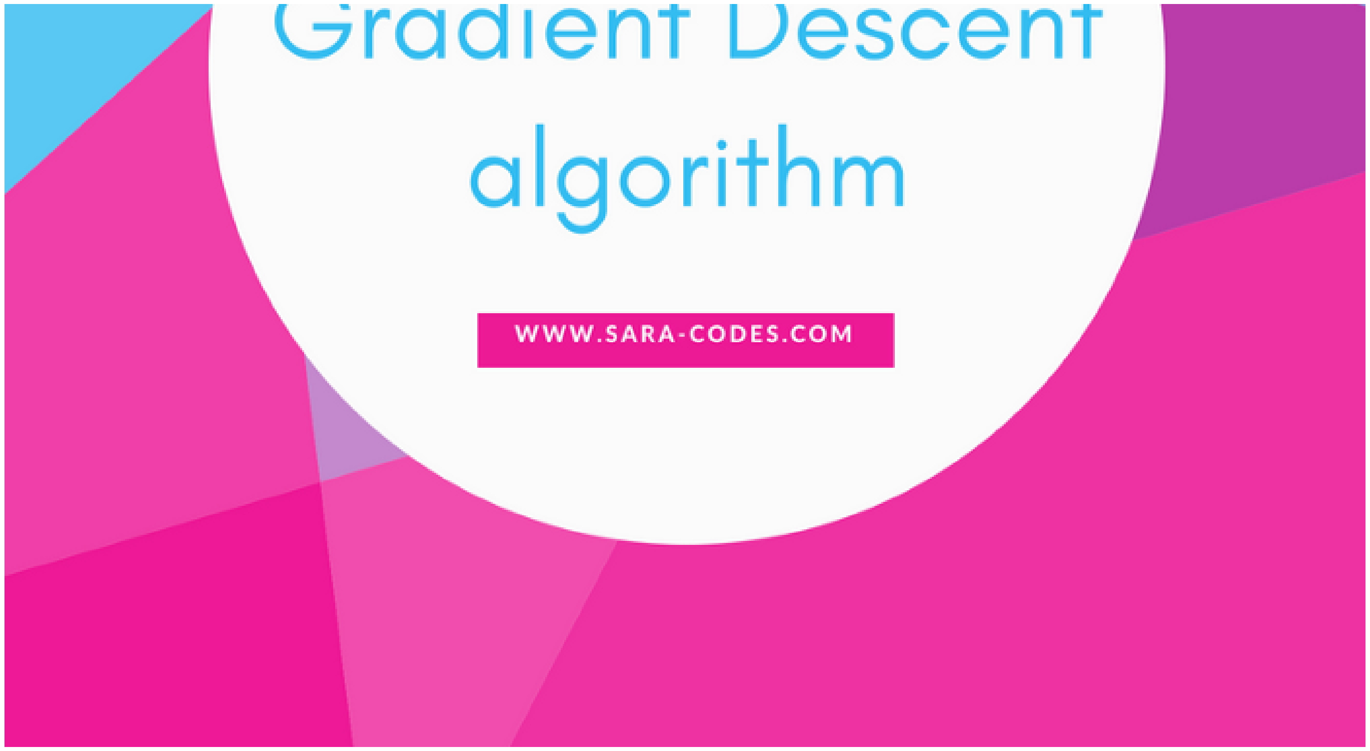
An introduction to Gradient Descent Algorithm

Sara Iris Garcia Follow
Jun 4, 2018 · 5 min read

Gradient Descent is one of the most used algorithms in Machine Learning and Deep Learning.



It is an optimization algorithm used in training a model. In simple words, Gradient Descent finds the parameters that minimize the cost function (error in prediction). Gradient Descent does this by iteratively moves toward a set of parameter values that minimize the function, taking steps in the opposite direction of the gradient.

What is a Gradient?

A gradient is a vector-valued function that represents the slope of the tangent of the graph of the function, pointing the direction of the greatest rate of increase of the function. It is a derivative that indicates the incline or the slope of the cost function.

At this point, I was completely lost, but what always helps me is to graphically imagine a problem, so imagine you are in the top of a mountain. Your goal is to reach the bottom field, but there is a problem: you are blind. How can you come with a solution? Well, you will have to take small steps around and move towards the direction of the higher incline. You do this iteratively, moving one step at a time until finally reach the bottom of the mountain.

That is exactly what Gradient Descent does. Its goal is to reach the lowest point of the mountain. The mountain is the data plotted in a space, the size of the step you move is the learning rate, feeling the incline around you and decide which is higher is calculating the gradient of a set of parameter values, which is done iteratively. The chosen direction is where the cost function reduces (the opposite direction of the gradient). The lowest point in the mountain is the value -or weights- where the cost of the function reached its minimum (the parameters where our model presents more accuracy).

What is the Learning rate?

Like we said, the gradient is a vector-valued function, and as a vector, it has both a direction and a magnitude. The Gradient descent algorithm multiplies the gradient by a number (Learning rate or Step size) to determine the next point.

For example: having a gradient with a magnitude of 4.2 and a learning rate of 0.01, then the gradient descent algorithm will pick the next point 0.042 away from the previous point.

Here is a cool explanation from the Machine Learning crash course from Google, where you can visually see the effects of the learning rate. Link here.

Image 1: Learning rate (Source: Towards data science)

At this point, can you guess what is the problem here?

If you played with the Learning rate of the link, then you clearly now the answer. One of the problems is in the value of the learning rate. The learning rate is a randomly chosen number that tells how far to move the weights. A small learning rate and you will take forever to reach the minima, a large learning rate and you will skip the minima.

Typically, the value of the learning rate is chosen manually, starting with 0.1, 0.01 or 0.001 as the common values, and then adapt it whether the gradient descent is taking too long to calculate (you need to increase the learning rate), or is exploding or being erratic (you need to decrease the learning rate).

Although the learning rate is usually chosen manually, there are several methods to automatically choose a fitting learning rate, such as AdaGram and RMSProp methods, but we will talk about them later.

A working example of Gradient Descent

Before we move forward, I believe a working example is worth gold for digesting new concepts, so **here** is an example of a linear regression using gradient descent written in python. The dataset used is the Student performance dataset, where we have to find an optimal line to predict the grade based on the number of hours studied. Of course, there is no point in using gradient descent in a linear regression problem, but this is just for explanation purposes.

Steps of gradient descent

Following the mountain example, we will call Agent as the blinded person. In each position, the agent only knows two things: the gradient (for that position, or parameters) and the width of the step to take (learning rate). With that information, the current value of each parameter is updated.

With the new parameter values, the gradient is re-calculated and the process is repeated until reach *convergence* or local minima.

Let’s revise how the gradient descent algorithm works at each step:

Repeat until hit convergence:

1. Given the gradient, calculate the change in the parameters with the learning rate.
2. Re-calculate the new gradient with the new value of the parameter.
3. Repeat step 1.

Here is the formula of gradient descent algorithm:

$$\begin{array}{l}\text{Repeat until convergence } \{ \\ \qquad \theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \\ \} \end{array}$$

Image 2: Gradient Descent algorithm. (Source: Samuel Trendler)[/caption]Convergence

Convergence is a name given to the situation where the loss function does not improve significantly, and we are stuck in a point near to the minima.

Gradient Descent variants

There are three variants of gradient descent based on the amount of data used to calculate the gradient:

- Batch gradient descent
- Stochastic gradient descent
- Mini-batch gradient descent

Batch Gradient Descent:

Batch Gradient Descent, aka Vanilla gradient descent, calculates the error for each observation in the dataset but performs an update only after all observations have been evaluated.

Batch gradient descent is not often used, because it represents a huge consumption of computational resources, as the entire dataset needs to remain in memory.

Stochastic Gradient Descent:

Stochastic gradient descent (SGD) performs a parameter update for each observation. So instead of looping over each observation, *it just needs one* to perform the parameter update. SGD is usually faster than batch gradient descent, but its frequent updates cause a higher variance in the error rate, that can sometimes jump around instead of decreasing.

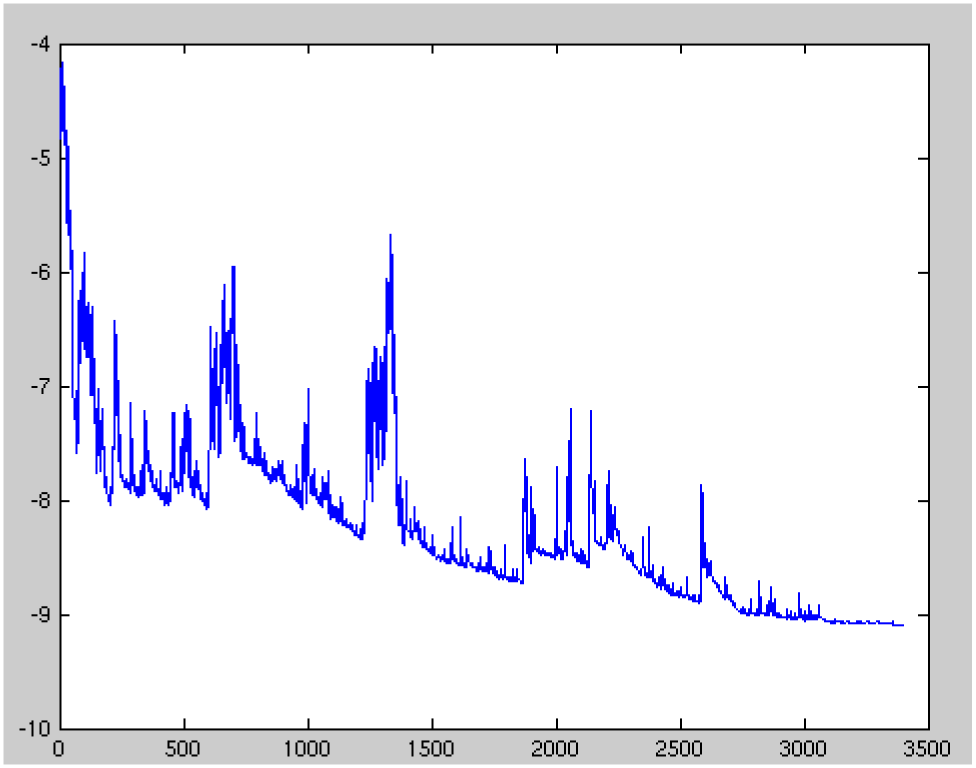


Image 3: SDG fluctuation (Source: Wikipedia)

Mini-Batch Gradient Descent:

It is a combination of both bath gradient descent and stochastic gradient descent. Mini-batch gradient descent performs an update for a batch of observations. It is the algorithm of choice for neural networks, and the batch sizes are usually from 50 to 256.

Image 4: Comparison of the 3 types of GD. (Source: Hackernoon)

That’s it!

There's still a lot more to cover here, but I promise to incorporate more useful concepts.

Useful Resources:

- An overview of gradient descent optimization algorithms
- Gradient checks
- Gradient descent working example

This post was originally published at my blog www.sara-codes.com

Machine Learning Gradient Descent Deep Learning Linear Regression Algorithms