Gradient Descent: How it works?

A.Asta*

Abstract

Gradient descent or gradient algorithm is an iterative process that calculates the minimum of a function. This algorithm is widely used in Machine Learning. In this article, we will present the gradient descent algorithm, explain how it works with a practical example implemented using Python and introduce its different three types.

Keywords: Optimisation, Gradient Descent Algorithm, Learning Rate, Batch Gradient Descent, Stochastic Gradient Descent, Mini-batch gradient descent, Cost Function.

1. Introduction

When an optimization problem is not deterministically soluble, there are algorithms used to find an approximate solution on the condition, however, that the function to be maximized or minimized is derivable, which is the case with many application fields such as neural networks. In a similar situation, many variants will be proposed to solve such problems and are grouped under the term gradient descent (GD).

The gradient descent is an optimization algorithm that calculates the local minimum of a (convex) function by changing (iterations) the parameters of this function. In other words, the gradient descent is an algorithm for finding the local minimum of a differentiable function. Gradient descent is simply used to find values for the parameters of a function to achieve this local minimum. This famous algorithm was invented by the French mathematician Louis Augustin Cauchy in 1847 in [1]. With the method becoming increasingly well-studied and used in the following decades, it is also often called by the name steepest descent.

*Corresponding author
Email address: amira.asta02@gmail.com (A.Asta)

This article is divided in two main parts. The first part aims at providing intuitions towards the behaviour of the gradient descent and explain how it works with a practical example implemented with Python framework. While in the second part, We are going to look at the different variants of the gradient descent algorithm and explain the difference between them.

2. Gradient Descent Algorithm

2.1. Basic Intuition

The Gradient Descent algorithm may seem complicated to understand, but the intuition behind it is quite simple. In order to understand the gradient descent algorithm you can imagine that you are on a hill, and you want to go down it. With each new step, you look around to find the best slope to move down. Once you have found the slope, you take a step forward with one magnitude α.
This is exactly the principle of Gradient Descent. The idea is to find the minimum of a mathematical function but without solving it in a "traditional" way. We will instead set up an iterative algorithm that will move step by step to approach the solution and therefore find our minimum. One can ask why we are doing this and quite simply because it is the only way to solve certain equations which are irresolvable as they are [2].
Now that we are familiar with an idea of gradient descent optimization, let's look at how we might implement this algorithm with Python.

2.2. How Gradient Descent Works: Example

In this section, we will take a closer look at the gradient descent algorithm. The gradient descent algorithm requires a target function that is being optimized, also called cost function and its derivative (the gradient). We will work through an example of applying gradient descent step by step to a simple test optimization function.
First, let's define a simple one-dimensional optimization function that squares the input and defines the range of the valid inputs from -1 to 1.

$$f(x) = x^2 \tag{1}$$

Where:
f: the cost function.
x: input values vector in the precised range from -1 to 1.

Next, our first step towards applying the gradient descent algorithm will be to calculate the derivative for this function also called gradient.

$$f'(x) = 2x \tag{2}$$

Where:
f': indicates the derivative (gradient) of the function f for the given set of inputs x.

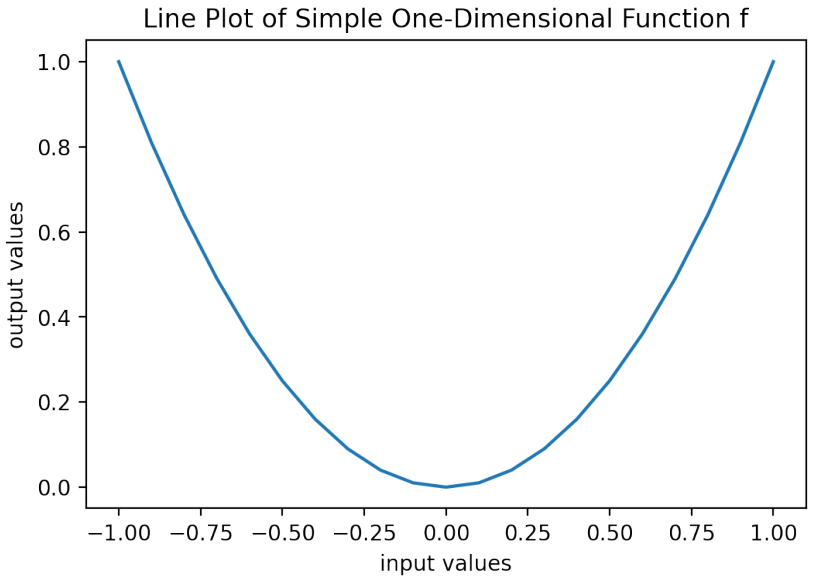

Figure 1: Plot of the function f.

A step is taken in the input space that is expected to result in a downhill movement in the optimized function as shown in Figure 1. A downhill movement is made by first calculating how far to move in the input space, calculated as the step size α, also named the learning rate, multiplied by the gradient. This is then subtracted from the current point, ensuring we move against the gradient, or down the optimized function.

$$x_{new} = x - \alpha f'(x) \tag{3}$$

Where:
α: Hyper-parameter that controls how far to move in the input space against

the gradient for each iteration of the algorithm.

The steeper the optimized function at a given point, the larger the magnitude of the gradient, and in turn, the larger the step taken in the input space. The size of the step taken is scaled using the step size hyper-parameter α. If the step size is too small, the movement in the input space will be small and the search will take a long time. However, If the step size is too large, the search may bounce around the input space and skip over the optima. We use a step size of 0.1 and 30 iterations, both found after a little experimentation. In this case, we can see from Figure 2 below, that the algorithm finds a good solution after about $20 - 30$ iterations with a function evaluation of about 0.0. Note that the optima for this function calculated manually is at $f(0.0) = 0.0$.

In our case, we can see that from Figure 3 shown below in the color red, the search started about halfway up the left part of the function and stepped downhill to the bottom of the basin. We can see that in the parts of the optimized function with the larger curve, the derivative (gradient) is larger, and in turn, larger steps are taken. Similarly, the gradient is smaller as we get closer to the optima, and in turn, smaller steps are taken.

This highlights that the step size is used as a scale factor on the magnitude of the gradient (curvature) of the optimized function.

3. Types of Gradient Descent Algorithms

Various variants of gradient descent are defined on the basis of how we use the data to calculate derivative of cost function in gradient descent. Depending upon the amount of data used, three popular types are discussed:

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

3.1. Batch Gradient Descent

Batch gradient descent, also called vanilla gradient descent [3]. It calculates the error for each example within the training data set, but only after all training examples have been evaluated does the model get updated. This whole process is like a cycle and it's called a training epoch. It is the first basic type of gradient descent in which we use the complete dataset available to compute the gradient of cost function.

```
>0 f([-0.2164454]) = 0.04685
>1 f([-0.17315632]) = 0.02998
>2 f([-0.13852505]) = 0.01919
>3 f([-0.11082004]) = 0.01228
>4 f([-0.08865603]) = 0.00786
>5 f([-0.07092483]) = 0.00503
>6 f([-0.05673986]) = 0.00322
>7 f([-0.04539189]) = 0.00206
>8 f([-0.03631351]) = 0.00132
>9 f([-0.02905081]) = 0.00084
>10 f([-0.02324065]) = 0.00054
>11 f([-0.01859252]) = 0.00035
>12 f([-0.01487401]) = 0.00022
>13 f([-0.01189921]) = 0.00014
>14 f([-0.00951937]) = 0.00009
>15 f([-0.0076155]) = 0.00006
>16 f([-0.0060924]) = 0.00004
>17 f([-0.00487392]) = 0.00002
>18 f([-0.00389913]) = 0.00002
>19 f([-0.00311931]) = 0.00001
>20 f([-0.00249545]) = 0.00001
>21 f([-0.00199636]) = 0.00000
>22 f([-0.00159709]) = 0.00000
>23 f([-0.00127767]) = 0.00000
>24 f([-0.00102213]) = 0.00000
>25 f([-0.00081771]) = 0.00000
>26 f([-0.00065417]) = 0.00000
>27 f([-0.00052333]) = 0.00000
>28 f([-0.00041867]) = 0.00000
>29 f([-0.00033493]) = 0.00000
Done!
f([-0.00033493]) = 0.000000
```

Figure 2: Results of the Gradient Descent after 30 iterations.

Some advantages of batch gradient descent are its computational efficient, it produces a stable error and a stable convergence. Some disadvantages are the stable error can sometimes result in a state of convergence that is not the best the model can achieve. It also requires the entire training dataset to be in memory and available to the algorithm.

3.2. Stochastic Gradient Descent

Batch Gradient Descent turns out to be a slower algorithm. So, for faster computation, we prefer to use the second type of GD called stochastic gradient descent [4]. By contrast, stochastic gradient descent (SGD) calculates

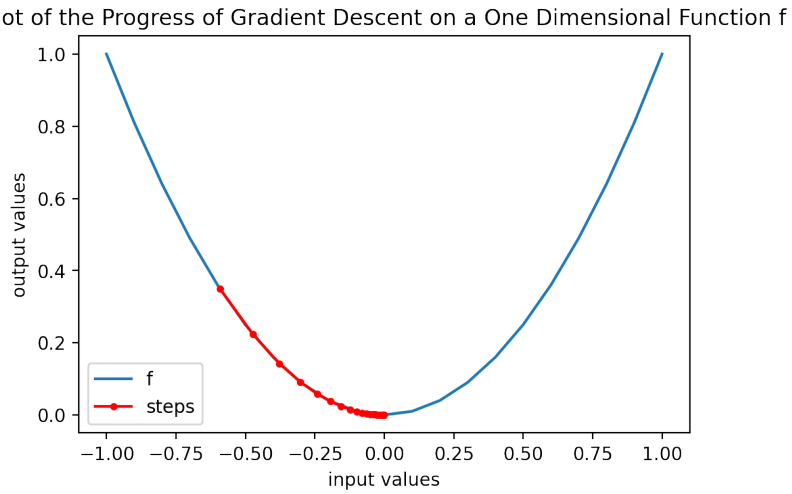

Figure 3: The Progress of Gradient Descent on a One Dimensional Function f.

the error for each training example within the dataset. It updates the parameters for each training example one by one. Depending on the problem, this can make SGD faster than batch gradient descent. One advantage is that the frequent updates allow us to have a pretty detailed rate of improvement.

The frequent updates, however, are more computationally expensive than the batch gradient descent approach. Additionally, the frequency of those updates can result in noisy gradients, which may cause the error rate to jump around instead of slowly decreasing.

3.3. Mini-Batch Gradient Descent

The third type of GD algorithm is called Mini Batch Gradient Descent. It is an often-preferred method since it uses a combination of both Stochastic Gradient Descent and Batch Gradient Descent [5]. It simply separates the training set into small batches and performs an update for each of these batches. It creates a balance between the efficiency of Batch Gradient Descent and the robustness of Stochastic Gradient Descent. Common numbers of examples per batch range between 30 and 500. But like for any other machine learning technique, there is no well-defined rule because the optimal number can vary for different problems. Mini Batch Gradient Descent is commonly used for deep learning problems. The table below shows a quick comparison for all three types of gradient descent algorithms discussed above:

Table 1: Comparison of 3 different Gradient Descent algorithms

Parameters	**Accuracy**	**Time Consuming**
Batch GD Algorithm	High	More
Mini-Batch GD Algorithm	Moderate	Moderate
Stochastic GD Algorithm	Low	Less

4. Conclusion

We have just seen how the Gradient Descent algorithm works. The latter is a must know in Machine Learning and deep learning fields. In this article, we learned about the basics of gradient descent algorithm and its types. These optimization algorithms are being widely used in neural networks these days. Hence, it's important to understand the difference between them.

References

[1] C. Lemaréchal, Cauchy and the gradient method, Doc Math Extra 251 (2012) 10.

[2] P. Baldi, Gradient descent learning algorithm overview: A general dynamical systems perspective, IEEE Transactions on neural networks 6 (1995) 182–195.

[3] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747 (2016).

[4] L. Bottou, Stochastic gradient descent tricks, in: Neural networks: Tricks of the trade, Springer, 2012, pp. 421–436.

[5] S. Khirirat, H. R. Feyzmahdavian, M. Johansson, Mini-batch gradient descent: Faster convergence under data sparsity, in: 2017 IEEE 56th Annual Conference on Decision and Control (CDC), IEEE, pp. 2880–2887.