1 Introduction

Gradient Descent is used while training a machine learning model. It is an optimization algorithm, based on a convex function, that tweaks it's parameters iteratively to minimize a given function to its local minimum.

It is simply used to find the values of a functions parameters (coefficients) that minimize a cost function as far as possible.

You start by defining the initial parameters values and from there on Gradient Descent iteratively adjusts the values, using calculus, so that they minimize the given cost-function. But to understand it's concept fully, you first need to know what a gradient is.

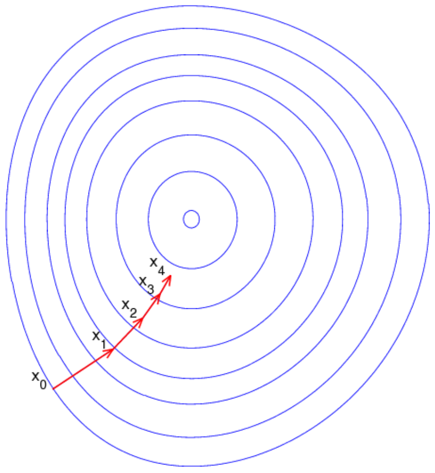
1.1 What is a Gradient?

"A gradient measures how much the output of a function changes if you change the inputs a little bit."

It simply measures the change in all weights with regard to the change in error. You can also think of a gradient as the 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. Said it more mathematically, a gradient is a partial derivative with respect to its inputs.

Imagine a blindfolded man who wants to climb a hill, with the fewest steps possible. He just starts climbing the hill by taking really big steps in the steepest direction, which he can do, as long as he is not close to the top. As he comes further to the top, he will do smaller and smaller steps, since he doesn't want to overshoot it. This process can be described mathematically, using the gradient.

Just take a look at the image below. Imagine it illustrates our hill from a top-down view, where the red arrows show the steps of our climber.Think of a gradient in this context as a vector that contains the direction of the steepest step the blindfolded man can go and also how long this step should be.



Note that the gradient ranging from X0 to X1 is much longer than the one reaching from X3 to X4. This is because the steepness/slope of the hill is less there, which determines the length of the vector. This perfectly represents the example of the hill, because the hill is getting less steep, the higher you climb it. Therefore a reduced gradient goes along with a reduced slope and a reduced step-size for the hill climber.

2 How it works?

Gradient Descent can be thought of climbing down to the bottom of a valley, instead of climbing up a hill. This is because it is a minimization algorithm that minimizes a given function.

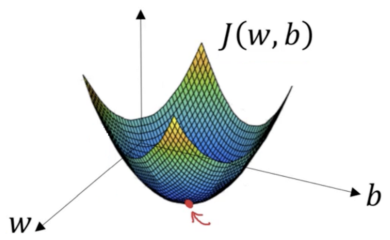
The equation below describes what Gradient Descent does: ”b” describes the next position of our climber, while ”a” represents his current position. The ”-” sign refers to the minimization part of gradient descent. The ”eta“ in the middle is a waiting factor and the gradient term $\Delta f(a)$ is simply the direction of the steepest descent.

$$\mathrm{b} = \mathrm{a} - \eta\Delta\mathrm{f(a)}$$

So this formula basically tells you the next position where you need to go, which is the direction of the steepest descent.

Another example, Imagine you are dealing with a machine learning problem and want to train your algorithm with gradient descent to minimize your cost-function J(w, b) and reach its local minimum by tweaking its parameters (w and b).

Let’s take a look at the picture below, which is an illustration of Gradient Descent. The horizontal axes represent the parameters (w and b) and the cost function J(w, b) is represented on the vertical axes. You can also see in the image that gradient descent is a convex function.



Like we already know that we want to find the values of W and B that correspond to the minimum of the cost function (marked with the red arrow).

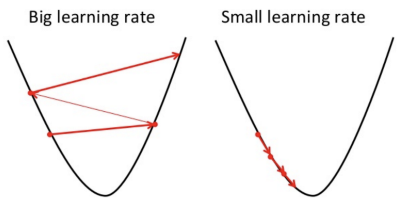
To start with finding the right values we initialize the values of W and B with some random numbers and Gradient Descent then starts at that point (somewhere around the top of our illustration). Then it takes one step after another in the steepest downside direction (e.g. from the top to the bottom of the illustration) till it reaches the point where the cost function is as small as possible.

3 Importance of the Learning Rate

How big the steps are that Gradient Descent takes into the direction of the local minimum are determined by the so-called learning rate. It determines how fast or slow we will move towards the optimal weights.

In order for Gradient Descent to reach the local minimum, we have to set the learning rate to an appropriate value, which is neither too low nor too high.

This is because if the steps it takes are too big, it maybe will not reach the local minimum because it just bounces back and forth between the convex function of gradient descent like you can see on the left side of the image below. If you set the learning rate to a very small value, gradient descent will eventually reach the local minimum but it will maybe take too much time like you can see on the right side of the image.



4 Types of Gradient Descent

There are three popular types of Gradient Descent, that mainly differ in the amount of data they use. We go through them one by one.

4.1 Batch Gradient Descent

Batch Gradient Descent, also called vanilla gradient descent, calculates the error for each example within the training dataset, but only after all training examples have been evaluated, the model gets updated. This whole process is like a cycle and called a training epoch.

Advantages of it are that it's computational efficient, it produces a stable error gradient and a stable convergence. Batch Gradient Descent has the disadvantage that the stable error gradient can sometimes result in a state of convergence that isn't the best the model can achieve. It also requires that the entire training dataset is in memory and available to the algorithm.

4.2 Stochastic Gradient Descent

Stochastic gradient descent (SGD) in contrary, does this for each training example within the dataset. This means that it updates the parameters for each training example, one by one. This can make SGD faster than Batch Gradient Descent, depending on the problem. One advantage is that the frequent updates allow us to have a pretty detailed rate of improvement.

The thing is that the frequent updates are more computationally expensive as the approach of Batch Gradient Descent. The frequency of those updates can also result in noisy gradients, which may cause the error rate to jump around, instead of slowly decreasing.

4.3 Mini Batch Gradient Descent

Mini-batch Gradient Descent is the go-to method since it's a combination of the concepts of SGD and Batch Gradient Descent. It simply splits the training dataset into small batches and performs an update for each of these batches. Therefore it creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent.

Common mini-batch sizes range between 50 and 256, but like for any other machine learning techniques, there is no clear rule, because they can vary for different applications. Note that it is the go-to algorithm when you are training a neural network and it is the most common type of gradient descent within deep learning.

References

The blog `https://medium.com/@koolanalytics/gradient-descent-simply-explained-1d2baa65c757` simple Gradient Descent
The blog `http://mccormickml.com/2014/03/04/gradient-descent-derivation/` Gradient Descent Derivation