CHAPTER 16

Optimization for Machine Learning: Gradient Descent

Gradient descent is an optimization algorithm that is used to minimize the cost function of a machine learning algorithm. Gradient descent is called an iterative optimization algorithm because, in a stepwise looping fashion, it tries to find an approximate solution by basing the next step off its present step until a terminating condition is reached that ends the loop.

Take the following convex function in Figure 16-1 as a visual of gradient descent finding the minimum point of a function space.

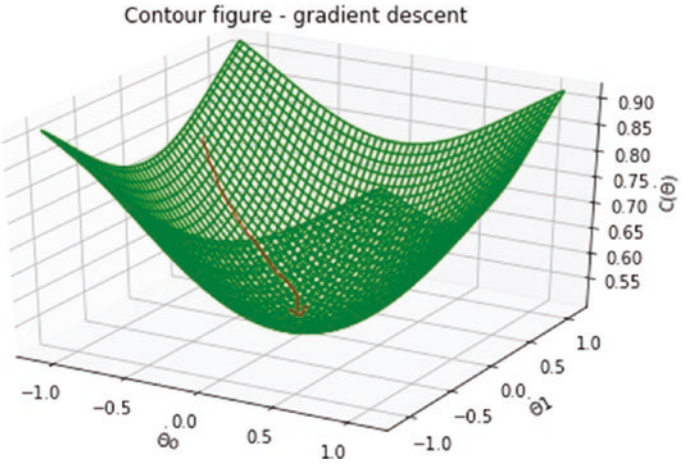

Figure 16-1. *Contour figure – gradient descent*

E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform*,
https://doi.org/10.1007/978-1-4842-4470-8_16

The image in Figure 16-1 is an example of a function space. This type of function is known as a ***convex or a bowl-shaped function***. The role of gradient descent in the function space is to find the set of values for the parameters of the function that minimizes the cost of the function and brings it to the global minimum. The global minimum is the lowest point of the function space.

For example, the mean squared error cost function for linear regression is nicely convex, so gradient descent is almost guaranteed to find the global minimum. However, this is not always the case for other types of non-convex function spaces. Remember, gradient descent is a global optimizer for minimizing any function space.

Some functions may have more than one minimum region; these regions are called local minima. The lowest region of the function space is called the global minimum.

The Learning Rate of Gradient Descent Algorithm

Learning rate is a hyper-parameter that controls how big a step the gradient descent algorithm takes when tracing its path in the direction of steepest descent in the function space.

If the learning rate is too large, the algorithm takes a large step as it goes downhill. In doing so, gradient descent runs faster, but it has a high propensity of missing the global minimum. An overly small learning rate makes the algorithm slow to converge (i.e., to reach the global minimum), but it is more likely to converge to the global minimum steadily. Empirically, examples of good learning rates are values in the range of 0.001, 0.01, and 0.1. In Figure 16-2, with a good learning rate, the cost function $C(\theta)$ should decrease after every iteration.

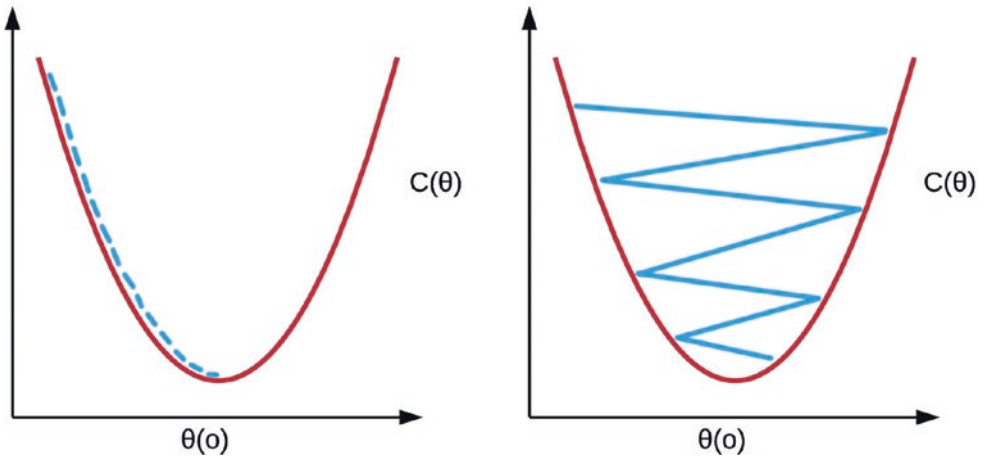

Figure 16-2. *Learning rates.* ***Left:*** *Good learning rate.* ***Right:*** *Bad learning rate.*