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.

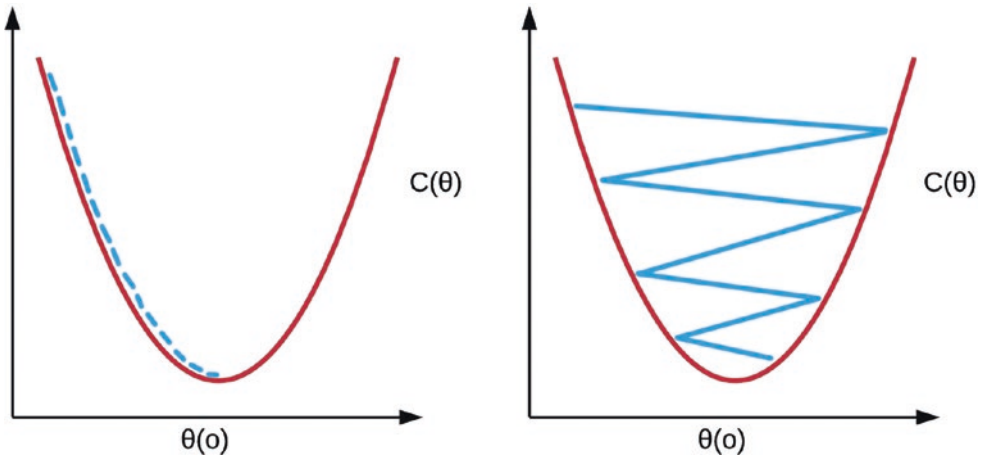

Figure 16-2. *Learning rates.* ***Left:*** *Good learning rate.* ***Right:*** *Bad learning rate.*

Classes of Gradient Descent Algorithm

The three types of gradient descent algorithms are

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

The ***batch gradient descent*** algorithm uses the entire training data in computing each step of the gradient in the direction of steepest descent. Batch gradient descent is most likely to converge to the global minimum. However, the disadvantage of this method is that, for massive datasets, the optimization process can be prolonged.

In ***stochastic gradient descent (SGD)***, the algorithm quickly learns the direction of steepest descent using a single example of the training set at each time step. While this method has the distinct advantage of being fast, it may never converge to the global minimum. However, it approximates the global minimum closely enough. In practice, SGD is enhanced by gradually reducing the learning rate over time as the algorithm converges. In doing this, we can take advantage of large step sizes to go downhill more quickly and then slow down so as not to miss the global minimum. Due to its speed when dealing with humongous datasets, SGD is often preferred to batch gradient descent.

Mini-batch gradient descent on the other hand randomly splits the dataset into manageable chunks called mini-batches. It operates on a mini-batch in each time step to learn the direction of steepest descent of the function. This method is a compromise between stochastic and batch gradient descent. Just like SGD, mini-batch gradient descent does not converge to the global minimum. However, it is more robust in avoiding local minimum. The advantage of mini-batch gradient descent over stochastic gradient descent is that it is more computational efficient by taking advantage of matrix vectorization under the hood to efficiently compute the algorithm updates.

Optimizing Gradient Descent with Feature Scaling

This process involves making sure that the features in the dataset are all on the same scale. Typically all real-valued features in the dataset should lie between $-1 \leq x_i \leq 1$ or a range around that region. Any range too large or arbitrarily too small can generate a contour plot that is too narrow and hence will take a longer time for gradient descent