2.1. Gradient Descent

The *gradient descent* (GD) method is employed to find the minimum of a differentiable, convex or non-convex function, commonly referred to as the "cost" or "loss" function (also known as the "objective" function). It stands out as one of the most popular algorithms to perform optimization and by far the most common way to optimize machine learning, deep learning, and various optimization problems. And this is particularly true for optimizing neural networks. In the context of machine learning, the cost function measures the difference between the predicted output of a model and the actual output. The neural networks or machine learning in general find the set of parameters $\boldsymbol{x} \in \mathbb{R}^d$ (also known as weights) in order to optimize an objective function $L(\boldsymbol{x})$. The gradient descent aims to find a sequence of parameters:

$$\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T, \tag{2.1.1}$$

such that as $T \to \infty$, the objective function $L(\boldsymbol{x}_T)$ attains the optimal minimum value. At each iteration t, a step $\Delta\boldsymbol{x}_t$ is applied to update the parameters. Denoting the parameters at the t-th iteration by $\boldsymbol{x}_t$. Then the update rule becomes

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \Delta\boldsymbol{x}_t. \tag{2.1.2}$$

The most straightforward gradient descents is the *vanilla update*: the parameters move in the opposite direction of the gradient, which finds the steepest descent direction since the gradients are orthogonal to level curves (also known as level surfaces, see Lemma 2.4.1):

$$\Delta\boldsymbol{x}_t = -\eta \boldsymbol{g}_t = -\eta \frac{\partial L(\boldsymbol{x}_t)}{\partial \boldsymbol{x}_t} = -\eta \nabla L(\boldsymbol{x}_t), \tag{2.1.3}$$

where the positive value η denotes the *learning rate* and depends on specific problems, and $\boldsymbol{g}_t = \frac{\partial L(\boldsymbol{x}^t)}{\partial \boldsymbol{x}^t} \in \mathbb{R}^d$ represents the gradient of the parameters. The learning rate η controls how large of a step to take in the direction of negative gradient so that we can reach a (local) minimum. While if we follow the negative gradient of a single sample or a batch of samples iteratively, the local estimate of the direction can be obtained and is known as the *stochastic gradient descent* (SGD) (Robbins and Monro, 1951). The SGD can be categorized into two types:

- **The strict SGD:** Computes the gradient for only one randomly chosen data point in each iteration.
- **The mini-batch SGD:** Represents a compromise between GD and strict SGD, using a subset (mini-batch) of the dataset to compute the gradient.

The SGD method is particular useful when the number of training entries (i.e., the data used for updating the model, while the data used for final evaluation is called the test entries or test data) are substantial, resulting in that the gradients from different input samples may cancel out and the final update is small. In the SGD framework, the objective function is stochastic, composed of a sum of subfunctions evaluated at different subsamples of the data. However, the drawback of the vanilla update (both GD and SGD) lies in its susceptibility to getting trapped in local minima (Rutishauser, 1959).