the second derivatives of the cost function. Finally, we conclude with a review of several optimization strategies that are formed by combining simple optimization algorithms into higher-level procedures.

8.1 How Learning Differs from Pure Optimization

Optimization algorithms used for training of deep models differ from traditional optimization algorithms in several ways. Machine learning usually acts indirectly. In most machine learning scenarios, we care about some performance measure P, that is defined with respect to the test set and may also be intractable. We therefore optimize P only indirectly. We reduce a different cost function $J(\boldsymbol{\theta})$ in the hope that doing so will improve P. This is in contrast to pure optimization, where minimizing J is a goal in and of itself. Optimization algorithms for training deep models also typically include some specialization on the specific structure of machine learning objective functions.

Typically, the cost function can be written as an average over the training set, such as

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\mathrm{y})\sim\hat{p}_{\mathrm{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y), \tag{8.1}$$

where L is the per-example loss function, $f(\boldsymbol{x};\boldsymbol{\theta})$ is the predicted output when the input is $\boldsymbol{x}$, $\hat{p}_{\mathrm{data}}$ is the empirical distribution. In the supervised learning case, y is the target output. Throughout this chapter, we develop the unregularized supervised case, where the arguments to L are $f(\boldsymbol{x};\boldsymbol{\theta})$ and y. However, it is trivial to extend this development, for example, to include $\boldsymbol{\theta}$ or $\boldsymbol{x}$ as arguments, or to exclude y as arguments, in order to develop various forms of regularization or unsupervised learning.

Equation 8.1 defines an objective function with respect to the training set. We would usually prefer to minimize the corresponding objective function where the expectation is taken across *the data generating distribution* p_{data} rather than just over the finite training set:

$$J^*(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\mathrm{y})\sim p_{\mathrm{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y). \tag{8.2}$$

8.1.1 Empirical Risk Minimization

The goal of a machine learning algorithm is to reduce the expected generalization error given by equation 8.2. This quantity is known as the **risk**. We emphasize here that the expectation is taken over the true underlying distribution p_{data}. If we knew the true distribution $p_{\mathrm{data}}(\boldsymbol{x}, y)$, risk minimization would be an optimization task