One way to define the problem of **model fitting** or **training** is to find a setting of the parameters that minimizes the empirical risk on the training set:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \mathcal{L}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(\boldsymbol{x}_n; \boldsymbol{\theta})) \tag{1.6}$$

This is called **empirical risk minimization**.

However, our true goal is to minimize the expected loss on *future data* that we have not yet seen. That is, we want to **generalize**, rather than just do well on the training set. We discuss this important point in Section 1.2.3.

1.2.1.5 Uncertainty

[We must avoid] false confidence bred from an ignorance of the probabilistic nature of the world, from a desire to see black and white where we should rightly see gray. — Immanuel Kant, as paraphrased by Maria Konnikova [Kon20].

In many cases, we will not be able to perfectly predict the exact output given the input, due to lack of knowledge of the input-output mapping (this is called **epistemic uncertainty** or **model uncertainty**), and/or due to intrinsic (irreducible) stochasticity in the mapping (this is called **aleatoric uncertainty** or **data uncertainty**).

Representing uncertainty in our prediction can be important for various applications. For example, let us return to our poisonous flower example, whose loss matrix is shown in Table 1.2. If we predict the flower is Virginica with high probability, then we should not eat the flower. Alternatively, we may be able to perform an **information gathering action**, such as performing a diagnostic test, to reduce our uncertainty. For more information about how to make optimal decisions in the presence of uncertainty, see Section 5.1.

We can capture our uncertainty using the following **conditional probability distribution**:

$$p(y = c|\boldsymbol{x}; \boldsymbol{\theta}) = f_c(\boldsymbol{x}; \boldsymbol{\theta}) \tag{1.7}$$

where $f : \mathcal{X} \to [0, 1]^C$ maps inputs to a probability distribution over the C possible output labels. Since $f_c(\boldsymbol{x}; \boldsymbol{\theta})$ returns the probability of class label c, we require $0 \leq f_c \leq 1$ for each c, and $\sum_{c=1}^{C} f_c = 1$. To avoid this restriction, it is common to instead require the model to return unnormalized log-probabilities. We can then convert these to probabilities using the **softmax function**, which is defined as follows

$$\text{softmax}(\boldsymbol{a}) \triangleq \left[\frac{e^{a_1}}{\sum_{c'=1}^{C} e^{a_{c'}}}, \ldots, \frac{e^{a_C}}{\sum_{c'=1}^{C} e^{a_{c'}}} \right] \tag{1.8}$$

This maps $\mathbb{R}^C$ to $[0, 1]^C$, and satisfies the constraints that $0 \leq \text{softmax}(\boldsymbol{a})_c \leq 1$ and $\sum_{c=1}^{C} \text{softmax}(\boldsymbol{a})_c = 1$. The inputs to the softmax, $\boldsymbol{a} = f(\boldsymbol{x}; \boldsymbol{\theta})$, are called **logits**. See Section 2.5.2 for details. We thus define the overall model as follows:

$$p(y = c|\boldsymbol{x}; \boldsymbol{\theta}) = \text{softmax}_c(f(\boldsymbol{x}; \boldsymbol{\theta})) \tag{1.9}$$