pattern recognition problem with $x_i \neq x_j$ for $i \neq j$, we could set

$$f(x) = \begin{cases} y_i & \text{if } x = x_i \text{ for some } i = 1, \ldots, m \\ 1 & \text{otherwise.} \end{cases} \tag{5.4}$$

This does not amount to any form of learning, however: suppose we are now given a test point drawn from the same distribution, $(x, y) \sim P(x, y)$. If $\mathcal{X}$ is a continuous domain, and we are not in a degenerate situation, the new pattern x will almost never be exactly equal to any of the training inputs x_i. Therefore, the learning machine will almost always predict that $y = 1$. *If we allow* all *functions from $\mathcal{X}$ to $\mathcal{Y}$, then the values of the function at points $x_1, \ldots, x_m$ carry no information about the values at other points.* In this situation, a learning machine cannot do better than chance. This insight lies at the core of the so-called *No-Free-Lunch Theorem* popularized in [608]; see also [254, 48].

The message is clear: if we make no restrictions on the class of functions from which we choose our estimate f, we cannot hope to learn anything. Consequently, machine learning research has studied various ways to implement such restrictions. In statistical learning theory, these restrictions are enforced by taking into account the *complexity* or *capacity* (measured by VC dimension, covering numbers, entropy numbers, or other concepts) of the class of functions that the learning machine can implement.[1]

In the Bayesian approach, a similar effect is achieved by placing *prior distributions $P(f)$* over the class of functions (Chapter 16). This may sound fundamentally different, but it leads to algorithms which are closely related; and on the theoretical side, recent progress has highlighted intriguing connections [92, 91, 353, 238].

5.2 The Law of Large Numbers

Let us step back and try to look at the problem from a slightly different angle. Consider the case of pattern recognition using the misclassification loss function. Given a fixed function f, then for each example, the loss $\xi_i := \frac{1}{2}|f(x_i) - y_i|$ is either

1. As an aside, note that the same problem applies to *training on the test set* (sometimes called *data snooping*): sometimes, people optimize tuning parameters of a learning machine by looking at how they change the results on an independent test set. Unfortunately, once one has adjusted the parameter in this way, the test set is not independent anymore. This is identical to the corresponding problem in training on the *training* set: once we have chosen the function to minimize the training error, the latter no longer provides an unbiased estimate of the test error. Overfitting occurs much faster on the training set, however, than it does on the test set. This is usually due to the fact that the number of tuning parameters of a learning machine is much smaller than the total number of parameters, and thus the capacity tends to be smaller. For instance, an SVM for pattern recognition typically has two tuning parameters, and optimizes m weight parameters (for a training set size of m). See also Problem 5.3 and [461].