

meaningful about these new configurations? Many traditional machine learning algorithms simply assume that the output at a new point should be approximately the same as the output at the nearest training point.

5.11.2 Local Constancy and Smoothness Regularization

In order to generalize well, machine learning algorithms need to be guided by prior beliefs about what kind of function they should learn. Previously, we have seen these priors incorporated as explicit beliefs in the form of probability distributions over parameters of the model. More informally, we may also discuss prior beliefs as directly influencing the *function* itself and only indirectly acting on the parameters via their effect on the function. Additionally, we informally discuss prior beliefs as being expressed implicitly, by choosing algorithms that are biased toward choosing some class of functions over another, even though these biases may not be expressed (or even possible to express) in terms of a probability distribution representing our degree of belief in various functions.

Among the most widely used of these implicit “priors” is the **smoothness prior** or **local constancy prior**. This prior states that the function we learn should not change very much within a small region.

Many simpler algorithms rely exclusively on this prior to generalize well, and as a result they fail to scale to the statistical challenges involved in solving AI-level tasks. Throughout this book, we will describe how deep learning introduces additional (explicit and implicit) priors in order to reduce the generalization error on sophisticated tasks. Here, we explain why the smoothness prior alone is insufficient for these tasks.

There are many different ways to implicitly or explicitly express a prior belief that the learned function should be smooth or locally constant. All of these different methods are designed to encourage the learning process to learn a function f^* that satisfies the condition

$$f^*(\mathbf{x}) \approx f^*(\mathbf{x} + \epsilon) \quad (5.103)$$

for most configurations \mathbf{x} and small change ϵ . In other words, if we know a good answer for an input \mathbf{x} (for example, if \mathbf{x} is a labeled training example) then that answer is probably good in the neighborhood of \mathbf{x} . If we have several good answers in some neighborhood we would combine them (by some form of averaging or interpolation) to produce an answer that agrees with as many of them as much as possible.

An extreme example of the local constancy approach is the k -nearest neighbors family of learning algorithms. These predictors are literally constant over each