Chapter 12

Nonparametric Regression

12.1 Introduction

In this chapter we consider regression techniques that allow to fit more flexible regression functions $f(X)$. On way to achieve this flexibility is by using basis expansions such as splines. These techniques are already discussed in Chapter 4. In this chapter we will focus on flexible regression methods that do not expand the predictor space but that fit a different simple local regression model at each query point x_0. The local fit is based on the observations close to the target point x_0 and the weights given to the observations are chosen such that the resulting estimating function $\hat{f}(X)$ is a *smooth* function. To determine the weights, a weighting or kernel function $K_\lambda(x_0, x)$ is used as in Section 3.8. Hence, the weight of an observation i depends on the distance of x_i to the target point x_0. As such, kernel based nonparametric regression can be seen as a smooth version of nearest neighbor regression, which gives weight one to all neighbors of the target point x_0. Nearest neighbors for regression is also discussed in this chapter.

12.2 Kernel Based Nonparametric Regression

12.2.1 Local averages

We will start by considering regression with only one predictor variable. The idea here is to estimate the response at the target point x_0 by a local weighted average of the responses of training points at small distance of x_0. The weight function $K_\lambda(x_0, x)$ decreases smoothly with the distance from the target point to ensure a smooth fit $\hat{f}$. The fit is given by

$$\hat{f}(x_0) = \frac{\sum_{i=1}^n K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^n K_\lambda(x_0, x_i)}, \tag{12.1}$$

which is called the *Nadaraya-Watson kernel-weighted average.* The kernel function $K_\lambda(x_0, x)$ often can be re-expressed as

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right). \tag{12.2}$$

Popular choices of D are

$$D(t) = \begin{cases} \frac{3}{4}(1-t^2) & \text{if } |t| \le 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{Epanechnikov quadratic kernel} \tag{12.3}$$

$$D(t) = \begin{cases} (1-|t|^3)^3 & \text{if } |t| \le 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{tri-cube kernel} \tag{12.4}$$

$$D(t) = \phi(t) \quad \text{Gaussian kernel.} \tag{12.5}$$

Figure 12.1 shows the three kernel functions. The Epanechnikov and tri-cube kernels have a compact support (positive on a finite interval) while the Gaussian kernel has a noncompact support. The tri-cube function is flatter on the top than the Epanechnikov kernel which yields more efficient results but can lead to more bias. The tri-cube function has the extra advantage of being differentiable at the boundary of the support.

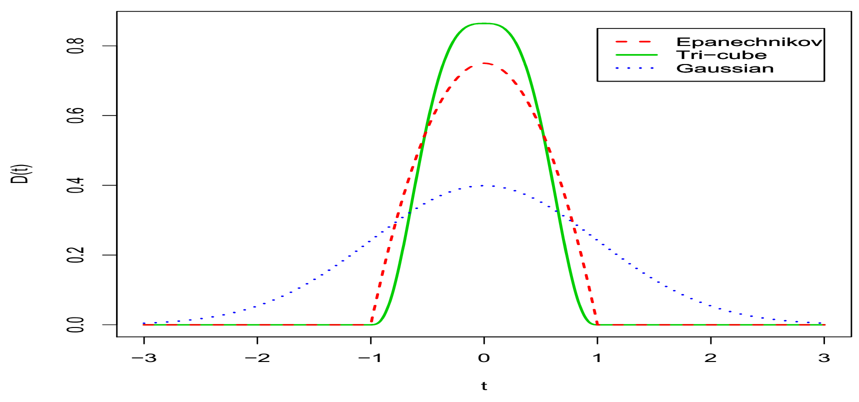

Figure 12.1: Epanechnikov, tri-cube, and Gaussian kernel functions. The tri-cube kernel function has been rescaled to integrate to 1.

The tuning parameter λ controls the window size, that is the minimal distance from the target point at which the weight becomes zero for the compact support kernels. For the noncompact Gaussian kernel, λ is the standard deviation as discussed in section 3.8.1. The choice of λ leads to the usual trade-off between bias (large λ) and variance (small λ).

Locally kernel-weighted averages can have problems at the boundary. At the boundary, the window tends to contain less points and the kernel window becomes asymmetric which can lead to severe bias. Bias can also occur in the interior if the training points are not approximately equally spaced. Fitting higher order local regression models can alleviate the bias effect.

12.2.2 Local linear regression

The bias that can occur with local weighted averages can be reduced by fitting straight lines locally in the window instead of weighted averages. It can be shown that fitting straight lines removes the dominant term of the bias (a first-order correction) and thus reduces the bias substantially.

The local linear fits are obtained by solving a weighted least squares problem at each target point x_0:

$$\min_{\beta_0,\beta_1} = \sum_{i=1}^{n} K_\lambda(x_0, x_i)(y_i - \beta_0 - \beta_1 x_i)^2, \tag{12.6}$$

which yields estimates $\hat{\beta}_0(x_0)$ and $\hat{\beta}_1(x_0)$ at each target point x_0. With these estimates, the fit at x_0 becomes $\hat{f}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0$. Note that at each target point x_0, the fit $\hat{f}(x_0)$ is obtained from a different linear model.

If we introduce the $n \times n$ diagonal weight matrix $\boldsymbol{W}(x_0)$ with diagonal elements $K_\lambda(x_0, x_i)$ $i = 1, \ldots, n$. Then, using the weighted least squares solution of (12.6) the fitted response can be written as

$$\begin{aligned} \hat{f}(x_0) &= \tilde{x}_0^t (\mathbf{X}^t \boldsymbol{W}(x_0)\mathbf{X})^{-1}\mathbf{X}^t \boldsymbol{W}(x_0)\mathbf{y} \\ &= \mathbf{l}(x_0)^t \mathbf{y}, \end{aligned}$$

with $\tilde{x}_0 = (1, x_0)^t$ and $\mathbf{X}$ an $n \times 2$ matrix where the first column is a column of ones and the second column is $\mathbf{x}$, the observed feature vector. Hence, again the fit is obtained by a linear combination of the observed response $\mathbf{y}$. If we consider $\hat{\mathbf{f}}$, the vector of fitted responses at the training points, then we obtain

$$\hat{\mathbf{f}} = \boldsymbol{S}_\lambda^{\text{kernel}} \mathbf{y},$$

where the $n \times n$ matrix $\boldsymbol{S}_\lambda^{\text{kernel}} = (\mathbf{l}(x_1), \ldots, \mathbf{l}(x_n))^t$ is the linear operator. The effective degrees of freedom, trace$(\boldsymbol{S}_\lambda^{\text{kernel}})$ can thus be used to determine the tuning parameter λ.

12.2.3 Local polynomial regression

A natural extension is to replace the local linear fit by a local polynomial fit of degree M, which leads to

$$\min_{\beta_0,\beta_1,\ldots,\beta_m} = \sum_{i=1}^{n} K_\lambda(x_0, x_i)\left(y_i - \beta_0 - \sum_{m=1}^{M} \beta_m x_i^m\right)^2, \tag{12.7}$$

which yields estimates $\hat{\beta}_0(x_0), \ldots, \hat{\beta}_M(x_0)$ and corresponding fit $\hat{f}(x_0) = \hat{\beta}_0(x_0) + \sum_{m=1}^{M} \hat{\beta}_m(x_0) x_0^m$ at each target point x_0. With local degree M ($M > 1$) polynomial fits, the bias will be reduced further. Especially in regions of high curvature, local polynomial fits will have much lower bias than local linear fits, that have a bias effect described by *trimming the hills and filling the valleys*. Of course the bias reduction does not come for free. A higher order model needs to be fit locally (using the same number of observations as the linear model) which yields a decrease of precision and thus an increase in variance.

12.3 Nearest Neighbors Regression

As in classification, *K-nearest-neighbors regression (K-NNR)* first determines the K training points closest to a target x_0 in Euclidean distance. The K-nearest-neighbors regression fit at X_0 is then the average response of these K nearest training points. Nearest neighbor regression thus is also a local regression method, but it uses an adaptive neighborhood size instead of the fixed width (by the choice of λ) neighborhood in the previous section. As each fit is a local average, it is a linear combination of the observed response $\mathbf{y}$. The vector of fitted responses at the training points $\hat{\mathbf{f}}$ can thus be written as

$$\hat{\mathbf{f}} = \boldsymbol{S}_K^{\mathrm{NNR}} \mathbf{y},$$

where $\boldsymbol{S}_K^{\mathrm{NNR}}$ is the linear operator. The effective degrees of freedom, trace($\boldsymbol{S}_K^{\mathrm{NNR}}$) can be used to determine the tuning parameter K.

Note that the kernel based local regression methods in the previous section can be extended by replacing the constant window width λ by a window width function $\lambda(x_0)$ that determines the width of the neighborhood at each target point x_0. For example, we can use the function

$$\lambda(x_0) = |x_0 - x_i|_{K:n} \tag{12.8}$$

which is the distance between x_0 and its Kth closest training point. If we construct an adaptive kernel by using (12.8) in (12.2) with $D(t) = I(|t| \leq 1)$, then the local average (12.1) just equals K-nearest neighbor regression. Straightforward extensions are obtained by using this kernel in (12.6) or (12.7) which yields nearest neighbor methods that locally fit linear or polynomial models instead of simple averages. Adaptive kernel based local regression methods are obtained by using the width function (12.8) in (12.2) and applying the Epanechnikov (12.3), tri-cube (12.4), or Gaussian (12.5) function.

12.4 Local Regression With More Predictors

When there is more than one predictor, linear models can still be fit locally. The weights to determine the local neighborhood are now determined by a d-dimensional kernel function. As in the one-dimensional case, local linear fits have better performance (lower bias) at the boundaries than local constant fits. Note that the boundary problem increases with dimension because the curse of dimensionality implies that a larger fraction of the points lies close to the boundary if the dimension increases. As in the one predictor case, local polynomial fits have smaller bias, especially in high curvature regions, but the variance starts to increase. Since higher order polynomials ($M > 1$) in d dimensions require the inclusion of interaction terms if no restrictions are imposed, the number of parameters, and thus the variance, grows fast.

The d-dimensional kernel functions are typically radial functions. That is, the absolute value $|x - x_0|$ in the right hand side of (12.2) is replaced by the Euclidean norm $\|x - x_0\|$,

$$K_\lambda(x_0, x) = D\left(\frac{\|x - x_0\|}{\lambda}\right), \tag{12.9}$$

which, depending on the choice of D leads to e.g. a radial Epanechnikov kernel or a radial tri-cube kernel. Note that since the Euclidean norm depends on the scaling of the variables, it is advisable to standardize the predictors. The curse of dimensionality implies that local methods as these become less useful in higher dimensions ($d > 3$) since the increasing sparsity of training samples implies that there are no local neighborhoods anymore.