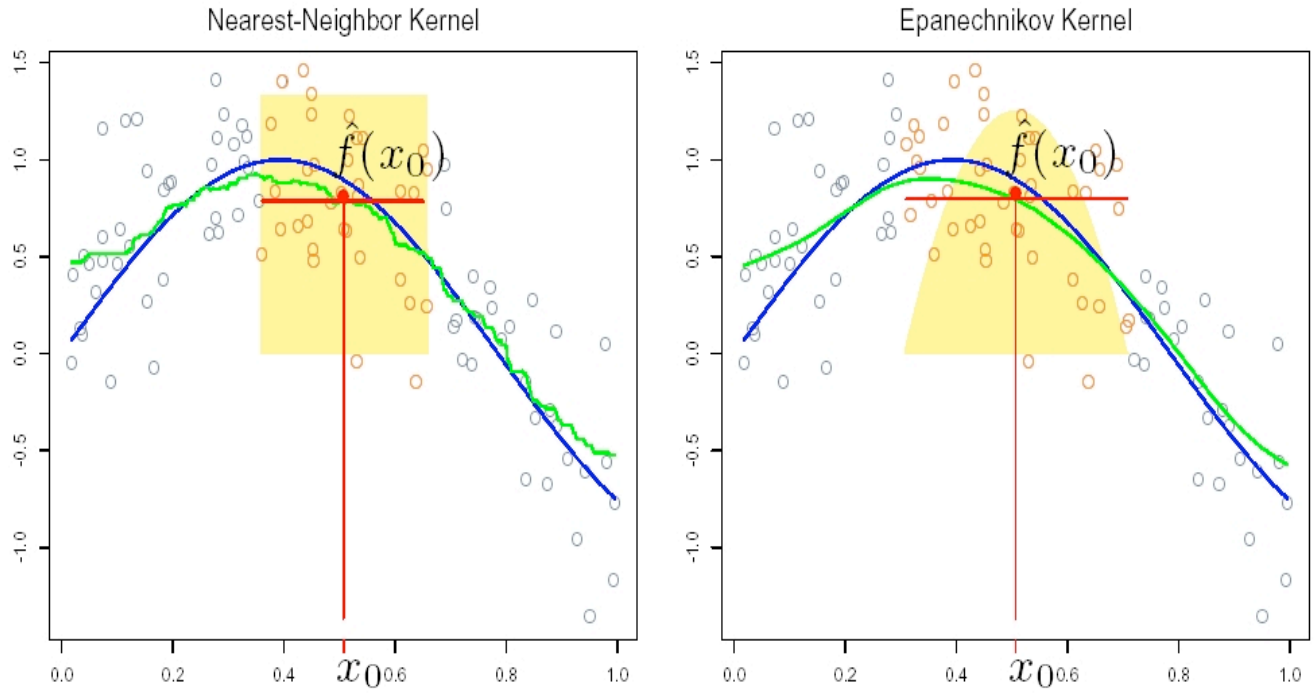
Local Smoothing

$$\hat{f}(x) = \text{Ave}(y_i | x_i \in \mathcal{N}_k(x))$$



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

Kernels

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

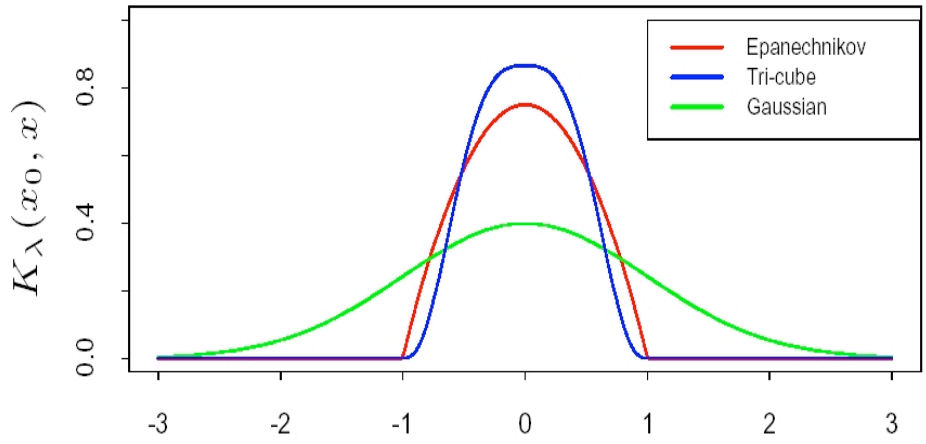


Figure 6.2: *A comparison of three popular kernels for local smoothing. Each has been calibrated to integrate to 1. The tri-cube kernel is compact and has two continuous derivatives at the boundary of its support, while the Epanechnikov kernel has none. The Gaussian kernel is continuously differentiable, but has infinite support.*

$$\text{Gaussian} \quad : \quad D(t) = \phi(t) = e^{-\frac{1}{2}t^2}$$

$$\text{Epanechnikov} \quad : \quad D(t) = \begin{cases} \frac{3}{4}(1-t^2) & if|t| < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Tri-Cube} \quad : \quad D(t) = \begin{cases} (1-t^3)^3 & if|t| < 1 \\ 0 & \text{otherwise} \end{cases}$$

Locally Linear Regression

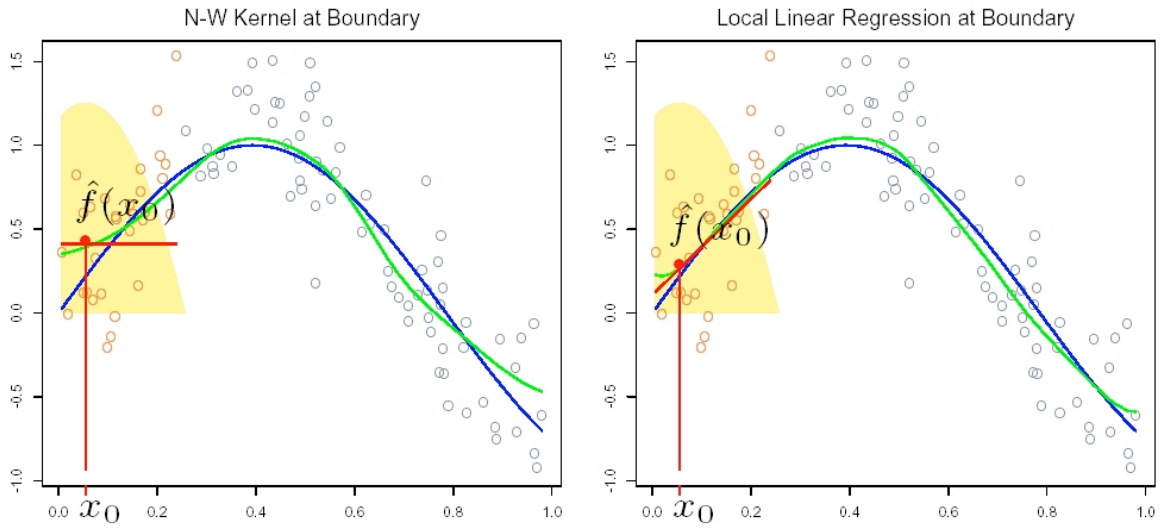


Figure 6.3: *The locally weighted average has bias problems at or near the boundaries of the domain. The true function is approximately linear here, but most of the observations in the neighborhood have a higher mean than the target point, so despite weighting, their mean will be biased upwards. By fitting a locally weighted linear regression (right panel), this bias is removed to first order*

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$$

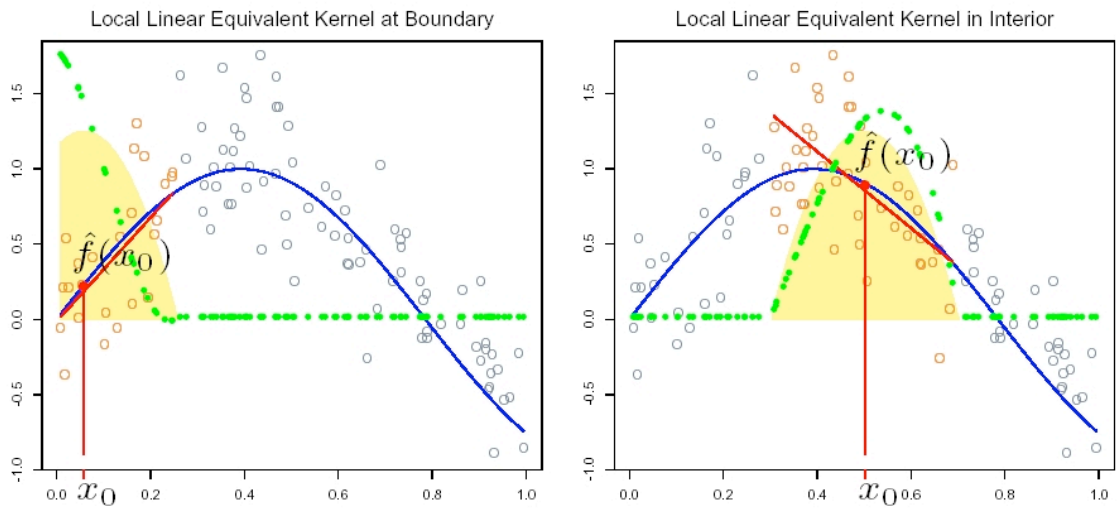


Figure 6.4: *The green points show the equivalent kernel $l_i(x_0)$ for local regression. These are the weights in $\hat{f}(x_0) = \sum_{i=1}^{N} l_i(x_0) y_i$, plotted against their corresponding x_i. For display purposes, these have been rescaled, since in fact they sum to 1. Since the orange shaded region is the (rescaled) equivalent kernel for the Nadaraya–Watson local average, we see how local regression automatically modifies the weighting kernel to correct for biases due to asymmetry in the smoothing window.*

$$\hat{f}(x_0) = b(x_0)^T \left(B^T W(x_0) B\right)^{-1} B^T W(x_0) y = \sum_{i=1}^{N} \ell_i(x_0) y_i$$

$b(x) = (1, x, \ldots)$ and diagonal weight matrix $\{W(x_0)\}_{ii} = K_\lambda(x_0, x_i)$

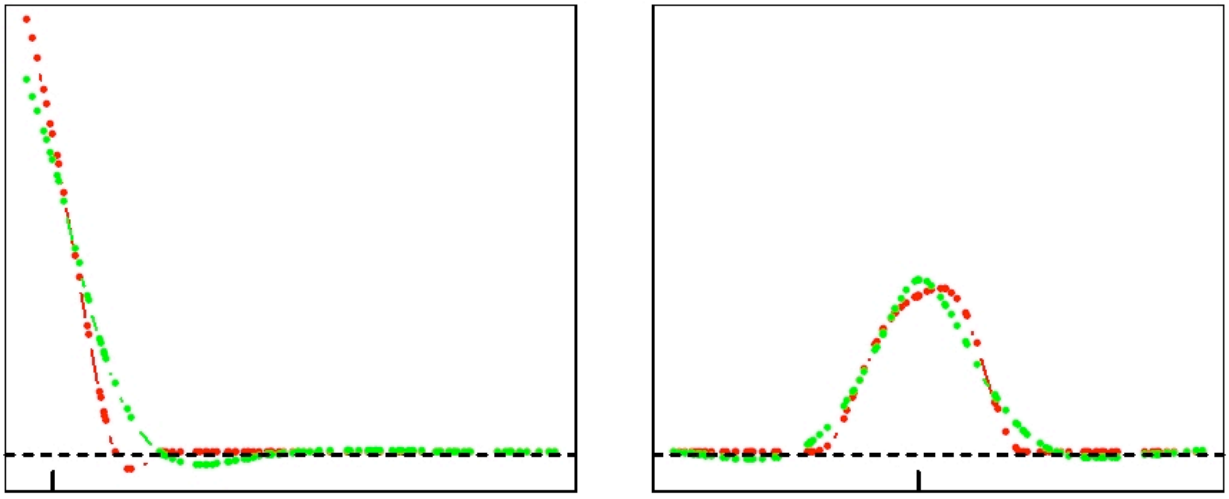


Figure 6.7: *Equivalent kernels for a local linear regression smoother (tri-cube kernel; red) and a smoothing spline (green), with matching degrees of freedom. The vertical spikes indicates the target points.*

$$\mathsf{df}(\lambda) = \mathsf{tr}(S_\lambda)$$

LOESS

The value of $\widehat{f}(x)$ is obtained as follows:

1. The αn points, those with $|x_i - x|$ smallest, are called the neighborhood of x "$\mathcal{N}(x)$".

2. A weighted least-squares linear regression

$$\widehat{f}(x) = \widehat{\beta}_{x,0} + \widehat{\beta}_{x,1} x$$

is fit in $\mathcal{N}(x)$. That is, choose $\widehat{\beta}_{x,0}$ and $\widehat{\beta}_{x,1}$ to minimize

$$\sum_{x_i \in \mathcal{N}(x)} w_{x,i}[y_i - (\beta_0 + \beta_1 x_i)]^2$$

where the weights $w_{x,i} = (1 - u_i^3)^3$ with

$$u_i = \frac{|x_i - x|}{\max_{\mathcal{N}(x)} |x_j - x|}.$$

Additive Models

- Multivariate linear models (Easy to fit, but less flexible):

$$y = \beta_0 + \sum_{j=1}^{n} \beta_j X_j + e.$$

- Nonparametric models (flexible, but impractical to fit such models due to large sample size requirements):

$$y = f(X_1, \ldots, X_p) + e.$$

- A good compromise between the above two

$$y = \beta_0 + \sum_{j=1}^{p} f_j(X_j) + e,$$

where the f_j's are smooth (univariate) functions.

Of course, the additive model will do poorly when strong interactions exist. In this case we might consider adding terms like $f_{ij}(x_i x_j)$ or $f_{ij}(x_i, x_j)$, if there is sufficient data.

Examples

- $y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + e$ (linear model)
- $y = \beta_0 + \left(\beta_{11} x_1 + \beta_{12} x_1^2\right) + \beta_2 \log(x_2) + e$ (linear model with transformed variables)
- $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + f_3(x_3) + e$ (semi-parametric model)

Fitting Additive Models

The backfitting algorithm for $y = \beta_0 + \sum_{j=1}^{p} f_j(X_j) + e$.

- We initialize by setting $\hat{\beta}_0 = \bar{y}.$ and $\hat{f}_j(x) = \hat{\beta}_j X_j$, where $\hat{\beta}_j$ is some initial estimate such as the LS estimate.

- We cycle $j = 1, \ldots, p$ until convergence

$$\hat{f}_j = S\Big(x_j,\ y - \hat{\beta}_0 - \sum_{i \neq j} \hat{f}_i(X_i)\Big),$$

  where $S(x, y)$ means the smooth on the data (x, y). For example, it could be a nonparametric smoother like splines or loess, or a parametric fit like linear or polynomial.

df of Nonparametric Models

- If we use the basis expansion approach, such as polynomial regression or regression splines,

$$df = \text{ number of basis functions.}$$

- For loess or smoothing splines, we can define df by an analogy to linear models. For linear models, we have $\hat{\mathbf{y}}_{n\times 1} = \mathbf{P}\mathbf{y}$, and df is equal to the trace of the projection matrix $\mathbf{P}$.

  Similarly, for nonparametric fitting, we have

$$\hat{\mathbf{y}}_{n\times 1} = \mathbf{S}_{n\times n}\mathbf{y}_{n\times 1}, \quad \text{then define } df = \text{tr}(\mathbf{S}).$$

The Ozone Data

We use data from a study of the relationship between atmospheric ozone concentration, O_3 and other meteorological variables in the Los Angeles Basin in 1976. To simplify matters, we will reduce the predictors to:

- temperature measured at E1 Monte, `temp`;
- inversion base height at LAX, `ibh`;
- inversion top temperature at LAX, `ibt`.

A number of cases with missing variables have been removed for simplicity. The data were first presented by Breiman and Friedman (1985).

R package: gam

```
gam(O3 ~ lo(temp) + lo(ibh) + lo(ibt))
gam(O3 ~ lo(temp, ibh) + ibt)
gam(O3 ~ lo(temp, 2) + lo(ibh) + s(ibt))
```

Choice of the smoother: loess or smoothing splines (with tuning parameters set by the user or by R using the default values). For more details on "lo" and "s", check the reference manual on R website.

Let's look at the R output for the ozone data.

R package: `mgcv`

```
gam(O3 ~ s(temp) + s(ibh) + s(ibt))
```

```
gam(O3 ~ s(temp) + s(ibh) + ibt)
```

```
gam(O3 ~ s(temp, ibh) + ibt)
```

Choice of the smoother: smoothing splines (with tuning parameters automatically chased by `R` via CV).

Let's look at the `R` output for the ozone data.

mgcv vs. gam

- Fittings are not exactly the same, but very similar.
- In gam, users can use either loess or smoothing.splines, and can control the tuning parameters.
- In mgcv, the fitting uses smoothing.splines, and the tuning parameters are automatically selected by CV.
- In both packages, users can extract fitted values and predictions, carry out F-tests to compare different models, and plot fitted curves.

Generalized Additive Models

- It's easy to extend the ordinary GLM to form generalized additive models, i.e.,

$$g(\mathbb{E}Y) = \beta_0 + \sum_{j=1}^{p} f_j(X_j).$$

  For example, for binary data, we can model

$$\log \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} = \beta_0 + f_1(x_1) + \cdots + f_p(x_p).$$

- Recall that the GLM is fitted via an iterative reweighed LS algorithm. So the back fitting algorithm (for LS shown on previous slide) can be easily extended to fit generalized additive model.