Statistics 8330: Data Analysis III
Beyond Linearity: Kernel and Local Polynomial Regression

Suggested Reading: James, Witten, Hastie, and Tibshirani (JWHT), 2013, Chapter 7

Supplemental Reading: Hastie, Tibshirani, and Friedman (HTF), 2009, Chapter 6

Christopher K. Wikle

University of Missouri
Department of Statistics

Kernel Smoothing Methods

CKW

The basic idea behind *kernel smoothing regression* is that we fit a separate, but simple, model at each selected point x_0 by using only those observations that are "close" to x_0 such that $\hat{f}(X)$ is *smooth* in $\mathbb{R}^p$.

This "localization" is accomplished by a *weighting function* or **kernel** $K_\lambda(x_0, x_i)$ that assigns a weight to x_i based on its distance from x_0. This kernel depends on a parameter, λ, that defines the "width" or "neighborhood" of the kernel.

These methods require less training than traditional regression models (typically, only λ need be chosen) but are *memory based methods* in that one needs *all* of the training data any time we want to make a prediction at a location x_0.

One-Dimensional Kernel Smoothers

Recall from early in the class the k-nearest neighbor approach to fitting a function

$$\hat{f}(x) = \text{Ave}(y_i | x_i \in N_k(x)),$$

where $N_k(x)$ refers to the set of k neighbors nearest the point x in squared distance.

Alternatively, we can consider a so-called "Nadaraya-Watson" kernel-weighted average for our estimate:

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

where the kernel is defined

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

The important point is that our fitted function is now continuous, unlike the nearest neighbor fit.

Kernel Smoothers (cont.)

CKW

Examples of the kernel function include the *Epanechnikov quadratic kernel*,

$$D(t) = \begin{cases} (3/4)(1-t^2) & \text{if } |t| \leq \lambda \\ 0 & \text{otherwise} \end{cases}$$

Or the *Tri-Cube kernel*,

$$D(t) = \begin{cases} (1-|t|^3)^3 & \text{if } |t| \leq \lambda \\ 0 & \text{otherwise} \end{cases}$$

There are others, e.g., the Gaussian, etc.

We can also use adaptive neighborhoods where the λ parameter depends on the location x_0, e.g.,

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

Kernel Smoothers (cont.)

CKW

Example Kernel Functions (HTF 2009):

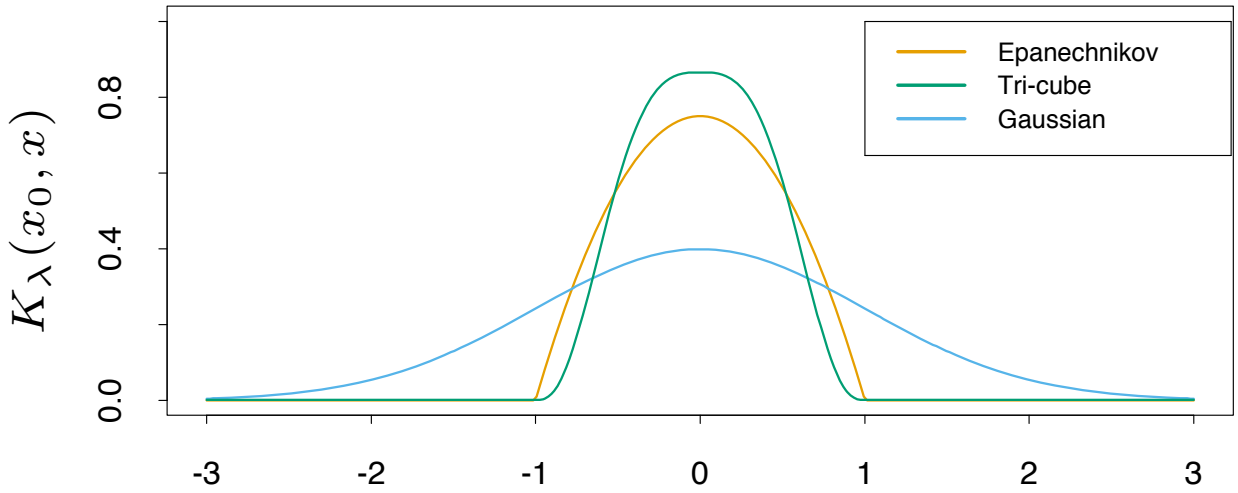

FIGURE 6.2. *A comparison of three popular kernels for local smoothing. Each has been calibrated to integrate to* 1. *The tri-cube kernel is compact and has two continuous derivatives at the boundary of its support, while the Epanechnikov kernel has none. The Gaussian kernel is continuously differentiable,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

Kernel Smoothers (cont.)

CKW

Kernel Smoothing Example (HTF 2009):

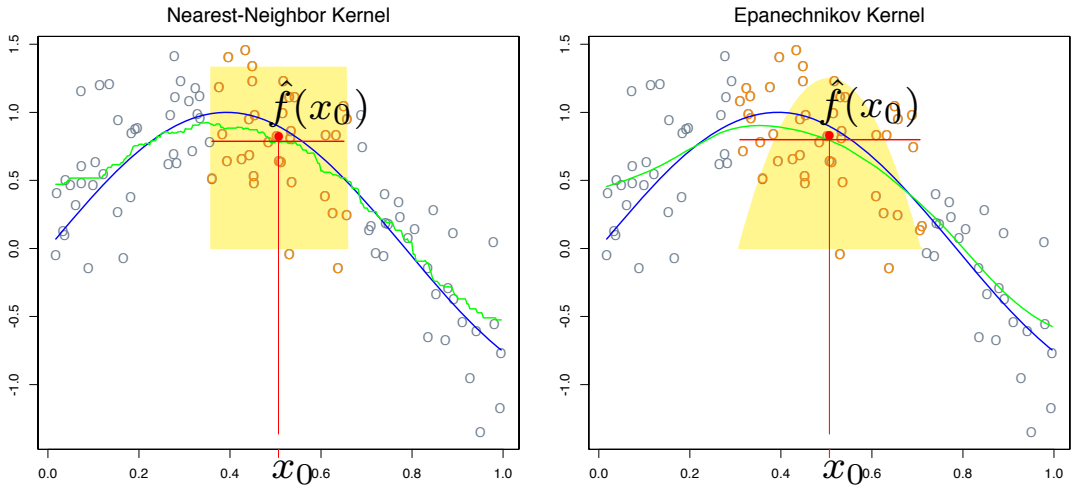

FIGURE 6.1. *In each panel* 100 *pairs* x_i, y_i *are generated at random from the blue curve with Gaussian errors:* $Y = \sin(4X) + \varepsilon$, $X \sim U[0,1]$, $\varepsilon \sim N(0, 1/3)$. *In the left panel the green curve is the result of a* 30*-nearest-neighbor running-mean smoother. The red point is the fitted constant* $\hat{f}(x_0)$, *and the red circles indicate those observations contributing to the fit at* x_0. *The solid yellow region indicates the weights assigned to observations. In the right panel, the green curve is the kernel-weighted average, using an Epanechnikov kernel with (half) window width* $\lambda = 0.2$.

Kernel Smoothers (cont.)

CKW

We note a couple of important points about kernel smoothers:

- Large λ implies lower variance (since we average over more observations) but higher bias (because we are approximating $f(x_0)$ with observations further away from x_0)
- An adaptive parameter $h_\lambda(x)$ tends to keep the bias constant but the variance is inversely proportional to the local density of observations;
- One of the biggest problems with kernel smoothers is their behavior near the boundaries (primarily due to the asymmetry of the kernels near the boundary); thus, the fits are often badly biased near the boundaries. We see below that local linear regression can remedy this.

Kernel Smoothers (cont.)

CKW

Boundary Bias Example (HTF 2009):

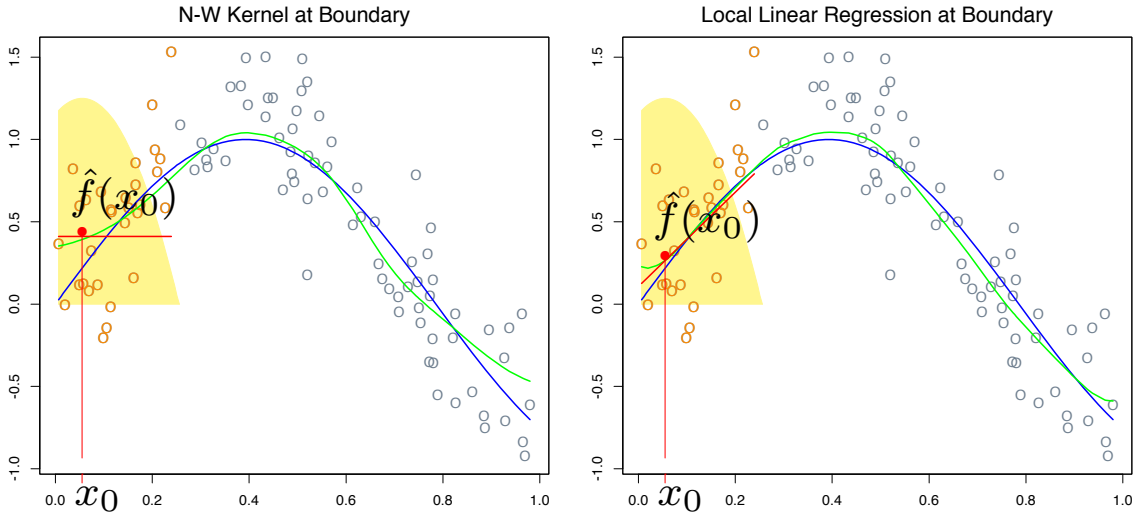

FIGURE 6.3. *The locally weighted average has bias problems at or near the boundaries of the domain. The true function is approximately linear here, but most of the observations in the neighborhood have a higher mean than the target point, so despite weighting, their mean will be biased upwards. By fitting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

Local Linear Regression

We can improve the bias issue at the boundaries by fitting straight lines instead of constants. That is, we can consider **locally weighted linear regression** which solves

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

and gives the estimate $\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$.

How do we fit this?

We let $b(x)^T = (1, x)$ and then define the $n \times 2$ matrix $\mathbf{B}$ to have ith row given by $b(x_i)^T$. Then, let $\mathbf{W}(x_0)$ be the $n \times n$ diagonal matrix with the ith diagonal given by $K_\lambda(x_0, x_i)$. Then,

$$\hat{f}(x_0) = b(x_0)^T(\mathbf{B}^T\mathbf{W}(x_0)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(x_0)\mathbf{y} = \sum_{i=1}^{n} \ell_i(x_0)y_i.$$

Local Linear Regression (cont.)

CKW

Thus, the fitted solution is again linear in the y_i with the weights in the linear combination given by $\ell_i(x_0)$, which are a combination of the kernel weights and least squares equations. In essence, this can also be thought of as another kernel – the **equivalent kernel**.

Critically, it can be shown that this corrects the boundary bias issues to first order (i.e., the bias is only a function of the second derivatives of $f(x_0)$ and so is fairly small on average). Thus, there is a small price that we pay in that this fit tends to be a little bit biased in regions of curvature in the true function.

Not surprisingly, we can improve on this bias by using a **locally quadratic regression**. In general, we can consider a **locally polynomial regression**

$$\min_{\alpha(x_0),\beta_j(x_0),j=1,\ldots,d} \sum_{i=1}^{n} K_\lambda(x_0,x_i)[y_i - \alpha(x_0) - \sum_{j=1}^{d} \beta_j(x_0)x_i^j]^2,$$

which gives the solution $\hat{f}(x_0) = \hat{\alpha}(x_0) + \sum_{j=1}^{d} \hat{\beta}_j(x_0)x_0^j$.

Local Linear Regression (cont.)

The bias reduction in the locally quadratic fit is offset by higher variance (of course!).

A couple of points are in order concerning local polynomial regression:

- In general, local linear fits help much better at the boundaries than higher order polynomials in reducing bias, without leading to excessive variance increase.
- Local quadratic fits are most helpful in reducing bias from curvature in the interior of the domain.
- Local polynomials of odd degree tend to be better than those of even degree because of better performance near the boundaries.

Local Linear Regression (cont.)

CKW

Equivalent Kernel Example (HTF 2009):

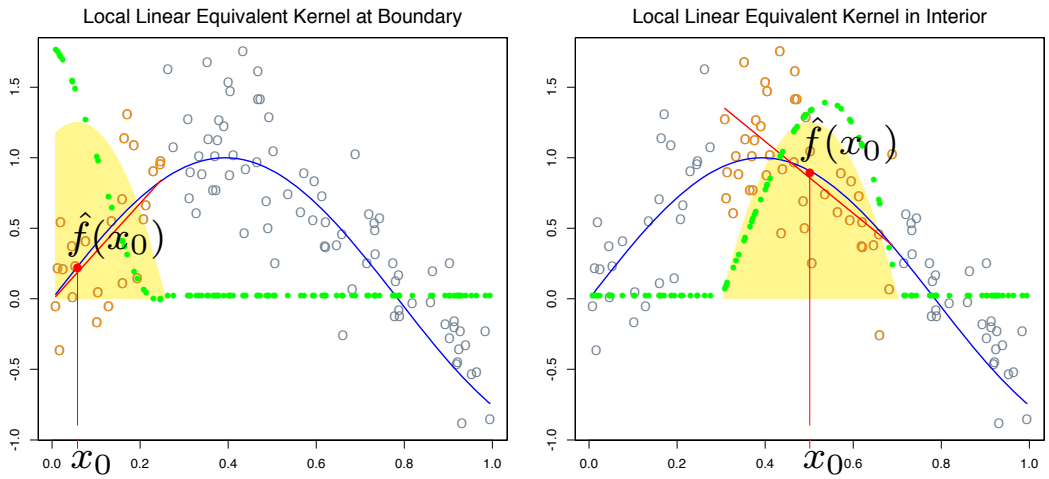

FIGURE 6.4. *The green points show the equivalent kernel $l_i(x_0)$ for local regression. These are the weights in $\hat{f}(x_0) = \sum_{i=1}^{N} l_i(x_0) y_i$, plotted against their corresponding x_i. For display purposes, these have been rescaled, since in fact they sum to 1. Since the yellow shaded 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.*

Local Regression (cont.)

CKW

Local Quadratic Regression Example (HTF 2009, Fig 6.5):

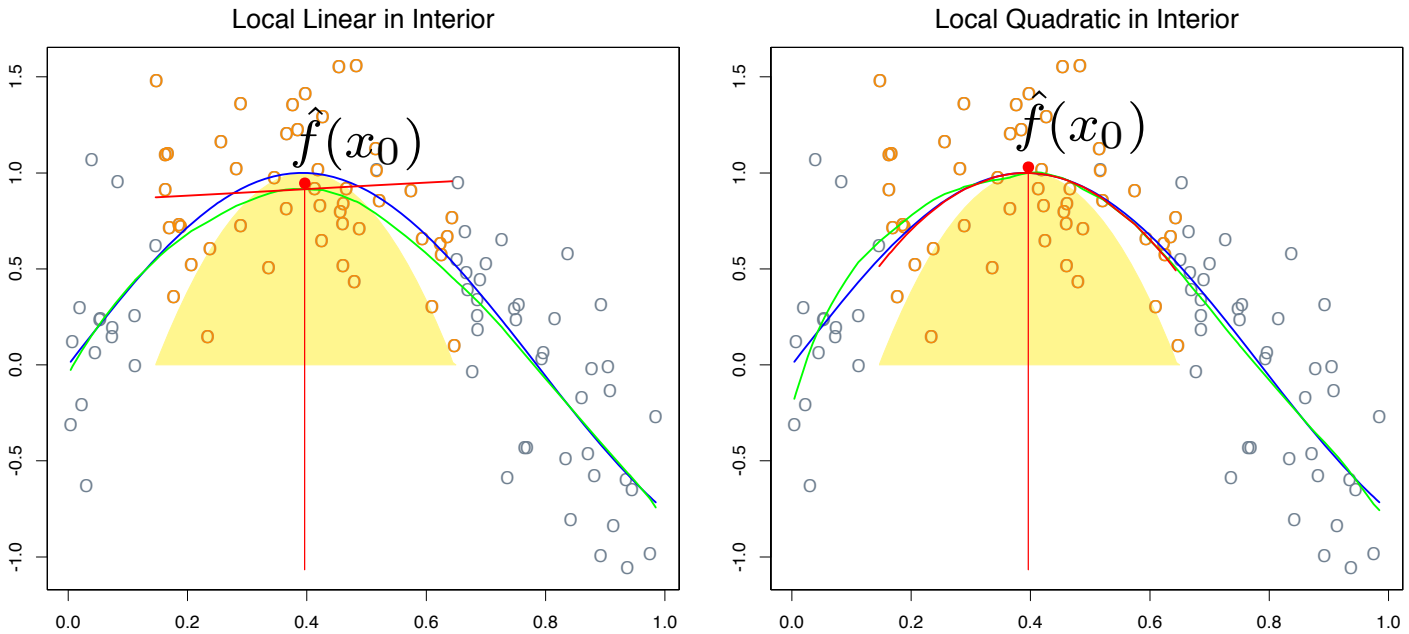

FIGURE 6.5. *Local linear fits exhibit bias in regions of curvature of the true function. Local quadratic fits tend to eliminate this bias.*

Selecting the Kernel Width

CKW

As we mentioned, λ controls the "width" of the kernel K_λ.

- **Epanechnikov/Tri-Cube:** λ is the radius of the support region
- **Gaussian:** λ is the standard deviation
- **Nearest Neighbor:** λ is the number k of nearest neighbors

smaller width: implies less averaging and thus higher variance and smaller bias

larger width: implies more averaging and thus less variance and larger bias

We can use cross validation to select this parameter. Also, given that we can write $\hat{\mathbf{f}} = \mathbf{S}_\lambda \mathbf{y}$, we can use the very useful (and efficient) LOOCV formula we have seen several times already.

Perhaps not surprisingly, given the smoothing matrix form given above, there is a close similarity between the equivalent kernels in smoothing spline fits and local linear regression fits.

Kernel Width (cont.)

Equivalent Kernels, Local Linear and Smoothing Spline Example (HTF 2009):

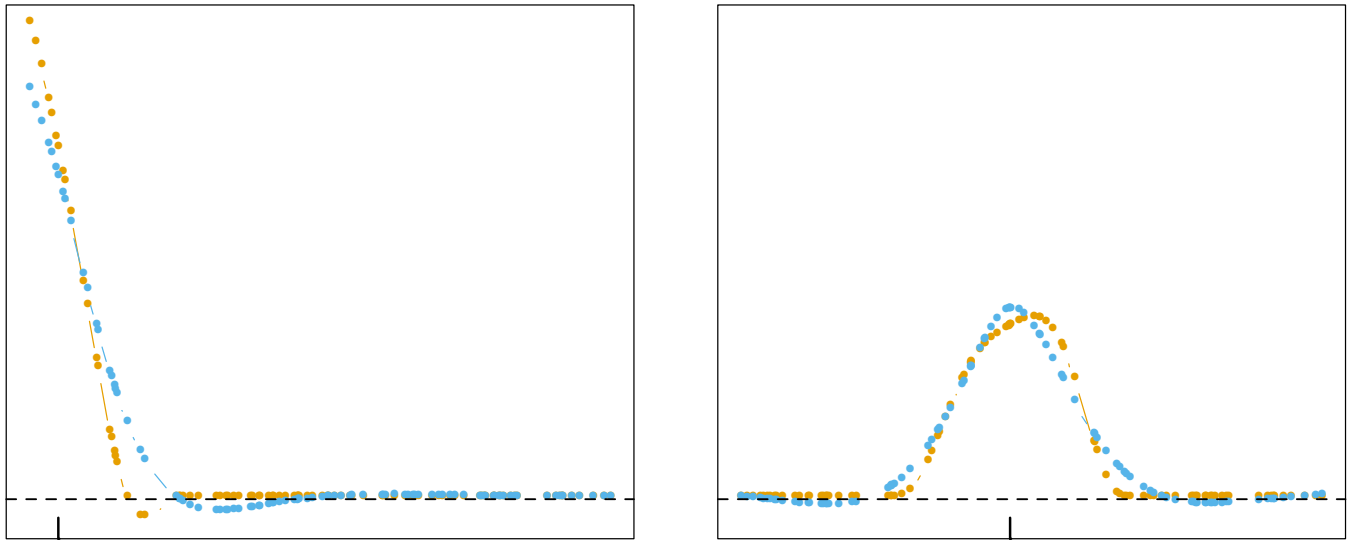

FIGURE 6.7. *Equivalent kernels for a local linear regression smoother (tri-cube kernel; orange) and a smoothing spline (blue), with matching degrees of freedom. The vertical spikes indicates the target points.*

Local Regression in Higher Dimensions

We can generalize the notions of kernel smoothing and local regression to higher dimensions. That is, we have a p-dimensional kernel and/or consider local hyperplanes.

For example, if $d = 1$ and $p = 2$ we have $b^T(X) = (1, X_1, X_2)$; for $d = 2$ and $p = 2$ we have $b^T(X) = (1, X_1, X_2, X_1^2, X_2^2, X_1X_2)$. So, we would solve

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

to fit $\hat{f}(x_0) = b(x_0)^T\hat{\beta}(x_0)$.

In this case, we would consider kernels of the general form

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

where $||\cdot||$ is the Euclidean norm. Given that the Euclidean norm is sensitive to the scale of the variables, we typically standardize the predictors in this setting.

Local Regression in Higher Dimensions

CKW

Two-Dimension Example (HTF 2009, Fig 6.8):

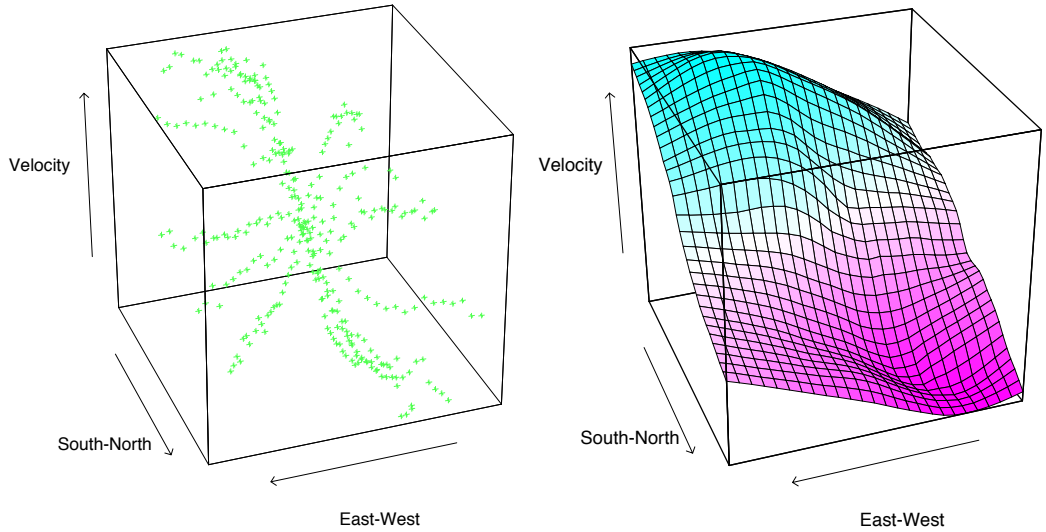

FIGURE 6.8. *The left panel shows three-dimensional data, where the response is the velocity measurements on a galaxy, and the two predictors record positions on the celestial sphere. The unusual "star"-shaped design indicates the way the measurements were made, and results in an extremely irregular boundary. The right panel shows the results of local linear regression smoothing in* $\mathbb{R}^2$*, using a nearest-neighbor window with* 15% *of the data.*

Local Regression in Higher Dimensions (cont.)

Note that boundary bias issues are more serious with higher dimensions as the curse of dimensionality implies that a larger fraction of the data locations are on or near the boundary.

As with the one-dimensional case, the local polynomial regression seamlessly accounts for this boundary bias issue to first order. However, it is always a problem to balance localness (low bias) and low variance as the dimension increases without the sample size increasing exponentially in p.

For this reason, one typically does not see smoothing or local polynomial regression used much in dimensions beyond two or three. (This is an important point!)

Structured Local Regression Models in Higher Dimensions

In the usual situation where the dimension to sample-size ratio is unfavorable, one can seek to improve fits by adding structural assumptions to the modeling framework. (Indeed, traditional multiple regression works because the model is very highly structured).

There are many ways to add structure to the kernel smoothing and local polynomial regression frameworks. For example, in the context of kernels, one can consider a form such as

$$K_{\lambda,\mathbf{A}}(x_0, x) = D\left(\frac{(x - x_0)^T \mathbf{A}(x - x_0)}{\lambda}\right),$$

where $\mathbf{A}$ is a $p \times p$ positive semidefinite matrix. This matrix can give different weights to the elements of $x - x_0$ and/or recombine the elements of this vector in various ways to shrink some dimensions and in some sense "compress" the points into a smaller region of space.

Structured Local Regression Models (cont.)

We could also consider structure through *varying coefficient models*. That is, assume we have p inputs and we seek a regression on the first q variables where the parameters then depend on the remaining inputs $Z = (X_{q+1}, X_{q+2}, \ldots, X_p)$. That is,

$$f(X) = \alpha(Z) + \beta_1(Z)X_1 + \ldots + \beta_q(Z)X_q,$$

where the coefficients are functions of Z. In the context of kernel weighted local regressions, we have

$$\min_{\alpha(z_0),\beta(z_0)} \sum_{i=1}^{n} K_\lambda(z_0, z_i)[y_i - \alpha(z_0) - x_{1i}\beta_1(z_0) - \ldots - x_{qi}\beta_q(z_0)]^2.$$

This can be a very effective way to let outside information inform the parameters (note the connection to the hierarchical mixed models we saw in DA II).

Kernel Density Estimation

CKW

Another use of kernels is concerned with density estimation, which precedes kernel regression in the history of statistics. The idea behind *kernel density estimation* is simple in principle. Assume we have a random sample $x_1, \ldots, x_n$ from some probability density $f_X(x)$ and our goal is to estimate f_X at a point x_0. [For now, assume we are in just one dimension, $X \in \mathbb{R}$].

The *Parzen* estimate is given by

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

where a popular choice of kernels is the Gaussian kernel with standard deviation λ. This easily extends to higher dimensional density estimation with the use of multivariate (Gaussian) kernels.

Kernel Density Estimation

Consider the kernel density estimation example from HTF (2009):

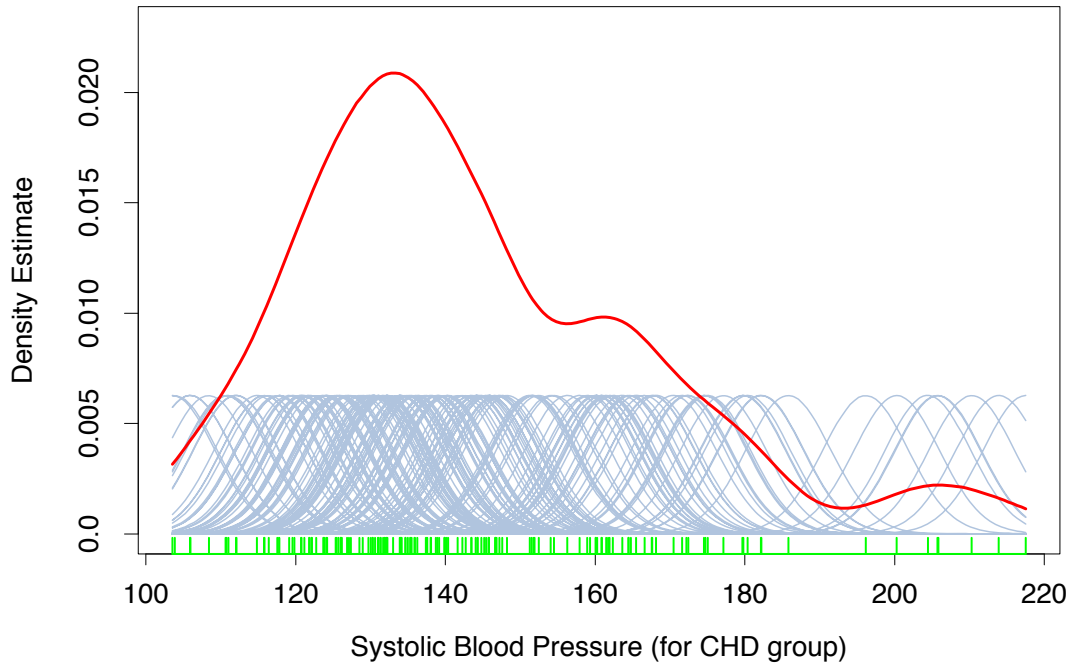

FIGURE 6.13. *A kernel density estimate for systolic blood pressure (for the CHD group). The density estimate at each point is the average contribution from each of the kernels at that point. We have scaled the kernels down by a factor of 10 to make the graph readable.*

Kernel Density Classification

CKW

Nonparametric density estimates can be useful for classification as well if we use Bayes' theorem.

Suppose we have J classes and fit the nonparametric densities $\hat{f}_j(X), j = 1, \ldots, J$ separately for each class. We also assume we have a prior probability for each class, say $\hat{\pi}_j$ (which might be from the sample proportions or some previous knowledge). Then

$$\widehat{Pr}(G = j | X = x_0) = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\sum_{k=1}^{J} \hat{\pi}_k \hat{f}_k(x_0)}.$$

See HTF (2009, Section 6.6) for more details.

Kernel Density Estimation

Consider the kernel density classification example from HTF (2009):

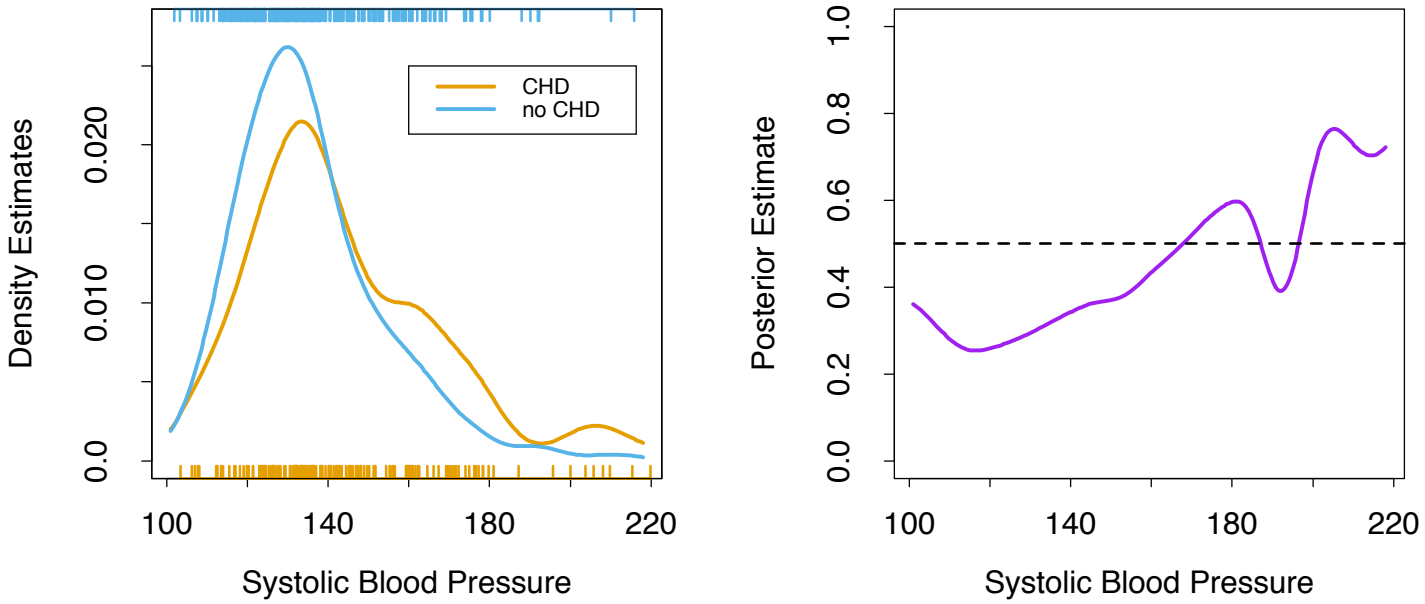

FIGURE 6.14. *The left panel shows the two separate density estimates for systolic blood pressure in the CHD versus no-CHD groups, using a Gaussian kernel density estimate in each. The right panel shows the estimated posterior probabilities for CHD, using (6.25).*