1.3 Suppose a graduation rule has a weight diagram with all positive weights $l_j \geq 0$ and that it reproduces constants (i.e. $\sum l_j = 1$). Also assume $l_j \neq 0$ for some $j \neq 0$. Show that graduation rule cannot be cubic reproducing. That is, there exists a cubic (or lower degree) polynomial that will not be reproduced by the graduation rule.

1.4 Compute the error reduction factors and coefficients of ∇^4 for Henderson's formula with $m = 5, \ldots, 10$. Make a scatterplot of the two components. Also compute and add the corresponding points for Spencer's 15- and 21-point rules, Woolhouse's rule and Higham's rule.

Remark. This exercise shows the bias-variance trade-off: As the length of the graduation rule increases, the variance decreases but the coefficient of $\nabla^4 y_j$ increases (in absolute value).

1.5 For each year in the age range 20 to 45, 1000 customers each wish to buy a \$10000 life insurance policy. Two competing companies set premiums as follows: First, estimate the mortality rate for each age, then set the premium to cover the expected payout, plus a 10% profit. For example, if the company estimates 40 year olds to have a mortality rate of 0.01, the expected (per customer) payout is $0.01 \times \$10000 = \$100$, so the premium is \$110. Both companies use Spencer's mortality data to estimate mortality rates. The Gauss Life Company uses a least squares fit to the data, while Spencer Underwriting applies Spencer's 15-point rule.

a) Compute for each age group the premiums charged by each company.

b) Suppose perfect customer behavior, so, for example, all the 40 year old customers choose the company offering the lowest premium to 40 year olds. Also suppose Spencer's 21-point rule provides the true mortality rates. Under these assumptions, compute the expected profit (or loss) for each of the two companies.

1.6 For large m, show the weights for Henderson's ideal formula are approximately $m^6 W(k/m)$ where $W(v) = (1 - x^2)_+^3$. Thus, conclude that the weight diagram is approximately $315/512 \times W(k/m)(3 - 11(k/m)^2)$. Compare with the $(0, 4, 3)$ kernel in Table 1 of Müller (1984).

2

Local Regression Methods

This chapter introduces the basic ideas of local regression and develops important methodology and theory. Section 2.1 introduces the local regression method. Sections 2.2 and 2.3 discuss, in a mostly nontechnical manner, statistical modeling issues. Section 2.2 introduces the bias-variance trade-off and the effect of changing smoothing parameters. Section 2.3 discusses diagnostic techniques, such as residual plots and confidence intervals. Section 2.4 introduces more formal criteria for model comparison and selection, such as cross validation.

The final two sections are more technical. Section 2.5 introduces the theory of linear estimation. This provides characterizations of the local regression estimate and studies some properties of the bias and variance. Section 2.6 introduces asymptotic theory for local regression.

2.1 The Local Regression Estimate

Local regression is used to model a relation between a predictor variable (or variables) x and response variable Y, which is related to the predictor variables. Suppose a dataset consists of n pairs of observations, $(x_1, Y_1), (x_2, Y_2), \ldots, (x_n, Y_n)$. We assume a model of the form

$$Y_i = \mu(x_i) + \epsilon_i \tag{2.1}$$

where $\mu(x)$ is an unknown function and ϵ_i is an error term, representing random errors in the observations or variability from sources not included in the x_i.

The errors ϵ_i are assumed to be independent and identically distributed with mean 0; $E(\epsilon_i) = 0$, and have finite variance; $E(\epsilon_i^2) = \sigma^2 < \infty$. Globally, no strong assumptions are made about μ. Locally around a point x, we assume that μ can be well approximated by a member of a simple class of parametric functions. For example, Taylor's theorem says that any differentiable function can be approximated locally by a straight line, and a twice differentiable function can be approximated by a quadratic polynomial.

For a fitting point x, define a bandwidth $h(x)$ and a smoothing window $(x-h(x), x+h(x))$. To estimate $\mu(x)$, only observations within this window are used. The observations weighted according to a formula

$$w_i(x) = W\left(\frac{x_i - x}{h(x)}\right) \tag{2.2}$$

where $W(u)$ is a weight function that assigns largest weights to observations close to x. For many of our examples, we use the tricube weight function

$$W(u) = (1 - |u|^3)^3. \tag{2.3}$$

Within the smoothing window, $\mu(u)$ is approximated by a polynomial. For example, a local quadratic approximation is

$$\mu(u) \approx a_0 + a_1(u - x) + \frac{1}{2}a_2(u - x)^2 \tag{2.4}$$

whenever $|u - x| < h(x)$. A compact vector notation for polynomials is

$$a_0 + a_1(u - x) + \frac{1}{2}a_2(u - x)^2 = \langle a, A(u - x)\rangle$$

where a is a vector of the coefficients and $A(\cdot)$ is a vector of the fitting functions. For local quadratic fitting,

$$a = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} \qquad A(v) = \begin{pmatrix} 1 \\ v \\ \frac{v^2}{2} \end{pmatrix}.$$

The coefficient vector a can be estimated by minimizing the locally weighted sum of squares:

$$\sum_{i=1}^{n} w_i(x)(Y_i - \langle a, A(x_i - x)\rangle)^2. \tag{2.5}$$

The local regression estimate of $\mu(x)$ is the first component of $\hat{a}$.

Definition 2.1 The **local regression estimate** is

$$\hat{\mu}(x) = \langle \hat{a}, A(0)\rangle = \hat{a}_0, \tag{2.6}$$

obtained by setting $u = x$ in (2.4).

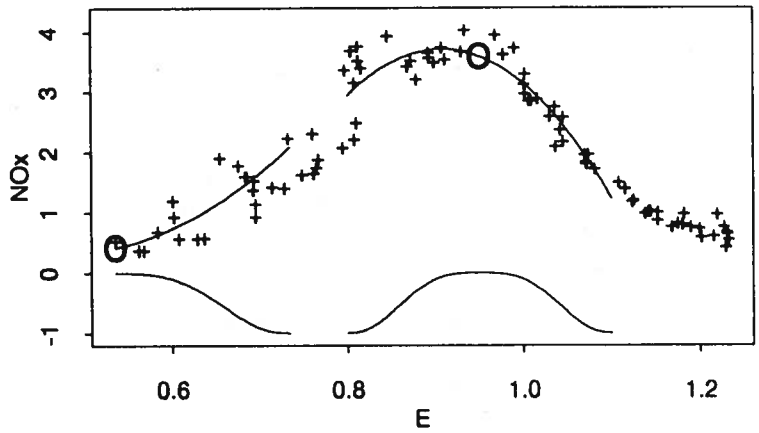

FIGURE 2.1. Local regression: Smoothing windows (bottom); local least squares fits (solid curves) and estimates $\hat{\mu}(x)$ (big circles).

The local regression procedure is illustrated in Figure 2.1. The ethanol dataset, measuring exhaust emissions of a single cylinder engine, is originally from Brinkman (1981) and has been studied extensively by Cleveland (1993) and others. The response variable, NOx, measures the concentration of certain pollutants in the emissions, and the predictor variable, E, is the equivalence ratio, measuring the richness of the air and fuel mix in the engine. Figure 2.1 illustrates the fitting procedure at the points E = 0.535 and E = 0.95. The observations are weighted according to the two weight functions shown at the bottom of Figure 2.1. The local quadratic polynomials are then fitted within the smoothing windows. From each quadratic, only the central point, indicated by the large circles in Figure 2.1, is retained. As the smoothing window slides along the data, the fitted curve is generated. Figure 2.2 displays the resulting fit.

The preceding demonstration has used local quadratic polynomials. It is instructive to consider lower order fits.

Example 2.1. (Local Constant Regression) For local constant polynomials, there is just one local coefficient a_0, and the local residual sum of squares (2.5) is

$$\sum_{i=1}^{n} w_i(x)(Y_i - a_0)^2.$$

The minimizer is easily shown to be

$$\hat{\mu}(x) = \hat{a}_0 = \frac{\sum_{i=1}^{n} w_i(x) Y_i}{\sum_{i=1}^{n} w_i(x)}. \tag{2.7}$$

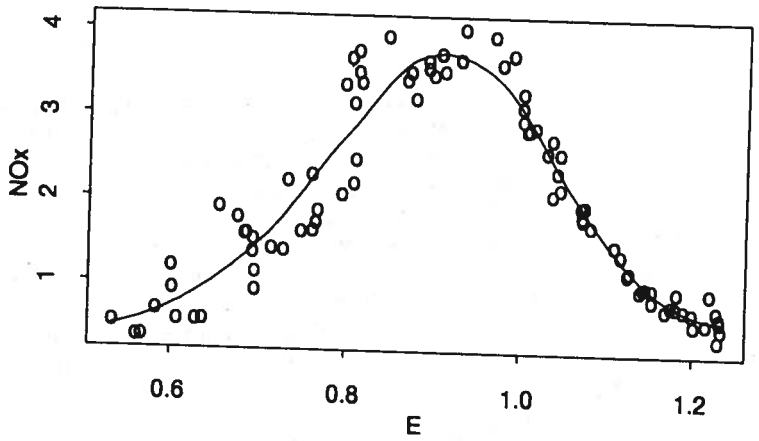

FIGURE 2.2. Local regression fit of the ethanol data.

This is the kernel estimate of Nadaraya (1964) and Watson (1964). It is simply a weighted average of observations in the smoothing window. A local constant approximation can often only be used with small smoothing windows, and noisy estimates result. The estimate is particularly susceptible to boundary bias. In Figure 2.1, if a local constant fit was used at $E = 0.535$, it would clearly lie well above the data.

Example 2.2. (Local Linear Regression) The local linear estimate, with $A(v) = (1 \quad v)^T$, has the closed form

$$\hat{\mu}(x) = \frac{\sum_{i=1}^n w_i(x) Y_i}{\sum_{i=1}^n w_i(x)} + (x - \bar{x}_w) \frac{\sum_{i=1}^n w_i(x)(x_i - \bar{x}_w) Y_i}{\sum_{i=1}^n w_i(x)(x_i - \bar{x}_w)^2} \tag{2.8}$$

where $\bar{x}_w = \sum_{i=1}^n w_i(x) x_i / \sum_{i=1}^n w_i(x)$. See exercise 2.1. That is, the local linear estimate is the local constant estimate, plus a correction for local slope of the data and skewness of the x_i. This correction reduces the boundary bias problem of local constant estimates. When the fitting point x is *not* near a boundary, one usually has $x \approx \bar{x}_w$, and there is little difference between local constant and local linear fitting. A local linear estimate exhibits bias if the mean function has high curvature.

2.1.1 *Interpreting the Local Regression Estimate*

In studies of linear regression, one often focuses on the regression coefficients. One assumes the model being fitted is correct and asks questions

such as how well the estimated coefficients estimate the true coefficients. For example, one might compute variances and confidence intervals for the regression coefficients, test significance of the coefficients or use model selection criteria, such as stepwise selection, to decide what coefficients to include in the model. The fitted curve itself often receives relatively little attention.

In local regression, we have to change our focus. Instead of concentrating on the coefficients, we focus on the fitted curve. A basic question that can be asked is "how well does $\hat{\mu}(x)$ estimate the true mean $\mu(x)$?". When variance estimates and confidence intervals are computed, they will be computed for the curve estimate $\hat{\mu}(x)$. Model selection criteria can still be used to select variables for the local model. But they also have a second use, addressing whether an estimate $\hat{\mu}(x)$ is satisfactory or whether alternative local regression estimates, for example, with different bandwidths, produce better results.

2.1.2 *Multivariate Local Regression*

Formally, extending the definition of local regression to multiple predictors is straightforward; we require a multivariate weight function and multivariate local polynomials. This was considered by McLain (1974) and Stone (1982). Statistical methodology and visualization for multivariate fitting was developed by Cleveland and Devlin (1988) and the associated LOESS method.

With two predictor variables, the local regression model becomes

$$Y_i = \mu(x_{i,1}, x_{i,2}) + \epsilon_i,$$

where $\mu(\cdot, \cdot)$ is unknown. Again, a suitably smooth function μ can be approximated in a neighborhood of a point $x = (x_{\cdot,1}, x_{\cdot,2})$ by a local polynomial; for example, a local quadratic approximation is

$$\begin{aligned} \mu(u_1, u_2) \approx\ & a_0 + a_1(u_1 - x_{\cdot,1}) + a_2(u_2 - x_{\cdot,2}) + \frac{a_3}{2}(u_1 - x_{\cdot,1})^2 \\ & + a_4(u_1 - x_{\cdot,1})(u_2 - x_{\cdot,2}) + \frac{a_5}{2}(u_2 - x_{\cdot,2})^2. \end{aligned}$$

This can again be written in the compact form

$$\mu(u_1, u_2) \approx \langle a, A(u - x) \rangle,$$

where $A(\cdot)$ is the vector of local polynomial basis functions:

$$A\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 \\ v_1 \\ v_2 \\ \frac{1}{2}v_1^2 \\ v_1 v_2 \\ \frac{1}{2}v_2^2 \end{pmatrix}. \tag{2.9}$$

Weights are defined on the multivariate space, so observations close to a fitting point x receive the largest weight. First, define the length of a vector v in $\mathcal{R}^d$ by

$$\|v\|^2 = \sum_{j=1}^{d} \left(\frac{v_j}{s_j}\right)^2, \tag{2.10}$$

where $s_j > 0$ is a scale parameter for the jth dimension. A spherically symmetric weight function gives an observation x_i the weight

$$w_i(x) = W\left(\frac{\|x_i - x\|}{h}\right). \tag{2.11}$$

As in the univariate case, the local coefficients are estimated by solving the weighted least squares problem (2.5). Following Definition 2.1, $\hat{\mu}(x)$ is the first component of $\hat{a}$.

2.2 The Components of Local Regression

Much work remains to be done to make local regression useful in practice. There are several components of the local fit that must be specified: the bandwidth, the degree of local polynomial, the weight function and the fitting criterion.

2.2.1 Bandwidth

The bandwidth $h(x)$ has a critical effect on the local regression fit. If $h(x)$ is too small, insufficient data fall within the smoothing window, and a noisy fit, or large variance, will result. On the other hand, if $h(x)$ is too large, the local polynomial may not fit the data well within the smoothing window, and important features of the mean function $\mu(x)$ may be distorted or lost completely. That is, the fit will have large bias. The bandwidth must be chosen to compromise this bias-variance trade-off.

Ideally, one might like to choose a separate bandwidth for each fitting point, taking into account features such as the local density of data and the amount of structure in the mean function. In practice, doing this in a sensible manner is difficult. Usually, one restricts attention to bandwidth functions with a small number of parameters to be selected.

The simplest specification is a **constant bandwidth,** $h(x) = h$ for all x. This is satisfactory in some simple examples, but when the independent variables x_i have a nonuniform distribution, this can obviously lead to problems with empty neighborhoods. This is particularly severe in boundary or tail regions or in more than one dimension.

Data sparsity problems can be reduced by ensuring neighborhoods contain sufficient data. A **nearest neighbor bandwidth** chooses $h(x)$ so that

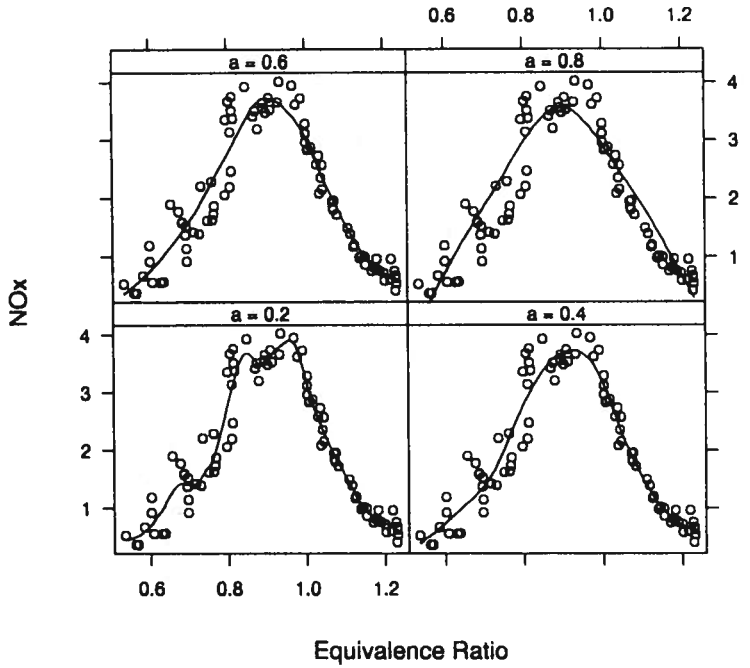

FIGURE 2.3. Local fitting at different bandwidths. Four different nearest neighbor fractions: $\alpha = 0.8, 0.6, 0.4$ and 0.2 are used.

the local neighborhood always contains a specified number of points. For a smoothing parameter α between 0 and 1, the nearest neighbor bandwidth $h(x)$ is computed as follows:

1. Compute the distances $d(x, x_i) = |x - x_i|$ between the fitting point x and the data points x_i.
2. Choose $h(x)$ to be the kth smallest distance, where $k = \lfloor n\alpha \rfloor$.

Example 2.3. Figure 2.3 shows local quadratic fits for the ethanol dataset using four different values of α. Clearly, the fit produced by the smallest fraction, $\alpha = 0.2$, produces a much noisier fit than the largest bandwidth, $\alpha = 0.8$. In fact, $\alpha = 0.8$ has oversmoothed, since it doesn't track the data well. For $1.0 < E < 1.2$, there is a sequence of 17 successive data points lying below the fitted curve. The leveling off at the right boundary is not captured. The peak for $0.9 < E < 1.0$ appears to be trimmed.

The fit with $\alpha = 0.2$ shows features - bimodality of the peak and a leveling off around $E = 0.7$ that don't show up at larger bandwidths. Are

these additional features real, or are they artifacts of random noise in the data? Our *apriori guess* might be that these are random noise; we hope that nature isn't too nasty. But *proving* this from the data is impossible. There are small clumps of observations that support both of the additional features in the plot with $\alpha = 0.2$, but probably not enough to declare statistical significance.

This example is discussed in more detail later. For now, we note the one-sided nature of bandwidth selection. While large smoothing parameters may easily be rejected as oversmoothed, it is much more difficult to conclude *from the data alone* that a small bandwidth is undersmoothed.

2.2.2 *Local Polynomial Degree*

Like the bandwidth, the degree of the local polynomial used in (2.5) affects the bias-variance trade-off. A high polynomial degree can always provide a better approximation to the underlying mean $\mu(u)$ than a low polynomial degree. Thus, fitting a high degree polynomial will usually lead to an estimate $\hat{\mu}(x)$ with less bias. But high order polynomials have large numbers of coefficients to estimate, and the result is variability in the estimate. To some extent, the effects of the polynomial degree and bandwidth are confounded. For example, if a local quadratic estimate and local linear estimate are computed using the same bandwidth, the local quadratic estimate will be more variable. But the variance increase can be compensated by increasing the bandwidth.

It often suffices to choose a low degree polynomial and concentrate on choosing the bandwidth to obtain a satisfactory fit. The most common choices are local linear and local quadratic. As noted in Example 2.1, a local constant fit is susceptible to bias and is rarely adequate. A local linear estimate usually produces better fits, especially at boundaries. A local quadratic estimate reduces bias further, but increased variance can be a problem, especially at boundaries. Fitting local cubic and higher orders rarely produces much benefit.

Example 2.4. Figure 2.4 displays local constant, local linear, local quadratic and local cubic fits for the ethanol dataset. Nearest neighbor bandwidths are used, with $\alpha = 0.25, 0.3, 0.49$ and 0.59 for the four degrees. These smoothing parameters are chosen so that each fit has about seven degrees of freedom; a concept defined in section 2.3.2. Roughly, two fits with the same degrees of freedom have the same variance $\text{var}(\hat{\mu}(x))$.

The local constant fit in Figure 2.4 is quite noisy, and also shows boundary bias: The fit doesn't track the data well at the left boundary. The local linear fit reduces both the boundary bias and the noise. A closer examination suggests the local constant and linear fit have trimmed the peak: For $0.8 < E < 1.0$, nearly all the data points are *above* the fitted curve. The

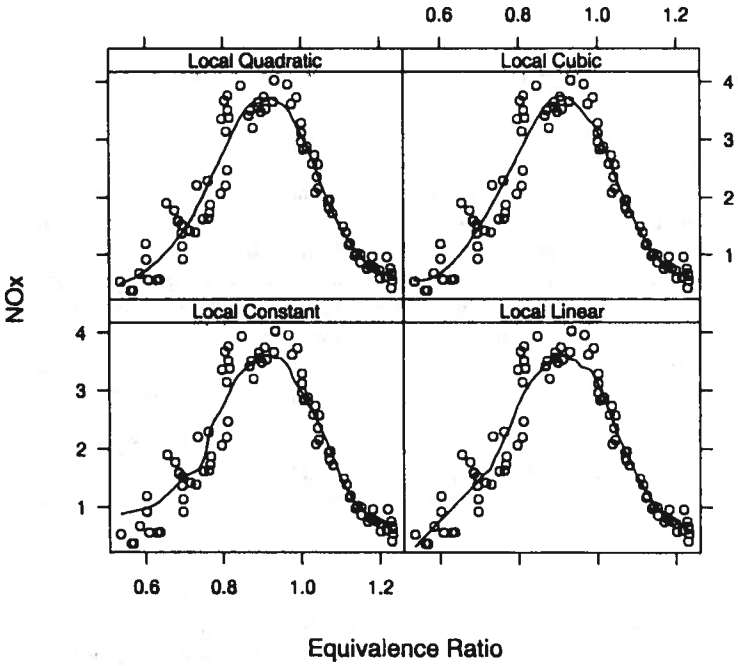

FIGURE 2.4. Ethanol data: Effect of changing the polynomial degree.

local quadratic and local cubic fits in Figure 2.4 produce better results: The fits show less noise and track the data better.

2.2.3 *The Weight Function*

The weight function $W(u)$ has much less effect on the bias-variance trade-off, but it influences the visual quality of the fitted regression curve. The simplest weight function is the rectangular:

$$W(u) = I_{[-1,1]}(u).$$

This weight function is rarely used, since it leads to discontinuous weights $w_i(x)$ and a discontinuous fitted curve. Usually, $W(u)$ is chosen to be continuous, symmetric, peaked at 0 and supported on $[-1, 1]$. A common choice is the tricube weight function (2.3).

Other types of weight function can also be useful. Friedman and Stuetzle (1982) use smoothing windows covering the same number of data points both before and after the fitting point. For nonuniform designs this is

asymmetric, but it can improve variance properties. McLain (1974) and Lancaster and Salkaus kas (1981) use weight functions with singularities at $u = 0$. This leads to a fitted smooth curve that interpolates the data. In Section 6.3, one-sided weight functions are used to model discontinuous curves.

2.2.4 *The Fitting Criterion*

The local regression estimate, as defined by (2.5) and (2.6), is a local least squares estimate. This is convenient, since the estimate is easy to compute and much of the methodology available for least squares methods can be extended fairly directly to local regression. But it also inherits the bad properties of least squares estimates, such as sensitivity to outliers.

Any other fitting criterion can be used in place of least squares. The local likelihood method uses likelihoods instead of least squares; this forms a major topic later in this book. Local robust regression methods are discussed in Section 6.4.

2.3 Diagnostics and Goodness of Fit

In local regression studies, one is faced with several model selection issues: Variable selection, choice of local polynomial degree and smoothing parameters. An ideal aim may be fully automated methods: We plug data into a program, and it automatically returns the best fit. But this goal is unattainable, since the best fit depends not only on the data, but on the questions of interest.

What statisticians (and statistical software) can provide is tools to help guide the choice of smoothing parameters. In this section we introduce some graphical aids to help the decision: residual plots, degrees of freedom and confidence intervals. Some more formal tools are introduced in Section 2.4. These tools are designed to help decide which features of a dataset are real and which are random. They cannot provide a definitive answer as to the best bandwidth for a (dataset,question) pair.

The ideas for local regression are similar to those used in parametric models. Other books on regression analysis cover these topics in greater detail than we do here; see, for example, chapter 3 of Draper and Smith (1981) or chapters 4, 5 and 6 of Myers (1990). Cleveland (1993) is a particularly good reference for graphical diagnostics.

It is important to remember that no one diagnostic technique will explain the whole story of a dataset. Rather, using a combination of diagnostic tools and looking at these in conjunction with both the fitted curves and the original data provide insight into the data. What features are real;

have these been adequately modeled; are underlying assumptions, such as homogeneity of variance, satisfied?

2.3.1 *Residuals*

The most important diagnostic component is the residuals. For local regression, the residuals are defined as the difference between observed and fitted values:

$$\hat{\epsilon}_i = Y_i - \hat{\mu}(x_i).$$

One can use the residuals to construct formal tests of goodness of fit or to modify the local regression estimate for nonhomogeneous variance. These topics will be explored more in Chapter 9. For practical purposes, most insight is often gained simply by plotting the residuals in various manners. Depending on the situation, plots that can be useful include:

1. Residuals vs. predictor variables, for detecting lack of fit, such as a trimmed peak.
2. Absolute residuals vs. the predictors, to detect dependence of residual variance on the predictor variables. One can also plot absolute residuals vs. fitted values, to detect dependence of the residual variance on the mean response.
3. Q-Q plots (Wilk and Gnanadesikan 1968), to detect departure from normality, such as skewness or heavy tails, in the residual distribution. If non-normality is found, fitting criteria other than least squares may produce better results. See Section 6.4.
4. Serial plots of $\hat{\epsilon}_i$ vs. $\hat{\epsilon}_{i-1}$, to detect correlation between residuals.
5. Sequential plot of residuals, in the order the data were collected. In an industrial experiment, this may detect a gradual shift in experimental conditions over time.

Often, it is helpful to smooth residual plots: This can both draw attention to any features shown in the plot, as well as avoiding any visual pitfalls. Exercise 2.6 provides some examples where the wrong plot, or a poorly constructed plot, can provide misleading information.

Example 2.5. Figure 2.5 displays smoothed residual plots for the four fits in Figure 2.3. The residual plots are much better at displaying bias, or oversmoothing, of the fit. For example, the bias problems when $\alpha = 0.8$ are much more clearly displayed from the residual plots in Figure 2.5 than from the fits in Figure 2.3. Of course, as the smoothing parameter α is reduced, the residuals generally get smaller, and show less structure.

The smooths of the residuals in Figure 2.5 are constructed with $\alpha_r = 0.2$ (this should be distinguished from the α used to smooth the original data).

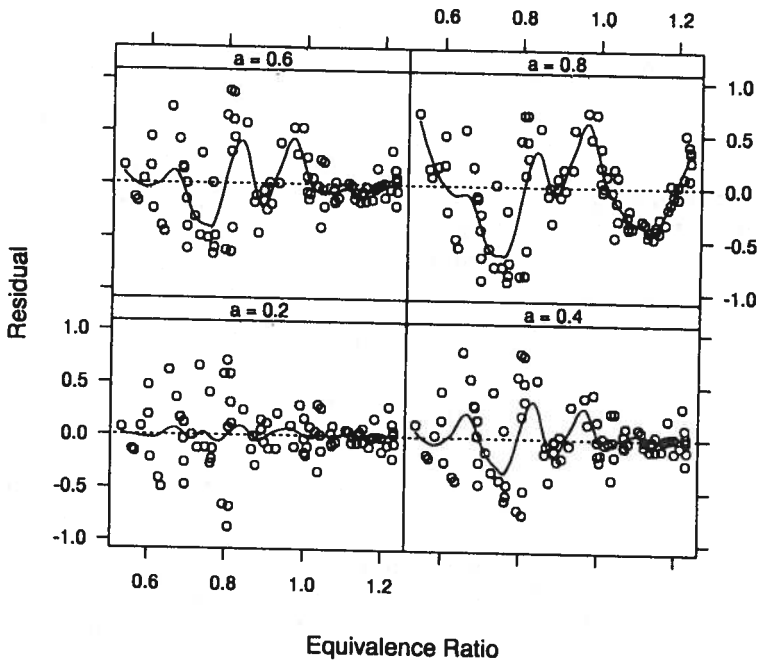

FIGURE 2.5. Residual plots for the ethanol dataset.

But α_r itself is not important. What is important is that the smooths help search for clusters of residuals that may indicate lack of fit. At $\alpha = 0.8$, the lack of fit is clear. At $\alpha = 0.6$ and $\alpha = 0.4$, the peaks in the smooth are generally supported by clumps of residuals, although generally not enough to indicate lack of fit.

This example shows that it is important not to look at the residual plots alone, but to use them in conjunction with plots of the fit. The object is to determine whether large residuals correspond to features in the data that have been inadequately modeled. The purpose of the plots can be related to the bias-variance trade-off:

- Plots of the fit help us detect noise in the fit.
- Residual plots help us detect bias.

It is important to note that the purpose of adding a smooth to a residual plot is not to provide a good estimate of the mean. Rather, it is to enhance our view of the residuals; by reducing the noise, our attention may be more

readily drawn to features that have been missed or not properly modeled by the smooth.

2.3.2 *Influence, Variance and Degrees of Freedom*

How can we characterize the amount of smoothing being performed? The bandwidth provides one characterization. But this is not ideal, since it takes no account of the other choices that go into the smooth, such as the degree of local polynomial and the weight function. Moreover, the bandwidth doesn't enable meaningful comparison with other smoothing methods or with parametric models.

What we need is unitless characterizations which allow comparison between methods. We discuss two types of characterization:

- Pointwise criteria, characterizing the amount of smoothing at a single point. These include the variance reducing factor and influence function.
- Global criteria, characterizing the overall amount of smoothing. This is the fitted degrees of freedom.

Before proceeding with definitions, the importance of the ideas presented here, both in theory and practice, must be emphasized. Throughout this book these concepts (and generalizations) will appear repeatedly. We already saw the variance reducing factor used in optimality results in Chapter 1; this will also appear in inference and confidence interval construction. The influence function and degrees of freedom will appear repeatedly in model selection criteria. The importance of these concepts has of course been emphasized elsewhere, both in local regression literature and elsewhere in the smoothing literature. See Craven and Wahba (1979), Cleveland and Devlin (1988), Buja, Hastie and Tibshirani (1989), Wahba (1990), Hastie and Tibshirani (1990) and Cleveland and Loader (1996). Ye (1998) contains a nice discussion of the importance, motivation, generalizations and applications of degrees of freedom.

Because the local regression estimate solves a least squares problem, $\hat{\mu}(x)$ is a **linear estimate**. That is, for each x there exists a **weight diagram** vector $l(x) = \{l_i(x)\}_{i=1}^n$ such that

$$\hat{\mu}(x) = \sum_{i=1}^{n} l_i(x) Y_i. \tag{2.12}$$

For local constant regression, (2.7) gives the explicit formula

$$l_i(x) = \frac{w_i(x)}{\sum_{j=1}^{n} w_j(x)}.$$

For more general local regression, the weight diagram is derived in Section 2.5. The weight diagram leads to compact forms for the mean and variance of the local regression estimate:

$$\begin{aligned} E(\hat{\mu}(x)) &= \sum_{i=1}^{n} l_i(x)\mu(x_i) = \langle l(x), \mu \rangle \\ \text{var}(\hat{\mu}(x)) &= \sigma^2 \sum_{i=1}^{n} l_i(x)^2 = \sigma^2 \|l(x)\|^2. \end{aligned} \tag{2.13}$$

The variance assumes the observations Y_i are independent and have constant variance σ^2. The **variance reducing factor** $\|l(x)\|^2$ measures the reduction in variance due to the local regression. Usually, this decreases as the bandwidth increases. Under mild conditions, one can show (see Theorem 2.3):

$$\frac{1}{n} \le \|l(x_i)\|^2 \le l_i(x_i) \le 1. \tag{2.14}$$

The extreme cases $1/n$ and 1 correspond, respectively, to $\hat{\mu}(x)$ being the sample average and interpolating the data.

The **hat matrix** is the $n \times n$ matrix $\mathbf{L}$ with rows $l(x_i)^T$, which maps the data to the fitted values:

$$\begin{pmatrix} \hat{\mu}(x_1) \\ \vdots \\ \hat{\mu}(x_n) \end{pmatrix} = \mathbf{L}Y. \tag{2.15}$$

The **influence** or **leverage** values are the diagonal elements $l_i(x_i)$ of the hat matrix. We denote these by $\text{infl}(x_i)$; these measure the sensitivity of the fitted curve $\hat{\mu}(x_i)$ to the individual data points.

The **degrees of freedom** of a local fit provide a generalization of the number of parameters of a parametric model. In fact, there are several possible definitions, but two of the most useful are

$$\begin{aligned} \nu_1 &= \sum_{i=1}^{n} \text{infl}(x_i) = \text{tr}(\mathbf{L}) \\ \nu_2 &= \sum_{i=1}^{n} \|l(x_i)\|^2 = \text{tr}(\mathbf{L}^T\mathbf{L}). \end{aligned} \tag{2.16}$$

The usefulness of the degrees of freedom is in providing a measure of the amount of smoothing that is comparable between different estimates applied to the same dataset. The concept was used in Example 2.4 to compare local polynomial fits of differing degrees, where we required smoothing parameters to perform the same amount of smoothing in each of the four cases.

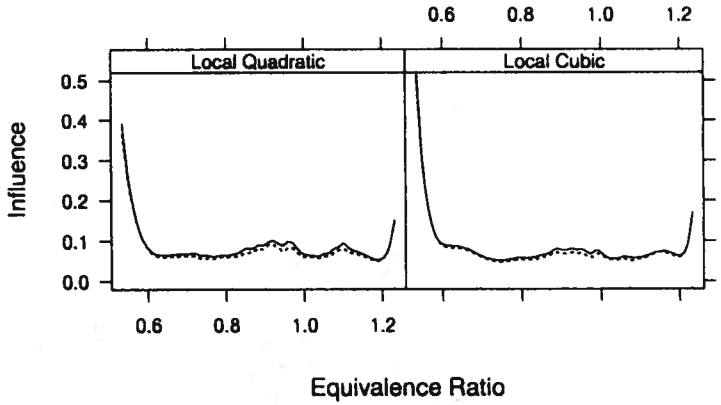

FIGURE 2.6. Influence functions (solid) and variance functions (dashed) for local quadratic and local cubic fits to the ethanol dataset.

For a parametric regression model, the hat matrix $\mathbf{L}$ is symmetric and idempotent, and the definitions coincide and usually equal the number of parameters. For local regression models, the two definitions are usually not equal; following (2.14), $1 \le \nu_2 \le \nu_1 \le n$. Both of these definitions arise naturally later.

Example 2.6. Figure 2.6 shows the influence and variance functions for the local quadratic and local cubic fits from Figure 2.4. Largely, the influence values are slightly less than 0.1, indicating that Y_i constitutes about 10% of the fitted value $\hat{\mu}(x_i)$. The variance function is slightly less than the influence. The degrees of freedom are $\nu_1 = 7.16$ and $\nu_2 = 6.60$ for the local quadratic fit, and $\nu_1 = 6.97$ and $\nu_2 = 6.53$ for the local cubic.

But the main feature is the boundary effect, particularly at the left, where the influence function shows a huge increase. This reflects the difficulty of fitting a polynomial at boundary regions. Note also that the effect is more pronounced for the local cubic fit: This shows that boundaries are a main concern when choosing the degree of the local fit.

2.3.3 *Confidence Intervals*

If $\hat{\mu}(x)$ is an unbiased estimate of $\mu(x)$, an approximate confidence interval for the true mean is

$$I(x) = (\hat{\mu}(x) - c\hat{\sigma}\|l(x)\|, \hat{\mu}(x) + c\hat{\sigma}\|l(x)\|),$$

where c is the appropriate quantile of the standard normal distribution ($c = 1.96$ for 95% confidence) and $\hat{\sigma}$ is an estimate of the residual standard deviation.

Prediction intervals provide interval estimates for a new observation Y_{new} at a point x_{new}. Assuming the new observation is independent of the estimation data, one has

$$\text{var}(Y_{\text{new}} - \hat{\mu}(x_{\text{new}})) = \sigma^2(1 + \|l(x_{\text{new}})\|^2).$$

Thus, a prediction interval has limits

$$\hat{\mu}(x_{\text{new}}) \pm c\hat{\sigma}(1 + \|l(x)\|^2)^{1/2}. \tag{2.17}$$

Note that prediction intervals assume normality: If Y_{new} is not normally distributed, the prediction interval will not be correct, even asymptotically.

In analogy with parametric regression, the variance σ^2 can be estimated using the normalized residual sum of squares:

$$\hat{\sigma}^2 = \frac{1}{n - 2\nu_1 + \nu_2} \sum_{i=1}^{n} (Y_i - \hat{\mu}(x_i))^2, \tag{2.18}$$

where ν_1 and ν_2 are defined by (2.16). The residual degrees of freedom, $n - 2\nu_1 + \nu_2$, are defined so that $\hat{\sigma}^2$ is unbiased. See Section 2.5.1.

The assumption that $\hat{\mu}(x)$ is unbiased is rarely exactly true, so variance estimates and confidence intervals are usually computed at small bandwidths where bias is small. Confidence intervals, bias correction and variance estimation are discussed in more detail in Chapter 9.

2.4 Model Comparison and Selection

2.4.1 *Prediction and Cross Validation*

How good is a local regression estimate? To formalize this question, we need to define criteria with which to assess the performance of the fit. One possible criterion is the prediction mean squared error for future observations:

$$\text{PMSE}(\hat{\mu}) = \text{E}(Y_{\text{new}} - \hat{\mu}(x_{\text{new}}))^2. \tag{2.19}$$

Clearly, PMSE($\hat{\mu}$) depends on assumptions made about x_{new}. For now, assume that the design points $x_1, \ldots, x_n$ are an independent sample from a density $f(x)$, and the new point x_{new} is sampled from the same density. The cross validation method provides an estimate of PMSE.

Definition 2.2 The **cross validation** estimate of the PMSE of an estimate $\hat{\mu}$ is

$$\text{CV}(\hat{\mu}) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{\mu}_{-i}(x_i))^2 \tag{2.20}$$

where $\hat{\mu}_{-i}(x_i)$ denotes the leave-x_i-out estimate of $\mu(x_i)$. That is, each x_i is deleted from the dataset in turn, and the local regression estimate computed from the remaining $n - 1$ data points.

The leave-one-out cross validation criteria was introduced for parametric regression models by Allen (1974) as the PRESS (prediction error sum of squares) procedure. Wahba and Wold (1975) applied the method to smoothing splines. Model validation based on splitting datasets into *estimation data* and *prediction data* has a long history, discussed by Stone (1974) and Snee (1977) among others.

The generalized cross validation criterion was first proposed in the context of smoothing splines by Craven and Wahba (1979). This provides an approximation to cross validation and is easier to compute. The motivation for the definition will appear in Section 2.5.

Definition 2.3 The **generalized cross validation score** for a local estimate $\hat{\mu}$ is

$$\text{GCV}(\hat{\mu}) = n \frac{\sum_{i=1}^{n} (Y_i - \hat{\mu}(x_i))^2}{(n - \nu_1)^2}, \tag{2.21}$$

where ν_1 is the fitted degrees of freedom defined by (2.16).

2.4.2 *Estimation Error and CP*

The cross validation methods are motivated by prediction error: How well does $\hat{\mu}(x)$ predict new observations? Alternatively, one can consider estimation error: How well does $\hat{\mu}(x)$ estimate the true mean $\mu(x)$? One possible loss criterion is the sum of the squared error over the design points;

$$L(\hat{\mu}, \mu) = \sum_{i=1}^{n} (\hat{\mu}(x_i) - \mu(x_i))^2. \tag{2.22}$$

The CP criterion, introduced by Mallows (1973) for parametric regression, provides an unbiased estimate of $L(\hat{\mu}, \mu)$ in the sense that $E(\text{CP}(\hat{\mu})) = E(L(\hat{\mu}, \mu))$. The CP statistic was extended to local constant fitting by Rice (1984) and to local regression by Cleveland and Devlin (1988).

Definition 2.4 The **CP estimate of risk** for a local regression estimate $\hat{\mu}(x)$ is

$$\text{CP}(\hat{\mu}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} (Y_i - \hat{\mu}(x_i))^2 - n + 2\nu_1.$$

Implementation of the CP method requires an estimate of σ^2. The usual use of CP is to compare several different fits (for example, local regression with different bandwidths or different polynomial degrees). One should use

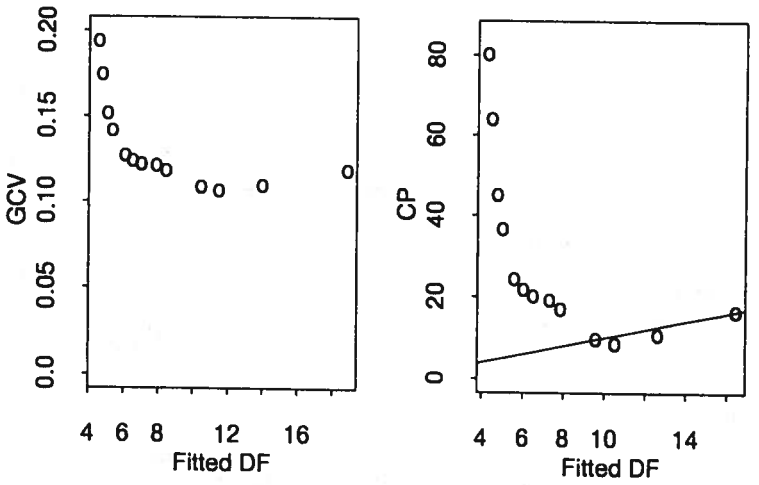

FIGURE 2.7. Generalized cross validation plot (left) and CP plot (right) for the ethanol dataset.

the *same* estimate $\hat{\sigma}^2$ for *all* fits being considered. The recommendation of Cleveland and Devlin is to compute the estimate (2.18) from a fit at the smallest bandwidth under consideration, at which one should be willing to assume that bias is negligible.

2.4.3 *Cross Validation Plots*

Frequently, the use of cross validation and CP is automated: A computer program computes $\mathrm{CV}(\hat{\mu})$ or $\mathrm{CP}(\hat{\mu})$ for several different fits and selects the fit with the lowest score. But, as argued strongly by Cleveland and Devlin (1988), this discards much of the information about the bias-variance trade-off that the statistics provide. Cleveland and Devlin introduce the CP (or M) plot as a graphical tool for displaying these statistics.

Example 2.7. The GCV and CP statistics are computed for local quadratic fits to the ethanol dataset and a range of smoothing parameters; $0.2 \leq \alpha \leq 0.8$. The results are shown in Figure 2.7 as a **cross validation plot** (left) and **CP plot** (right). These plots use the fitted degrees of freedom $\mathrm{tr}(\mathbf{L}^T\mathbf{L})$ as the horizontal axis and the GCV and CP statistics as the vertical axis. The smoothing parameter is $\alpha = 0.8$ on the left, decreasing in steps of 0.05 to $\alpha = 0.2$ on the right.

Both plots show a similar profile. The first four points, with fewer than five fitted degrees of freedom (or $\alpha > 0.65$), produce large GCV and CP scores, indicating these fits are inadequate. For larger degrees of freedom,

the plots (especially GCV) are flat, indicating there is little to choose between the fits. As α is decreased from 0.6 to 0.2, the fitted degrees of freedom increases from 5.6 to 16.4, and the GCV score ranges from 0.107 to 0.127.

An important point in the construction of Figure 2.7 is the use of the fitted degrees of freedom, rather than the smoothing parameter, as the horizontal axis. This aids interpretation: Four degrees of freedom represents a smooth model with very little flexibility, while 16 degrees of freedom represents a noisy model showing many features. It also aids comparability. For example, CP scores could be computed for other polynomial degrees or for other smoothing methods and added to the plot.

The cross validation and CP plots must be emphasized as *a graphical aid in choosing smoothing parameters*. Flat plots, such as Figure 2.7, occur frequently, and any model with a GCV score near the minimum is likely to have similar predictive power. *The flatness of the plot reflects the uncertainty in the data, and the resultant difficulty in choosing smoothing parameters.* We concluded earlier that $\alpha = 0.8$ was too large for the ethanol dataset; the lack of fit is reflected as the sharp increase in the GCV and CP scores at the left boundary of Figure 2.7. At the other end, we are unsure whether the additional features at $\alpha = 0.2$ in Figure 2.3 were real. The flat GCV plot reflects this uncertainty.

A consequence of Figure 2.7 is that going to extensive lengths to minimize GCV is very data-sensitive and can produce an unsatisfactory fit. In general, minimizing GCV (or CP, or CV) is highly variable: two visually similar datasets could produce very different results. Most importantly, just minimizing GCV discards significant information provided by the whole profile of the GCV curve, as displayed by the cross validation plot.

We should emphasize that the points raised here are *not* problems with cross validation and CP, but a reflection of the difficulty of model selection. This issue is explored further in Chapter 10, where cross validation methods are compared with bandwidth selectors claimed to be less variable. Such selectors are found to reflect the model selection difficulty in other ways; in particular, missing features when applied to difficult smoothing problems.

2.5 Linear Estimation

As noted previously, local regression is a linear estimate. The linear representation (2.12) provides the basis for a theoretical development of local regression estimation. Simple mean and variance expressions have already been derived; further properties are developed in this section. Results are also derived for the influence function and model selection criteria.

The first task is to identify the weight diagram. Let $\mathbf{X}$ be the $n \times (p+1)$ design matrix with rows $A(x_i - x)^T$, $\mathbf{W}$ be the diagonal matrix with entries

$w_i(x)$ and $Y = (Y_1, Y_2, \ldots, Y_n)^T$ be the response vector. The weighted sum of squares (2.5) can be written in matrix form

$$(Y - \mathbf{X}a)^T \mathbf{W}(Y - \mathbf{X}a).$$

If $\mathbf{WX}$ has full column rank, least squares theory gives the explicit expression

$$\hat{a} = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}Y \tag{2.23}$$

for the minimizer of (2.5).

The representation (2.23) identifies the weight diagram for the local polynomial smooth, defined by (2.12):

$$l(x)^T = e_1^T(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}. \tag{2.24}$$

Here, e_1 is the unit vector; $e_1 = (1, 0, \ldots, 0)^T$.

The following theorem, originally from Henderson (1916) for local cubic fitting, provides a characterization of the weight diagrams for local polynomial regression.

Theorem 2.1 (Henderson's Theorem) The weight diagram for a local polynomial fit of degree p has the form

$$l_i(x) = W\left(\frac{x_i - x}{h(x)}\right)\langle \alpha, A(x_i - x)\rangle; \tag{2.25}$$

that is, the least squares weights multiplied by a polynomial of degree p. This representation is unique, provided $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is non-singular.

Conversely, if a linear estimate reproduces polynomials of degree p, and the weight diagram has at most p sign changes, then the estimate can be represented as a local polynomial fit of degree p.

Proof: The representation (2.24) immediately yields (2.25), with $\alpha^T = e_1^T(\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}$, providing $\mathbf{X}^T\mathbf{W}\mathbf{X}$ is non-singular.

For the converse, define a polynomial $P(u-x)$ of degree $\leq p$, whose roots match the sign changes of the weight diagram. Then, the smoother is reconstructed as local polynomial smoothing with weights $w_i(x) = l_i(x)/P(x_i - x)$. □

Despite its innocuous simplicity, Henderson's theorem has profound consequences. For example, the polynomial reproduction property implies the local regression method achieves exact bias correction in finite samples. The bias of a local linear estimate cannot depend on the slope $\mu'(x)$. See Section 2.5.2 for more discussion. As noted in Section 1.4, this contrasts sharply with kernel smoothing literature, where considerable effort has been expended in achieving asymptotic bias corrections.

For an immediate illustration of the power of Henderson's theorem, we derive a simplification of the leave-one-out cross validation statistic (2.20).

Theorem 2.2 If $\text{infl}(x_i) < 1$, the leave-one-out cross validation estimate $\hat{\mu}_{-i}(x)$ is

$$\hat{\mu}_{-i}(x_i) = \frac{\hat{\mu}(x_i) - \text{infl}(x_i)Y_i}{1 - \text{infl}(x_i)}$$

and

$$\text{CV}(\hat{\mu}) = \frac{1}{n}\sum_{i=1}^{n}\frac{(Y_i - \hat{\mu}(x_i))^2}{(1 - \text{infl}(x_i))^2}. \tag{2.26}$$

This result can be proved directly using (2.12), (2.24) and some tedious matrix algebra. See Exercise 2.2 or Appendix B.4 of Myers (1990) for the same result for parametric regression. The following proof derives the result directly from Henderson's theorem.

Proof: Let

$$m_j(x_i) = \frac{l_j(x_i)}{1 - \text{infl}(x_i)}; j = 1, \ldots, n; j \neq i.$$

Using Henderson's theorem, we show that $\{m_j(x_i)\}$ is the weight diagram for $\hat{\mu}_{-i}(x_i)$ and thus

$$\hat{\mu}_{-i}(x_i) = \sum_{\substack{j=1\\ j\neq i}}^{n} m_j(x_i)Y_j = \frac{\hat{\mu}(x_i) - \text{infl}(x_i)Y_i}{1 - \text{infl}(x_i)}.$$

(2.26) then follows directly from (2.20).

For fixed x_i, $\{m_j(x_i)\}$ is a polynomial multiplied by the weights $W((x_j - x_i)/h)$, because $\{l_j(x_i)\}$ is. It remains to show that $\{m_j(x_i)\}$ reproduces polynomials $P(x)$ of degree $\leq p$:

$$\begin{aligned}
\sum_{\substack{j=1\\ j\neq i}}^{n} m_j(x_i)P(x_j) &= \frac{1}{1 - \text{infl}(x_i)}\left(\sum_{j=1}^{n} l_j(x_i)P(x_j) - l_i(x_i)P(x_i)\right) \\
&= \frac{1}{1 - \text{infl}(x_i)}\left(P(x_i) - \text{infl}(x_i)P(x_i)\right) \\
&= P(x_i)
\end{aligned}$$

where the second line follows from the polynomial reproducing property of $\{l_j(x_i)\}_{j=1}^n$. □

Theorem 2.2 assumes the bandwidth $h(x)$ does not change when the fit is carried out on the reduced dataset. This assumption can fail for a nearest neighbor bandwidth, but for estimating prediction error, this is the correct assumption to make, since the leave-one-out problem should mimic, as closely as possible, the true prediction problem.

The motivation for GCV also follows from the approximation (2.26), simply replacing $\text{infl}(x_i)$ by the average value $\text{tr}(\mathbf{L})/n$.

b) Show that the weight diagram for $\hat{\mu}(i)$ is quadratic reproducing, and hence by Henderson's theorem $\hat{\mu}$ is a local quadratic smooth. Does the weight diagram look sensible?

2.6 Some visual experiments:

a) Construct a dataset with a mean function having flat and steep regions. For example, let x_i be uniform on the interval $[-5, 5]$ and $Y_i = \Phi(x_i) + \epsilon_i$ where $\Phi(x)$ is the standard normal distribution function and ϵ_i is normally distributed with $\sigma = 0.1$. Plot the dataset. Does the residual variance *look* constant?

b) Construct a nonuniform predictor variable. For example, in S-Plus, `x <- sqrt(runif(100))`. Generate standard normal observations as the response variable. Plot the data. Does the residual variance *look* constant? This experiment may take two or three attempts; eventually, large residuals in the high density region should be distracting.

Remark. A real data example where these visual distractions occur is provided in Exercise 3.2.

2.7 *Estimation under the L_1 loss function.*

a) Suppose $X \sim N(\mu, \sigma^2)$. Show that

$$\mathrm{E}|X| = \mu\left(\Phi(\frac{\mu}{\sigma}) - \Phi(-\frac{\mu}{\sigma})\right) + 2\sigma\phi(\frac{\mu}{\sigma}).$$

Here, $\phi(\cdot)$ and $\Phi(\cdot)$ denote the standard normal density and distribution function.

b) Consider the risk function $R_1(\hat{\mu}, \mu) = n^{-1}\sum_{i=1}^{n}|\hat{\mu}(x_i) - \mu(x_i)|$. Derive an explicit expression for $R_1(\hat{\mu}, \mu)$ in terms of the bias and variance.

2.8 *L_1 Cross Validation.* Suppose the risk function for predicting a future observation $(x_{\text{new}}, Y_{\text{new}})$ is $E|Y_{\text{new}} - \hat{\mu}(x_{\text{new}})|$. This is estimated by the L_1 cross validation criterion

$$\mathrm{CV}_1(\hat{\mu}) = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{\mu}_{-i}(x_i)|.$$

Show that

$$\mathrm{CV}_1(\hat{\mu}) = \frac{1}{n}\sum_{i=1}^{n}\frac{|Y_i - \hat{\mu}(x_i)|}{1 - \mathrm{infl}(x_i)}.$$

Propose an L_1 version of generalized cross validation.

3

Fitting with LOCFIT

The examples in this book are implemented using the local regression software LOCFIT. This can be used either as a stand-alone program or as a library within the S (Becker, Chambers and Wilks 1988), S-Plus or R (Ihaka and Gentleman 1996) languages. See Appendix A for details of how to obtain the LOCFIT code and installation.

The code examples in this book are designed for S version 4; most will also work in S-Plus and R. The syntax for the stand-alone C-LOCFIT version is different. For many examples, the corresponding code for the stand-alone version can be obtained using the `example` command:

```
locfit> example 3.1

Example 3.1. Local Regression

  locfit NOx~E data=ethanol alpha=0.5
  plotfit data=T
```

prints the corresponding code on the screen. Typing

```
locfit> example 3.1 run
```

results in the code being executed and plots being produced as appropriate.

3.1 Local Regression with LOCFIT

LOCFIT provides two functions, `locfit()` and `locfit.raw()`, to perform local regression. The `locfit()` function uses the S model language to specify the local regression model, while `locfit.raw()` has separate arguments for the predictor and response variables. In other respects, the two functions are identical.

Example 3.1. We fit a local quadratic model to the ethanol dataset and plot the result:

```
> fit <- locfit(NOx~E, data=ethanol, alpha=0.5)
> fit
Call:
locfit(formula = NOx ~ E, data = ethanol, alpha = 0.5)

Number of observations:          88
Fitted Degrees of freedom:       6.485
Residual scale:                  0.336
> plot(fit, get.data=T)
```

The plot was displayed in Figure 2.2.

The first argument to `locfit()` is the *model formula* `NOx~E` specifying the local regression model and is read as "`NOx` is modeled by `E`". The `data=ethanol` argument specifies a data frame where the variables in the model formula may be found; if the `data` argument is omitted, currently attached data directories are searched. The use of model formulae and data frames follows chapter 2 of Chambers and Hastie (1992). The third argument to the `locfit()` function, `alpha`, controls the bandwidth. Here, a nearest neighbor based bandwidth covering 50% of the data is used. The fit could also be generated by

```
> fit <- locfit.raw(ethanol$E, ethanol$NOx, alpha=0.5)
```

The fit returned by the `locfit()` call is an S object, with the `"locfit"` class. Printing the fit then shows a short summary. The `plot(fit)` command then calls the plot method `plot.locfit()`. The `get.data=T` argument adds the original data to the plot.

Confidence intervals can be added to the plot with the `band=` argument, for example,

```
> plot(fit,band="global")
```

adds confidence intervals under the assumption that the residual variance σ^2 is constant. If `band="local"`, an attempt is made to estimate σ^2 locally. If `band="pred"`, prediction bands (2.17) are computed under the constant variance assumption. Variance estimation and confidence bands are discussed in more detail in Chapter 9.

3.2 Customizing the Local Fit

The `locfit()` function has additional arguments to control the fit. The most important are described in this section; others are introduced throughout the book as they are needed.

Smoothing Parameter. The `alpha` argument, used in Example 3.1, controls the bandwidth. When `alpha` is given as a single number, it represents a *nearest neighbor* fraction, as described in section 2.2.1.

Example 3.2. (Changing the Smoothing Parameter.) We compute local regression fits for the ethanol dataset, with four different smoothing parameters:

```
> alp <- c(0.8,0.6,0.4,0.2)
> for(a in alp) {
+   fit <- locfit(NOx~E, data=ethanol, alpha=a)
+   plot(fit, get.data=T, main=paste("alpha =",a))
+ }
```

The fits are as shown in Figure 2.3 (For the actual code producing the trellis display, see section B.4).

More generally, `alpha` can be specified as a vector with two components. The second component represents a constant bandwidth, so `alpha=c(0,1)` implies $h(x) = 1$ is used everywhere. If both the nearest neighbor and fixed components are nonzero, both bandwidths are computed, and $h(x)$ will be chosen as the larger component. Specifically, if $\alpha = (\alpha_0, \alpha_1)$, the bandwidth $h(x)$ is computed as follows:

1. $k = \lfloor n\alpha_0 \rfloor$.
2. Compute $d_i = |x - x_i|; i = 1, \ldots, n$ and find the kth smallest $d_{(k)}$.
3. Return $h(x) = \max(d_{(k)}, \alpha_1)$.

The default smoothing parameter is `alpha=c(0.7,0)`.

Degree of Local Polynomial. The degree of local polynomial is specified through the `deg` argument: `deg=1` specifies a local linear fit; `deg=2` species local quadratic (the default). For univariate fits, LOCFIT supports any degree, although there's usually little reason to use degrees greater than 3. For multivariate fits, `deg=3` is the maximum.

The Weight Function. LOCFIT supports several weight functions, listed in Table 3.1. These are selected with the `kern` argument to the `locfit()` function. For example,

```
> locfit(...,kern="gauss")
```

selects the Gaussian weight function. The default weight function is the tricube. Note also the factor of 2.5 in the Gaussian weight function; this

makes the scaling for the Gaussian weight function more comparable to the compact weight functions.

`rect`	Rectangular	$W(x) = 1, \|x\| < 1$
`tria`	Triangular	$W(x) = 1 - \|x\|, \|x\| < 1$
`epan`	Epanechnikov	$W(x) = 1 - x^2, \|x\| < 1$
`bisq`	Bisquare	$W(x) = (1 - x^2)^2, \|x\| < 1$
`tcub`	Tricube	$W(x) = (1 - \|x\|^3)^3, \|x\| < 1$
`trwt`	Triweight	$W(x) = (1 - x^2)^3, \|x\| < 1$
`gauss`	Gaussian	$W(x) = \exp(-(2.5x)^2/2)$
`expl`	Exponential	$W(x) = \exp(-3\|x\|)$
`minm`	Minimax	See Section 13.3
`macl`	McLain	$W(x) = 1/(x + \epsilon)^2$

TABLE 3.1. The LOCFIT weight functions.

3.3 The Computational Model

The definition of local regression formally requires solving a weighted least squares problem for each fitting point x. But for large datasets, or the iterative procedures discussed in later chapters, this becomes computationally expensive.

The idea of a computational model began with the LOWESS algorithm (Cleveland 1979) and was developed considerably by LOESS (Cleveland and Grosse 1991). The local regression method is carried out at a small set of fitting points. The fitted values and local slopes at these fitting points are then used to define a fitted surface, which can be evaluated rapidly at any point. LOCFIT uses a similar computational model but differs in the way the fitting points are chosen. In particular, the LOCFIT computational model is bandwidth adaptive, choosing the most fitting points in regions where the smallest bandwidths are used. The algorithm is described more fully in Chapter 12.

The determination of fitting points and the direct fitting are performed by the `locfit()` function. The `predict.locfit()` and `preplot.locfit()` methods are used to interpolate the fits. These functions have a similar set of arguments but differ in the returned objects: `predict.locfit()` returns a vector of the predicted values, while `preplot.locfit()` returns an object with the `"preplot.locfit"` class. This object contains prediction points, predicted values and other information required to produce a plot.

Example 3.3. For the fit to the ethanol dataset, the fitted surface is evaluated at $E = 0.6, 0.8$ and 1.0:

```
> fit <- locfit(NOx~E, data=ethanol, alpha=0.5)
```



```
> predict(fit, c(0.6,0.8,1.0))
[1] 0.7239408 2.7544413 3.1183651
```

The two arguments to `predict()` are the `"locfit"` object and the `newdata` of prediction points. The latter can have one of several forms, including a vector, data frame, matrix or grid margins. See Appendix B.1 for more details.

3.4 Diagnostics

3.4.1 Residuals

The residuals of a LOCFIT model can be found with the command

```
> res <- residuals(fit)
```

which calls the residuals method `residuals.locfit()`.

Example 3.4. Smoothed residual plots are constructed for the four fits in Figure 2.3:

```
> alp <- c(0.8,0.6,0.4,0.2)
> for(a in alp) {
+   fit <- locfit(NOx~E, data=ethanol, alpha=a)
+   res <- residuals(fit)
+   fit2 <- locfit.raw(ethanol$E, res, alpha=0.2)
+   plot(ethanol$E, res, main=paste("alpha =",a),
+     ylim = c(-1,1))
+   lines(fit2)
+   abline(h=0, lty=2)
+ }
```

Figure 2.5 showed the smoothed residual plots. Note that `locfit.raw()` is used to smooth the residuals, since the residuals are not stored on the `ethanol` data frame.

3.4.2 Cross Validation

The cross validation and CP criteria can be computed from information stored on a `"locfit"` object. We begin with GCV, since this is most direct.

Example 3.5. From the ethanol fit, we extract a `dp` component that contains information about the fit:[1]

```
> fit <- locfit(NOx~E, data=ethanol, alpha=0.5)
```

[1] Here, and elsewhere, users of S version 3 must substitute `$` for `@`.

```
> fit@dp
 nnalph fixh adpen cut       lk      df1      df2        rv
    0.5    0     0 0.8 -4.53376 7.013307 6.487448 0.1126948
```

The components of interest to us are `lk` (-0.5 times the residual sum of squares), `df1` (tr(**L**)) and `df2` (tr(**L**'**L**)). Since this dataset contains 88 points, the GCV score is computed as

```
> gencv <- 88*(-2*fit@dp["lk"])/(88-fit@dp["df1"])^2
> gencv
       lk
0.1216522
```

In fact, LOCFIT provides two functions, `gcv()` and `gcvplot()`, to simplify this. `gcv()` automatically calls `locfit()`, and returns a vector with four components: the log-likelihood (-0.5 times the residual sum of squares), the degrees of freedom according to the influence and variance definitions, and the GCV score. The arguments for `gcv()` are exactly the same as for `locfit()`.

`gcvplot()` is a wrapper function for `gcv()`. It is provided a vector of smoothing parameters, and calls `gcv()` in turn for each parameter. It returns an object with the `"gcvplot"` class; the plot method defined for this class produces cross validation plots such as those in Figure 2.7.

Example 3.6. The `gcvplot()` function is applied to the ethanol dataset for a range of smoothing parameters:

```
> alpha <- seq(0.2, 0.8, by=0.05)
> plot(gcvplot(NOx~E, data=ethanol, alpha=alpha),
+   ylim=c(0,0.2))
```

Figure 2.7 showed the result. Note that each smoothing parameter here is a nearest neighbor fraction; to use constant bandwidths, `alpha` should be a two-column matrix with the first column 0.

The cross validation approach is only slightly more complicated. We can use the definition directly by using a special cross validation evaluation structure, `ev="cross"`:

```
> fit <- locfit(NOx~E, data=ethanol, alpha=0.5, ev="cross")
> -2*fit@dp["lk"]/88
       lk
0.1171337
```

This deletes each observation in turn and computes the fit, so should only be used for fairly small datasets. For large datasets, an approximation is

```
> fit <- locfit(NOx~E, data=ethanol, alpha=0.5)
> r <- residuals(fit)
> infl <- fitted(fit,what="infl")
> mean((r/(1-infl))^2)
[1] 0.1190185
```

The small discrepancy here is because the fitted values and influence function are being interpolated rather than computed directly at each point. A simpler alternative is

```
> mean(residuals(fit,cv=T)^2)
[1] 0.1177989
```

When provided with the `cv=T` argument, the `residuals.locfit()` function computes the values

$$(1 + \text{infl}(x_i))(Y_i - \hat{\mu}(x_i)). \tag{3.1}$$

Thus the sum of squares in this case is

$$\sum_{i=1}^{n} ((1 + \text{infl}(x_i))(Y_i - \hat{\mu}(x_i)))^2 \tag{3.2}$$

rather than the exact cross validation. Clearly, the two approaches are asymptotically equivalent in large samples, when $\text{infl}(x_i)$ is small. The motivation for (3.1) will become clear in Chapter 4, where (3.1) generalizes naturally to local likelihood problems. Droge (1996) argued that (3.2) provides a better estimate of the prediction mean squared error in finite samples. A pair of functions, `lcv()` and `lcvplot()`, are provided to implement this cross validation method.

The pair of functions, `cp()` and `cpplot()`, implement the CP method. The implementation is again similar to `gcv()`, but now requires an estimate of the residual variance σ^2. By default, `cpplot()` takes the variance estimate (2.18) from the fit with the largest degrees of freedom ν_2.

3.5 Multivariate Fitting and Visualization

To specify a multivariate local regression model, multiple terms are specified on the right-hand side of the model formula.

Example 3.7. We consider the ethanol dataset used in Figure 2.1. A second predictor variable, `C`, was not considered previously and measures the compression ratio of the engine. The fit is computed by:

```
> fit <- locfit(NOx~E+C, data=ethanol, alpha=0.5, scale=0)
> plot(fit, get.data=T)
> plot(fit, type="persp")
```

The formula can be given either as `NOx~E+C` or `NOx~E*C`; both will give the same results. Figure 3.1 shows the resulting contour and perspective plots. If `type="image"`, the plot is produced using the S-Plus `image()` function.

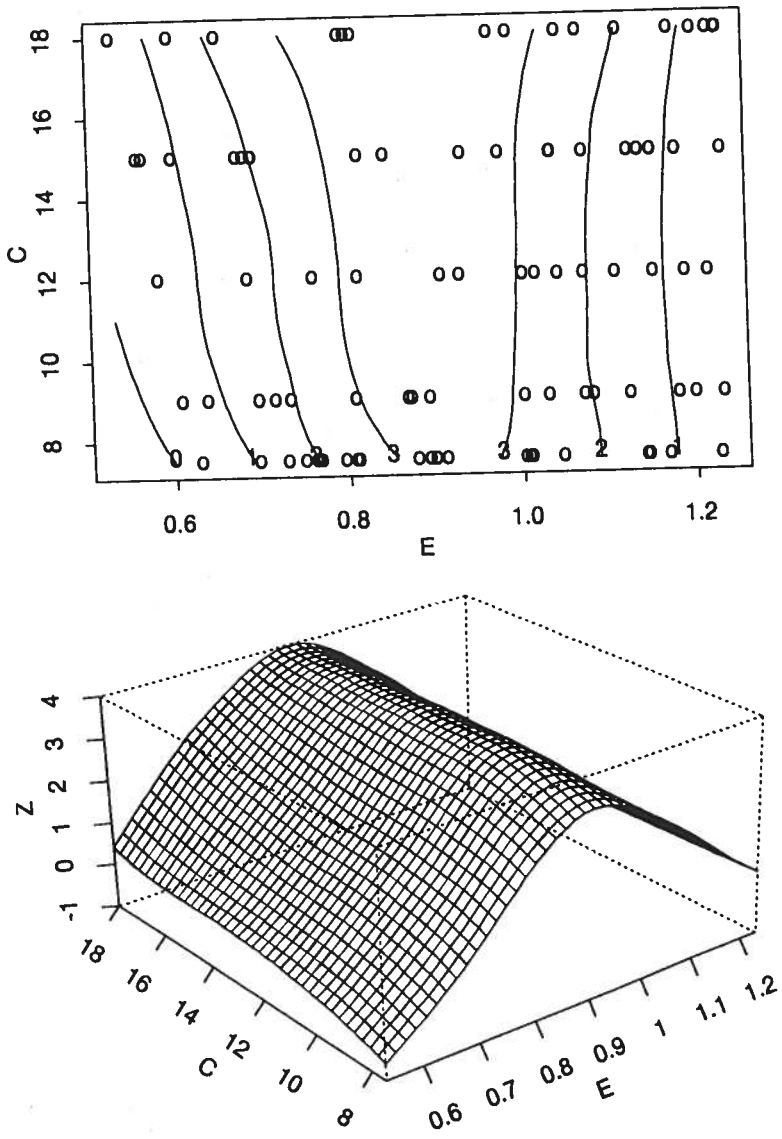

FIGURE 3.1. Bivariate local regression for the ethanol data.

An important argument in the multivariate case is `scale`. This provides a set of scales s_j to use in the distance computation (2.10), thereby controlling the relative amounts of smoothing in each variable. Specifying `scale=0` results in the sample standard deviations of each of the variables being computed and used as scales. One can also compare fits with different choices of scales using cross validation methods.

The contour plot and perspective plot in Figure 3.1 both display the fitted surface, but serve different visualization purposes. No general claim can be made that one of these displays is better. Either display shows that `NOx` is much more heavily dependent on `E` than on `C`. The general nature of this dependence - first increasing, then decreasing, as `E` increases - is more readily apparent in the perspective plot. On the other hand, values of the fitted surface are more readily judged from the contour plot. As an exercise, try to estimate $\hat{\mu}(0.7, 13)$ from the perspective plot, and then do the same from the contour plot.

What about the dependence on `C`? Both plots appear to show some increase in `NOx` with `C`, although it is difficult to preceive the precise nature of the relationship. Is there any interaction between the two variables? Neither plot is good for answering this type of question. An alternative display is of one dimensional cross sections of the fit. The S Trellis library (Becker, Cleveland, Shyu and Kaluzny 1994) provides a convenient mechanism for producing such plots. An interface is provided in the `plot.locfit()` function, by specifying a *panel variable* `pv`, which is varied within a panel of the trellis display, and a `tv`, which is varied between panels of the trellis display. A final argument, `mtv`, specifies the number of panels for the display.

Example 3.8. To run this example, trellis graphics must be initialized, using the `trellis.device()` function. We plot the bivariate fit to the ethanol dataset, using `E` as the trellis variable:

```
> fit <- locfit(NOx~E+C, data=ethanol, alpha=0.5, scale=0)
> plot(fit, pv="C", tv="E", mtv=9, get.data=T)
```

Figure 3.2 display the results. The slight dependence of `NOx` on `C` is much easier to see in this plot than in Figures 3.1 and 3.3.

3.5.1 *Additive Models*

The definition of multivariate local regression extends to any number of dimensions. But beyond two or three dimensions, a local regression model is difficult to fit, due to both the rapid increase in the number of parameters in the local model and the sparsity of data in high dimensional spaces. In addition, visualization of a high dimensional surface is difficult.

Because of these problems, a number of simplified models have been proposed. Typically, these methods build a fitted surface by applying local regression (or other smoothers) to low dimensional projections of the data.

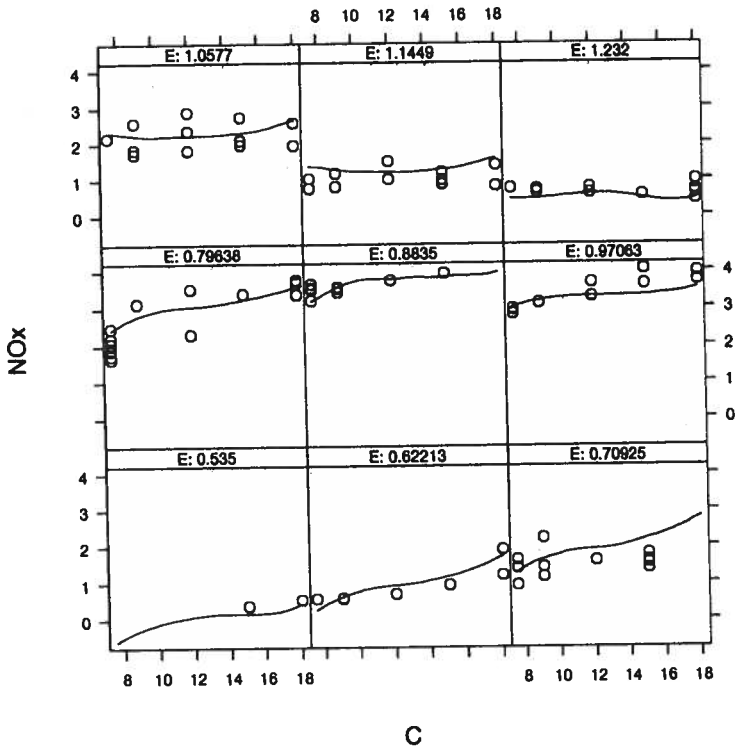

FIGURE 3.2. Bivariate local regression: Ethanol data sectioned by equivalence ratio E.

The most widely studied model of this type is the additive model, which for two predictors x and z has the form

$$\mu(x,z) = \mu_1(x) + \mu_2(z)$$

where $\mu_1(x)$ and $\mu_2(z)$ are smooth functions. The backfitting algorithm can be used to fit the model and alternately estimates the components. A thorough account of additive models and the backfitting algorithm can be found in Hastie and Tibshirani (1990). Opsomer and Ruppert (1997) discuss some theoretical properties of the backfitting algorithm.

Additive models are fitted in S using the `gam()` function described in Hastie (1992). To use LOCFIT for the additive component, functions `lf()` and `gam.lf()` are provided. The `lf()` function is used in the model formula; at the time of writing it accepts the `alpha`, `deg`, `ev` and `kern` arguments. We remark that the `gam` library also includes a `lo()` function for fitting additive terms using local regression and LOESS. The `lf()` function has considerably more flexibility.

Example 3.9. An additive model is fitted to the ethanol dataset. We use a local quadratic term with `alpha=0.5` for the equivalence ratio and a local linear term for the compression ratio.

```
> library("locfit",first=T)
> fit <- gam(NOx~lf(E,alpha=0.5)+lf(C,deg=1),data=ethanol)
> plot(fit)
```

Figure 3.3 plots the two additive components, showing nearly linear dependence on C and the peaked dependence on E. One has to look closely at the scales to see that the E dependence is much stronger.

Important: For this example to work properly, you *must* specify `first=T` when attaching the LOCFIT library, or otherwise ensure `"lf"` appears in your `gam.slist` variable.

A special case of the additive model is the partially linear model:

$$\mu(x,z) = \mu_1(x) + \langle \beta, z \rangle .$$

This model is particularly attractive since the backfitting algorithm has a closed form limit:

$$\hat{\beta} = \left(\mathbf{X}_2^T(\mathbf{I}-\mathbf{L}_1)\mathbf{X}_2\right)^{-1}\mathbf{X}_2^T(\mathbf{I}-\mathbf{L}_1)Y, \tag{3.3}$$

where $\mathbf{X}_2$ is the design matrix for the parametric component and $\mathbf{L}_1$ is the hat matrix for the smooth component. See Hastie and Tibshirani ((1990), page 118). This model can be fitted using `gam()`. For example,

```
> gam(NOx~lf(E,alpha=0.5)+C, data=ethanol)
```

produces a fit that is smooth in E and linear in C.

Using a slightly different motivation, Robinson (1988) and Eubank and Speckman (1993a) arrive at a modified form of (3.3), using $(\mathbf{I}-\mathbf{L}_1)^T(\mathbf{I}-\mathbf{L}_1)$ in place of $\mathbf{I}-\mathbf{L}_1$. Other references on partially linear models include Engle, Granger, Rice and Weiss (1986), Green (1987) and Severini and Staniswalis (1994). A more thorough review is provided by Ichimura and Todd (1999).

3.5.2 *Conditionally Parametric Models*

The conditionally parametric model is similar to the partially linear model, in that the fit is smooth in some variables and parametric in others. But the conditionally parametric fit allows all coefficients of the parametric variables to depend on the smoothing variables. A conditionally quadratic fit has the form

$$\mu(x,z) = a_0(x) + a_1(x)z + a_2(x)z^2.$$

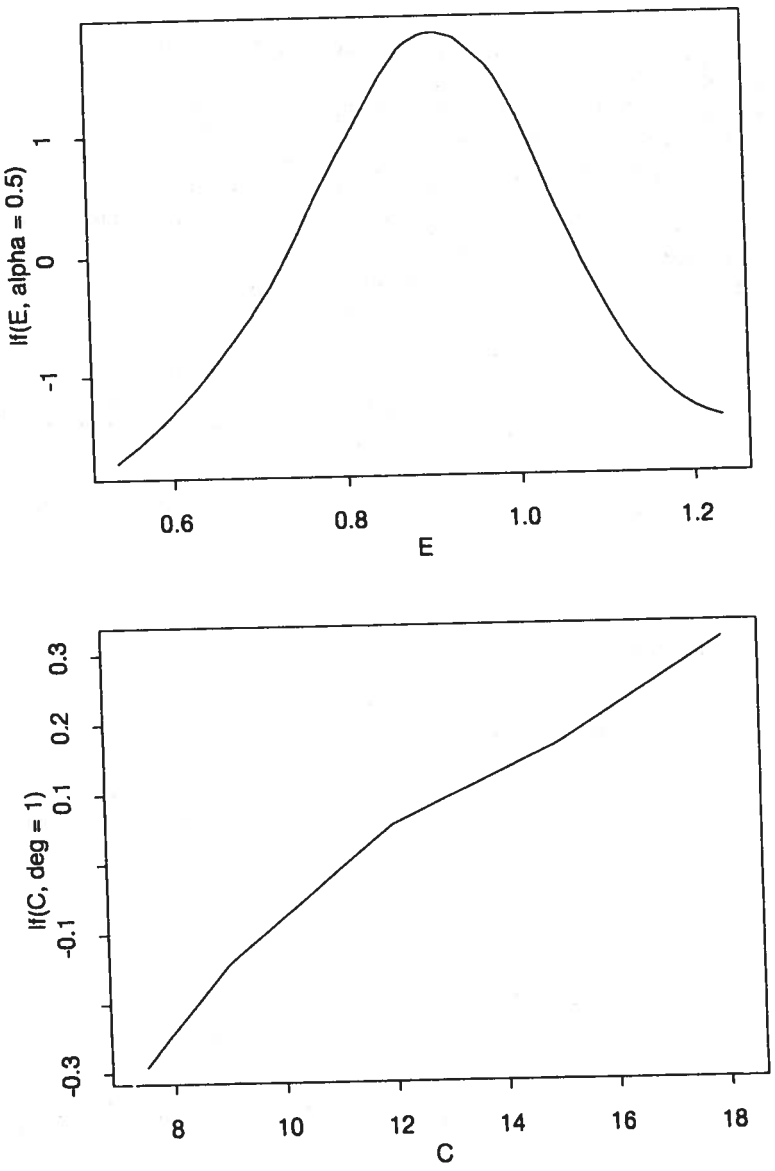

FIGURE 3.3. Components of an additive fit to the ethanol dataset.

For fixed x, $\mu(x,z)$ is a quadratic function of z. But all three coefficients, $a_0(x)$, $a_1(x)$ and $a_2(x)$, are allowed to vary as a function of x. This differs from the partially linear model in which only $a_0(x)$ is allowed to depend on x.

The conditionally parametric fit was considered in detail by Cleveland, Grosse and Shyu (1992) and Cleveland (1994). In particular, they provide a conceptually straightforward way to fit the model: Simply fit a bivariate local regression in x and z but ignore the z variable when computing the distances and smoothing weights. The varying coefficient model (Hastie and Tibshirani 1992) is a broad class of models that encompasses both conditionally parametric fits and partially linear models.

To fit a conditionally parametric model in LOCFIT, one uses the special `cpar()` function in the model formula. For example,

```
> locfit(NOx~E+cpar(C), data=ethanol, alpha=0.5)
```

produces a fit that is conditionally quadratic in `C`.

3.6 Exercises

3.1 a) Try to fit the ethanol dataset using local constant and local linear fitting. By varying the bandwidth (using both the fixed and nearest neighbor components, if necessary), can a fit comparable to the local quadratic fit in Figure 2.2 be obtained? Pay attention to both proper modeling of the peak and the leveling off at the boundaries, and to the roughness of the estimates.

b) Compute the GCV scores for local linear fitting, and compare with the results of local quadratic fitting in Figure 2.7.

3.2 The diabetes dataset used by Hastie and Tibshirani (1990) consists of a predictor variable `age` and response `lcp`.

a) Produce a scatter plot of the data. Does the residual variance look constant?

b) Fit a local regression model. Construct appropriate smoothed residual plots to investigate the nonhomogeneous variance further. (You'll probably conclude that the nonhomogeneity is real, but much less than might have been guessed from the scatterplot).

Note: the dataset can be accessed using `data=diab` in LOCFIT.

3.3 Consider the L_1 cross validation of Exercise 2.8. Write a modified version of `gcv()` to implement the L_1 generalized cross validation (use `residuals()` to get $Y_i - \hat{\mu}(x_i)$). Apply this to the `NOx~E` model for the ethanol dataset, and compare with Figure 2.7.