17

Spatially Adaptive Smoothing

17.1 Introduction

The penalty parameter λ of a penalized spline controls the trade-off between bias and variance. In this chapter we introduce a method of fitting penalized splines wherein λ varies spatially in order to accommodate possible spatial nonhomogeneity of the regression function. In other words, λ is allowed to be a function of the independent variable x. Allowing λ to be a function of spatial location can improve mean squared error (MSE; see Section 3.11) and the accuracy of inference.

Suppose we are using a quadratic spline that has constant curvature between knots. If the regression function has rapid changes in curvature, then a small value of λ is needed so that the second derivative of the fitted spline can take jumps large enough to accommodate these changes in curvature. Conversely, if the curvature changes slowly, then a large value of λ will not cause large bias and will reduce df_{fit} and the variance of the fitted values. The problem is that a regression function may be spatially nonhomogeneous with some regions of rapidly changing curvature and other regions of little change in curvature. A single value of λ is not suitable for such functions. The inferiority – in terms of MSE – of splines having a single smoothing parameter is shown in a simulation study by Wand (2000). In that study, for regression functions with significant spatial inhomogeneity, penalized splines with a single smoothing parameter were not competitive with knot selection methods. In an empirical study by Ruppert (2002), the spatially adaptive penalized splines described in this chapter were found to be at least as efficient as knot selection methods.

Another problem with having only a single smoothing parameter concerns inference. As seen in Chapters 4 and 16, penalized splines are BLUPs if one assumes a certain mixed model or Bayes estimates for particular priors. A single smoothing parameter corresponds to a spatially homogeneous mixed model distribution or Bayesian prior. For a penalized spline, the mixed model assumption (or prior) is that the knot coefficients $u_1, \ldots, u_K$ are independent $N(0, \sigma_u^2)$ for a single parameter σ_u^2. Such priors on $u_1, \ldots, u_K$ are not appropriate for spatially heterogeneous f. Consider the confidence intervals based on the posterior variance of $f(\cdot)$ discussed in Section 16.2.2. As Nychka (1988) shows, the resulting confidence bands have good *average* (over x) coverage probabilities but do not have accurate pointwise coverage probabilities in areas of high oscillations or other "features" in f.

https://doi.org/10.1017/CBO9780511755453.019 Published online by Cambridge University Press

17.2 A Local Penalty Method

Here is a simple approach to spatially varying λ. Choose another set of the knots, $\{\kappa_k^*\}_{k=1}^M$, where M is smaller than K and such that $\{\kappa_1^* = \kappa_1 < \cdots < \kappa_M^* = \kappa_K\}$. The penalty at one of these "subknots", say κ_k^*, is set equal to a parameter λ_k^*. The penalties at the original knots, $\{\kappa_k\}_{k=1}^K$, are determined by interpolation of the penalties at the $\{\kappa_k^*\}_{k=1}^M$. The interpolation is on the log-penalty scale to ensure positivity of the penalties. Thus we have a penalty $\lambda(\kappa_k)$ at each κ_k, but these penalties depend only upon $\boldsymbol{\lambda} = (\lambda_1^*, \ldots, \lambda_M^*)^\mathsf{T}$. Therefore, $\{\lambda(\kappa_1), \ldots, \lambda(\kappa_K)\}$ is a function of $\boldsymbol{\lambda}$. This function need not be derived explicitly but rather can be computed by using an interpolation algorithm; we used MATLAB's built-in linear interpolator. One could, of course, use other interpolation methods (e.g., cubic interpolation). If linear interpolation is used, then $\log\{\lambda(\cdot)\}$ is a linear spline with knots at $\{\kappa_k^*\}_{k=1}^M$.

As in Chapter 3, let $\mathbf{y} = [y_1, \ldots, y_n]^\mathsf{T}$ and let $\mathbf{C}$ be a matrix with ith row equal to

$$\mathbf{C}_i = [1 \;\; x_i \;\; \cdots \;\; x_i^p \;\; (x_i - \kappa_1)_+^p \;\; \cdots \;\; (x_i - \kappa_K)_+^p]. \tag{17.1}$$

Let $\mathbf{D}(\boldsymbol{\lambda})$ be a diagonal matrix whose first $p+1$ diagonal elements are 0 and whose remaining diagonal elements are $\lambda^2(\kappa_1), \ldots, \lambda^2(\kappa_K)$, which depend only on $\boldsymbol{\lambda}$. Then, as in Section 3.5, penalized spline estimates of

$$\boldsymbol{\nu} \equiv [\boldsymbol{\beta}^\mathsf{T} \;\; \mathbf{u}^\mathsf{T}]^\mathsf{T}$$

are given by

$$\hat{\boldsymbol{\nu}}(\boldsymbol{\lambda}) = \{\mathbf{C}^\mathsf{T}\mathbf{C} + \mathbf{D}(\boldsymbol{\lambda})\}^{-1}\mathbf{C}^\mathsf{T}\mathbf{y}. \tag{17.2}$$

The smoothing parameter $\boldsymbol{\lambda} = (\lambda_1^*, \ldots, \lambda_M^*)$ can be determined by minimizing the generalized cross-validation statistic

$$\mathrm{GCV}(\boldsymbol{\lambda}) = \frac{\|\mathbf{y} - \mathbf{C}\hat{\boldsymbol{\nu}}(\boldsymbol{\lambda})\|^2}{\{1 - df_{\mathrm{fit}}(\boldsymbol{\lambda})/n\}^2}.$$

Here

$$df_{\mathrm{fit}}(\boldsymbol{\lambda}) = \mathrm{tr}[\{\mathbf{C}^\mathsf{T}\mathbf{C} + \mathbf{D}(\boldsymbol{\lambda})\}^{-1}\mathbf{C}^\mathsf{T}\mathbf{C}] \tag{17.3}$$

is the degrees of freedom of the fit.

A search over an M-dimensional grid is not recommended because of computational cost. Rather, we recommend that one start with $\lambda_1^*, \ldots, \lambda_M^*$, each equal to the best global value of λ chosen by minimizing GCV as in Chapter 5. Then each λ_k^* is varied, with the others fixed, over a 1-dimensional grid centered at the current value of λ_k^*. On each such step, λ_k^* is replaced by the λ-value minimizing GCV on this grid. This minimizing of GCV over each λ_k^* is repeated a total of N_{iter} times. Although minimizing GCV over the λ_k^* one at a time in this manner does not guarantee finding the global minimum of GCV over $\lambda_1^*, \ldots, \lambda_M^*$, our simulations show that this procedure is effective in selecting a satisfactory amount of local smoothing. The minimum GCV global λ is a reasonably good starting value for the smoothing parameters, and each step of our algorithm improves on this start in the sense of lowering GCV. Since each λ_k^* controls the penalty over only a small range of x, the optimal value of one λ_k^* should depend only slightly upon the other λ_k^*. We believe this is why the one-at-a-time search strategy works effectively.

17.3 Completely Automatic Algorithm

The local penalty method has three tuning parameters: the number of knots, K; the number of subknots, M; and the number of iterations, N_{iter}. The exact values of the tuning parameters are not crucial provided they are within certain acceptable ranges – the crucial parameter is $\boldsymbol{\lambda}$, which is selected by GCV, not the user. However, users may want a completely automatic algorithm that requires no user-specified parameters and attempts to ensure that the tuning parameters are within acceptable ranges. An automatic algorithm would have to balance the need for the tuning parameters to be large enough to obtain a good fit with the need that the tuning parameters not be so large that the computation time is excessive; overfitting is not a concern because it is controlled by $\boldsymbol{\lambda}$.

In this section, we propose two methods for choosing the tuning parameters, the myopic and the full-search algorithms, which are similar to algorithms of the same names in Section 5.6.3. The two algorithms are based on the following principle: As the complexity of f increases, each of K, M, N_{iter} should increase. The algorithms use a sequence of values of (K, M, N_{iter}) where each parameter is nondecreasing in the sequence. The myopic algorithm stops when there is no appreciable decrease in GCV between two successive values of (K, M, N_{iter}). Monte Carlo experimentation, discussed in Section 17.5, shows that the values of N_{iter} and M have relatively little effect on the fit, at least within the ranges studied. However, it seems reasonable to increase N_{iter} and M slightly with K. On the other hand, computation time for given K is roughly proportional to $M \times N_{\text{iter}}$, so we avoid $N_{\text{iter}} > 2$ and $M > 6$.

The full-search algorithm computes GCV for all tuning parameter sets and chooses the set that minimizes GCV. Specifically, the sequence of values of (K, M, N_{iter}) that we use are $(10, 2, 1)$, $(20, 3, 2)$, $(40, 4, 2)$, $(80, 6, 2)$, and $(120, 6, 2)$. For each set of tuning parameters, $\boldsymbol{\lambda}$ is chosen to minimize GCV. The full-search algorithm chooses the set that has the smallest of these minimized GCV values. The myopic algorithm first compares (10, 2, 1) and (20, 3, 2) using GCV. If the value of GCV for (20, 3, 2) is more than a constant C times the GCV value of $(10, 2, 1)$, then we conclude that further increases in the tuning parameters will not appreciably decrease GCV – in the simulations we used $C = 0.98$ and that choice worked well. Therefore, we stop and use (10, 2, 1) or (20, 3, 2) as the final value of the three tuning parameters, whichever has the smallest GCV value. Otherwise, we fit using (40, 4, 2) and compare its GCV value to that of $(20, 3, 2)$. If the value of GCV for (40, 4, 2) is more than C times the GCV value of (20, 3, 2) then we stop and use either (20, 3, 2) or (40, 4, 2), whichever has the smallest GCV, as the final tuning parameters. Otherwise, we continue in this manner, comparing (40, 4, 2) to $(80, 6, 2)$, and so on. If very complex f were contemplated then one could, of course, continue using increasingly larger values of the tuning parameters.

17.4 Bayesian Inference

Bayesian inference for local penalty splines differs little from the Bayesian inference for global penalty splines that was discussed in Section 16.2.2. Suppose that $\varepsilon_1, \ldots, \varepsilon_n$ are independent $\mathrm{N}(0, \sigma_\varepsilon^2)$ and that the prior on $\boldsymbol{\nu}$ is $\mathbf{N}\{\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{\lambda})\}$, where $\boldsymbol{\Sigma}(\boldsymbol{\lambda})$ is a covariance matrix that depends on $\boldsymbol{\lambda}$. Here $\mathbf{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate normal distribution with mean and covariance matrix $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. For now, assume that σ_ε^2 and $\boldsymbol{\lambda}$ are known. Then the posterior log density of $\boldsymbol{\nu}$ given $\mathbf{y}$ is, up to an additive function of $\mathbf{y}$ and $(\sigma_\varepsilon^2, \boldsymbol{\lambda})$, given by

$$-\frac{1}{2}\left(\frac{1}{\sigma_\varepsilon^2}\|\mathbf{y} - \mathbf{C}\boldsymbol{\nu}\|_2^2 + \boldsymbol{\nu}^{\mathsf{T}}\boldsymbol{\Sigma}(\boldsymbol{\lambda})^{-1}\boldsymbol{\nu}\right). \tag{17.4}$$

The maximum a posteriori (MAP) estimator of $\boldsymbol{\nu}$ – that is, the mode of the posterior density – maximizes (17.4). Now let $\beta_0, \ldots, \beta_p$ have improper uniform $(-\infty, \infty)$ priors and let $\{u_k\}_{k=1}^K$ be independent, with β_{p+k} having a $\mathrm{N}(0, \sigma_\varepsilon^2/\lambda_k)$ distribution. Then

$$\boldsymbol{\Sigma}^{-1}(\boldsymbol{\lambda}) = \sigma_\varepsilon^2 \,\mathrm{diag}(0, \ldots, 0, \lambda_1, \ldots, \lambda_K). \tag{17.5}$$

More precisely, we let $\beta_0, \ldots, \beta_p$ have a $\mathrm{N}(0, \sigma_\beta^2)$ prior and then (17.5) holds in the limit as $\sigma_\beta \to \infty$. The MAP estimator of $\boldsymbol{\nu}$ minimizes

$$\sum_{i=1}^n \{y_i - f(x_i; \boldsymbol{\nu})\}^2 + \sum_{k=1}^K \lambda(\kappa_k) u_k^2.$$

Of course, the $\boldsymbol{\lambda} = (\lambda_1^*, \ldots, \lambda_M^*)^{\mathsf{T}}$ that determines the λ_k will not be known in practice. Empirical Bayes methods replace unknown "hyperparameters" in a prior by estimates. For example, if $\boldsymbol{\lambda}$ is estimated by GCV and then considered fixed, one is using empirical Bayes methods. Standard calculations show that when $\boldsymbol{\lambda}$ and σ_ε^2 are known, the posterior distribution of $\boldsymbol{\nu}$ is

$$\mathbf{N}[\hat{\boldsymbol{\nu}}(\boldsymbol{\lambda}), \sigma_\varepsilon^2\{\mathbf{C}^{\mathsf{T}}\mathbf{C} + \boldsymbol{\Sigma}(\boldsymbol{\lambda})\}^{-1}]. \tag{17.6}$$

Also, the posterior distribution of $\mathbf{f} = \{f(x_1), \ldots, f(x_n)\}^{\mathsf{T}}$ is

$$\mathbf{N}[\mathbf{C}\hat{\boldsymbol{\nu}}(\boldsymbol{\lambda}), \sigma_\varepsilon^2\mathbf{C}\{\mathbf{C}^{\mathsf{T}}\mathbf{C} + \boldsymbol{\Sigma}(\boldsymbol{\lambda})\}^{-1}\mathbf{C}^{\mathsf{T}}]. \tag{17.7}$$

An approximate Bayes posterior replaces $\boldsymbol{\lambda}$ and σ_ε^2 in (17.6) and (17.7) by estimates. Assuming that $\boldsymbol{\lambda}$ has been estimated by GCV, one need only estimate σ_ε^2 by $\|\mathbf{y} - \mathbf{C}\hat{\boldsymbol{\nu}}(\hat{\boldsymbol{\lambda}})\|^2/\{n - df_{\mathrm{fit}}(\hat{\boldsymbol{\lambda}})\}$, where $df_{\mathrm{fit}}(\boldsymbol{\lambda})$ is defined by (17.3). This gives the approximate posterior distribution for $\mathbf{f}$:

$$\mathbf{f} \sim \mathbf{N}[\mathbf{C}\hat{\boldsymbol{\nu}}(\hat{\boldsymbol{\lambda}}), \hat{\sigma}_\varepsilon^2\mathbf{C}\{\mathbf{C}^{\mathsf{T}}\mathbf{C} + \boldsymbol{\Sigma}(\hat{\boldsymbol{\lambda}})\}^{-1}\mathbf{C}^{\mathsf{T}}]. \tag{17.8}$$

The approximate Bayes $100(1-\alpha)\%$ confidence interval for $f(x_i)$ is

$$\hat{f}(x_i) \pm z(1 - \alpha/2)\,\widehat{\mathrm{st.dev.}}\{\hat{f}(x_i) - f(x_i)\},$$

where $\hat{f}(x_i) = \mathbf{C}_i\hat{\boldsymbol{\nu}}(\hat{\lambda})$ is the ith element of the posterior mean in (17.8) and $\widehat{\mathrm{st.dev.}}\{\hat{f}(x_i) - f(x_i)\}$ is the square root of the ith diagonal entry of the posterior covariance matrix in (17.8).

Because they estimate hyperparameters yet then pretend that the hyperparameters were known, these approximate Bayesian methods do not account for extra variability in the posterior distribution caused by estimation of hyperparameters in the prior; for discussion see, for example, Morris (1983), Laird and Louis (1987), Kass and Steffey (1989), or Carlin and Louis (2000). Everything else held constant, the underestimation of posterior variance should become worse as M increases, since each λ^*_m $(m = 1, \ldots, M)$ will be determined by fewer data and will therefore be more variable. As Nychka (1988) has shown empirically, this underestimation does not appear to be a problem for a global penalty that has only one hyperparameter. However, the local penalty has M hyperparameters. For local penalty splines we have found that the pointwise approximate posterior variance of $\hat{f}$ is too small in the sense that it noticeably underestimates the frequentist MSE.

A simple ad hoc correction to this problem is to multiply the pointwise posterior variances of the local penalty $\hat{f}$ from (17.8) by a constant so that the average pointwise posterior variance of $\hat{f}$ is the same for the global and local penalty estimators. The reasoning behind this correction is as follows. As stated previously, the global penalty approximate posterior variance from (17.8) is nearly equal to the frequentist's MSE on average. The local penalty estimate has an MSE that varies spatially but should be close, on average, to the MSE of the global penalty estimate and hence also close, on average, to the estimated posterior variance of the global penalty estimator. We found that this adjustment is effective in guaranteeing coverage probabilities at least as large as nominal, though in extreme cases of spatial heterogeneity the adjustment can be conservative; see Section 17.5. The reason for the latter is that, in cases of severe spatial heterogeneity, the local penalty MSE will be less, on average, than that of the global penalty estimate. Then, there will be an overcorrection and the local penalty MSE will be overestimated by this adjusted posterior variance. The result is that confidence intervals constructed with this adjustment should be conservative. The empirical evidence in Section 17.5 supports this conjecture. In that section, we refer to these adjusted intervals as *local penalty, conservative* intervals.

Another correction would be to use a fully Bayesian hierarchical model, where the hyperparameters are given a prior. Deely and Lindley (1981) first considered such empirical Bayes methods. An exact Bayesian analysis for penalized splines would require Gibbs sampling or other computationally intensive techniques (discussed in Chapter 16) and would be an interesting area for further research.

There are intermediate positions between the quick, ad hoc, conservative adjustment just proposed and an exact, fully Bayesian analysis. One that we now describe is an approximate fully Bayesian method that uses a small bootstrap experiment and a delta-method correction adopted from Kass and Steffey's (1989) "first-order approximation". Kass and Steffey considered conditionally independent hierarchical models, which are also called empirical Bayes models, but their ideas apply directly to more general hierarchical Bayes models.

The Kass and and Steffey approximation is applied to penalized splines as follows. Let $f_i = f(x_i) = \mathbf{C}_i\boldsymbol{\nu}$. The posterior variance of f_i is calculated from the joint posterior distribution of $(\boldsymbol{\nu}, \boldsymbol{\lambda})$ and by a standard identity is

$$\mathrm{var}(f_i) = \mathsf{E}\{\mathrm{var}(f_i|\boldsymbol{\lambda})\} + \mathrm{var}\{\mathsf{E}(f_i|\boldsymbol{\lambda})\}.$$

Note that $\mathsf{E}\{\mathrm{var}(f_i|\boldsymbol{\lambda})\}$ is well approximated by the posterior variance of f_i when $\boldsymbol{\lambda}$ is treated as known and fixed at its posterior mode (Kass and Steffey 1989). Thus, $\mathrm{var}\{\mathsf{E}(f_i|\boldsymbol{\lambda})\}$ is the extra variability in posterior distribution of f_i that the approximate posterior variance given by (17.8) does not account for. We estimate $\mathrm{var}\{\mathsf{E}(f_i|\boldsymbol{\lambda})\}$ by the following three steps and add this estimate to the posterior variance given by (17.8).

(1) Use a parametric bootstrap to estimate $\mathrm{Cov}\{\log(\hat{\boldsymbol{\lambda}})\}$. Here the log function is applied elementwise to the vector $\boldsymbol{\lambda}$.
(2) Numerically differentiate $\mathbf{C}\hat{\boldsymbol{\nu}}(\boldsymbol{\lambda})$ with respect to $\log(\boldsymbol{\lambda})$ at $\boldsymbol{\lambda} = \hat{\boldsymbol{\lambda}}$. We use one-sided numerical derivatives with a step length of 0.1.
(3) Put the results from (1) and (2) into the delta-method formula:

$$\mathrm{var}\{\mathsf{E}(f_i|\boldsymbol{\lambda})\} \simeq \left\{\frac{\partial \mathbf{C}\hat{\boldsymbol{\nu}}(\hat{\boldsymbol{\lambda}})}{\partial \log(\boldsymbol{\lambda})}\right\}^{\mathsf{T}} \mathrm{Cov}\{\log(\hat{\boldsymbol{\lambda}})\} \left\{\frac{\partial \mathbf{C}\hat{\boldsymbol{\nu}}(\hat{\boldsymbol{\lambda}})}{\partial \log(\boldsymbol{\lambda})}\right\}. \tag{17.9}$$

When (17.9) is added to the approximate posterior variance from (17.8), we refer to the corresponding confidence intervals as *local penalty, corrected* intervals. Since the correction (17.9) is a relatively small portion of the corrected posterior variance, it need not be estimated by the bootstrap with as great a precision as when a variance is estimated entirely by a bootstrap. In our simulations, we used only 25 bootstrap samples in step (1).

In the simulations of the next section, the local penalty, conservative intervals are close to the more computationally intensive local penalty, corrected intervals. Since the latter have a theoretical justification, this closeness is some justification for the former.

17.5 Simulations

17.5.1 *Effects of the Tuning Parameters*

A Monte Carlo experiment was conducted to learn how the tuning parameters affect the accuracy of the local penalized spline. The regression function

$$f(x; j) = \sqrt{x(1-x)}\,\sin\left(\frac{2\pi(1+2^{(9-4j)/5})}{x+2^{(9-4j)/5}}\right) \tag{17.10}$$

was used, where j varied as a factor with levels 3, 4, 5, and 6. Larger values of j imply greater spatial heterogeneity. The sample size was 400, the x were all equally spaced on [0, 1], and σ_ε was 0.2. The three other factors besides j were tuning parameters: K with levels 20, 40, 80, and 120; M with levels 3, 4, 6, and 8; and N_{iter} with levels 1, 2, and 3. A full four-factor design was used with two replications for a total of 384 runs.

The response was log(MASE), where MASE is defined in Section 5.6.3. Because of interaction between j and the tuning parameters, quadratic response surfaces in the three tuning parameters were fit with j fixed at each of its four levels. It was found that for $j = 3$, 4, or 5, the tuning parameters had no appreciable effects on log(MASE). For $j = 6$, only the number of knots, K, had an effect on log(MASE). That effect is nonlinear: log(MASE) decreases rapidly as K increases up to about 80, but then log(MASE) levels off.

In summary, for the scenario we simulated, of three tuning parameters only K has a detectable effect on log(MASE). It is important that K be at least a certain minimum value depending on the regression function; however, after K is sufficiently large, further increases in K do not affect accuracy.

17.5.2 *The Automatic Algorithms*

The algorithms in Section 17.3 that choose all tuning parameters automatically were tested on simulated data. As previously mentioned, it is important that the number of knots, K, be sufficiently large that all significant features of the regression function can be modeled. Thus, the main function of the automatic algorithm is to ensure that K is sufficiently large. As reported in Section 17.5.1, the number of subknots and the number of iterations were not noticed to affect accuracy, but in our proposed algorithm we allowed them to increases slightly with K.

There were three simulations that differed in the regression function and the sample size. All three simulations used 300 simulated data sets and a standard deviation of all the ε equal to 0.2. The regression function was always of form (17.10) but with different values of j. The estimators were compared by MASE.

The first simulation used $j = 3$ and $n = 150$. Panel (a) of Figure 17.1 shows a typical data set and the true regression function. Recall that the algorithm can choose as the final value of the tuning parameters (K, M, N_{iter}) one of the vectors $(10, 2, 1)$, $(20, 3, 2)$, $(40, 4, 2)$, $(80, 6, 2)$, or $(120, 6, 2)$. We coded these tuning parameter sets as 1, 2, 3, 4, and 5. Panel (b) shows the MASE for each fixed tuning parameter set and for the myopic and full-search algorithms. Panel (c) gives histograms of the tuning parameter sets chosen by the myopic (left) and full-search (right) algorithms. Finally, panel (d) plots the ASE for tuning parameter set 5 versus ASE for tuning parameter set 1.

In this example, MASE increases with the tuning parameter set number, so that larger values of K, M, and N_{iter} lead to worse estimates. The myopic algorithm chooses the best (i.e., the first) tuning parameter set over two thirds of the time and performs better than the full-search algorithm. In panel (d) one sees that the difference in ASE between the best and the worst tuning parameter sets is quite small for over 90% of the data sets. But there are about 20 out of 300 data sets where the fifth tuning parameter set has a substantially higher ASE than the first tuning parameter set. This behavior explains why one often finds that, for a given data set, the tuning parameters have little effect on the fit – provided that they exceed a certain minimal value – yet MASE still depends on the tuning parameters even when they are above this threshold.

The second simulation used $j = 3$ and $n = 400$. See Figure 17.2. The results are similar to those of the first simulation, except that now the myopic algorithm chooses the first tuning parameter set nearly 100% of the time and has a MASE value nearly as small as the best of the fixed tuning parameter set estimators.

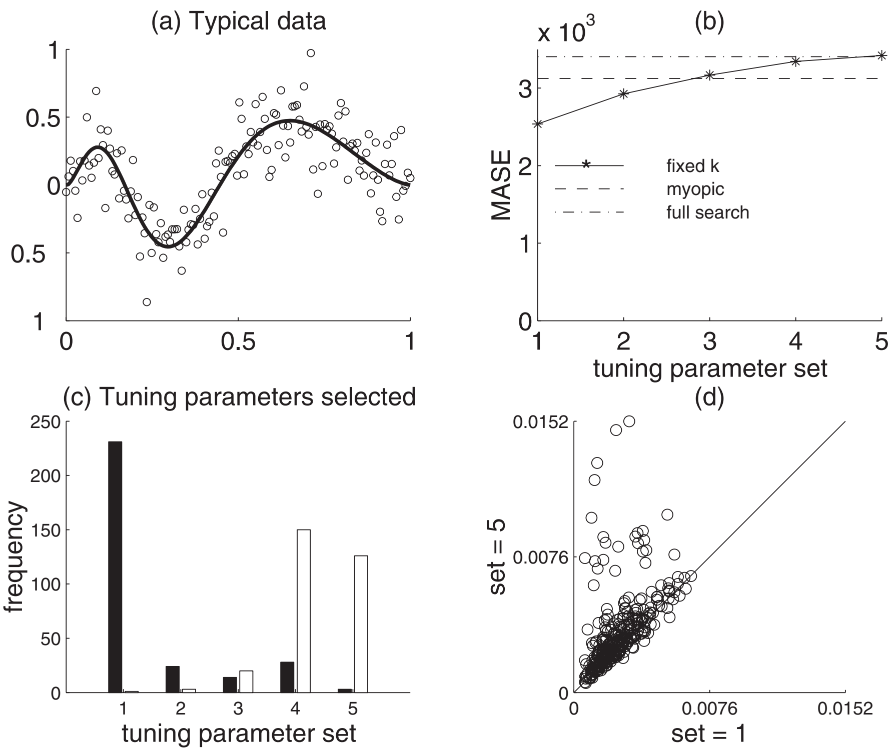

Figure 17.1 Study of the automatic algorithm for choosing the tuning parameters with $n = 150$ and $j = 3$ so that there is low spatial variability. (a) Typical data set and true regression function. (b) MASE of estimates using each of the five sets of fixed tuning parameter values and for the myopic and full-search algorithms. (c) Histograms of the tuning parameter sets chosen by the myopic (left) and full-search (right) algorithm. (d) ASE for tuning parameter set 5 versus tuning parameter set 1.

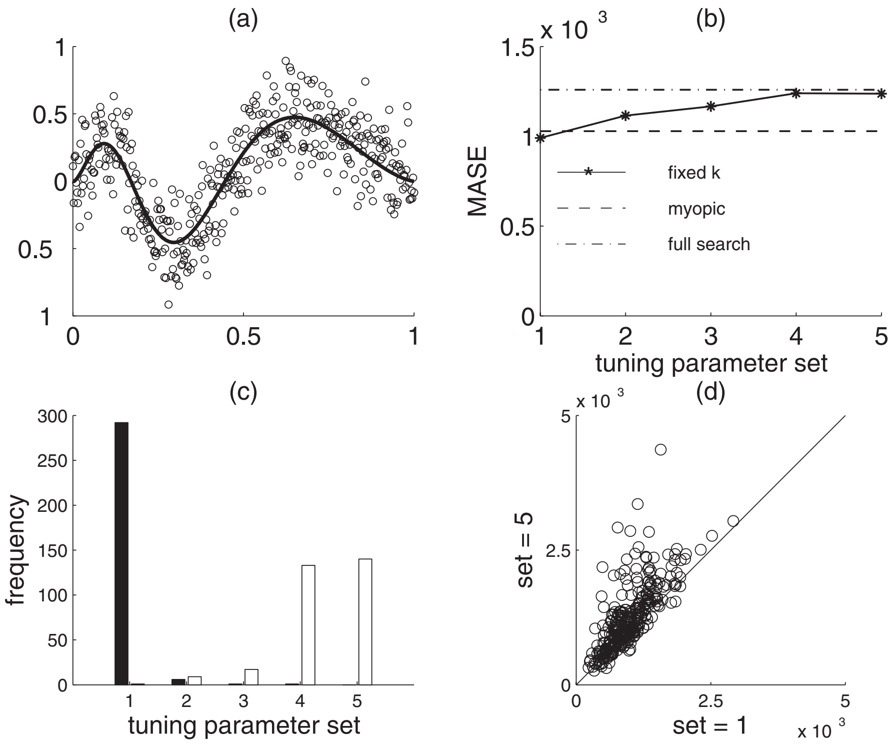

Figure 17.2 Study of the automatic algorithm for choosing the tuning parameters with $n = 400$ and $j = 3$ so that there is low spatial variability; (a)–(d) as in Figure 17.1.

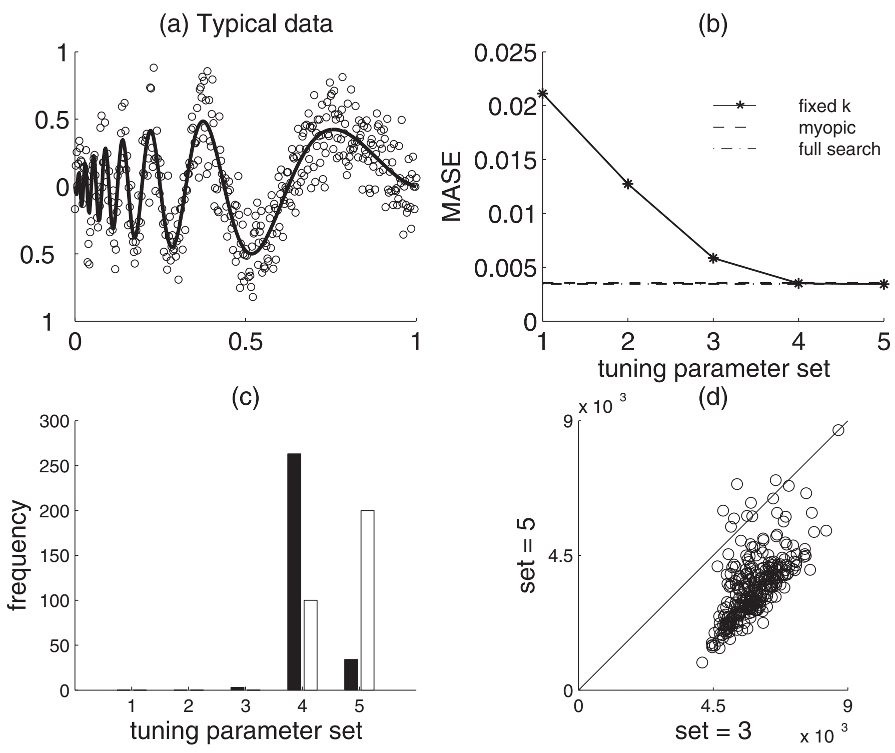

Figure 17.3 Study of the automatic algorithm for choosing the tuning parameters with $n = 400$ and $j = 6$ so that there is substantial spatial variability; (a)–(c) as in Figure 17.1. (d) ASE for tuning parameter set 5 versus tuning parameter set 3.

The third simulation (Figure 17.3) used $n = 400$ and $j = 6$. With this value of j there is serious spatial heterogeneity. The MASE is rather large unless the fourth or fifth tuning parameter set is used. Both the myopic and full-search algorithms choose either the fourth or fifth tuning parameter set in nearly all samples. The bias for tuning parameter sets 1 and 2 is so large that in panel (d) the comparison is between sets 3 and 5, not sets 1 and 5 as in the previous figures. The plot in panel (d) is similar to those in the previous figures except shifted to the right owing to the bias with parameter set 3.

We conclude that the automatic algorithms can supply reasonable, but not optimal, values of the tuning parameters when the user has little idea how to choose them. In particular, for complex functions such as (17.10), both of the automatic algorithms choose K and M large enough to obtain good estimates. However, for less complex functions such as (17.10) with $j = 3$, the MASE is somewhat reduced by choosing K and M small, but the automatic algorithms are not likely to achieve this reduction in MASE.

The myopic algorithm performed better than the full-search algorithm for the examples studied here, but we have seen in Section 5.6.3 that myopic algorithms can stop prematurely if the regression function is particularly nasty. The myopic algorithm can be used provided one exercises caution and checks whether it might have stopped prematurely. The full-search and myopic algorithms can easily be computed together and, in fact, our MATLAB routine does this. It is useful to compute both and to check whether they give vastly different estimates.

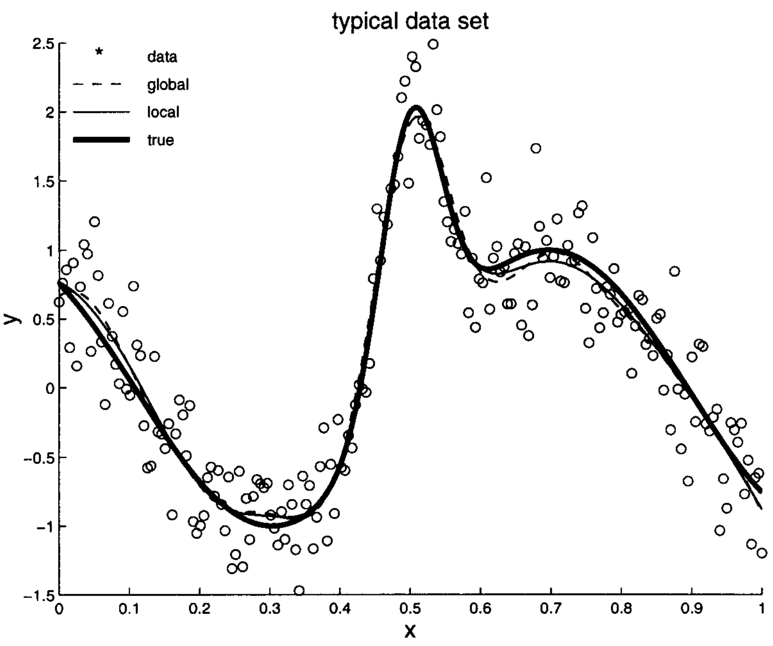

Figure 17.4 Typical data in the Bayesian inference study: local and global penalty splines.

If they do, that is evidence that the myopic algorithm may have stopped prematurely. If the myopic and full-search estimates are similar, we recommend using the myopic algorithm because it is likely to be slightly closer to the true regression function.

17.5.3 *Bayesian Inference*

To compare posterior distribution with and without a local penalty, we used a spatially heterogeneous regression function

$$f(x) = \sin\{8(x - 0.5)\} + 2\exp\{-16^2(x - 0.5)^2\}. \qquad (17.11)$$

The x_i were equally spaced on [0, 1], the sample size was $n = 200$, and the ε_i were normally distributed with $\sigma_\varepsilon = 0.3$. We used quadratic splines with $K = 40$ knots, the number of subknots was $M = 5$, and the number of iterations to minimize GCV using the local penalty was $N_{\text{iter}} = 1$.

Figure 17.4 shows a typical data set and the global and local penalty estimates. The global and local penalty estimate are difficult to distinguish visually. In cases with more extreme spatial heterogeneity, this is not true; see an example in Ruppert and Carroll (2000, p. 215), where the global penalty estimate is noticeably undersmoothed but where the local penalty estimate is much smoother in a region where f is nearly constant.

Figure 17.5 shows the pointwise MSE and squared bias of the global penalty estimator calculated from 300 Monte Carlo samples. Also shown is the pointwise posterior variance given by (17.8) averaged over the 300 repetitions. The posterior variance should be estimating the MSE. We see that the posterior variance is

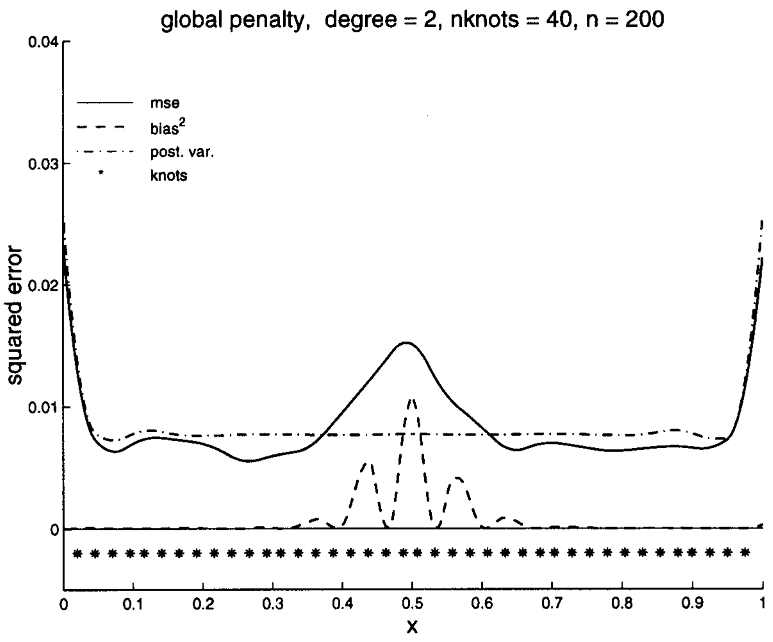

Figure 17.5 Bayesian inference study: behavior of the global penalty estimator. Plots of pointwise MSE, squared bias, and average (over Monte Carlo trials) posterior variance. The MSE and the posterior variance have been smoothed to reduce Monte Carlo variance. The posterior variance assumes that λ is known, so the variability in $\hat{\lambda}$ is not taken into account.

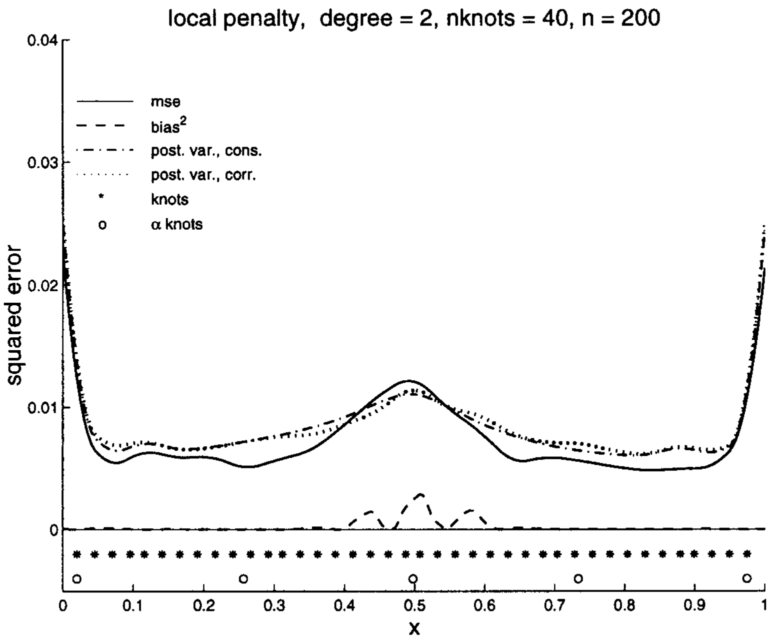

Figure 17.6 Bayesian inference study: behavior of the local penalty estimator. Plots of pointwise MSE, squared bias, and average (over Monte Carlo trials) posterior variance. The MSE and the posterior variance have been smoothed to reduce Monte Carlo variance.

constant, except for boundary effects, and cannot detect the spatial heterogeneity in the MSE.

Figure 17.6 is similar to Figure 17.5 but is for the local penalty estimator. Two posterior variances are shown, the conservative adjustment and the Kass–Steffey correction. One can see that the MSE is somewhat different than in Figure 17.5

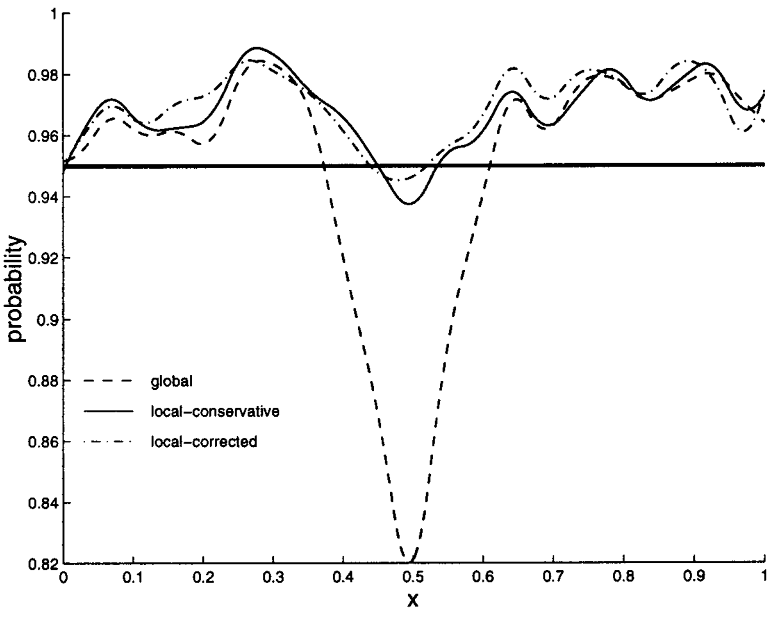

Figure 17.7 Bayesian inference study using function (17.11). Pointwise coverage probabilities of 95% Bayesian confidence intervals for $f(x_i)$ using global and local penalties. The probabilities have been smoothed to remove Monte Carlo variability. The local penalty intervals use the conservative adjustment to the posterior variance and the Kass–Steffey correction.

since the estimator reduces bias by adapting to spatial heterogeneity. Also, the posterior variance tracks the MSE better than for the global penalty estimator.

In Figure 17.7 we present the Monte Carlo estimates of the pointwise coverage probabilities of nominal 95% Bayesian confidence intervals based on the global and local penalty estimators. These coverage probabilities have been smoothed by penalized splines to remove some of the Monte Carlo variability. All three confidence interval procedures achieve pointwise coverage probabilities close to 95%. Because the local penalty methods are somewhat conservative, the global penalty method is, on average, the closest to 95%, but the local penalty methods avoid low coverage probabilities around features in f.

17.6 LIDAR Example

As mentioned in Section 6.8.3, an interesting feature of the LIDAR example is that there is more scientific interest in the first derivative (f') than in f itself because $-f'(x)$ is proportional to concentration at range x; see Ruppert et al. (1997) for further discussion. For the estimation of f, a global penalty works satisfactorily. Visually, the local penalty estimate of f is difficult to distinguish from the global penalty fit.

However, for the estimation of f', a local penalty appears to improve upon a global penalty. Figure 6.13 (p. 155) shows the derivatives (times −1) of fitted splines and their confidence intervals using global and local penalties. Notice that the confidence intervals using the local penalty are generally narrower than for the global penalty – except at the peak, where the extra width should be reflecting real uncertainty. The local penalty estimate has a sharper peak and less noise in the flat areas.

17.7 Additive Models

17.7.1 *An Algorithm for Additive Models*

Spatially adaptive penalties can be easily extended to additive models. Suppose we have d predictor variables and that $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})^\mathsf{T}$ is the vector of predictor variables for the ith case. The additive model is

$$y_i = \beta_0 + \sum_{j=1}^{d} f_j(x_{i,j}) + \varepsilon_i .$$

Let the jth predictor variable have K_j knots, $\kappa_{1,j}, \ldots, \kappa_{K_j,j}$. Then the additive spline model is

$$f(\mathbf{x}, \boldsymbol{\nu}) = \beta_0 + \sum_{j=1}^{d}\left(\beta_{1,j}x_j + \cdots + \beta_{p,j}x_j^p + \sum_{k=1}^{K_j} u_{k,j}(x_j - \kappa_{k,j})_+^p\right).$$

The parameter vector is $\boldsymbol{\nu} = [\beta_0, \beta_{1,1}, \ldots, u_{K_1,1}, \ldots, u_{K_d,d}]^\mathsf{T}$. Let $\lambda_j(\cdot)$ be the penalty function for the jth predictor. Then the penalized criterion to minimize is

$$\sum_{i=1}^{n}\{y_i - f(\mathbf{x}_i; \boldsymbol{\nu})\}^2 + \sum_{j=1}^{d} \lambda_j^2(\kappa_{k,j})u_{k,j}^2 .$$

Consider three levels of complexity of the penalty:

(1) $\lambda_j(\cdot) \equiv \lambda$ (a common global penalty);
(2) $\lambda_j(\cdot) \equiv \lambda_j$ (separate global penalties);
(3) $\lambda_j(\cdot)$ is a linear spline (separate local penalties).

The following algorithm allows one to fit separate local penalties using only 1-dimensional grid searches for minimizing GCV. First minimize GCV using a common global penalty. For this penalty to be reasonable, one should standardize the predictors so that they have common standard deviations. Then, using the common global penalty as a starting value, minimize GCV over separate global penalties. The d global penalty parameters are varied one at a time during minimization, with the rationale that the optimal value of λ_j depends only slightly on the $\lambda_{j'}$, $j' \neq j$. Finally, using separate global penalties as starting values, minimize GCV over separate local penalties. The jth local penalty has M_j parameters, so there are a total of $M_1 + \cdots + M_d$ penalty parameters. These parameters are then varied in succession to minimize GCV.

17.7.2 *Simulations of an Additive Model*

To evaluate the practicality of this algorithm, we used an example in which we added two spatially homogeneous component functions to a spatially heterogeneous function. Thus, there were three predictor variables, which for each case were independently distributed as Uniform(0, 1) random variables. The components of f were spatially homogeneous, $f_1(x_1) = \sin(4\pi x_1)$ and $f_2(x_2) = x_2^3$, and f_3 was the spatially heterogeneous function

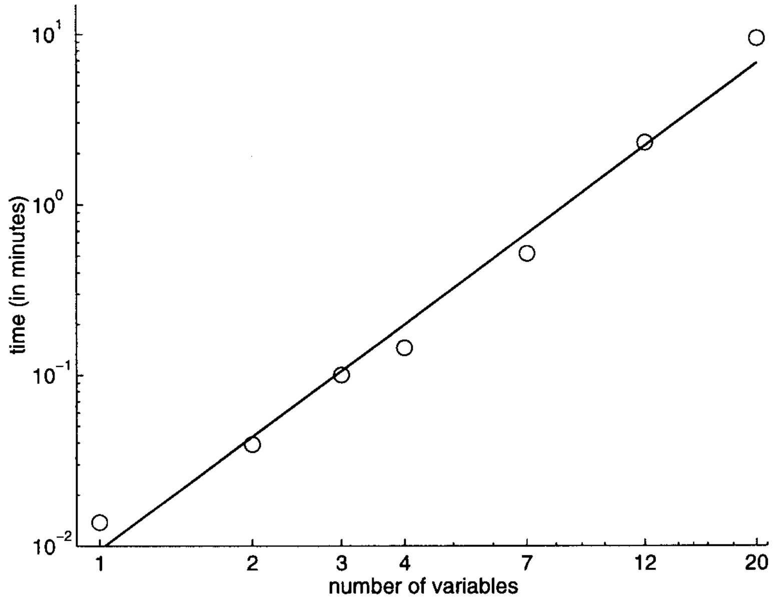

Figure 17.8
Log-log plot of the computation time for fitting an additive model with local penalties as a function of the number of variables, d. A linear fit (slope = 2.45) is also shown.

$$f_3(x_3) = \exp\{-400(x_3 - 0.6)^2\} + \tfrac{5}{3}\exp\{-500(x_3 - 0.75)^2\} + 2\exp\{-500(x_3 - 0.9)^2\}.$$

As in Section 17.5.3, $n = 300$ and the ε were independent N(0, 0.25). We used quadratic splines and 10, 10, and 40 knots for f_1, f_2, and f_3, respectively. The local penalty estimate had four subknots for all three functions.

First consider computation time. For a single data set and using our MATLAB program on a SUN Ultra 1 computer, the common global penalty estimate took 2.1 seconds to compute, the separate global penalty estimate took an additional 1.5 seconds, and the separate local penalties estimate took an additional 10.4 seconds. Thus, local penalties are more computationally intensive than global penalties but still feasible for small values of the number of predictor variables d – for example, for $d = 3$ here.

Now consider larger values of d. Everything else held constant, the number of parameters of an additive model grows linearly in d and, since matrix inversion time is cubic in dimension, the time for a single fit should grow cubically in d. Since the number of fits needed for the sequential grid searching just described will grow linearly in d, the total computation time for local penalties should be roughly proportional to d^4. To test this rough calculation empirically, we found the computation time for fitting additive models with 300 data points, 10 knots per variable, and 4 subknots per variable. The number of predictor variables d took seven values from 1 to 20. Figure 17.8 is a log-log plot of computation time versus d. A linear fit on the log scale is also shown; its slope is 2.2, not 4 as the quartic model predicts. The actual data show log-times that are nonlinear in $\log(d)$ with an increasing slope. Thus, a quartic model of time as a function of d may work for large values of d, but a quadratic or cubic model would be better for d in the "usual" range of 1 to 20. A likely reason that the quartic model doesn't fit well for smaller d is that it ignores parts of the computation that are linear, quadratic, and cubic in d. The computation time for 7 variables is about 0.5 minutes but for 20 variables is about 9.4 minutes. It seems clear that local additive fitting is feasible up to at least 8–10 variables and perhaps 20 variables, but it is only "interactive" up to 4 variables.

An important point to keep in mind is that computation times are largely independent of the sample size n. The reason for this is that once $\mathbf{C}^\mathsf{T}\mathbf{C}$ and $\mathbf{C}^\mathsf{T}\mathbf{y}$ have been computed, all computation times are independent of n, and the computation of $\mathbf{C}^\mathsf{T}\mathbf{C}$ and $\mathbf{C}^\mathsf{T}\mathbf{y}$ is quite fast unless n is enormous.

Now consider statistical efficiency. The MSEs that were computed over 500 Monte Carlo samples for the separate local penalties estimator were 0.010, 0.0046, and 0.0165 for f_1, f_2, and f_3, respectively. Thus, f_2 is relatively easy to estimate and f_3 is slightly more difficult to estimate than f_1. Ratios of the MSE for common global penalties to separate local penalties were 1.26, 2.36, and 1.23 for f_1, f_2, and f_3, respectively, whereas ratios of the MSE for separate global penalties to separate local penalties were 0.85, 0.88, and 1.20 for f_1, f_2, and f_3, respectively. Thus, for all three component functions, the common global penalty estimator with a single smoothing parameter is less efficient than the fully adaptive estimator with separate local penalties. For the spatially homogeneous functions f_1 and f_2, there is some loss of efficiency when using local penalties rather than separate global penalties, but the spatially heterogeneous f_3 is best estimated by separate local penalties. These results are somewhat different than what we found for univariate regression, where no efficiency loss was noticed when a local penalty was used even though a global penalty would have been adequate. There may be practical situations where one knows that a certain component function is spatially heterogeneous but the other component functions are not. Then greater efficiency should be achievable by using global penalties for the spatially homogeneous component functions and local penalties for the spatially heterogeneous ones.

The results in this section provide evidence that sequential 1-dimensional grid searches to find the smoothing parameter vector are effective. The reason for this is that the optimal value of one tuning parameter depends only weakly upon the other tuning parameters. The result is that searches over a rather large number of tuning parameters (up to 60 when $d = 15$ and there are four subknots per variable) do appear to be feasible.

17.8 Bibliographical Notes

The spatially adaptive local penalty estimators were introduced by Ruppert and Carroll (2000). Other methods for spatially adaptive smoothing include automatic knot selection, local polynomial regression with local bandwidths (e.g. Ruppert 1997b), and wavelets. The automatic knot selection literature is discussed in Section 3.4. For wavelets, see Cai (1999) and the bibliographic notes at the end of Chapter 3.