Chapter 4

Regression Model Results

4.1 Introduction

In this chapter, I describe the Bayesian Gaussian process (GP) nonparametric regression model in detail. I discuss the model implementation via Markov chain Monte Carlo (MCMC). I then evaluate the nonstationary GP model by comparing it to competing Bayesian nonparametric methods in the literature as well as to a stationary GP nonparametric regression model. The competing methods assessed here use free-knot splines to estimate the regression function. I compare the methods on datasets whose covariates range from one- to three-dimensional and for which a normal likelihood is assumed. Some of the datasets are simulated, while others are real. I compare the methods using mean squared error to evaluate point predictions and predictive density calculations to evaluate overall fit. I also fit the model to a dataset for which a binomial likelihood is appropriate, assessing the success of the posterior mean centering sampling scheme on non-Gaussian data.

4.2 Model Structure

The simplest form of the nonstationary GP nonparametric regression model (see Figure 4.1 for a directed acyclic graph of the model) is based on a normal likelihood with a nonstationary GP prior for the regression function,

$$Y_i \sim \mathrm{N}(f(\boldsymbol{x_i}), \eta^2)$$

$$f(\cdot) \quad \sim \quad \text{GP}\left(\mu_f, \sigma_f^2 R_f^{NS}(\cdot,\cdot)\right),$$

where $R_f^{NS}(\cdot,\cdot)$ is the Matérn-based nonstationary correlation function defined by

$$R_f^{NS}(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{2^{\frac{P}{2}} \mid \Sigma_i \mid^{\frac{1}{4}} \mid \Sigma_j \mid^{\frac{1}{4}}}{\mid \Sigma_i + \Sigma_j \mid^{\frac{1}{2}}} \frac{1}{\Gamma(\nu_f) 2^{\nu_f - 1}} (2\sqrt{\nu_f Q_{ij}})^{\nu} K_{\nu_f}(2\sqrt{\nu_f Q_{ij}}).$$

ν_f is the smoothness parameter of the Matérn function, and Q_{ij} is the quadratic form,

$$Q_{ij} = (\boldsymbol{x_i} - \boldsymbol{x_j})^T \left(\frac{\Sigma_i + \Sigma_j}{2}\right)^{-1} (\boldsymbol{x_i} - \boldsymbol{x_j}).$$

For the smoothness parameter, ν_f, I place a uniform prior on $(0.5, 30)$, which allows the smoothness to vary between non-differentiable (0.5) and very smooth. Recall that as $\nu_f \to \infty$, one recovers the original Higdon kernel-based nonstationary covariance function. The results in this chapter suggest that the data do not contain much information about this parameter, apart from some indication that smoothness approaching the exponential correlation function is less likely than more smooth functions. I use vague but proper Gaussian priors: $\mu_f \sim \text{N}(0, 10^2)$, $\log \sigma_f \sim \text{N}(0, 9^2)$, and $\log \eta \sim \text{N}(0, 9^2)$. The kernel matrices, Σ_i, are modelled using the eigendecomposition described in Section 3.2.3. Each location (training or test), $\boldsymbol{x_i}$, has a Gaussian kernel with mean $\boldsymbol{x_i}$ and covariance matrix, Σ_i, whose eigendecomposition is

$$\Sigma_i = \Gamma(\gamma_1(\boldsymbol{x_i}), \ldots, \gamma_Q(\boldsymbol{x_i})) D(\lambda_1(\boldsymbol{x_i}), \ldots, \lambda_P(\boldsymbol{x_i})) \Gamma(\gamma_1(\boldsymbol{x_i}), \ldots, \gamma_Q(\boldsymbol{x_i}))^T,$$

where D is a diagonal matrix of eigenvalues and Γ is an eigenvector matrix constructed as described in Section 3.2.3.2. Using that parameterization for the eigenvector matrix, there are $Q = \frac{P(P-1)}{2} + P - 1$ spatial processes that determine the eigenvector matrices. $\gamma_q(\cdot), q = 1, \ldots, Q$, and $\lambda_p(\cdot), p = 1, \ldots, P$, are spatial processes, since implicitly there are eigenvalues and eigenvectors at all points in the covariate space. I will refer to these as the eigenvalue and eigenvector processes, or to them collectively as the eigenprocesses.

The next level of the hierarchy gives the prior distribution for the kernel matrices in terms of the eigenprocesses, $\phi(\cdot) \in \{\log(\lambda_1(\cdot)), \ldots, \log(\lambda_P(\cdot)), \gamma_1(\cdot), \ldots, \gamma_Q(\cdot)\}$. In order for the kernels to vary smoothly in covariate space, I model the eigenvalues and eigenvectors using Gaussian process priors as well, but use stationary covariance functions for simplicity. In one dimension, I need only parameterize one eigenvalue process. I do this by taking the prior for the eigenvalue process to be

$$\log(\lambda(\cdot)) \sim \text{GP}(\mu_\lambda, \sigma_\lambda^2 R_\lambda^S(\cdot)).$$

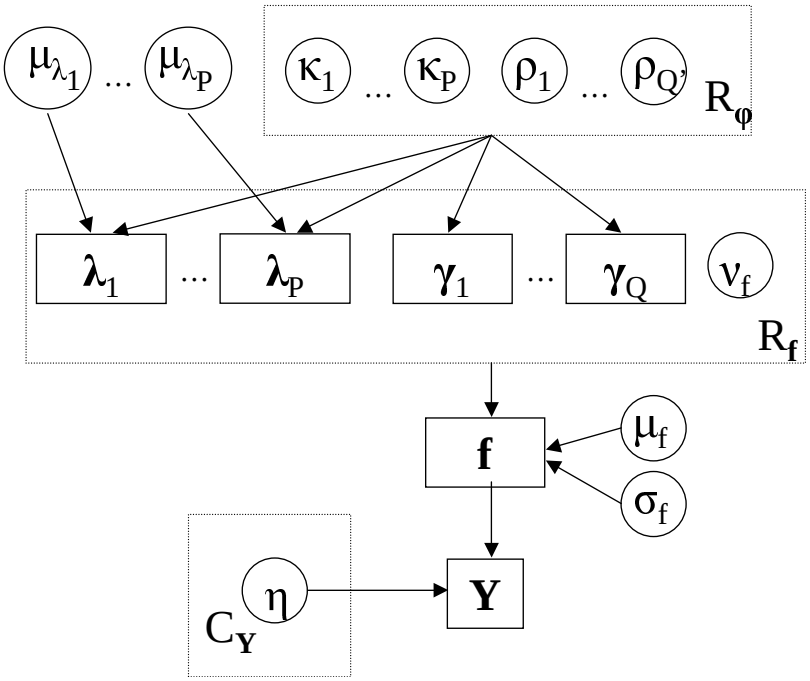

Figure 4.1. Directed acyclic graph for the normal likelihood nonstationary Gaussian process regression model. Bold letters indicate vectors.

In higher dimensions, it is difficult to model the eigenvectors simply and effectively. In part, this is simply the curse of dimensionality; the number of eigenvalue processes is the same as the dimension of the covariate space, P, while the number of eigenvector processes grows more rapidly, with a minimum of $\frac{P(P-1)}{2}$ processes (if one were able to use the Givens angle parameterization). For each of the eigenprocesses, the number of hyperparameters grows with P as well. I focus on the simpler parameterization described in detail in Section 3.2.3.3. I model each of the eigenprocesses with a stationary GP prior,

$$\phi(\cdot) \sim \text{GP}(\mu_\phi, \sigma_\phi^2 R_\phi^S(\cdot)).$$

$R_\phi^S(\tau_{ij})$ is a Matérn stationary correlation function, common to all the processes and defined by the Mahalanobis distance,

$$\tau_{ij} = \sqrt{(\boldsymbol{x_i} - \boldsymbol{x_j})\Sigma_\phi(\boldsymbol{x_i} - \boldsymbol{x_j})},$$

where

$$\Sigma_{\boldsymbol{\phi}} = \Gamma(\rho_1, \ldots, \rho_{Q'})D(\kappa_1^2, \ldots, \kappa_P^2)\Gamma(\rho_1, \ldots, \rho_{Q'})^T,$$

Γ is an eigenvector matrix parameterized by the $Q' = \frac{P(P-1)}{2}$ Givens angles, $\rho_q, q = 1, \ldots, Q'$, and D is a diagonal matrix of P eigenvalues, $\kappa_p^2, p = 1, \ldots, P$. In addition to the $P + \frac{P(P-1)}{2}$ hyperparameters for this shared stationary correlation, there are separate hyperparameters, μ_ϕ and σ_ϕ, for each process. I fix the smoothness parameter of the eigenprocesses, $\nu_\phi = 30$, so that the processes are very smooth. Following the results in Section 2.5.5, this ensures that the smoothness of the regression function will be determined by ν_f and not influenced by the smoothness of the kernels. I choose not to let ν_ϕ vary because this parameter should have minimal impact on the estimate for the regression function, and the data are likely to only weakly, if at all, inform the parameter. The priors for the Givens angles follow the development in Section 3.2.3.1.

To simplify the prior specification, I assume without loss of generality, $\boldsymbol{x_i} \in [0,1]^P$, the P-dimensional unit cube. For each eigenvalue process, $\log \lambda_p(\cdot)$, I let μ_{λ_p} be a free parameter, with a vague but proper Gaussian prior, $\mathrm{N}(-4, 2.5^2)$, informed by my knowledge of the reasonable range of values for the size of the kernels. I fix $\sigma^2_{\lambda_p} = 3^2$ (2.5^2 for one-dimensional covariates), because the coarseness and range of the covariates determine a reasonable range for the size of the eigenvalues, since it is the eigenvalues that determine the size of the kernel matrices and therefore the correlation scale. For the eigenvector processes, $\gamma_q(\cdot)$, I take $\mu_{\gamma_q} = 0$ and $\sigma_{\gamma_q} = 1$ without loss of generality, because these processes are used only to determine the directions of the eigenvectors. The κ_p values parameterize the scale of the correlation. I use a Beta$(1, 3)$ prior,

$$\Pi(\log(\kappa_p)) \propto (1 - g(\log(\kappa_p)))^2,$$

where the function $g(\cdot)$ maps the range I chose for $\log \kappa_p$, (-3.5,1.5), to (0,1). A uniform prior would favor small values of κ_p and unsmooth eigenprocesses, which I do not think is reasonable, nor necessary, to model the data. The upper limit on $\log \kappa_p$ prevents κ_p from wandering into a part of the parameter space where large changes in κ_p have little effect on the correlation of $\phi(\cdot)$, while the lower limit is set based on not wanting the correlation scale of the eigenprocesses to become smaller than the distances between covariates. I force $\lambda_p(\boldsymbol{x_i}) < 9P$ to avoid having the chain wander off to parts of the parameter space in which the eigenvalues are very large and from which it would take a long time to return because of the flatness of the likelihood in that region. This limit varies with P because as the number of covariates increases, in order for the model to ignore a covariate, it is necessary for the size of the kernels to become increasingly large in that direction of

covariate space. However, this limit still prevents the model from completely ignoring a covariate, as discussed in Section 3.5.1.3.

The nonstationary covariance is non-negative and decays with distance, but can decay at a different rate in different parts of the covariate space, allowing the degree of smoothing to vary. For most applications, the restriction to non-negative correlation is sensible and probably desirable, but perhaps not for some functions, such as oscillating functions.

To assess the effectiveness of the nonstationary correlation model relative to the simpler stationary alternative, I also implement the Gaussian process-based nonparametric regression method using a stationary correlation model, replacing $R_f^{NS}(\cdot,\cdot)$ with $R_f^S(\cdot)$, where the latter is of the same form as $R_\phi^S(\cdot)$ in the nonstationary model. This stationary GP prior for the function takes the same form as the stationary GP priors for the eigenprocesses described above. However, I let ν_f in the stationary Matérn correlation vary in the same way that ν_f varies in the nonstationary model.

4.3 MCMC Sampling Scheme

The sampling scheme for the regression model is built on the proposals discussed in Section 3.6.2.2. In particular, any time I propose a hyperparameter of $\boldsymbol{f}$ or new process values for any of the eigenprocesses, $\boldsymbol{\phi}$, I also propose $\boldsymbol{f}^*$conditional on the proposed hyperparameter or eigenprocess values. Here I describe the proposal scheme step by step, starting at the bottom of the model hierarchy (Figure 4.1). In all cases, v indicates a user-tuneable proposal standard deviation that differs between parameters and $\boldsymbol{\omega}_\phi = (\sigma_\phi L_{\boldsymbol{\phi}})^{-1}(\boldsymbol{\phi} - \mu_\phi)$, where $\phi(\cdot)$ is an eigenprocess and $L_{\boldsymbol{\phi}}$ is the Cholesky factor of $R_{\boldsymbol{\phi}}^S$.

First let me describe two prototype steps that are used in the sampling scheme. The first prototype step, S1, is the basic posterior mean centering proposal involving $\boldsymbol{f}$.

1. Propose either μ_f, σ_f, or ν_f using a Metropolis-Hastings proposal or propose a single process vector, $\boldsymbol{\phi}$, or propose all of the eigenprocess vectors simultaneously, as will be described below. In the notation that follows, I will indicate that all of the potential parameters have been proposed, but this is merely for notational convenience.

2. Propose $\boldsymbol{f}$ conditionally on $\boldsymbol{\theta}^* = \{\mu_f, \sigma_f, \nu_f, \boldsymbol{\phi}\}$ as

$$\boldsymbol{f}^* \mid \boldsymbol{\theta}^* \sim \mathrm{N}(\widetilde{\boldsymbol{f}(\boldsymbol{\theta}^*)} + \sigma_f^* L_{\boldsymbol{f}}^* \boldsymbol{\chi}, v^2 R_{\boldsymbol{f}}^*),$$

where $\boldsymbol{\chi} = (\sigma L_{\boldsymbol{f}})^{-1}(\boldsymbol{f} - \widetilde{\boldsymbol{f}(\boldsymbol{\theta})})$, $\widetilde{\boldsymbol{f}(\boldsymbol{\theta})}$ is the posterior mean of $\boldsymbol{f}$ conditional on the current parameter values, and $\widetilde{\boldsymbol{f}(\boldsymbol{\theta}^*)}$ is the posterior mean conditional on the proposed values,

$$\begin{aligned} \widetilde{\boldsymbol{f}(\boldsymbol{\theta})} &= \sigma_f^2 R_{\boldsymbol{f}} \left(\sigma_f^2 R_{\boldsymbol{f}} + C_{\boldsymbol{Y}}\right)^{-1} \boldsymbol{y} + C_{\boldsymbol{Y}} \left(\sigma_f^2 R_{\boldsymbol{f}} + C_{\boldsymbol{Y}}\right)^{-1} \mu_f \\ &= \mu_f + \sigma_f^2 R_{\boldsymbol{f}} \left(\sigma_f^2 R_{\boldsymbol{f}} + C_{\boldsymbol{Y}}\right)^{-1} (\boldsymbol{y} - \mu_f). \end{aligned}$$

Note that $C_{\boldsymbol{Y}}$ is the diagonal residual covariance matrix. Note again that, as mentioned in Chapter 3, it is allowable to set $v = 0$, so long as one includes the Jacobian of the deterministic mapping, $\boldsymbol{f} \to \boldsymbol{f}^*$, which is the same as the Hastings ratio one uses if $v > 0$.

3. The Hastings ratio for the proposal is a ratio of determinants, which cancels the determinant ratio from the prior for $\boldsymbol{f}$ as described in Section 3.6.2.2.

The second prototype step, S2, is a joint proposal for an eigenprocess hyperparameter with the values of the process. Since I do not know the posterior mean of $\boldsymbol{\phi}$ conditional on the other parameters, I am forced to make a joint proposal that takes account only of the prior correlation of the process and not the posterior correlation.

1. This proposal applies to $\theta \in \{\mu_{\lambda_1}, \ldots, \mu_{\lambda_P}, \log\kappa_1, \ldots, \log\kappa_P, \rho_1, \ldots, \rho_{Q'}\}$. Propose θ using a Metropolis-Hastings proposal (for $\log\kappa_p$ I change the proposal variance as a function of the current value, requiring a Hastings correction).

2. If $\theta = \mu_{\lambda_p}$, I next propose only $\boldsymbol{\lambda}_p$. If θ is any of the hyperparameters involved in Σ_ϕ in the stationary correlation matrix $R_{\boldsymbol{\phi}}^S$, propose all of the eigenprocesses. For an individual eigenprocess, take

$$\boldsymbol{\phi}^* \mid \theta^* \sim \mathrm{N}(\mu_\phi^* + \sigma_\phi L_{\boldsymbol{\phi}}^* \boldsymbol{\omega}_\phi, v^2 R_{\boldsymbol{\phi}}^*).$$

Note again that, as mentioned in Chapter 3, it is allowable to set $v = 0$, so long as one includes the Jacobian of the deterministic mapping, $\boldsymbol{\phi} \to \boldsymbol{\phi}^*$, which is the same as the Hastings ratio one uses if $v > 0$.

3. The Hastings ratio for this proposal is a ratio of determinants, which cancels the determinant ratio in the prior for $\boldsymbol{\phi}$.

Now let's consider the steps in the proposal scheme for the regression model, making use of prototype steps S1 and S2 as necessary.

1. Sample $\log(\eta)$ using a simple Metropolis step.

2. Separately, for each of the pairs, $(\nu_f, \boldsymbol{f})$, $(\mu_f, \boldsymbol{f})$, and $(\sigma_f, \boldsymbol{f})$, sample the pair jointly using a proposal of type S1. For ν_f I increase the proposal variance when ν_f is large, so the proposal requires a Hastings correction to the acceptance ratio.

3. Separately, for each of the eigenprocesses, propose $(\boldsymbol{\phi}, \boldsymbol{f})$ using a joint proposal of type S1. The marginal proposal for $\boldsymbol{\phi}$ is
$$\boldsymbol{\phi}^* \sim \mathrm{N}(\boldsymbol{\phi}, v^2 R_{\boldsymbol{\phi}}).$$

4. Separately, for each hyperparameter involved in the eigenprocesses, propose the hyperparameter using a proposal of type S2. As part of the same proposal, propose $\boldsymbol{f}$ conditionally on the new eigenprocess(es) values using a proposal of type S1.

5. Propose $\boldsymbol{f}$ using a simple Metropolis step with correlation amongst the elements of $\boldsymbol{f}$: $\boldsymbol{f}^* \sim \mathrm{N}(\boldsymbol{f}, v^2 R_{\boldsymbol{f}})$. It is also straightforward to do a Langevin update here, however I did not use such an update in the model runs reported in this thesis.

With a Gaussian likelihood, I can of course integrate $\boldsymbol{f}$ out of the model, which simplifies the scheme above and avoids using the PMC proposal scheme. In the development of the model, I was interested in being able to extend the model to non-Gaussian data, so I have sampled from the model without integrating the function out. To sample from a model for non-Gaussian data, the steps are as above, but with the modifications given in Section 3.6.2.3.

The sampling scheme for the stationary GP prior model is a simplified version of this sampling scheme, with proposals of type S1 for $(\log \kappa_p, \boldsymbol{f}), p = 1, \ldots, P$ and $(\gamma_q, \boldsymbol{f}), q = 1, \ldots, Q'$.

Since the function at test locations is conditionally independent of the observations given the function at training locations, I can generate samples of the function at test locations based on the

usual conditional normal calculation,

$$\boldsymbol{f_2} \mid \boldsymbol{f_1} \sim \mathrm{N}(\mu_f + C_{\mathbf{21}}C_{\mathbf{11}}^{-1}(\boldsymbol{f_1} - \mu_f), C_{\mathbf{22}} - C_{\mathbf{21}}C_{\mathbf{11}}^{-1}C_{\mathbf{12}}),$$

where the $\mathbf{1}$ subscript indicates the training set and the $\mathbf{2}$ subscript the test set. In practice, one can compute $\boldsymbol{f_1}$ and $\boldsymbol{f_2}$ as a single vector, $\boldsymbol{f}' = \mu_f + \sigma_f L'_{\boldsymbol{f}}\boldsymbol{\omega}'_f$, where $L_{\boldsymbol{f}}$ is the Cholesky factor of the full covariance of all the locations, with the training locations in the first block and the test locations in the second, and $\boldsymbol{\omega}'_f$ is the concatenation of the original $\boldsymbol{\omega}_f = (\sigma_f L_{\boldsymbol{f}})^{-1}(\boldsymbol{f} - \mu_f)$ vector, used in sampling the training locations, with m standard normal deviates, where m is the number of test locations. These samples are used in calculating the evaluation criteria (Section 4.4) for test locations.

4.4 Evaluation Procedures

To compare the GP model to other methods, I use three criteria. In this section I use $\mathbf{1}$ to indicate the set of training locations and $\mathbf{2}$ to indicate a set of test locations.

4.4.1 Mean squared error

The first criterion is mean squared error (MSE), which judges the accuracy of the posterior mean of the model. For simulated data, the MSE is calculated with respect to the true mean function, $\check{\boldsymbol{f}}$, used to generate the data,

$$\mathrm{MSE}_{\mathbf{1}} = \frac{\sum_{i=1}^{n}(\check{f}_i - \widetilde{f}_i)^2}{n},$$

where $\widetilde{f}_i$ is the posterior mean, calculated by averaging over the MCMC simulations. Following DiMatteo et al. (2002) I do not calculate the MSE for test covariates in the one-dimensional simulated datasets because I am using the true mean function in the calculation of MSE rather than using the data values, and because the training data are rather dense in the covariate space and the posterior means are smooth. In the two-dimensional simulated dataset, with the training data less dense in covariate space, and the possibility of extrapolation errors, I calculate MSE for both the training and test sets.

For real datasets, I calculate MSE with respect to both test and training data as

$$\begin{aligned} \text{MSE}_{\mathbf{1}} &= \frac{\sum_{i=1}^{n}(y_i - \widetilde{f}_i)^2}{n}, \\ \text{MSE}_{\mathbf{2}} &= \frac{\sum_{j=1}^{m}(y_j - \widetilde{f}_j)^2}{m}. \end{aligned}$$

Finally to facilitate comparison of MSE across datasets, I normalize the MSE by the variance of the function values (for simulated data), $V_{\mathbf{1}} = \sum_{i=1}^{n}(\check{f}_i - \bar{f})^2/n$, or variance of the data (for real data), $V_{\mathbf{2}} = \sum_{i=1}^{n}(y_i - \bar{y})^2/n$, to calculate the Fraction of Variance Unexplained (FVU),

$$\text{FVU}_{\mathbf{1}} = \frac{\text{MSE}_{\mathbf{1}}}{V_{\mathbf{1}}},$$

which is used in Denison et al. (1998a) and Holmes and Mallick (2001), among others. I calculate the analogous quantity for test data, $\text{FVU}_{\mathbf{2}}$, based on $V_{\mathbf{2}}$.

4.4.2 Predictive density

The MSE only assesses the point predictions of the model, not the distribution of the data under the model. As an alternative when analyzing the real datasets, I also calculate the usual predictive density on held-out data. This measure assesses how well the model estimates the residual variance and how well the individual draws from the posterior explain the data. I average the conditional predictive density over the posterior distribution of the parameters using the MCMC draws from the posterior. The log predictive density (LPD) for test data is

$$\begin{aligned} \text{LPD}_{\mathbf{2}} &= \log h(\boldsymbol{y_2}|\boldsymbol{y_1}) \\ &= \log \int h(\boldsymbol{y_2} \mid \boldsymbol{f}, \eta, \boldsymbol{y_1}) d\Pi(\boldsymbol{f}, \eta|\boldsymbol{y_1}) \\ &\approx \log \frac{1}{K} \left(\sum_{k=1}^{K} \frac{1}{\sqrt{2\pi}\eta_{(k)}^m} \exp\left(-\frac{\sum_{j=1}^{m}(y_j - f_{j,(k)})^2}{2\eta_{(k)}^2} \right) \right), \end{aligned} \tag{4.1}$$

where $h(\cdot)$ is the normal density function and (k) indicates the kth MCMC draw from the posterior. I also calculate (4.1) at the training observations, y_i, $i = 1, \ldots, n$, which should give a sense for how well the model predicts the data on which it was trained, although (4.1) cannot be interpreted as a predictive density, since $h(\boldsymbol{y_1}|\boldsymbol{y_1}) = 1$. I calculate $\text{LPD}_{\mathbf{2}}$ for the two real datasets by calculating the LPD on each data point when held out as a test point and averaging over the data points to

report one value for the whole dataset. For the training set, I average over both the training points and the data splits of the cross-validation procedure.

4.4.3 Kullback-Leibler divergence

For simulated data, I know the distribution of the data, so I can calculate the Kullback-Leibler (KL) divergence between the true distribution of the data and the posterior predictive distribution for the data. I do this by generating many test observations from the true model and evaluating the KL divergence between the density of the observations under the true distribution and the predictive density of the observations under the model. I can calculate this divergence for both the set of training covariates and a set of test covariates. In either case, we have

$$\begin{aligned}
\text{KL} &= \int \log\left(\frac{h(\boldsymbol{y_2} \mid \check{\boldsymbol{f}}, \check{\eta})}{h(\boldsymbol{y_2} \mid \boldsymbol{y_1})}\right) h(\boldsymbol{y_2} \mid \check{\boldsymbol{f}}, \check{\eta}) d\boldsymbol{y_2} \\
&= \int \log\left(\frac{h(\boldsymbol{y_2} \mid \check{\boldsymbol{f}}, \check{\eta})}{\int h(\boldsymbol{y_2} \mid \boldsymbol{f}, \eta) d\Pi(\boldsymbol{f}, \eta | \boldsymbol{y_1})}\right) h(\boldsymbol{y_2} \mid \check{\boldsymbol{f}}, \check{\eta}) d\boldsymbol{y_2} \\
&= -\frac{\log(2\pi) + \log(\check{\eta}^2) + 1}{2} - \int \log\left(\int h(\boldsymbol{y_2} \mid \boldsymbol{f}, \eta)) d\Pi(\boldsymbol{f}, \eta | \boldsymbol{y_1})\right) \\
&\quad \times h(\boldsymbol{y_2} \mid \check{\boldsymbol{f}}, \check{\eta}) d\boldsymbol{y_2} \qquad (4.2) \\
&\approx -\frac{\log(2\pi) + \log(\check{\eta}^2) + 1}{2} - \int \frac{1}{K} \sum_{k=1}^{K} \log(h(\boldsymbol{y_2} \mid \boldsymbol{f}_{(k)}, \eta_{(k)})) h(\boldsymbol{y_2} \mid \check{\boldsymbol{f}}, \check{\eta}) d\boldsymbol{y_2} \\
&\approx -\frac{\log(\check{\eta}^2) + 1}{2} - \frac{1}{JK} \sum_{j=1}^{J} \sum_{k=1}^{K} \left(-\log \eta_{(k)} - \frac{(y_j - f_{j,(k)})^2}{2\eta_{(k)}^2}\right), \qquad (4.3)
\end{aligned}$$

where the first term in (4.2) is found by integrating the log of the true normal density against itself. In calculating (4.3) I use 500 observations drawn at each covariate of interest, hence for the training covariates, I have $J = 500n$, and for test covariates, I have $J = 500m$. I use $K = 1000$ subsamples from the 20000 MCMC draws (only so as to save on storage space). Note that the KL divergence is equivalent to the LPD for comparing models because the two differ only in the first term in (4.3). However, the KL divergence allows me to compare the quantity to the absolute baseline of zero, which is the KL divergence between the true distribution and itself. In all cases I calculate the KL divergence for new observations generated from the true model and not based on training observations. However, as I discussed in the case of MSE, for the simulated data in one dimension, the KL divergence calculated for test observations generated at test covariates is

unlikely to differ substantially from that calculated for test observations at training covariates, so I calculate it only for the training covariates. For the simulated dataset in two dimensions, I calculate the KL divergence separately for test observations at the test and training covariates. Note that in the papers describing the free-knot spline methods, the authors compare their methods to other methods solely on the basis of MSE (or equivalently, FVU) and do not use predictive density estimates.

4.5 Test Datasets and Competing Methods

I compare the GP model to several successful methods from the literature. For one-dimensional functions, I compare the method to the Bayesian free-knot spline model (Bayesian Adaptive Regression Splines or BARS) of DiMatteo et al. (2002). DiMatteo et al. (2002) assess the performance of BARS relative to other popular methods using three test functions, one a smoothly varying function, the second a spatially inhomogeneous function, and the third a function with a sharp jump. I use these same functions to compare the nonstationary GP regression model (NSGP) developed here to BARS. I also compare the results to the stationary GP model (SGP) to evaluate the importance of including nonstationarity in the GP model. For each test function, I generate 50 sets of noisy data from the underlying function, using the error variance given in DiMatteo et al. (2002). I then run BARS, NSGP, and SGP on these data. Since BARS outperformed the other methods to which DiMatteo et al. (2002) compared it, I simply compare the GP methods to BARS and then compare the GP methods to the other methods using DiMatteo et al. (2002, Table 1). I follow DiMatteo et al. (2002) in calculating the assessment criteria at the training locations, since in one dimension, the training locations are dense and the function estimates are smooth enough that evaluation based on training locations should not differ from that based on test locations. I will refer to these three test functions as examples 1, 2, and 3.

For higher-dimensional functions, I compare the GP methods to two free-knot spline methods, Bayesian Multivariate Linear Splines (BMLS) (Holmes and Mallick 2001) and Bayesian Multivariate Automatic Regression Splines (BMARS) (Denison et al. 1998b), a Bayesian version of the original MARS algorithm of Friedman (1991). BMLS uses piecewise, continuous linear splines, while BMARS uses tensor products of univariate splines; both are fit via reversible jump MCMC

(RJMCMC). For both methods, I consider interactions between the spline basis functions up to order two; Holmes and Mallick (2001) suggest that even when $P > 2$, using higher-order interactions in the basis functions may not be necessary. To compare the methods, I use three datasets. The first dataset is a two-dimensional test function first introduced by Hwang, Lay, Maechler, Martin, and Schimert (1994). Both Holmes and Mallick (2001) and Denison et al. (1998b) assess the performance of their methods using 225 training locations and 10000 test locations on a single simulated dataset. I also use 225 training locations, but because of the computational difficulty of working with large covariance matrices (I would need to calculate the Cholesky decomposition of matrices of size 10000 by 10000), for each simulation, I use only 225 test locations. Once again, I simulate 50 sets of noisy data from the test function. By using different test locations for each of the 50 simulations, I assess the method at $50 \cdot 225 = 11250$ test locations. The second dataset is a two-dimensional dataset of real air temperature anomalies (departures from the mean) for December 1993 used by Wood et al. (2002). Fitting the NSGP model to the full 445 observations is very slow, and I was unable to achieve convergence in reasonable time, so I chose a 109 observation subset of the original data, focusing on the Western hemisphere, $222.5°$-$322.5°$E and $62.5°$S-$82.5°$N. This allows me to assess the methods on a spatial dataset and on a dataset with clear inhomogeneous smoothness. I fit the models on 55 test/training splits of the 109 observations, with 54 splits of 107 training examples and two test examples and one split of 108 training examples and one test example, thereby including each data point as a test point once. Finally, I use a real dataset of ozone measurements (Bruntz, Cleveland, Kleiner, and Warner 1974) included in the S-plus statistical software package as the 'air' dataset. This dataset has been used frequently in the past (Cleveland and Devlin 1988; Denison et al. 1998b; Holmes and Mallick 2001) for methodological evaluation. The data are 111 daily records of ozone from the New York metropolitan area from the summer of 1973, and the goal is to predict the cube root of ozone based on three covariates: radiation, temperature, and wind speed. Here I do 56 test/training splits of the 111 observations, with 55 splits of 109 training examples and two test examples and one split of 110 training examples and one test example, thereby including each data point as a test point once. I will refer to these three datasets as the Hwang, Wood, and ozone datasets.

For the evaluation runs for all the methods, I use 5000 iterations for burn-in and then collect

20000 iterations for evaluation. For each dataset, I separately scale and translate each covariate so that the values lie in $[0, 1]$.

In running the GP models for these evaluations, I use different random seeds for each run as well as different initial values and different permutations of the covariate vectors. In running the BARS model, I use different random seeds, but use the default initial values in the code of DiMatteo et al. (2002). In running the BMLS and BMARS models, I use different random seeds and different permutations of the covariate vectors, the latter because knots are positioned on the data points and are added randomly during the MCMC. I use the default initial values given in the BMLS and BMARS code, which is available from the website of Dr. Chris Holmes. Note that for some runs, the acceptance rate of the RJMCMC was as high as 90%, although it's not clear if this is a problem.

In Section 3.5.1.2 I discuss the possibility of imposing a prior that constrains the degrees of freedom of the conditional posterior mean of the regression function in the GP model based on the trace of the smoothing matrix. In one dimension, this does not seem to be necessary in practice as the Occam's razor effect is sufficient to avoid undersmoothing. In higher dimensions, smoothness becomes more of a concern, and I do impose a prior on the degrees of freedom. For the simulations that follow I use a $\Gamma(2, 5)$ prior for the Hwang and ozone datasets, which gives a mean of 10 and variance of 50. For the temperature dataset of Wood et al. (2002) I use a $\Gamma(2, 10)$ prior, since I think that the function may not be very smooth.

In adjusting the proposal variances in the GP models, I attempted to achieve the acceptance rates recommended in Roberts and Rosenthal (2001), namely, 0.44 for scalar parameters and 0.23 for vector parameters. Because I used the same proposal variances for all of the simulations for a given dataset, there were parameters for which it was difficult to find proposal variances that worked well for all the simulations, but I do not believe this has much impact on the broad comparisons with other methods, although this may have had some impact on individual simulations. Of course in practice with real data, one could tune these proposal variances in a more optimal way than done in these evaluation runs. For the initial values of the GP models, I use a combination of dispersed values and estimates based on the data. I set $\mu_f = \bar{\boldsymbol{Y}}$ and $\sigma_f = \eta = S_{\boldsymbol{Y}}$. I take $\boldsymbol{\omega} \sim \mathrm{N}(0, I)$ and construct $\boldsymbol{f} = \mu_f + \sigma_f L_{\boldsymbol{f}} \boldsymbol{\omega}_f$. For the parameters determining the covariance structure, on

which $L_{\boldsymbol{f}}$ is based, I generate initial values from the following distributions, $\nu_f \sim \text{U}(0.5, 30)$, $\log \kappa_\phi \sim \text{U}(-3.5, 1.5)$, $\gamma_\phi \sim \text{U}(-\pi, \pi)$, while setting $\boldsymbol{\omega}_\phi = \mathbf{0}$. For the eigenvalue processes, $\lambda_p(\cdot)$, I take $\mu_{\lambda_p} \sim \text{U}(-7, 2)$.

For the simulated data in one dimension, I report the 50 values (from the 50 data draws from the underlying distribution) of the MSE and KL for each of the methods for the training set. For the simulated two-dimensional Hwang dataset, I report the 50 values of MSE and KL for both the training and test sets. For the real datasets, I calculate the MSE and LPD for each observation when held out of the fitting and average over the data points to report one value for each of the criteria. For the training set, I average the MSE and LPD over both the data splits and the observations.

4.6 Results

4.6.1 One-dimensional assessment

On the slowly-varying smooth function (example 1), BARS and both the stationary and nonstationary GP models give very similar results for MSE (Figure 4.2). However, the KL divergences for the GP models are much lower than for BARS (Figure 4.2), because BARS systematically overestimates the error variance when $\boldsymbol{y}^T\boldsymbol{y}$ is large. This occurs because the unit information prior on the regression spline coefficients used by DiMatteo et al. (2002) is conditional on the error variance, η^2, and the resulting conditional posterior mean of η^2 is

$$\text{E}(\eta^2|\boldsymbol{f}, \boldsymbol{y}, k, \boldsymbol{\xi}) = \frac{\boldsymbol{y}^T\boldsymbol{y} - \frac{n}{n+1}\boldsymbol{y}^T\boldsymbol{f}}{n} = \frac{\frac{n+2}{n+1}\boldsymbol{f}^T\boldsymbol{\epsilon} + \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} + \frac{1}{n+1}\boldsymbol{f}^T\boldsymbol{f}}{n},$$

where $\boldsymbol{\epsilon} = \boldsymbol{f} - \boldsymbol{y}$, while the conditional posterior mean without the unit information prior would be

$$\text{E}(\eta^2|\boldsymbol{f}, \boldsymbol{y}, k, \boldsymbol{\xi}) = \frac{\boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T\boldsymbol{f}}{n} = \frac{\boldsymbol{f}^T\boldsymbol{\epsilon} + \boldsymbol{\epsilon}^T\boldsymbol{\epsilon}}{n}.$$

When the function mean is far from zero, as in example 1, the term $\frac{1}{n+1}\boldsymbol{f}^T\boldsymbol{f}$ can have a large impact on the estimate of η^2; this is a general effect in models in which the prior for the mean structure is dependent on the variance parameter. A better approach would be to subtract off $\bar{\boldsymbol{y}}$ from the data before running BARS, although I have not done that here. Interestingly, for samples of data in which it was difficult for the algorithms to determine the underlying function (i.e., the

MSE was high for both methods), the nonstationary GP method tended to perform better than BARS, while for samples in which the MSE was low, BARS tended performed better. In Figure 4.3, I show two data samples with BARS and NSGP fit to the data. Next I compare the results with those from Table 1 of DiMatteo et al. (2002), in which they compare BARS to the SARS and DMS methods (Figure 4.4). We see that based on a 95% confidence interval for the mean MSE over simulated datasets, BARS and the GP methods seem to outperform SARS and DMS, but that we cannot be certain of this conclusion relative to DMS because of high variability.

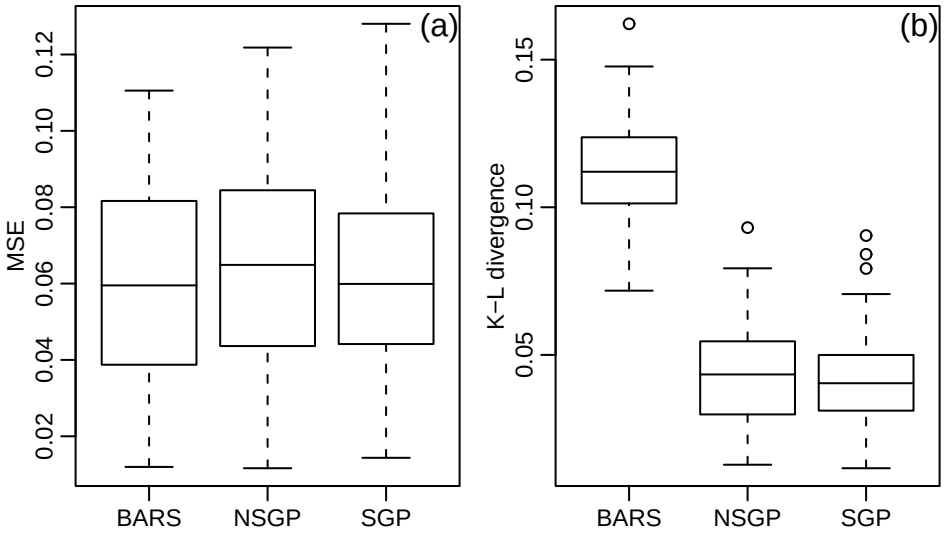

Figure 4.2. Boxplots of (a) MSE and (b) KL divergence for the three methods over 50 simulated datasets of example 1: Bayesian adaptive regression splines (BARS), nonstationary GP (NSGP), and stationary GP (SGP).

On the spatially inhomogeneous function (example 2), the nonstationary GP appears slightly worse than BARS in terms of MSE, while the stationary GP is noticeably worse than either BARS or the nonstationary model (Figure 4.5). In most (42 of 50) data samples, BARS has lower MSE than NSGP. The KL divergence for BARS may be slightly better than for the nonstationary GP (Figure 4.5), although this is difficult to interpret in light of the poor KL divergence for BARS in example 1. In Figure 4.6, I show BARS and NSGP fit to one of the 50 data samples. The nonstationary GP model smooths the data somewhat less than BARS, resulting in undersmoothing, apart

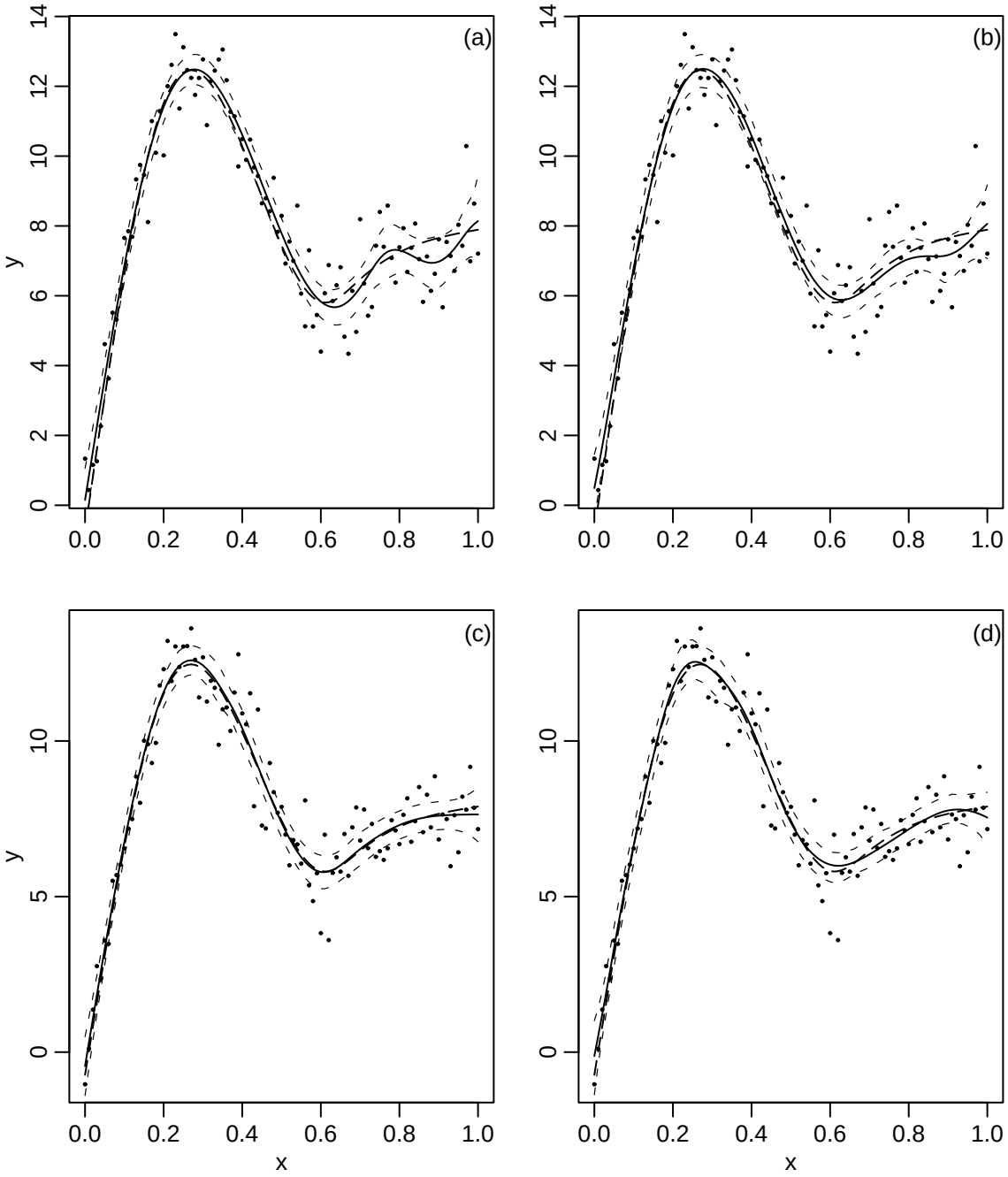

Figure 4.3. (a) BARS and (b) NSGP fit to one data sample in which NSGP has lower MSE than BARS. (c) BARS and (d) NSGP fit to a second data sample in which BARS has lower MSE. The thick dashed line is the true function, the solid line is the posterior mean estimate, and the thin dashed lines are 95% pointwise credible intervals.

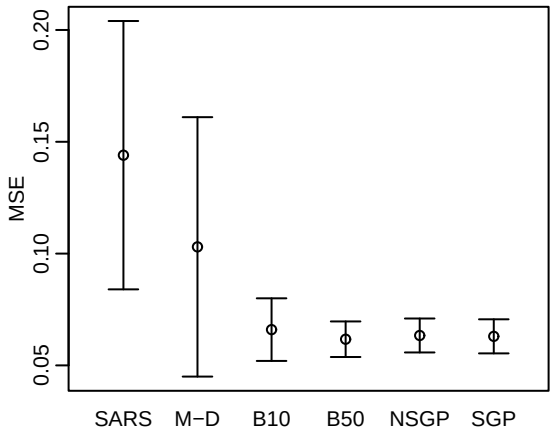

Figure 4.4. 95% confidence intervals for the mean MSE over simulated datasets of example 1. SARS, M-D (Modified-DMS) and B10 (BARS) are based on 10 sample datasets as calculated in DiMatteo et al. (2002), while B50 (BARS), NSGP (nonstationary GP) and SGP (stationary GP) are based on 50 sample datasets as calculated here.

from the jump at zero. Interestingly, for this data sample NSGP better captures the peak than does BARS. Comparing 95% confidence intervals for the mean MSE over simulated datasets (Figure 4.7), we see that the nonstationary GP may be comparable to what DiMatteo et al. (2002) term "Modified-DMS", namely BARS without the use of the locality heuristic for locating knots, and better than SARS, although the variability in the modified-DMS estimates makes this conclusion tentative. This result suggests the importance of the locality heuristic for the success of BARS. Note that the stationary GP is clearly worse than the various adaptive methods, illustrating the danger in using a non-adaptive method when the function is inhomogeneous. In the stationary GP model, κ_f and ν_f trade off, with at least one at a very low value during each iteration of the Markov chain. The model overfits in the regions where the true function is smooth, producing a very jagged posterior mean, and somewhat underfits the jump in the function.

Example 3 has a sharp jump at $x = 0.4$, where the function is not differentiable. On this example, BARS clearly outperforms NSGP in terms of both MSE and KL divergence, while the stationary GP again performs poorly relative to the nonstationary GP (Figure 4.8). It's clear what is happening based on fits from BARS and NSGP (Figure 4.9). The NSGP model captures the

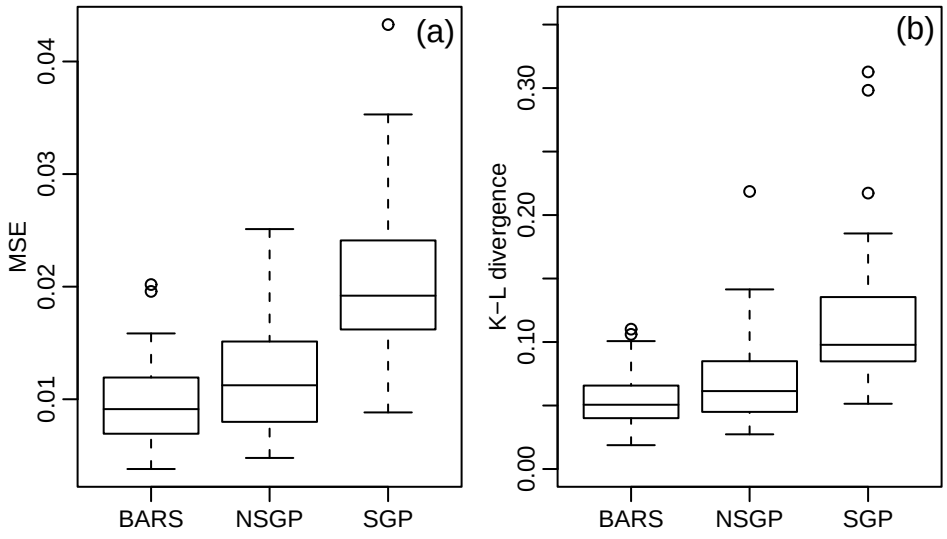

Figure 4.5. Boxplots of (a) MSE and (b) KL divergence for the three methods over 50 simulated datasets of example 2: Bayesian adaptive regression splines (BARS), nonstationary GP (NSGP), and stationary GP (SGP). For SGP, one outlier with KL divergence of 0.58 is not plotted.

sharp jump by using small kernels at the jump point, but because the kernels are forced to vary smoothly, the model undersmooths the data in the vicinity of the jump. Comparing 95% confidence intervals for the mean MSE over simulated datasets (Figure 4.10), we see that the nonstationary GP is roughly comparable to and probably somewhat better (the two outlying MSE values for NSGP appear to be caused by poor MCMC mixing) than modified-DMS, namely BARS without the use of the locality heuristic for location knots, and worse than SARS. As in example 2, this result suggests the importance of the locality heuristic for the success of BARS. The relatively poor performance of the nonstationary GP method is expected because the kernels are forced to vary smoothly, limiting the ability of the method to model sharp function changes. In Figure 4.11 I show how the GP model defines an implicit kernel smoother (locally-weighted averaging of the observations) based on the smoothing matrix,

$$S = C_{\boldsymbol{f}}(C_{\boldsymbol{f}} + C_{\boldsymbol{Y}})^{-1},$$

as I discussed in Section 1.4.3. We see that the kernel at $x = 0.4$, the location of the sharp jump

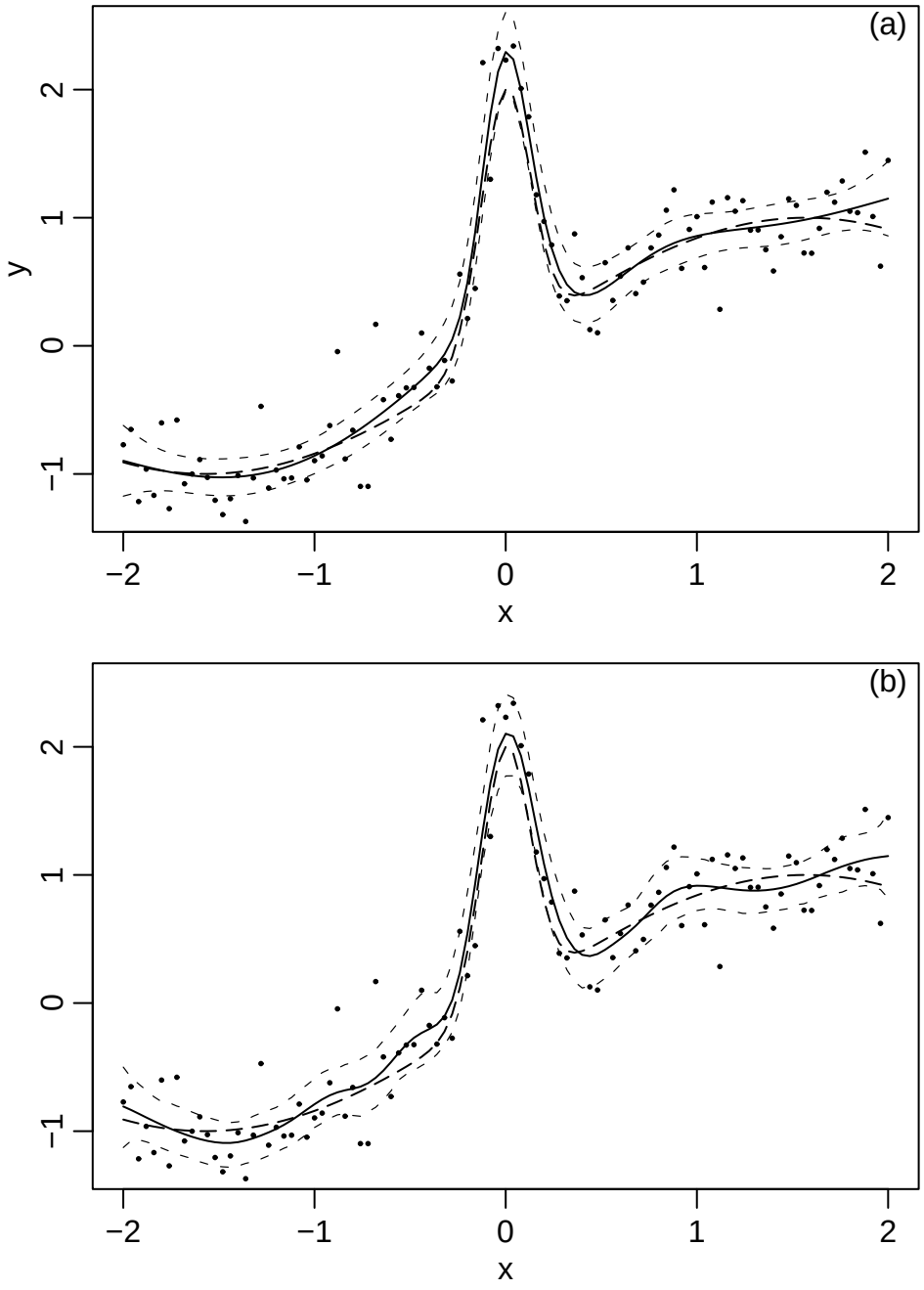

Figure 4.6. (a) BARS and (b) nonstationary GP fit to one data sample of example 2. The thick dashed line is the true function, the solid line is the posterior mean estimate, and the thin dashed lines are 95% pointwise credible intervals.

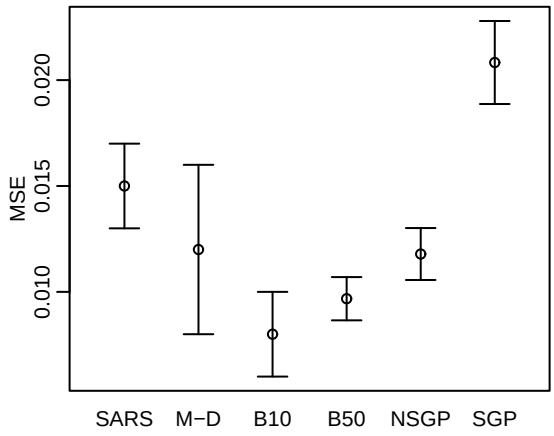

Figure 4.7. 95% confidence intervals for the mean MSE over simulated datasets of example 2. SARS, M-D (Modified-DMS) and B10 (BARS) are based on 10 sample datasets as calculated in DiMatteo et al. (2002), while B50 (BARS), NSGP (nonstationary GP) and SGP (stationary GP) are based on 50 sample datasets as calculated here.

in the function, is quite narrow, resulting in very local averaging, while the kernels elsewhere are much broader, resulting in more smoothing. As in example 2, the stationary GP model overfits in the regions where the true function is smooth, producing a very jagged posterior mean, and somewhat underfits the jump in the function. In this example, either κ_f or ν_f, or both, are at very low values during each iteration of the Markov chain.

The one-dimensional examples allow me to assess the GP model relative to BARS qualitatively. In general BARS sample paths are much smoother than GP sample paths. Once the MCMC in BARS settles on a small number of knots, it does not visit the parts of the function space with many knots. This may be partly because BARS is known to oversmooth to some extent, although to its credit, in these simulations, the method performs better than the GP method in areas where the function is less smooth, through its ability to place multiple knots in small intervals. The GP method shows less smoothness for several reasons. The main reason seems to be that the nonstationary GP model inherently samples less smooth functions. However, other reasons contribute as well. First, on the occasions when it samples from small values of ν_f, such as $\nu_f \leq 2$, which give non-differentiable sample paths, the sample paths are locally unsmooth, as expected. We can see

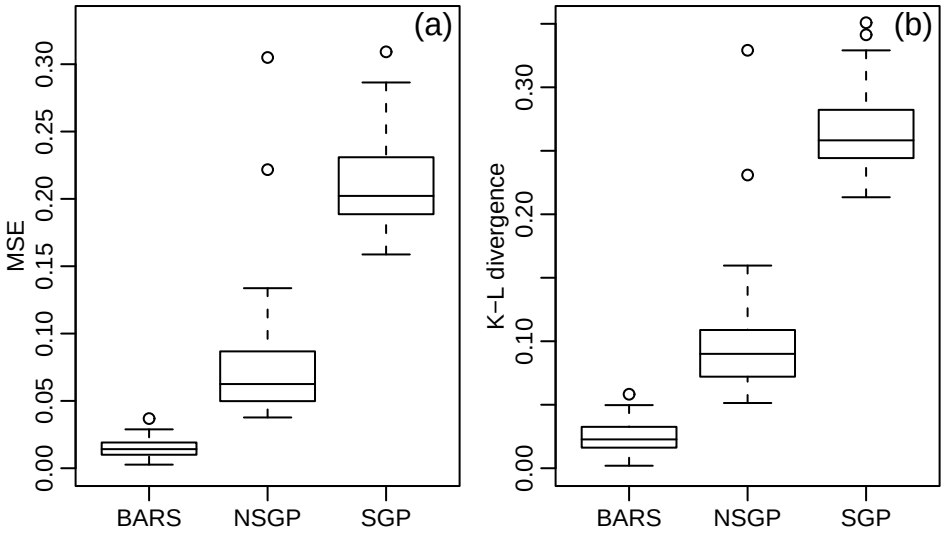

Figure 4.8. Boxplots of (a) MSE and (b) KL divergence for the three methods over 50 simulated datasets of example 3: Bayesian adaptive regression splines (BARS), nonstationary GP (NSGP), and stationary GP (SGP). One outlying value for SGP is omitted in both plots (MSE=0.87, KL=0.54). The two outliers for NSGP appear to be datasets for which the MCMC did not mix well because the proposal variances were the same for the MCMCs for all 50 data samples.

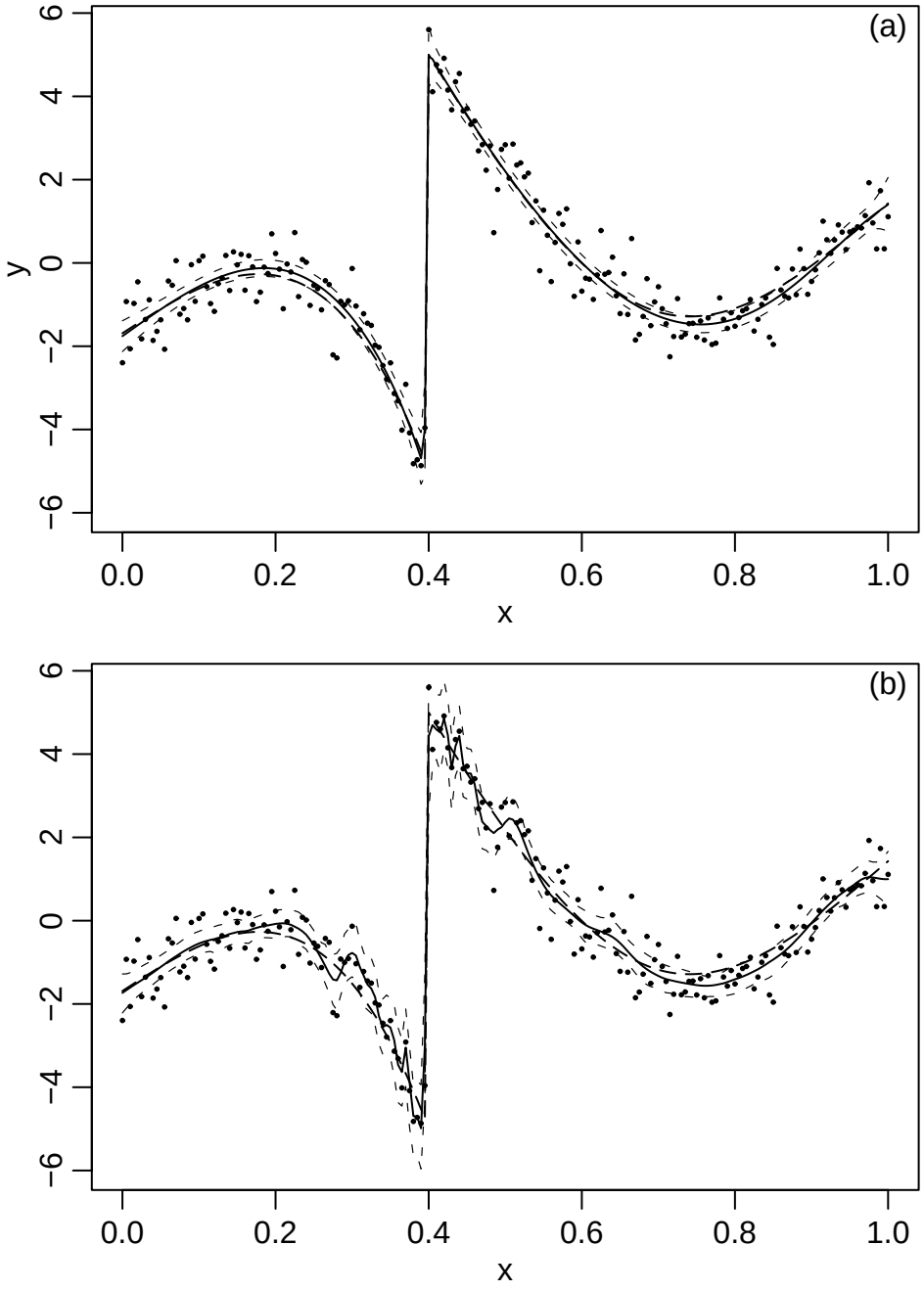

Figure 4.9. (a) BARS and (b) nonstationary GP fit to one data sample of example 3. The thick dashed line is the true function, the solid line is the posterior mean estimate, and the thin dashed lines are 95% pointwise credible intervals.

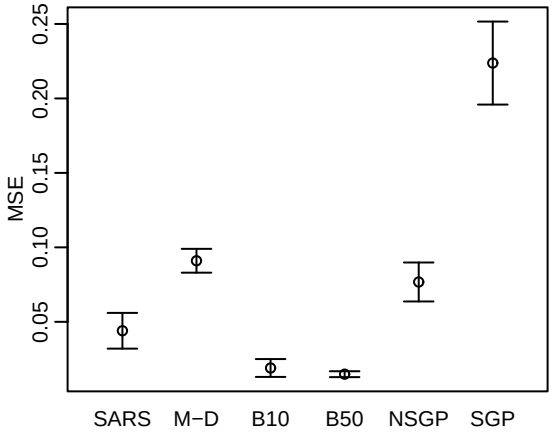

Figure 4.10. 95% confidence intervals for the mean MSE over simulated datasets of example 3. SARS, M-D (Modified-DMS) and B10 (BARS) are based on 10 sample datasets as calculated in DiMatteo et al. (2002), while B50 (BARS), NSGP (nonstationary GP) and SGP (stationary GP) are based on 50 sample datasets as calculated here.

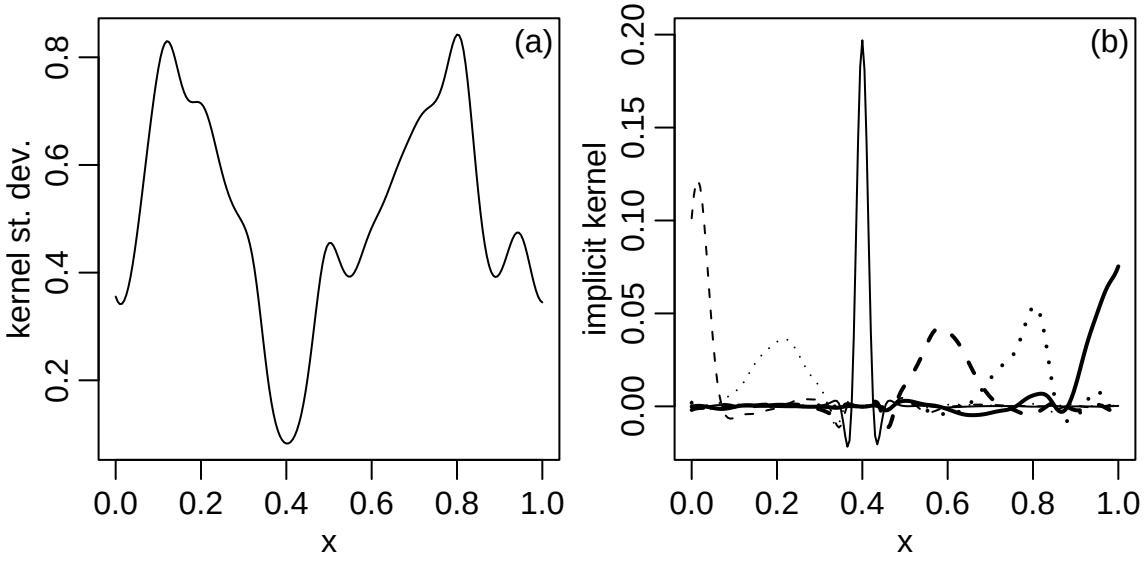

Figure 4.11. (a) Kernel size (standard deviation of kernels) as a function of the covariate for one posterior sample for example 3. (b) Implicit smoothing kernels at six covariate values, with different line types and/or widths for each implicit kernel.

an example of this in Figure 4.12 where the regression function is locally rough when $\nu_f = 1.3$ and locally smooth when $\nu_f = 4.5$, even though the other hyperparameter values are similar and the kernel sizes are larger in the case of $\nu_f = 1.27$. In other cases with small ν_f, the sample function can be even more jagged. A second reason relates to the observation of Gibbs (1997), namely that when the kernel sizes change rapidly, the correlation drops off relatively quickly in an intermediate neighborhood before levelling off and declining slowly at larger distances (Section 2.2). When the chain samples kernel sizes that are relatively large in magnitude, but which change rapidly, the sample functions tend to be rather unsmooth at intermediate scales (though still smooth at small scales). In Figure 4.13, we see an example of a sample function in which sharp changes in the kernel size induce lack of smoothness in the function, even though the kernel size is quite large. Also in Figure 4.13, we see a different example of a sample function in which the sharp change in the kernel size induces lack of smoothness in the function, although on this occasion, the model is using this feature of the correlation model to capture the jump in the function at $x = 0$ by having an unintuitively large kernel size right at the point of the jump. A third reason relates to a numerical problem that also occurred in some of my development runs. On occasion, the generalized Cholesky algorithm creates a Cholesky factor for which $LL^T \approx R$ did not hold. The result is localized jaggedness in the sample paths accounted for by not using a reasonable approximation to the Cholesky of the covariance matrix. It is not entirely clear why this happens, but it seems to occur because of very small errors, $O(10^{-14})$, in the calculation of elements of the covariance matrix.

For the stationary GP method on the inhomogeneous functions, the posterior for the Matérn smoothness parameter, ν_f, tends to concentrate on very small values in the range $(0.5, 2)$. These values are much lower than the bulk of the posterior for ν_f in the nonstationary model or in the stationary model for the homogeneous function. This suggests that the model is attempting to capture sharp changes in the function using the smoothness parameter, because it is unable to adequately model the data with the constant scale parameter, κ_f. In Figure 4.14 we see a demonstration of this for example 2 in which $\nu_f = 0.69$, verging on the exponential correlation function, but $\kappa_f = 4.0$, which is extremely large. (But note that κ_f is also quite small in many of the data samples and Markov chain iterations.) Instead of using a small value of κ_f to be able to capture the jump at

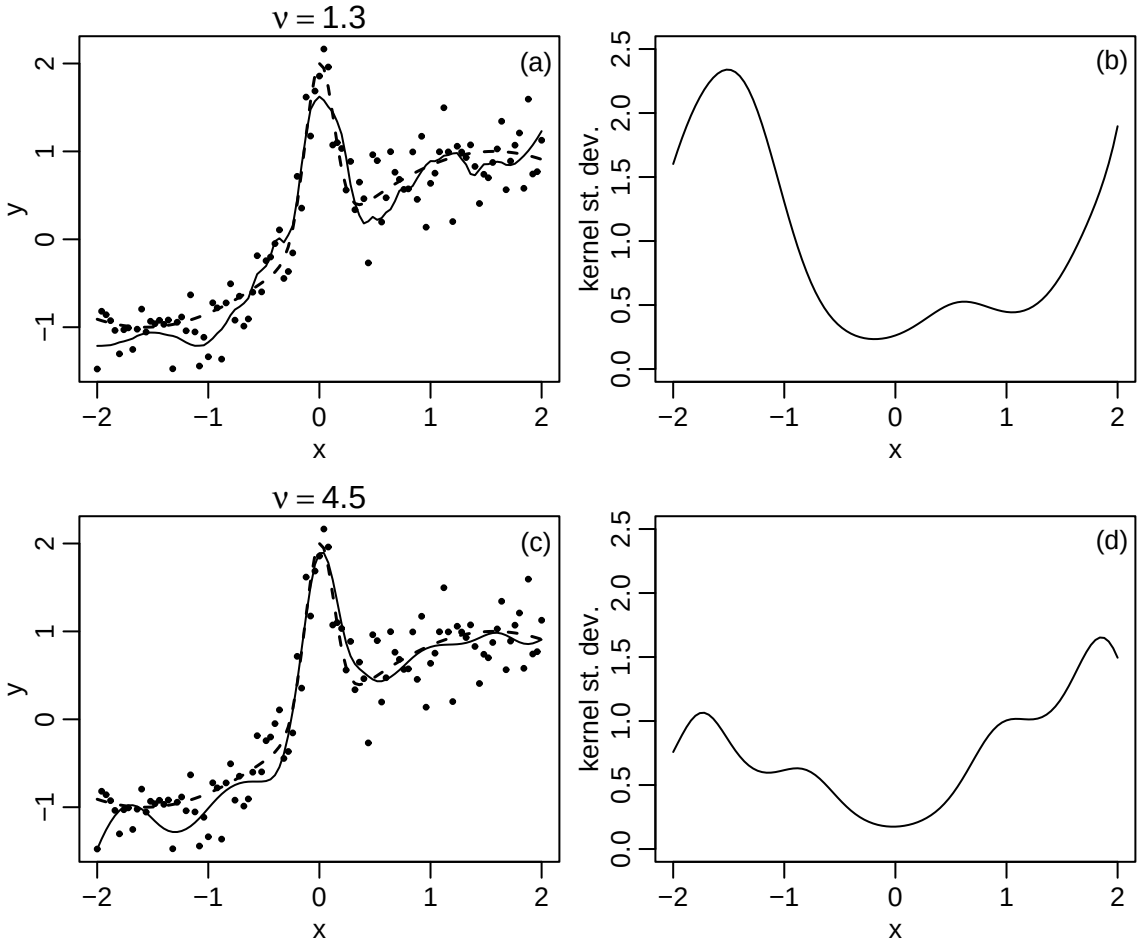

Figure 4.12. (a) Sample posterior regression function with $\nu_f = 1.3$ (solid line) and true function (dashed line) from example 2. (b) Kernel size (standard deviation of kernel) as a function of the covariate for the sample shown in (a). (c) Sample posterior regression function with $\nu_f = 4.5$ (solid line) and true function (dashed line). (d) Kernel size (standard deviation of kernel) as a function of the covariate for the sample shown in (c).

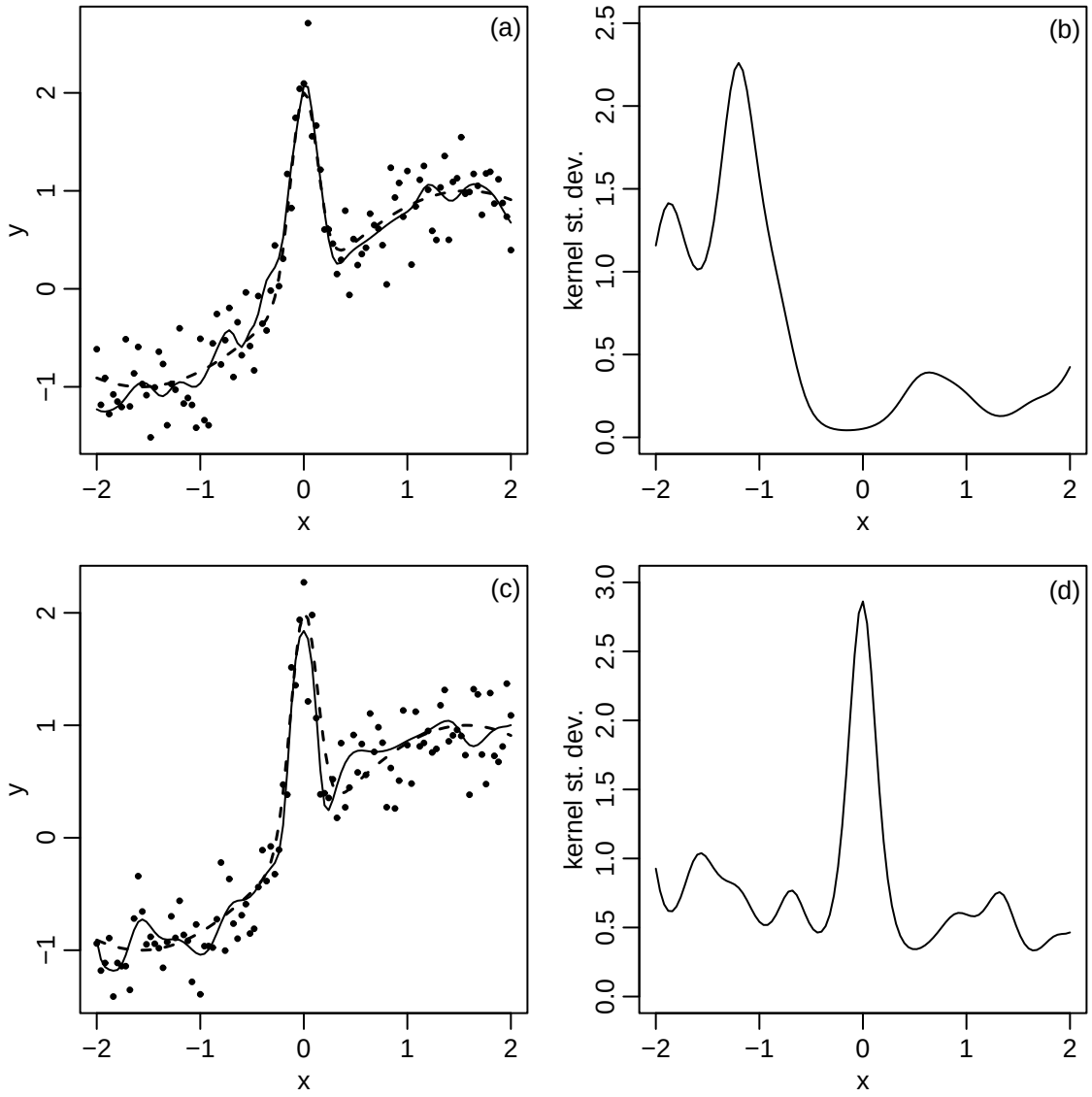

Figure 4.13. (a) Sample posterior regression function from one data sample (solid line) and true function (dashed line) for example 2. (b) Kernel size (standard deviation of kernel) as a function of the covariate for the sample shown in (a). Notice the lack of smoothness in the function for $-2 < x < -0.5$*, where the kernel sizes are large but variable. (c) Sample posterior regression function for a different data sample of example 2 (solid line) and true function (dashed line). (d) Kernel size (standard deviation of kernel) as a function of the covariate for the sample shown in (c). Notice that the nonintuitive sharp increase in the kernel size is what allows the model to capture the function jump at* $x = 0$*.*

$x = 0$, the model uses a very small value of ν_f. This suggests that one should be wary of interpreting the Matérn smoothness parameter as the degree of differentiability of the underlying function, since the parameter may just be compensating for misspecification of the correlation structure at coarser scales.

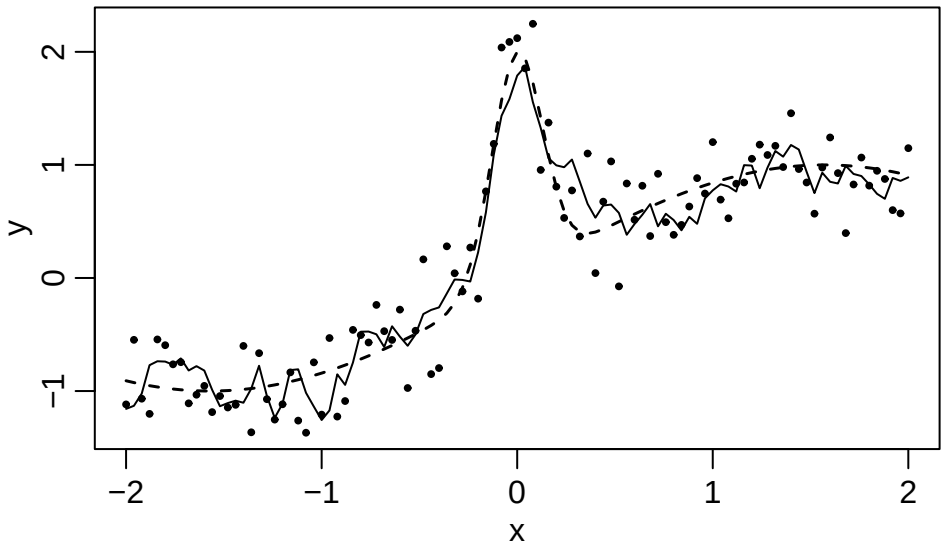

Figure 4.14. Sample posterior regression function from example 2 using a stationary GP model; here $\nu_f = 0.69$ *and* $\kappa_f = 4.0$.

Note that in earlier runs, I set $\nu_\lambda = 4$ rather than the value of 30 used here, and the performance of the nonstationary GP was very similar, suggesting that differentiability properties of the kernel elements may be of limited practical import, although I suspect that specifying ν approaching $\frac{1}{2}$ (the exponential correlation) might have an effect.

4.6.2 Higher-dimensional assessment

The simulated two-dimensional covariate dataset of Hwang et al. (1994) is based on a complex interaction surface. In Figure 4.15, I show perspective plots of the true function and the posterior mean estimate from the nonstationary GP model based on 225 training observations. The model does a good job of estimating the surface. Note that the residual standard deviation is only 0.25, so it is a fairly easy smoothing problem. Figure 4.16 shows contour plots of the true function

and posterior mean estimate, as well as the difference between the two, which highlights that the model performs worst at the extremes of the covariate values, as one would expect because of extrapolation error. Both the stationary and nonstationary Gaussian process models perform well in comparison with the spline methods, based on both the training (Figure 4.17) and test (Figure 4.18) locations. For FVU, both the stationary and nonstationary GP models perform better than either spline model, while the stationary and nonstationary models are very similar, reflecting the relative homogeneity of the true function. All methods show higher FVU and KL divergence on the test covariates than the training covariates, probably because of poor extrapolation for outlying test covariates. The average test FVU estimate for BMLS is essentially the same (0.033) as reported in Holmes and Mallick (2001) based on a single simulated dataset. They also report test FVU estimates of 0.07 for a neural network model and 0.043 for a projection pursuit model, so it appears that the GP models do well relative to these methods as well, since the mean FVU for the NSGP is 0.023 and that for the SGP is 0.024. Denison et al. (1998b) report a training FVU of 0.0575 and a test FVU of 0.0411 for BMARS on this function, both somewhat different than the averages over the 50 datasets that I found (training FVU of 0.0288 and test FVU of 0.0761), presumably because I used different simulated data samples than they did.

The temperature dataset used by Wood et al. (2002) allows me to compare the methods on a spatial dataset, for which inhomogeneity in the underlying surface seems likely. NSGP outperforms the other methods in terms of both MSE (Table 4.1) and LPD (Table 4.2). Comparing the data and the model estimates (not shown), it appears that the success of the NSGP relative to the other methods occurs because the other methods oversmooth the data and don't capture real peaks and troughs in the surface. Note that in my initial fits of the models to the full 445 observations in the dataset, the MSE of the NSGP on test data was poor, but given its success on the reduced dataset and the very slow mixing of the MCMC for the full dataset, it appears that this occurs because the NSGP model has not burned in, rather than because the model is unable to fit the data. Using the prior on the degrees of freedom of the conditional posterior mean regression function appeared to help in the NSGP model. The MSE on test data without the df prior was 1.24, higher than the 1.10 with the prior and the LPD was slightly lower, -0.43 compared to -0.40.

For the real ozone dataset of Bruntz et al. (1974), the NSGP does a better job of prediction

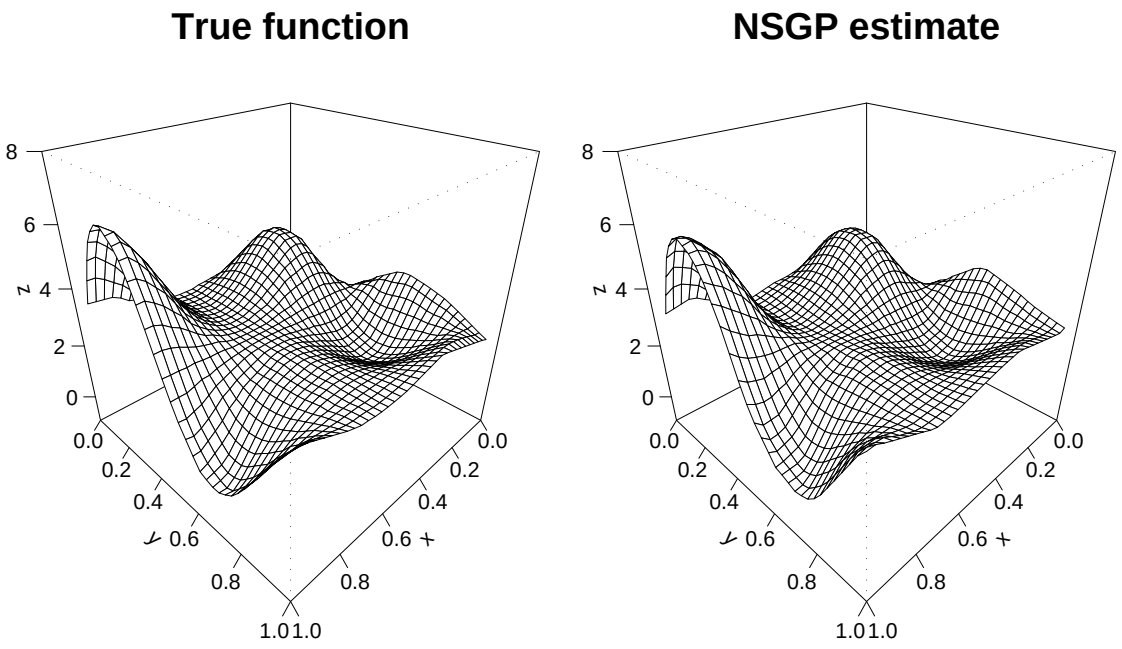

Figure 4.15. Perspective plots of (left) true Hwang function and (right) posterior mean function using the nonstationary GP model. Note that these plots involve interpolation by the interp and persp functions in the R statistical package; the interp function is found in the akima library.

Table 4.1. MSE for training and test sets for the four methods on a portion of the Wood dataset..

model	train	test
BMARS	0.55	1.74
BMLS	0.97	2.40
SGP	0.62	1.40
NSGP	0.25	1.10

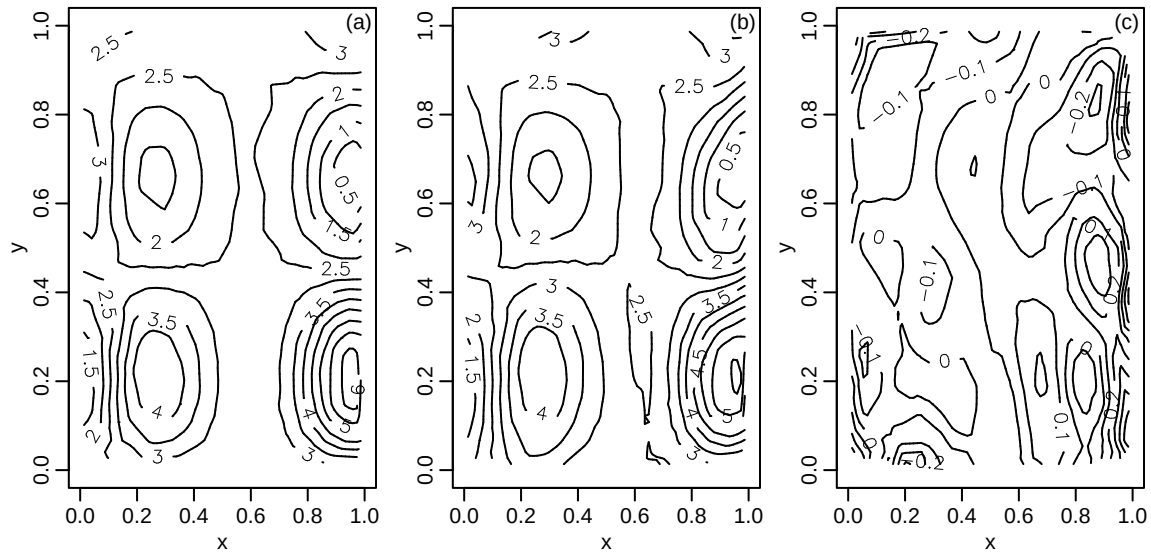

Figure 4.16. Contour plots of (a) true Hwang function, (b) posterior mean function using the nonstationary GP model, and (c) difference between the true function and the estimated function. Note that these plots involve interpolation by the interp and contour functions in the R statistical package; the interp function is found in the akima library.

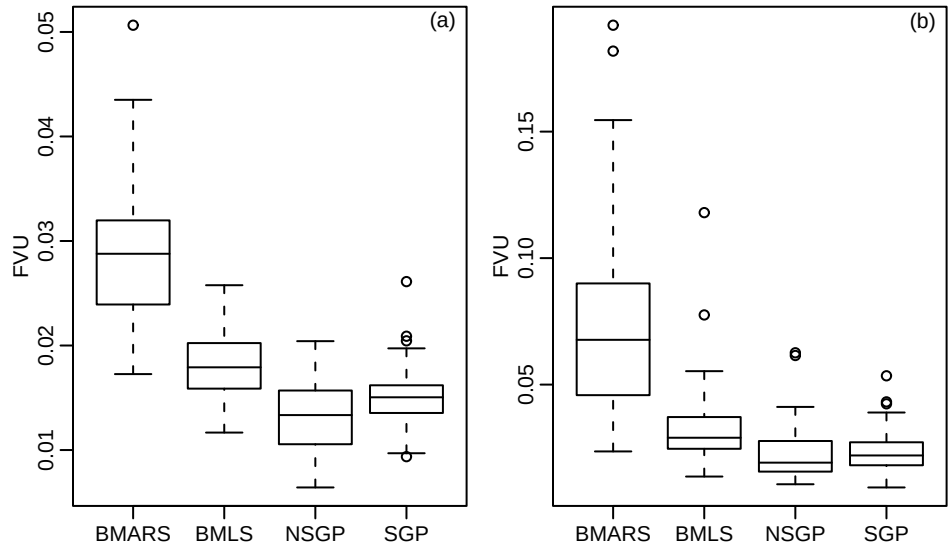

Figure 4.17. Boxplots of FVU for (a) training covariates and (b) test covariates over 50 simulated datasets of the Hwang function for the four methods

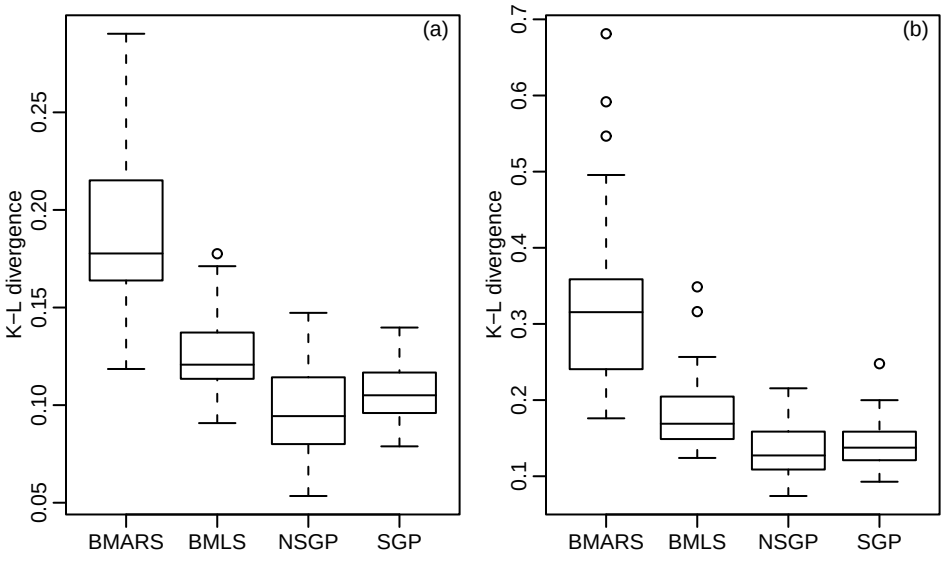

Figure 4.18. Boxplots of KL divergence for (a) training covariates and (b) test covariates over 50 simulated datasets of the Hwang function for the four methods.

Table 4.2. LPD for training and test sets for the four methods on a portion of the Wood dataset, averaged across observations.

model	train	test
BMARS	-0.22	-0.66
BMLS	-0.40	-0.86
SGP	-0.25	-0.65
NSGP	0.11	-0.40

on held-out data than the stationary model or the spline models (Table 4.3). Note that all of the GP and spline models do much better than linear regression or even a generalized additive model (GAM) with a four degree of freedom smooth for each covariate, suggesting the presence of non-additive structure in the data. The NSGP also does better in terms of predictive density than the other methods (Table 4.4), but the relative differences are small (a change of 0.08 in LPD means that an observation is 1.08 times as likely under the NSGP as under BMARS, for example). The similarity between the stationary and nonstationary GP results for MSE suggests that the underlying regression function is relatively homogeneous, but there does appear to be some heterogeneity. For this dataset, the use of the prior on the degrees of freedom seems to have little impact on predictive performance (MSE of 0.0055 and LPD of 2.12 without the prior).

Table 4.3. MSE for training and test sets for the four methods, as well as linear regression and a generalized additive model (GAM) on the ozone dataset.

model	train	test
Lin Regr	not computed	0.021
GAM	not computed	0.020
BMARS	0.0039	0.0062
BMLS	0.0048	0.0062
SGP	0.0037	0.0062
NSGP	0.0027	0.0054

In the code for BMARS and BMLS, available from the website of Dr. Chris Holmes, the spline basis functions in the model are standardized so that they have mean zero and standard deviation one over the basis function values evaluated at the training covariates. However, when a basis function has support only on a small portion of the covariate space, this tends to result in the basis function having very high values in extreme parts of the space. This is discouraged by the likelihood when there is a training covariate in this part of the space, but when there is only a test covariate, it tends to result in very poor prediction on the test covariates. For this reason, I altered the code to avoid the standardization, and achieved better predictive results. I do not know why this

Table 4.4. LPD for training and test sets for the four methods on the ozone data, averaged across observations.

model	train	test
BMARS	2.31	2.05
BMLS	2.17	2.04
SGP	2.13	2.05
NSGP	2.45	2.13

standardization step is included in the code. In particular, this problem arose for the BMLS model in the Wood et al. (2002) dataset, with the MSE on test data being 17.2 compared to 2.4 without the standardization.

4.6.3 Convergence

Based on diagnostic plots for the MCMC runs for the GP models, as well as the free-knot spline models in the higher-dimensional datasets, it appears that while the models have burned in, they have not fully mixed and that longer runs would be necessary to ensure convergence, in particular for the hyperparameters of the kernel eigenvalue processes. However, the evaluation criteria appear to be stable, which is sufficient for my purpose here. My general approach of running many chains with some of the initial parameter values differing between runs is in the spirit of Gelman and Rubin (1992), and the comparable results for the various runs offer evidence that the MCMC is giving reasonable results. Since I know the underlying function in 4 of the 6 cases, it would be clear if the MCMC were performing terribly, and my comparison with other methods gives further evidence that the chain is producing reasonable results, even though I do not directly assess the convergence for each simulation.

4.6.4 Example using non-Gaussian data

As a final evaluation of the method, I consider a dataset with non-Gaussian response, the Tokyo rainfall dataset used by Biller (2000) and others. I do not carry out a full evaluation of the GP

method in comparison with other methods nor do I assess whether a nonstationary model is useful for this dataset, but rather I use the dataset to illustrate the ability of the nonstationary GP method to model non-Gaussian data. In particular, I compare two MCMC algorithms with this dataset, one the centered parameterization with joint sampling and the other the PMC sampling scheme. I also compare sampling with and without the Langevin-style update for $\boldsymbol{f}$, which uses the gradient of the log posterior. The model and sampling are described in more detail in Section 3.6.2.3, with details of the Langevin sampling in Section 3.6.2.1.

Biller (2000) uses the Tokyo rainfall dataset, originally from Kitagawa (1987), to illustrate his Bayesian free-knot spline method for non-Gaussian data. The data are the presence or absence of rainfall greater than 1 mm for every calendar day in the years 1983 and 1984. Assuming independence between years, the likelihood for a given calendar day, x_i, is binomial with two trials and unknown probability of rainfall, $p(x_i)$. Following Biller (2000) I take the response at different calendar days to be independent, conditional on $f(\cdot) = \text{logit}(p(\cdot))$. In a careful analysis, one might want to constrain the function and the eigenvalue function to be continuous between calendar day 365 and calendar day 1 by using distance on the circle rather than Euclidean distance, but I have not done so here.

I sampled 20000 iterations during the MCMC, following a burn-in period of 5000 iterations. Based on time series plots (not shown) and the effective sample size (ESS) approach (described in Section 3.6.2.3), the PMC and PMC with Langevin sampling schemes mix better than using the centered-joint scheme, either with or without Langevin sampling (Table 4.5). For reasons that are unclear, the PMC-Langevin scheme mixed much better when I attempted to have an acceptance rate of 0.23 for $\boldsymbol{f}$ than when I attempted to achieve a rate of 0.57, the theoretically optimal value given in Roberts and Rosenthal (2001).

Since the PMC-Langevin scheme with acceptance rate of approximately 23% mixed best, I focus on the results from that scheme. In Figure 4.19, I show time series plots for the hyperparameters and in Figure 4.20, time series plots for the function values and kernel eigenvalues at four locations. These plots indicate that the MCMC needs more iterations to mix fully but that it does seem to be exploring the parameter space, suggesting that the model is being fit successfully. In Figure 4.21 I show the posterior mean of the modelled probability of rainfall, $p(\cdot)$, with pointwise

Table 4.5. Effective sample size (ESS) by sampling scheme for key model parameters for the Tokyo rainfall dataset. $\bar{f}$ is the mean ESS for the function values, averaged over 10 randomly sampled calendar days, and $\bar{\lambda}$ is the mean ESS for the log of the kernel eigenvalues, averaged over 10 randomly sampled calendar days.

sampling method	μ_f	σ_f	ν_f	μ_λ	κ_λ	$\bar{f}$	$\bar{\lambda}$
centered-joint	37	47	21	39	26	47	17
centered-joint, Langevin, acc.≈23%	32	68	23	30	37	46	12
PMC	1114	81	241	48	59	85	62
PMC, Langevin, acc.≈57%	902	34	646	27	47	55	29
PMC, Langevin, acc.≈23%	1269	91	639	72	87	101	71

95% credible intervals. The response function seems to reasonably follow the data. Note that the data in some areas seem quite clustered, which explains why the response function is not estimated to be smoother. This response function closely matches the response function in Biller (2000, Fig. 1) using natural splines with a Poisson(60) prior on the number of knots, but is much less smooth than the spline models with Poisson(30) priors on the number of knots. Biller (2000) prefers the Poisson(30) curves, saying that the Poisson(60) curve is too rough and shows too many details. If one's goal is to see long-term patterns, then a smoother function is desirable, but my assessment is that the less smooth curves (namely the Poisson(60) curve and the GP curve here) are more true to the apparent clustering in the underlying data.

In Figure 4.22 I show the geometric mean kernel size as a function of calendar day, indicating that the model is detecting inhomogeneity in the underlying function, with more smoothness in the first few months and less smoothness later in the year. I use the geometric mean since the kernel eigenvalues are fit on the log scale.

The GP method is clearly much slower than the spline approach. Biller (2000) reports a running time of about 5 minutes for 15000 iterations on a Pentium II 333 MHz machine, while the runs here took about 30 hours for 15000 iterations on a Pentium 4 2.2 GHz machine.

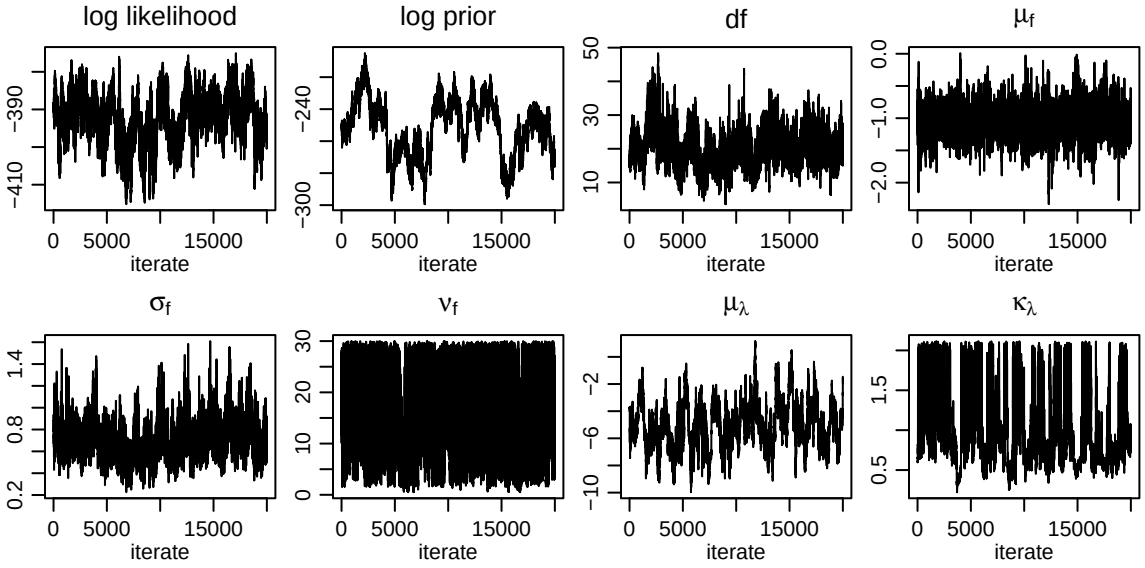

Figure 4.19. Time series plots for the Tokyo rainfall dataset for model log likelihood, log prior density, degrees of freedom of the conditional posterior mean function, and hyperparameters.

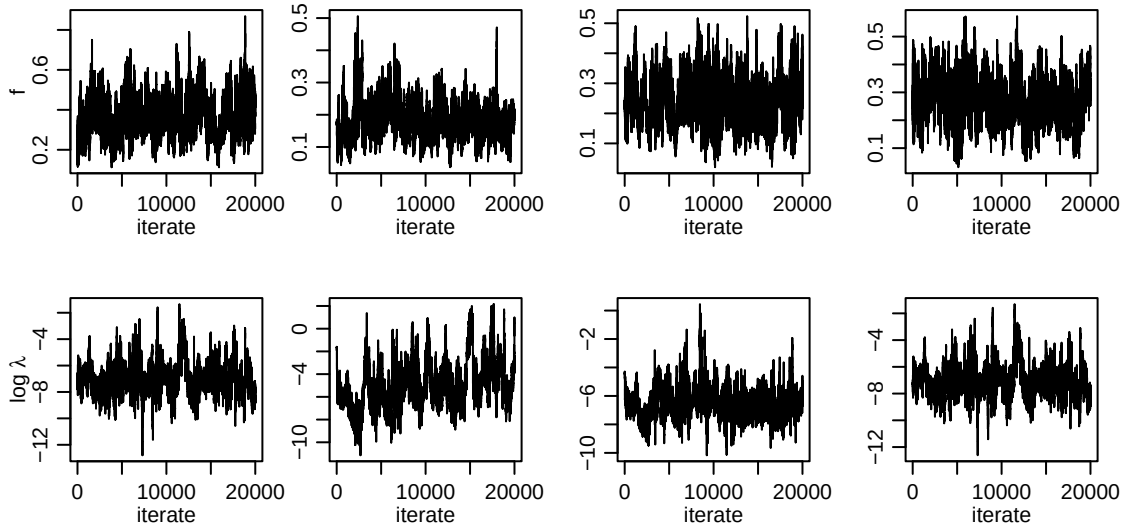

Figure 4.20. Time series plots for the Tokyo rainfall dataset for function values, $f(\cdot)$ (first row), and log of kernel eigenvalues, $\log \lambda(\cdot)$ (second row), at four covariate values.

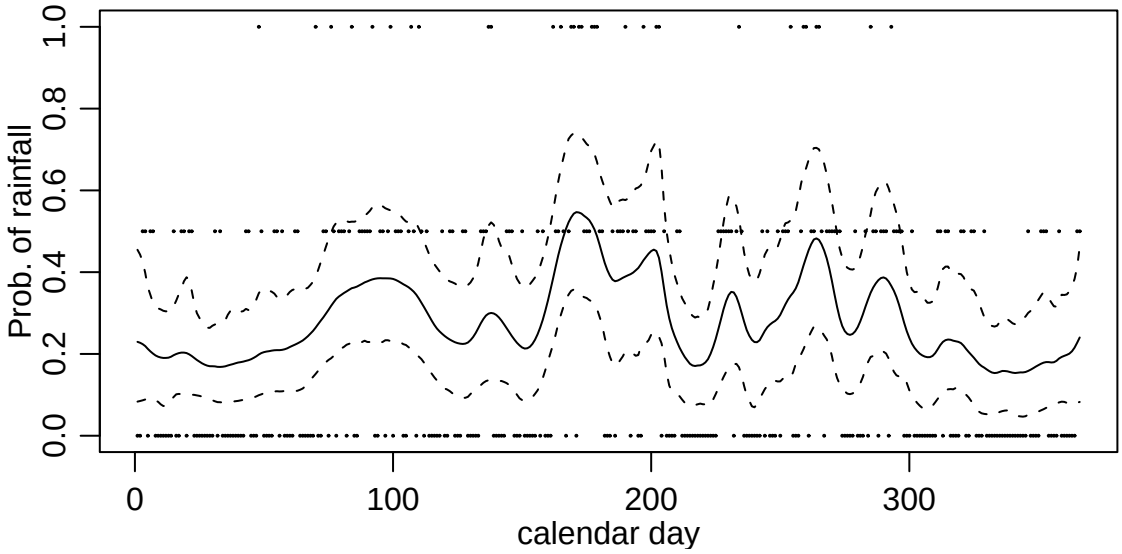

Figure 4.21. Posterior mean estimate of $p(\cdot)$, the probability of rainfall as a function of calendar day, with 95% pointwise credible intervals. Dots are empirical probabilities of rainfall based on the two binomial trials.

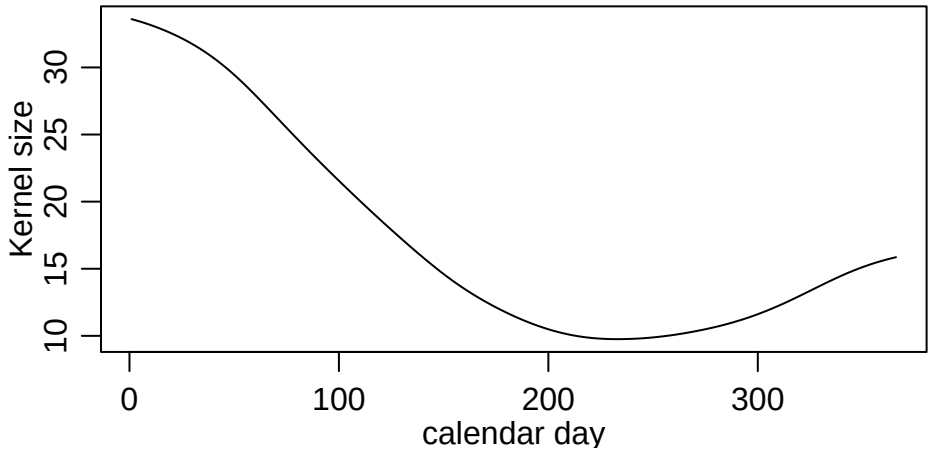

Figure 4.22. Posterior geometric mean kernel size as a function of calendar day. The kernel sizes are plotted as the square roots of the geometric means of the kernel eigenvalues, and hence can be thought of as correlation scale parameters, with units of days.

4.7 Discussion

The results for the nonstationary GP model in fitting nonparametric regression models are mixed. In one dimension, the nonstationary GP does indeed improve the fit compared to a stationary GP model when the true function is inhomogeneous, as for examples 2 and 3 of DiMatteo et al. (2002). However, for very sharp jumps, such as in example 3, the smoothness constraints on the kernel structure cause the nonstationary model to undersmooth in the vicinity of the jump. While the nonstationary GP method generally outperforms some earlier spline methods, in all three one-dimensional examples examined here, the free-knot spline method, BARS, which employs a locality heuristic for locating knots, outperforms the nonstationary GP method. For one-dimensional problems, I would suggest the use of BARS rather than a GP model, although for functions that are not too inhomogeneous, the nonstationary GP method performs quite well. An added advantage of BARS is that it is much faster than the GP method and allows one to easily compute the function estimate in closed form based on the linear model representation conditional on the location of the knots.

The story changes somewhat in higher dimensions. Here I have compared the GP methods to free-knot spline methods that generalize the usual one-dimensional spline representation. If one is willing to use an additive model, then an additive model version of BARS is probably one's best choice. In particular, based on some incomplete experimentation, I believe the nonstationary GP method has trouble ignoring unimportant covariates, and so should probably be employed only in situations in which one has reason to believe that all or most of the covariates are important. Employing a non-additive model such as the multivariate spline models or the GP models is likely to be most useful if important interactions are present. The evidence presented in this chapter suggests that GP models are effective competitors to the spline-based methods, provided the sample size and number of covariates are not too large, allowing the GP model to be fit in reasonable time. Additional comparisons with standard nonparametric regression models such as kernel smoothers, wavelets, radial basis function networks, and neural networks, if successful, would strengthen the case for the nonstationary GP model. It would also be useful to compare model performance excluding possible boundary effects by assessing in the interior portion of the covariate space, although as the covariate dimension increases, the importance of good estimation near the boundaries

increases as well.

In more standard regression problems, to contrast with what one might think of as surface-fitting problems, nonstationarity in higher dimensions may not be an important concern; in particular the curse of dimensionality limits our ability to detect such features of the data even if they are present. This suggests that a stationary GP model may be a good choice. For two of the three examples here, which seem to be relatively homogeneous, the stationary GP model outperforms the spline-based models, and in the ozone example, the performance is comparable to the spline-based model. The stationary GP model is also simpler conceptually and in its parameterization than the nonstationary GP model or the spline-based models. The MCMC methods outlined in Chapter 3, possibly in conjunction with the work of Christensen et al. (2003) will help in fitting the stationary model when the likelihood is not normal. However, full MCMC is still slow and subject to poor mixing. Thoughtful choices about hyperparameters that can be fixed in some way, such as via an empirical Bayes approach, may be useful. The relative success of the stationary GP model should not be too surprising, given that work in the machine learning literature has reported success with the stationary model on a variety of tasks. For example, Rasmussen (1996) reports that, along with a Bayesian neural network model, GP models were the most successful of the models he used for a variety of regression problems.