The Report Committee for Shuling Guo Malloy
Certifies that this is the approved version of the following report:

Nonparametric Regression Analysis

APPROVED BY
SUPERVISING COMMITTEE:

Supervisor:

Lizhen Lin

Maggie Myers

Nonparametric Regression Analysis

by

Shuling Guo Malloy, Ph.D., M.E., B.E.

Report

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science in Statistics

The University of Texas at Austin

May 2015

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Dr. Lizhen Lin for her guidance and encouragement throughout the course of this report. It would be impossible for me to finish this report without her in-depth knowledge and patient guidance. I would also like to thank Dr. Maggie Myers, the reader of this report, for her patience and valuable comments.

Secondly, I would like to thank all the professors who have taught my courses, including Dr. Mathew Hersh, Dr. Mary Parker, Dr. Maggie Myers, Dr. Karen Wickett, and Dr. Michael Mahometa, Dr. Joydeep Ghosh and Dr. Carlos Carvalho. Thank you all for leading me into the world of statistics.

My special thanks are given to Dr. Vicki Keller for her endless support and valuable advice.

Finally I would like to thank my family and friends who are always there for me.

Abstract

Nonparametric Regression Analysis

Shuling Guo Malloy, M.S. Stat

The University of Texas at Austin, 2015

Supervisor: Lizhen Lin

Nonparametric regression uses nonparametric and flexible methods in analyzing complex data with unknown regression relationships by imposing minimum assumptions on the regression function. The theory and applications of nonparametric regression methods with an emphasis on kernel regression, smoothing spines and Gaussian process regression are reviewed in this report. Two datasets are analyzed to demonstrate and compare the three nonparametric regression models in R.

Table of Contents

List of Tables

List of Figures

1. Introduction

Regression analysis focuses on estimating the relationships among variables. The aim of a regression analysis is to produce a reasonable analysis to the unknown response function m, where for n data points (X_i, Y_i), in a regression model:

$$Y_i = f(X_i) + \varepsilon_i, \qquad i = 1, \dots n$$

Many techniques for carrying out regression analysis have been developed in the literature. Familiar methods such as linear regression and logic regression are parametric, in that the regression function is defined in terms of a finite number of unknown parameters that are estimated from the data. Parametric regression is about finding the optimum parameters from data by assuming a parametric model. Parametric regression methods are global methods since all data instances affect the final global estimate [1].

Nonetheless, there are cases when parametric regression is not desirable. For example, the data can be generated from a very complicated model with large number of parameters; a model for distribution densities can't be assumed. Assuming a wrong model leads to inconsistent estiamtes and large errors. In these cases, a preselected parametric model might be too restricted or too low-dimensional to fit unexpected features, whereas the nonparametric approach offers a flexible alternative in analyzing unknown regression relationships.

Nonparametric regression is a form of regression analysis in which the predictor does not take a predetermined form but is constructed according to information derived from the data [2]. Unlike parametric approaches where the function m is fully described by a finite set of parameters, nonparametric modeling accommodates a very flexible form of the regression curve. Nonparametric estimation methods are instance-based or memory-based methods, since nonparametric estimation is affected only by nearby instances and local models are created as needed.

The theory and applications of nonparametric regression methods with an emphasis on kernel regression, smoothing spines and Gaussian process regression are reviewed in this report. Two datasets are analyzed to demonstrate and compare the three nonparametric regression models in R.

2. Nonparametric regression

The nonparametric regression approach has four main purposes [3]. First, it provides a versatile method of exploring a general relationship between variables. Second, it gives predictions of observations yet to be made without reference to a fixed parametric model. Third, it provides a tool for finding spurious observations by studying the influence of isolated points. Fourth, it constitutes a flexible method of substituting for missing values or interpolating between adjacent X values. Nonparametric regression can be used as a benchmark for linear models against which to test the linearity assumption. Nonparametric regression also provides a useful way to enhance scatterplots to display underlying structure in the data.

A reasonable approximation to the regression curve f(x) will be the mean of response variables near a point x. This local averaging procedure can be defined as

$$\hat{f}(x) = n^{-1} \sum_{i=1}^{n} W_{ni}(x) Y_i$$

Every smoothing method can be described this way. The amount of averaging is controlled by a smoothing parameter [4]. The choice of smoothing parameter is related to the balances between bias and variance.

2.1 Kernel regression

2.1.1 Model

Kernel regression as a nonparametric technique involves weighting each neighboring data point according to a kernel function giving a decreasing weight with distance and then computing a weighted local mean of linear or polynomial regression model [5]. The primary tuning parameter is the bandwidth of the kernel function, which is generally specified in a relative fashion so that the same value can be applied along all predictor variables. Larger bandwidths result in smoother functions. The smoothing parameter can be chosen using generalized cross-validation (GCV) methods [4]. A further discussion on choice of the bandwidth parameter is given in section (4) below. The form of the kernel function is of secondary importance.

Kernel smoothing describes the shape of the weight function by a density function *K* with a scale parameter that adjusts the size and the form of the weights near x. The kernel function K is a continuous and bounded (usually symmetric around zero) real function which integrates to 1. Most common kernel functions are listed in Table. 2.1.

Uniform kernel:	$K(z)$	$= 0.5$	for $\|z\| \leq 1$
		$= 0$	for $\|z\| > 1$
Epanechnikov kernel:	$K(z)$	$= 0.75(1-z^2)$	for $\|z\| \leq 1$
		$= 0$	for $\|z\| > 1$
Gaussian (normal) kernel:	$K(z)$	$= 1/\text{sqrt}(2\pi)\exp(-z^2/2)$	
Quartic (biweight) kernel:	$K(z)$	$= 15/16\ (1-z^2)^2$	for $\|z\| \leq 1$
		$= 0$	for $\|z\| > 1$
Triweight kernel:	$K(z)$	$= 35/32\ (1-z^2)^3$	for $\|z\| \leq 1$
		$= 0$	for $\|z\| > 1$

Table. 2.1: The most common kernel functions.

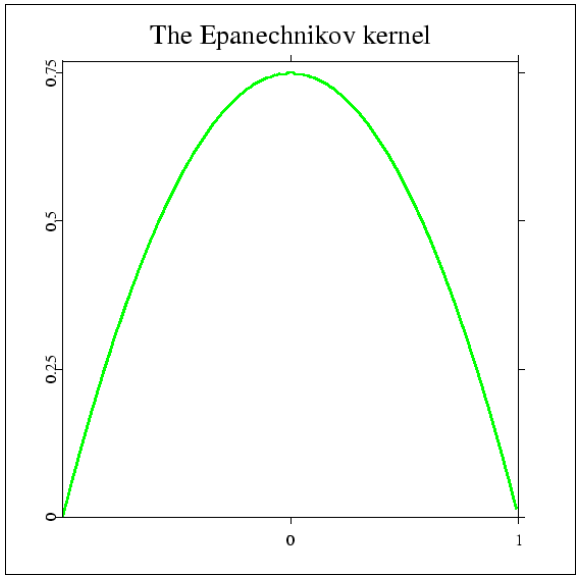

Figure 2.1: The Epanechnikov kernel $K(u) = 0.75(1-u^2)\ I(|u| <= 1)$.

Nadaraya and Watson (1964) proposed to estimate f(x) as a locally weighted average, using a kernel as a weighting function [6]. The Nadaraya-Watson estimator is:

$$\hat{f}_h(x) = \frac{n^{-1}\sum_{i=1}^{n} K_h(x - X_i)Y_i}{n^{-1}\sum_{i=1}^{n} K_h(x - X_i)}$$

where K is a kernel with a bandwidth *h*. The fraction is a weighting term with sum 1. The choice of the kernel *K* is not too important. The bandwidth *h* controls the amount of smoothing. In general the bandwidth depends on the sample size (*h*n).

2.1.2 Statistical inferences

1) Consistency

Assume the predictors or the covariates X are random, in general, for an *i.i.d.* sample of the random variable X, for any value x_0, $\hat{f}(x_0)$ is a biased estimate of $f(x_0)$. The bias goes to zero if $h \to 0$ as $N \to \infty$. The bias depends on h, the curvature of f(.), and K(.):

$$bias(\hat{f}(x_0)) = E[\hat{f}(x_0)] - f(x_0) = \frac{1}{2}h^2 f''(x_0)\int_{-\infty}^{\infty} z^2 K(z)dz$$

The size of this bias is $O(h^2)$. Assuming that $h \to 0$ as $N \to \infty$, the variance of $\hat{f}(x_0)$ is:

$$\mathrm{Var}[\hat{f}(x_0)] = [1/(Nh)]\, f(x_0) \int (K(z))^2 dz + o(1/Nh)$$

The variance depends on the N, *h*, f(.) and K(.). It will go to 0 as $Nh \to \infty$, so *h* must converge to 0 at a slower rate than N goes to ∞.

The above results were derived by approximating integrals by a Taylor expansion of f(x+*h*u) in the argument h*u* →0. The kernel estimator $\hat{f}(x_0)$ is pointwise consistent at any point x_0 if both the variance and bias disappear as $N \to \infty$, which requires that $h \to 0$ and $Nh \to \infty$. The uniform convergence (stronger) property holds if $Nh/\ln(h) \to \infty$.

2) Asymptotic normality

Since the kernel estimator is the sample average, a central limit theorem (CLT) can be applied. Given the order of the variance, the rate of convergence is $\mathrm{sqrt}(N_h)$ as in standard regression estimates. As the estimator is biased, $\hat{f}(x_0)$ is centered around its

expectation, therefore by the CLT:

$$\text{sqrt}(Nh)\ (\hat{f}(x_0) - E[\hat{f}(x_0)]) \rightarrow^{d} N(0, f(x_0)\int (K(z))^2 dz)$$

Given the bias, $[\hat{f}(x_0) - E(\hat{f}(x_0))]$ is also asymptotically normally distributed, but with a non-zero mean.

3) Confidence Intervals

The conventional confidence intervals (C.I.) for estimates of $f(x_0)$ for any point x_0 can be obtained by using the variance formula above:

$$f(x_0) \in \hat{f}(x_0) - \text{bias}(x_0) \pm z_{\alpha/2}\ \text{sqrt}\{(1/Nh)\ \hat{f}(x_0)\ [\int (K(z))^2\ dz]\}$$

where $\text{bias}(x_0)$ is given above and $\hat{f}(x_0)$ is assumed asymptotically normal. For the problem with C.I. containing negative values, the solution is to consider constructing the C.I. by inverting a test statistic.

$$C(x) = \{f: |\hat{f}(x_0) - \text{bias}(x_0)| / [\text{sqrt}\{(1/Nh)\ \hat{f}(x_0) \int (K(z))^2\ dz]\}$$

This set must be found numerically. In practice, it is hard to calculate the bias, and there may not be a reason to calculate the C.I. for $\hat{f}(x_0)$.

4) Bandwidth

There is a genuine trade-off between bias and variance of the estimate at any given point x. In general, large *h* reduce the variance by smoothing over a large number of points, but this is likely to lead to bias because the points are "averaged" in a mechanical way that does not account for the particular shape of the distribution. In contrast, small *h* gives higher variance but have less bias. In the limit, $h \rightarrow 0$, the kernel reproduced the data.

A natural approach to deal with the trade-off between bias and variance is to minimize the MSE:

$$\text{MSE}(\hat{f}(x_0)) = \text{Var}[\hat{f}(x_0)] + [\text{bias}(\hat{f}(x_0))]^2$$

As shown in previous formulas, the bias is $O(h^2)$ and the variance is $O(1/Nh)$. Intuitively, h should be chosen to that the $(\text{bias})^2$ and the variance are of the same order. The square of the bias is $O(h^4) \Rightarrow h^4 = 1/Nh \Rightarrow h = (1/N)^{1/5}$. That is, $h = O(N^{-0.2})$ and $\text{sqrt}(Nh) =$

$O(N^{0.4})$. Since the MSE is approximated using asymptotic expansion, it is called AMSE (asymptotic MSE).

Another approach developed by Rosenblatt (1956) to find optimal bandwidth is minimizing the SSE at a very large number of hypothetical points [7]. As the number of points goes to infinity, this amounts to minimizing the mean of the integrated squared error (MISE). If the previous asymptotic approximations are used, the MISE becomes AMISE. That is, an optimal bandwidth minimizes

$$\mathrm{MISE}(h) = E[\int(\hat{f}(x_0)-f(x_0))^2\,dx_0] = E[\int \mathrm{MSE}(\hat{f}(x_0))\,dx_0]$$

Differentiating AMISE(h) with regard to h yields the optimal bandwidth:

$$h^* = \delta\,[\int (f''(x_0))^2\,dx_0]^{-0.2}\,N^{-0.2}$$

where δ depends on the kernel function used:

$$\delta = [\int (K(z))^2\,dz]^{0.2}\,[\int z^2 K(z)\,dz]^{-0.4}$$

The optimal bandwidth, h^*:

$$h^* = \delta\,[\int (f''(x_0))^2\,dx_0]^{-0.2}\,N^{-0.2}$$

The optimal bandwidth decreases (very slowly) as N increases. Then, $h^* \to 0$ as $N \to \infty$ (as required for consistency). h^* depends on δ, which is a function of the kernel K(.). For example, if K(.) is Gaussian,

$$\delta=[\int (K(z))^2\,dz]^{0.2}\,[\int z^2 K(z)\,dz]^{-0.4}=[1/(2*\mathrm{sqrt}(\pi))]^{0.2}\,[\sigma^2=1]^{-0.4}$$
$$= [1/(2*\mathrm{sqrt}(\pi))]^{0.2}\,(=.776388)$$

This result also shows that if the true density function has a lot of curvature, the bandwidth should be smaller. Since the optimal h^* is unknown, we do not know $f(x_0)$ or $f''(x_0)$. Approximations methods are required. In practice, a normal density is commonly used instead of $f(x_0)$.

The choice of the kernel matters very little, since MISE(h^*) varies little across the different kernels. Technically speaking the best kernel can be selected to minimize the AMISE. It is a calculus of variation problem, but the advantage is tiny. Since the Epanechnikov kernel (1969) is optimal, it is used to judge the efficiency of a kernel [8].

Cross-validation (CV) is another approach to find optimal bandwidth. CV

attempts to make a direct estimate of the squared error, and pick the *h* which minimizes this estimate. As MISE(*h*) is unknown, CV replaces it with an estimate. The goal is to find an estimate of MISE(*h*), and find the *h*, which minimizes this estimate.

2.2. Spline Regression

2.2.1 Regression splines

Regression can be performed on splines by estimating the regression function by fitting a *k*th order spline with knots at some pre-specified locations. A *k*th order spline is a piecewise polynomial function of degree k, that is continuous and has continuous derivatives of orders 1, … *k*-1, at its knot points. The continuity in all of their lower order derivatives makes splines very smooth. The most common case considered in practice is $k = 3$, cubic splines. It is claimed that a cubic spline is so smooth that the discontinuity at the knots cannot be noticed by human eyes. The discovery that splines could be used in place of polynomials occurred in the early twentieth century. Splines have since become one of the most popular ways of approximating nonlinear functions [9].

Considering functions of the form $\sum_{j=1}^{m+k+1} \beta_j g_j$, where $\beta_1, \ldots \beta_{m+k+1}$ are coefficients and $g_1, \ldots g_{m+k+1}$, are the truncated power basis functions for *k*th order splines over the knots $t_1, \ldots t_m$,

$$g_1(x) = 1,\ g_2(x) = x,\ \ldots\ g_{k+1}(x) = x^k,$$
$$g_{k+1+j}(x) = (x - t_j)_+^k, \quad j = 1, \ldots m.$$

Here x_+ denotes the positive part of x, i.e., $x_+ = \max\{x, 0\}$. The coefficients $\beta_1, \ldots \beta_{m+k+1}$ are just estimated by least squares. That is, $\hat{\beta}_1, \ldots \hat{\beta}_{m+k+1}$ are first found to minimize the criterion

$$\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} \beta_j g_j(x_i) \right)^2$$

then the regression spline is defined as

$$\hat{r}(x) = \sum_{j=1}^{m+k+1} \hat{\beta}_j g_j(x)$$

Regression splines are linear smoothers since the regression spline estimate at x is

$$\hat{r}(x) = g(x)^T \hat{\beta} = g(x)^T (G^T G)^{-1} G^T y$$

a weighted combination of y_i , i =1, … n. Spline regression as a classic tool can work well if good knot points t_1, … t_m are chosen, but in general choosing knots is a tricky business. One problem with regression splines is that the estimates tend to display erratic behavior, i.e., they have high variance at the boundaries of the domain. This gets worse as the order *k* gets larger. A way to remedy this problem is to force the piecewise polynomial function to have a lower degree to the left of the leftmost knot, and to the right of the rightmost knot, this is exactly what natural splines do.

A natural spline of order *k*, with knots at $t_1 < t_2 <\ldots < t_m$, is a piecewise polynomial function such that 1) function is a polynomial of degree k on each of $[t_1: t_2]$, … $[t_{m-1}: t_m]$; 2) function is a polynomial of degree (k-1)/2 on $(-\infty, t_1]$ and $[t_m, \infty)$; 3) function is continuous and has continuous derivatives of orders 1, … *k*-1 at its knots t_1, … t_m. There is a variant of the truncated power basis for natural splines (and a variant of the B-spline basis for natural splines). The m basis functions, g_1, … g_m, are only need to span the space of *k*th order natural splines with knots at t_1, … t_m.

For smoothing splines, knots don't have to be chosen. These estimators perform a regularized regression over the natural spline basis, placing knots at splines circumvent the problem of knot selection as they just use the inputs as knots, and simultaneously they control for over-fitting by shrinking the coefficients of the estimated function in its basis expansion.

2.2.2 Smoothing splines

Smoothing splines often deliver similar fits to those from kernel regression. Both have a tuning parameter: the bandwidth *h* for kernel regression, and the smoothing parameter λ for smoothing splines, which we would typically need to choose by cross validation. However, a choice of kernel is not required for smoothing splines. Smoothing splines are generally much more computationally efficient [10].

Considering functions of the form $\sum_{j=1}^{n} \beta_j g_j$, where $g_1, \ldots g_n$ are the truncated power basis functions for natural cubic splines with knots at $x_1, \ldots x_n$, the coefficients are specifically chosen to minimize

$$\|y - G\beta\|_2^2 + \lambda \beta^T \Omega \beta, \tag{1}$$

where $G \in R^{n \times n}$ is the basis matrix defined as

$$G_{ij} = g_j(x_i), \quad i, j = 1, \ldots n$$

and $\Omega \in R^{n \times n}$ is the penalty matrix defined as

$$\Omega_{ij} = \int g_i''(t) g_j''(t)\, dt, \quad i, j = 1, \ldots n$$

Given the optimal spline estimate at $\hat{\beta}$ minimizing (1), the smoothing spline estimate at x is defined as

$$\hat{r}(x) = \sum_{j=1}^{n} \hat{\beta}_j g_j(x)$$

The exact form of the penalty matrix Ω is actually not so important. The extra term $\lambda \beta^T \Omega \beta$ is called a regularization term which has the effect of shrinking the components of the solution $\hat{\beta}$ towards zero. The smoothing parameter $\lambda \geq 0$ is a tuning parameter, and the higher the value of λ, the more shrinkage. Each computed coefficient $\hat{\beta}_j$ corresponds to a particular basis function g_j. The term $\beta^T \Omega \beta$ imparts more shrinkage on the coefficients $\hat{\beta}_j$ that correspond to wigglier functions g_j. With increasing λ, the wiggly basis functions g_j are being shrunk away.

Similar to least squares regression, the coefficient $\hat{\beta}$ minimizing (1) are

$$\hat{\beta} = (G^T G + \lambda \Omega)^{-1} G^T y$$

Then smoothing splines are seen to be linear smoothers. With $g(x) = (g(x_1), \ldots, g(x_n))$

$$\hat{r}(x) = g(x)^T \hat{\beta} = g(x)^T (G^T G + \lambda \Omega)^{-1} G^T y$$

which is linear combination of the y_i, $i = 1, \ldots n$. What makes smoothing splines even more interesting is that they can be alternatively motivated directly from a functional minimization perspective. Consider minimizing over all functions f,

$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int (f''(x))^2\, dx$$

This criterion trades off the least squares error over (x_i, y_i), $i = 1, \ldots n$ with a regularization term that grows large when the second derivative of f is wiggly. Remarkably, it so happens that there is a unique function minimizing this criterion, and further, this function is exactly the cubic smoothing spline estimator $\hat{r}$.

Smoothing Splines involve fitting a sequence of local polynomial basis functions to minimize an objective function involving both model fit and model curvature, as measured by the second derivative. The smoothing parameter, λ, controls the trade-off between data fit and smoothness, with larger values leading to smoother functions but larger residuals (on the training data). The smoothing parameter can be selected through automated cross-validation, choosing a value that minimizes the average error on the withheld data. The approach can be generalized to higher dimensions.

For one input variable and one output variable, smoothing splines can basically do everything which kernel of splines can do. Their advantages of splines are their computational speed and simplicity (once the basis functions have been calculated), as well as the clarity of controlling curvature directly. Kernels however are easier to program (if slower to run), easier to analyze extend more straightforwardly to multiple variables, and to combinations of discrete and continuous variables.

2.3. Gaussian Process Regression

Bayesian nonparametric regressions are Bayesian models where the underlying finite-dimensional random variable is replaced by a stochastic process. This replacement allows much richer nonparametric modeling in a Bayesian framework. In Gaussian process regression, a Gaussian process prior is assumed for the regression curve. The errors are assumed to have a multivariate normal distribution and the regression curve is estimated by its posterior distribution. The Gaussian prior may depend on unknown hyperparameters, which are usually estimated via empirical Bayes or MCMC methods. Many methods for model selection and hyperparameter selection in Bayesian methods are immediately applicable to Gaussian processes [11].

Gaussian process regression models provide a natural way to introduce kernels

into a regression modeling framework. By careful choice of kernels, Gaussian process regression models can sometimes take advantage of structure in the data. Gaussian process regression models quantify uncertainty in predictions resulting not just from intrinsic noise in the problem but also the errors in the parameter estimation procedure [12].

A Gaussian process is a stochastic process where any finite number of random variables have a joint Gaussian distribution. Let $x \in R^d$ be a random vector, and f be a stochastic process such that $f(x) \in R$, f is the stochastic process. A Gaussian process is specified by a mean function

$$m(x) = E[f(x)]$$

and a covariance function (positive definite, a.k.a. kernel function)

$$k(x, x_0) = E[(f(x) - m(x))(f(x_0) - m(x_0))]$$

The Gaussian process is written as

$$f(x) \sim GP(m(x), k(x, x_0))$$

A draw from a GP is a function f(). A common choice of the mean function is $m(x) = 0, \forall x$. This greatly simplifies calculations without loss of generality and allows the mean square properties of the process to be entirely determined by the covariance function k. A common choice of the covariance function is $k(x, x') = \exp(-\frac{1}{2\sigma^2}||x - x'||^2)$, though other covariance functions might be more appropriate for specific tasks.

By the definition of Gaussian process, for any finite set $x_1, \ldots x_n$,

$$(f(x_1), \ldots f(x_n)) \sim N(\mu, \Sigma)$$

where

$$\mu = (m(\mathbf{x}_1), \ldots, m(\mathbf{x}_n))$$

and

$$\Sigma = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \ldots & k(\mathbf{x}_1, \mathbf{x}_n) \\ & \ldots & \\ k(\mathbf{x}_n, \mathbf{x}_1) & \ldots & k(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix}$$

Any finite subset of those random variables follow a Gaussian distribution with the mean vector and the covariance matrix determined pointwise by m and k, respectively.

Therefore, Gaussian process can be thought of as an infinite-dimensional Gaussian distribution.

Bayesian nonparametric modeling with Gaussian process is basically realized by that the prior is a GP for the infinite set of random variables x, and the posterior upon observing some finite subset of the random variables is another GP.

2.3.1 Gaussian process for regression without noise [13]

Consider the standard regression setting given a training set $\{(x_i, y_i)\}$, i = 1, ... n, where $x_i \in R^d$ and $y_i \in R$. In order to predict y_* on a test set x_*, the key assumption made is that $y_x = f(x)$, $\forall x$ (noiseless), and $f \sim GP(0, k)$ for some covariance function k. This is the prior. This implies that the finite set of training and test outputs follow a (prior) Gaussian distribution:

$$\begin{pmatrix} f_{1:n} \\ f_* \end{pmatrix} \sim N\left(0, \begin{pmatrix} K_{nn} & K_{n*} \\ K_{*n} & K_{**} \end{pmatrix}\right)$$

where K_{nn} is the n×n covariance matrix defined on the training set and K_{n*} is the n×|test| covariance matrix, and so on.

Now the noiseless values are $f_1 = y_1, \ldots f_n = y_n$, and the posterior on f is another slightly degenerate GP. For the purpose of regression, however, it is important to compute the finite-dimensional conditional distribution $p(f_*|x_*, x_{1:n}, f_{1:n})$. This follows from the property of the Gaussian distribution:

$$p(f_* \mid \mathbf{x}_*, \mathbf{x}_{1:n}, f_{1:n}) = N(K_{*n}K_{nn}^{-1}f_{1:n}, K_{**} - K_{*n}K_{nn}^{-1}K_{n*}) \qquad (2)$$

In particular, the Bayesian prediction for f_* is, i.e., the mean in (2), and the uncertainty is encoded in the covariance matrix above.

2.3.2 Gaussian Process for Regression with Noise [14]

More often, the observed output is assumed noisy:

$$y_i = f_i + \epsilon,$$

where $\epsilon \sim N(0, \sigma_n^2)$. However, the underlying f is still assumed a GP. In this case,

$$cov(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{ij}$$

The joint distribution between $y_{1:n}$ and f_* is:

$$\begin{pmatrix} y_{1:n} \\ f_* \end{pmatrix} \sim N\left(0, \begin{pmatrix} K_{nn} + \sigma_n^2 I & K_{n*} \\ K_{*n} & K_{**} \end{pmatrix}\right)$$

A similar conditioning results is

$$p(f_* \mid \mathbf{x}_*, \mathbf{x}_{1:n}, y_{1:n}) = N(K_{*n}(K_{nn} + \sigma_n^2 I)^{-1} y_{1:n}, K_{**} - K_{*n}(K_{nn} + \sigma_n^2 I)^{-1} K_{n*})$$

The above is the predictive distribution for f_*. The predictive distribution for y_* has the same mean but wider spread: its covariance can be obtained by adding $\sigma_n^2 I$ to the covariance. A quantity of interest is the marginal likelihood

$$p(y_{1:n} \mid \mathbf{x}_{1:n}) = \int p(y_{1:n} \mid f_{1:n}) p(f_{1:n} \mid \mathbf{x}_{1:n}) df_{1:n}$$

where the function values $f_{1:n}$ are integrated out (marginalized over). Using the fact that $y_{1:n} \sim N(0, K_{nn} + \sigma_n^2 I)$, we have

$$\log p(y_{1:n} \mid \mathbf{x}_{1:n}) = -\frac{1}{2} y_{1:n}^\top (K_{nn} + \sigma_n^2 I)^{-1} y_{1:n} - \frac{1}{2} \log |K_{nn} + \sigma_n^2 I| - \frac{n}{2} \log 2\pi.$$

The marginal likelihood is used for model selection, e.g., tuning the kernel bandwidth σ in k.

3. Data

3.1. Old faithful geyser dataset

The first dataset involves old faithful geyser dataset with 272 observations of waiting time between eruptions and the duration of the eruption for the Old Faithful geyser in Yellowstone National Park, Wyoming, USA [15]. The X variable here is the length of time in minutes that it takes for the geyser to erupt. The Y variable is the waiting time until the next eruption. It is believed that the waiting time depends on the eruption's length of time.

Figure 3.1 shows the scatter plot of waiting time to the next eruption against duration of the previous eruption. The relationship between the variables seems somewhat linear, but that may not be the best fit. This seemingly linear data is chosen to demonstrate the features of different nonparametric regression models to represent the relationship between the waiting time to the duration of the previous eruption.

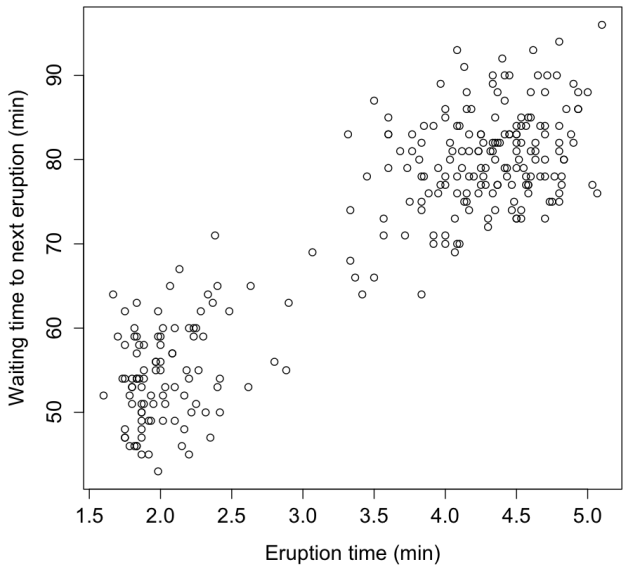

Figure 3.1: Scatter plot of waiting time to the next eruption against duration of the previous eruption for 272 observations from the old faithful geyser.

3.2. Internet traffic dataset

The second dataset is time series data about Internet traffic data with 1231 observations collected from a private ISP with centers in 11 European cities. The data corresponds to a transatlantic link hourly and was collected from 06:57 hours on 7 June to 11:17 hours on 31 July 2005. The dataset is available at https://datamarket.com/data/list/?q=cat:ecd%20provider:tsdl

Figure 3.2 is the scatter plot of Internet traffic in bits (y = Internet traffic) against the time in hour (x = time) for 1231 observations from the Internet traffic data. This example is to demonstrate how nonparametric regression models can be applied on time series data with periodic trends.

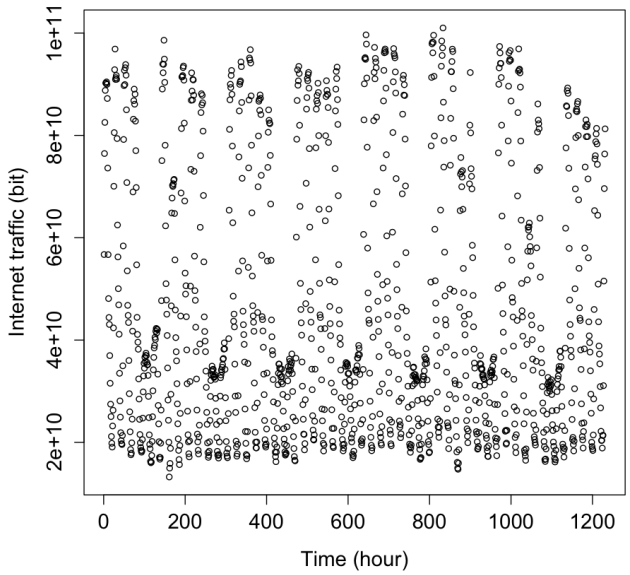

Figure. 3.2: Scatter plot of Internet traffic in bits against the time in hours for 1231 observations from the Internet traffic data.

4. Results and discussion

4.1. Kernel regression

For kernel regression, the primary tuning parameter is the bandwidth of the kernel function. The ksmooth() function in R is used to apply Nadaraya-Watson kernel regression with the choice of "normal" kernels. Figure 4.1 shows the effect of bandwidths *h* on kernel regression estimates for the geyser dataset. As indicated in Figure 4.1, a relatively small bandwidth *h* of 0.1 produces a very wiggly curve. The default bandwidth *h* of 0.5 seems a good fit curve. With increasing the bandwidth *h*, the estimates become smoother. A larger bandwidth *h* of 4 produces an overly smooth curve. It is evident that larger bandwidths *h* result in smoother functions.

Figure 4.2 shows the effect of bandwidths *h* on kernel regression estimated curve for the Internet traffic data. For very small bandwidths *h* between 0.1 and 0.5, the kernel regression estimates almost reproduce the data. With the increment of bandwidth *h*, a larger number of points are smoothed over. A larger bandwidth *h* of 6 produces an overly smooth curve. This proves again that larger bandwidth *h* results in smoother functions.

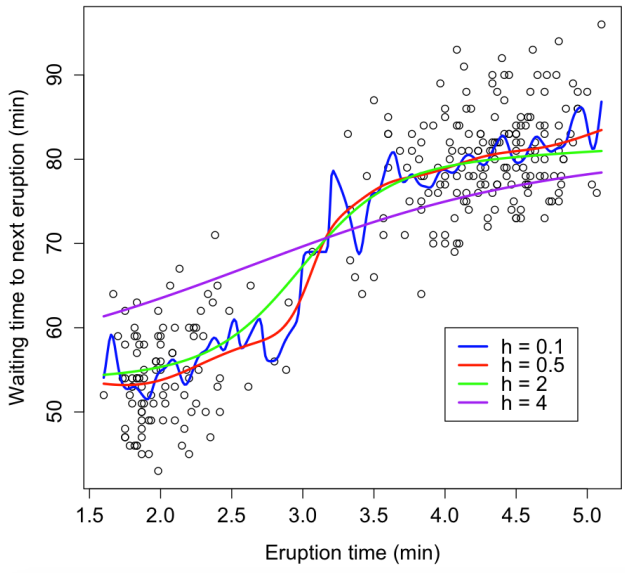

Figure. 4.1: Effect of bandwidths *h* on kernel regression estimates for faithful geyser data.

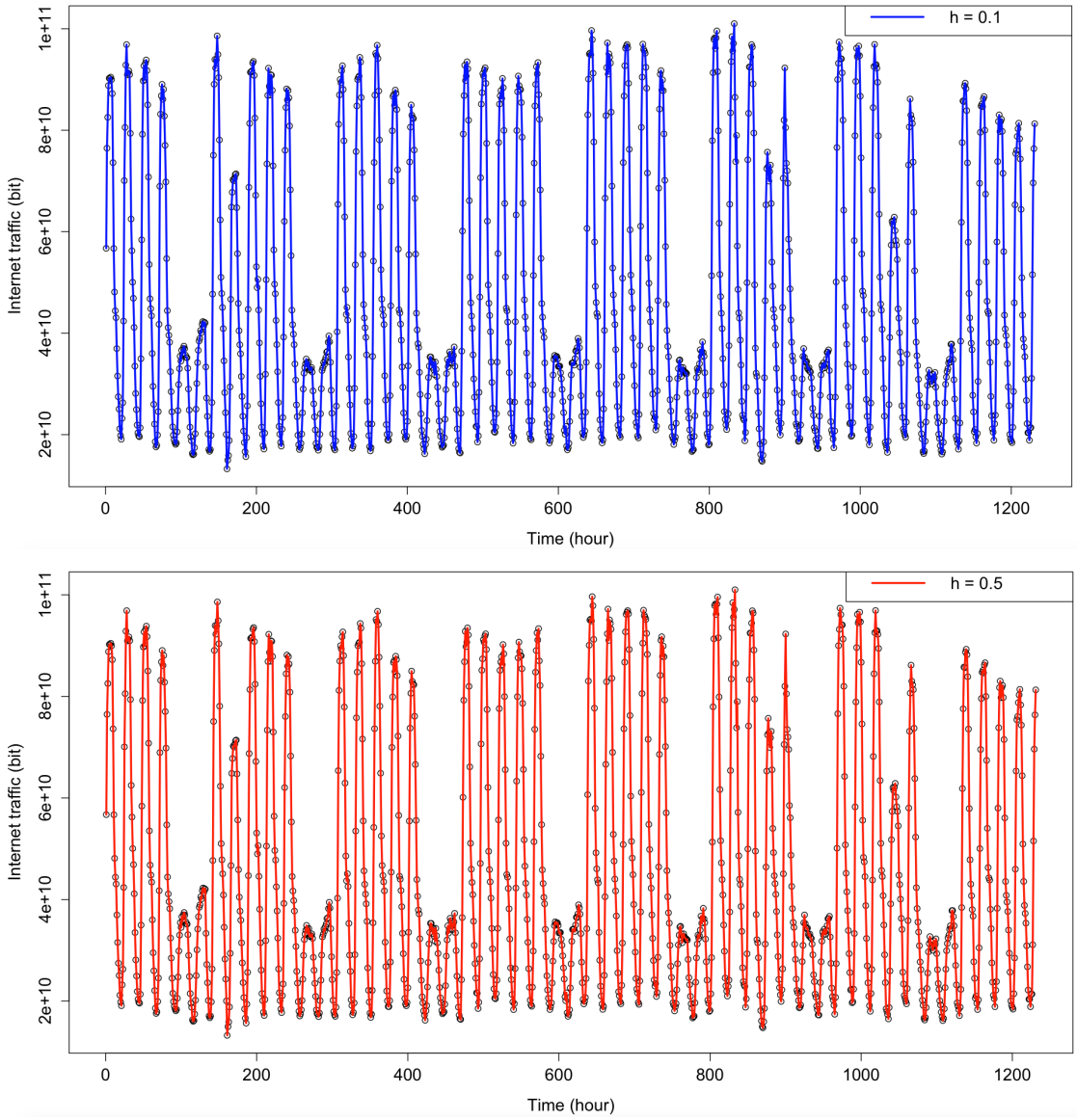

Figure. 4.2(a): Effect of bandwidths *h* on kernel regression estimates for the Internet traffic data.

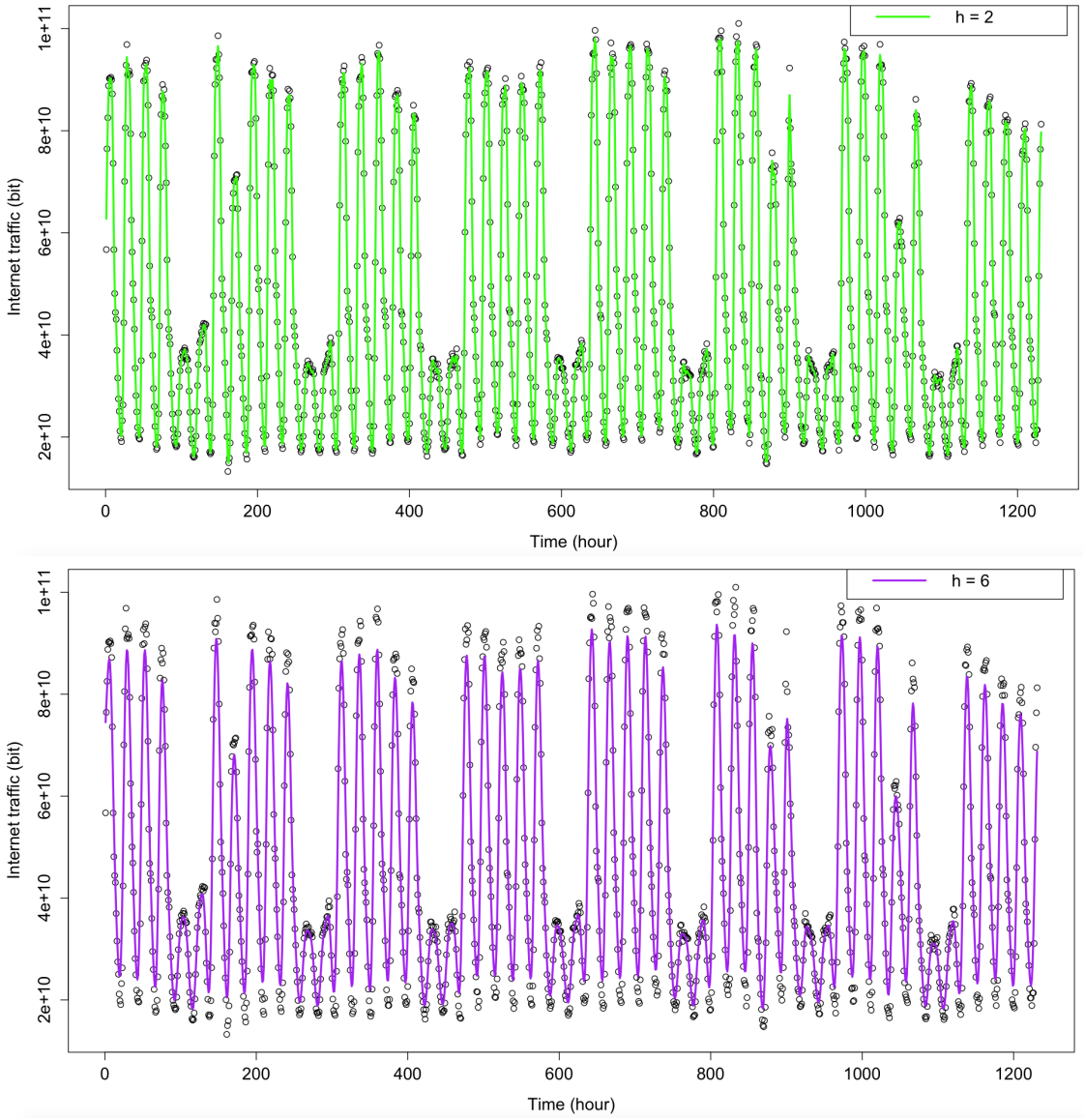

Figure. 4.2(b): Effect of bandwidths *h* on kernel regression estimates for the Internet traffic data.

The trade-off between bias and variance of the estimate always exists at any given point x. Small bandwidths give higher variance but have less bias. In contrast, high bandwidths *h* reduce the variance but increase bias because the points are "average" mechanical way that does not account for the particular shape of the distribution. An optimum bandwidth *h* can be specified by minimizing MSE or SSE. Cross validation is another way to get the optimum bandwidth *h*.

4.2. Splines

Spline regression works well on the condition that good knot points are chosen. In this report, an R package called splines is used to perform spline regression and smoothing splines. The bs( ) function is applied to produce B-splines, a computationally efficient way to compute cubic regression splines. Four and eight equally spaced knots as well as unequally spaced knots are specified for the spline regression on the geyser dataset.

Figure 4.3 shows the effect of knots on the spline regression estimates for the geyser dataset. It is obvious that the spline regression estimated curve becomes wiggly with more knots. There isn't much difference between the estimated curves with equally spaced knots or not equally spaced 4 knots. Both regression splines with equally spaced and not equally spaced knots do a good job of smoothing the data.

Figure 4.4 illustrates the effect of knots on the spline regression estimates for the Internet traffic dataset. Since the drastic change of Internet traffic in bits during every period of 24 hours and the cyclic nature of 7 days, a big number of knots are needed to have a reasonable spline regression. With fewer knots, the estimate is getting smoother. It eventually looks like a linear regression line due to too few knots. With increasing the number of knots, the smoothing spline estimated curves start to catch the weekly peak features and then daily peak features. When the number of knots is large enough, the spline regression estimate is just reproducing data.

The two examples illustrate the importance of knots for regression splines. Splines with fewer knots are generally smoother than splines with more knots. Splines

with no knots are generally smoother than splines with knots, which are generally smoother than splines with multiple discontinuous derivatives. However, increasing the number of knots usually increases the fit of the spline function to the data. Knots give the curve freedom to bend to more closely follow the data.

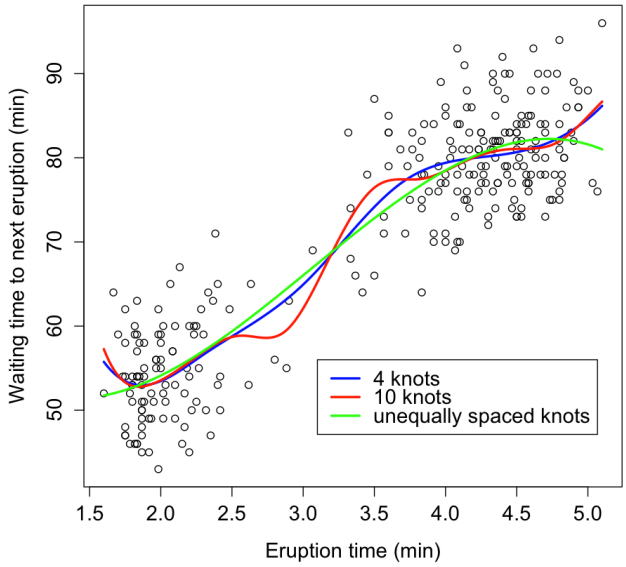

Figure 4.3: Effect of knots on spline regression estimates for the geyser dataset.

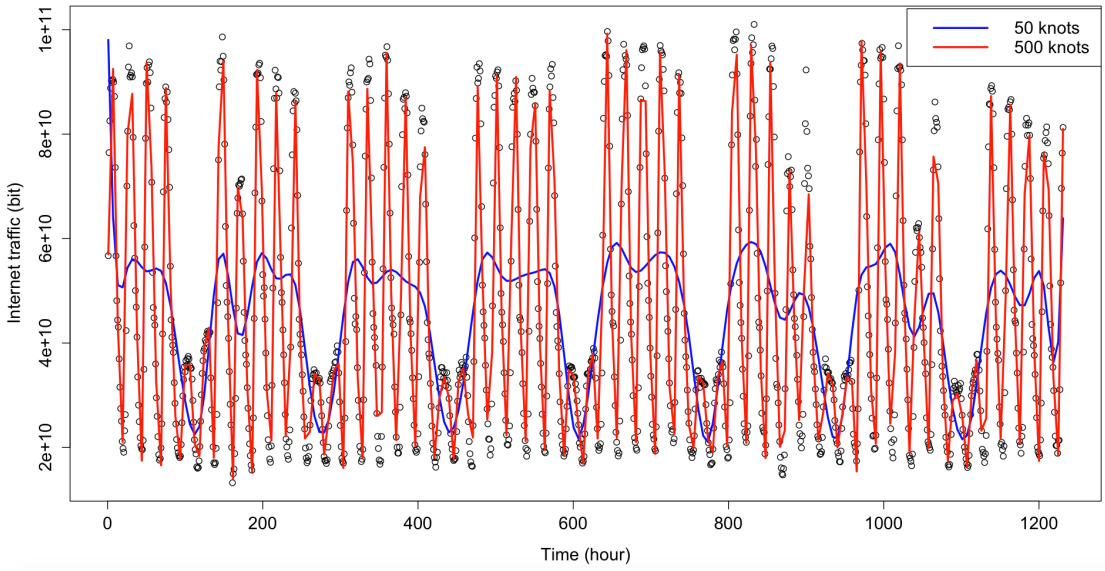

Figure 4.4: Effect of knots on spline regression estimates for the Internet traffic dataset.

A smoothing spline has a knot at each data point, but introduces a penalty for lack of smoothness. The smooth.spline() function in R is used to computes cubic smoothing splines. By default, the value of the smoothing parameter is determined by cross-validation. The default smoothing parameters are 0.9018476 and 0 for the geyser dataset and the Internet traffic dataset, respectively. Larger and smaller smoothing parameter values than the default are specified to illustrate the effect of smoothing parameters on smoothing spline estimates.

Figure 4.5 shows the effect of smoothing parameters on spline smoothing estimates for the geyser dataset. With a small smoothing parameter of 0.3, the smoothing spline is very wiggly. The smoothing spline with the default smoothing parameter of 0.9 is doing a good job. With a much larger smoothing parameter of 1.5, the estimate is overly smooth appealing as a straight line.

Figure 4.6 presents the effect of smoothing parameters on spline smoothing estimates for the Internet traffic dataset. Smoothing parameter of 0 is the default smoothing parameter due to the nature of time series data with radical changes every 24 hours. With the increasing smoothing parameter, the smoothing spline is getting more overly smooth. Smoothing spline does not work well for the Internet traffic dataset. This confirms the fact that the higher the value of smoothing parameter, the more shrinkage.

Spline smoothing quantifies the competition between producing a good fit to the data and producing a curve without too much rapid local variation. If the penalty is zero you get a function that interpolates the data points. If the penalty is infinite, you get an OLS fit to the data. Usually a nice compromise can be found somewhere in between. For data with too much rapid local variation, it is difficult to have a reasonable compromise.

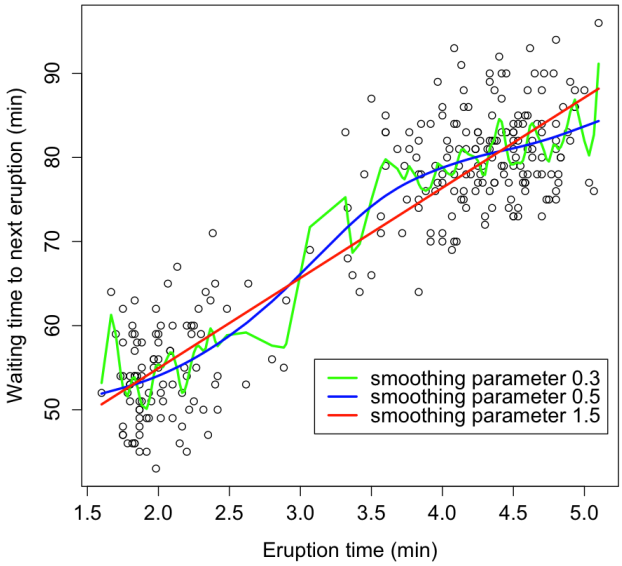

Figure. 4.5: Effect of smoothing parameters on spline smoothing estimates for the geyser dataset.

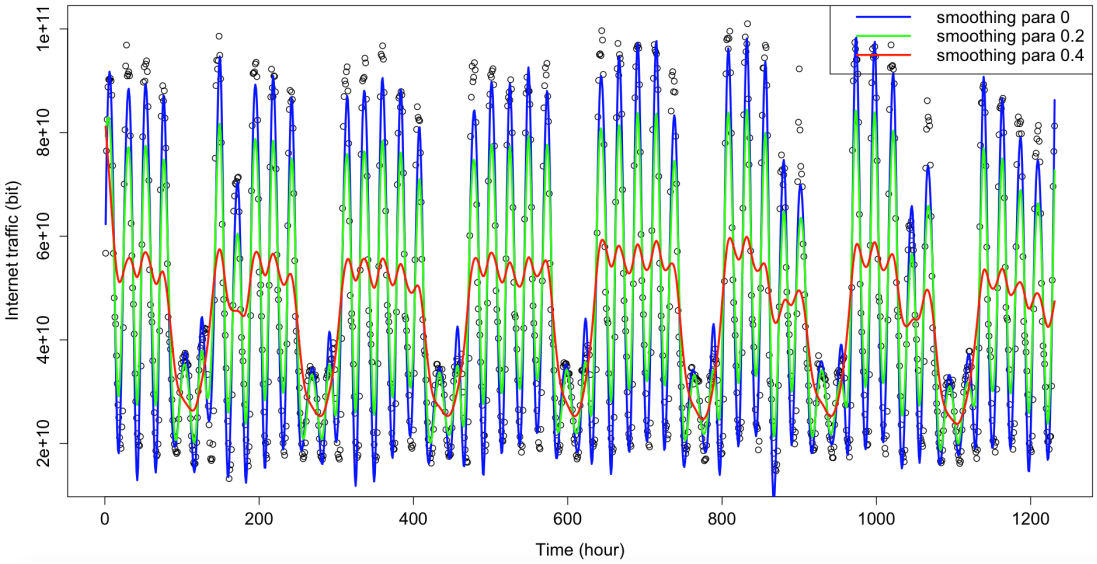

Figure. 4.6: Effect of smoothing parameters on smoothing spline estimates for the Internet traffic dataset.

4.3. Gaussian process

The R package of GPfit is used to apply Gaussian process regression on both the geyser and Internet traffic datasets. GP_fit() function uses a novel parameterization of the spatial correlation function for the ease of optimization. The deviance optimization is achieved through a multi-start L-BFGS-B algorithm [16]. GP_fit() function returns the object of class GP that contains the data set X, Y and the estimated model parameters $\widehat{\beta}$, $\hat{\sigma}^2$, $\delta_{lb}(\hat{\beta})$.

Figure 4.3.1 shows the results of a Gaussian process regression on observations of waiting time between eruptions and the duration of the eruption for the Old Faithful geyser. The prediction of the waiting time from Gaussian process regression is reasonable.

Figure 4.3.2 presents a Gaussian process regression on observations of Internet traffic and time (x) for the Internet traffic dataset. The prediction of seasonal time series data by the conventional Gaussian process regression is not satisfactory. Other covariance functions might be more appropriate for this specific task.

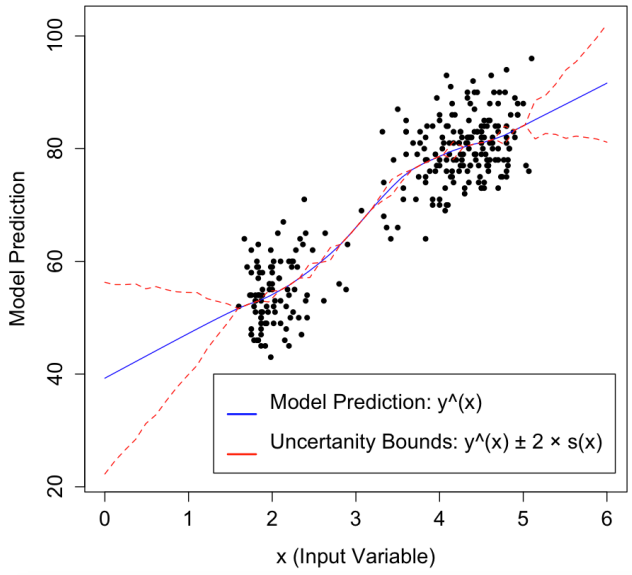

Figure 4.7: A Gaussian process regression to observations of waiting time between eruptions and the duration of the eruption for the Old Faithful geyser.

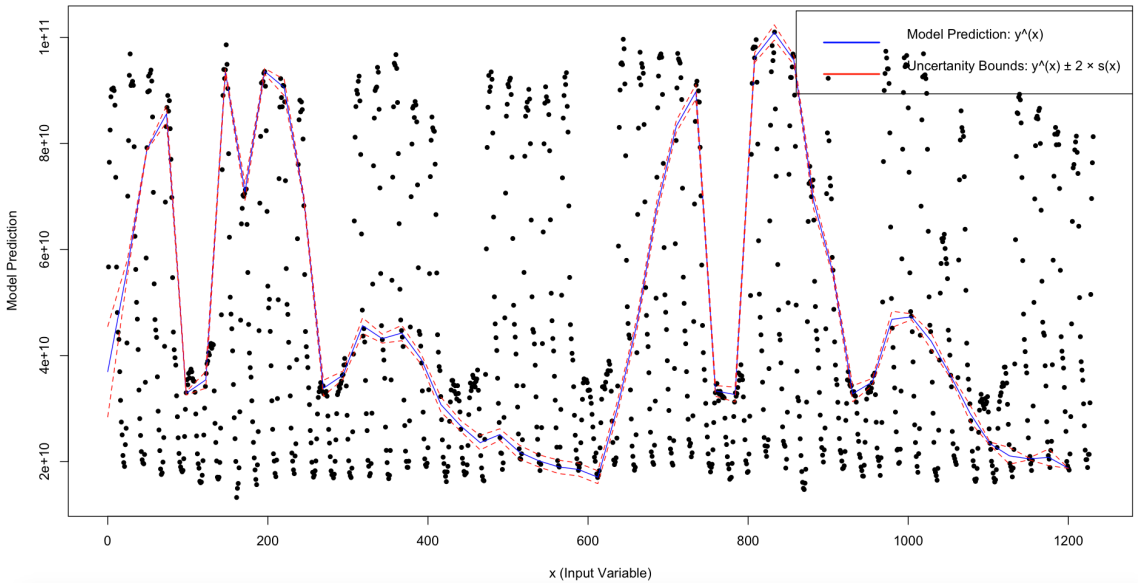

Figure 4.8: A Gaussian process regression to observations of Internet traffic and time (x) for the Internet traffic dataset.

5. Conclusions and future work

Three nonparametric regression methods, including kernel regression, smoothing spines and Gaussian process regression, have been applied on two datasets. For the seemly linear data from the geyser dataset, nonparametric regression models produce good regression estimates after optimal bandwidth *h*, knots, and smoothing parameters are specified for kernel regression, spline regression and smoothing splines, respectively. Gaussian process regression model also produces good regression estimates for the geyser dataset.

For the Internet traffic dataset, time series data with cyclic nature, kernel regression still works well. However, due to the drastic change of data with time and seasonal vibrations, it is hard to specify optimal smoothing parameters for smoothing splines in order to produce a good regression estimate. Conventional Gaussian process regression on these seasonal time series data doesn't have a satisfactory prediction. Other covariance functions are needed for this specific task.

Other Gaussian regression methods have been proposed for modeling time series data, such as Gaussian process sequences [17] and Gaussian process with change point detection [18]. In the future, those Gaussian regression models will be applied on these time series data aiming for reasonable regression estimates and good predictions. It would be ideal that a new Gaussian regression model could be developed for time series data with drastic changes seasonally.

Appendix

R code for the old faithful geyser dataset

```
x<-faithful$eruptions
y<-faithful$waiting

###Kernel regression###
# Default bandwidth of 0.5:
oldfaithful.reg.default <- ksmooth(x, y, kernel="normal")
# A smaller bandwidth of 0.1
oldfaithful.reg.smallbw <- ksmooth(x, y, kernel="normal", bandwidth=0.1)
# A larger bandwidth of 2:
oldfaithful.reg.largebw <- ksmooth(x, y, kernel="normal", bandwidth=2);
# An extreme bandwidth of 4:
oldfaithful.reg.extremebw <- ksmooth(x, y, kernel="normal", bandwidth=4);

# Plotting the estimated curve on top of the data:
par(ps = 16, cex = 1, cex.main = 1)
plot(x,y, xlab = "Eruption time (min)", ylab = "Waiting time to next eruption (min)")
lines(oldfaithful.reg.smallbw, col="blue", lwd=2.5)
lines(oldfaithful.reg.default, col="red", lwd=2.5)
lines(oldfaithful.reg.largebw, col="green", lwd=2.5)
lines(oldfaithful.reg.extremebw, col="purple", lwd=2.5)
legend(4, 60, c("h = 0.1 ","h = 0.5","h = 2", "h = 4" ), lty=c(1,1,1,1), lwd=c(2.5,2.5, 2.5,
 2.5), col=c("blue","red", "green", "purple"))

###Splines###
library(splines)
# Specifying 4 equally spaced knots:
smallnumber.knots <- 4
spacings<seq(from=min(x),to=max(x),length=smallnumber.knots+2)[2:(smallnumber.kn
 ots+1)]
smallregr.spline <- lm(y ~ bs(x, df = NULL, knots=spacings, degree = 3, intercept=T))
# Specifying 10 equally spaced knots:
largenumber.knots <- 10
spacings <-
 seq(from=min(x),to=max(x),length=largenumber.knots+2)[2:(largenumber.knots+1)]
largeregr.spline <- lm(y ~ bs(x, df = NULL, knots=spacings, degree = 3, intercept=T))
# Specifying unequally spaced knots:
uneqregr.spline <- lm(y ~ bs(x, df = NULL, knots=c(0.25, 0.5, 0.6, 0.7, 0.8, 0.9), degree
 = 3, intercept=T))

# plotting the data with the regression spline overlain:
```

```
x.values <- seq(from=min(x), to=max(x), length=200)
par(ps = 16, cex = 1, cex.main = 1)
plot(x,y, xlab = "Eruption time (min)", ylab = "Waiting time to next eruption (min)")
lines(x.values, predict(smallregr.spline, data.frame(x=x.values)), col="blue", lwd=2.5)
lines(x.values, predict(largeregr.spline, data.frame(x=x.values)),col="red", lwd=2.5 )
lines(x.values, predict(uneqregr.spline, data.frame(x=x.values)), col="green", lwd=2.5)
legend(3.1, 56, c("4 knots", "10 knots", "unequally spaced knots"), lty=c(1, 1, 1),
 lwd=c(2.5, 2.5, 2.5), col=c("blue", "red", "green"))

###(Cubic) Smoothing Splines ###
# The smooth.spline() function in R computes (cubic) smoothing splines
smoothspline.reg <- smooth.spline(x, y)

# By default, the value of the smoothing parameter is:
smoothspline.reg$spar # The default choice is 0.9018476.
#Specifying a larger smoothing parameter value:
smoothspline.reg.large <- smooth.spline(x, y, spar = 1.5)
#Specifying a smaller smoothing parameter value:
smoothspline.reg.small <- smooth.spline(x, y, spar = 0.3)

# plotting the data with the smoothing spline overlain:
par(ps = 16, cex = 1, cex.main = 1)
plot(x,y, xlab = "Eruption time (min)", ylab = "Waiting time to next eruption (min)")
lines(smoothspline.reg.small, col="green", lwd=2.5)
lines(smoothspline.reg, col="blue", lwd=2.5)
lines(smoothspline.reg.large, col="red", lwd=2.5)
legend(3.1, 56, c("smoothing parameter 0.3", "smoothing parameter 0.5", "smoothing
 parameter 1.5"), lty=c(1, 1, 1), lwd=c(2.5, 2.5, 2.5), col=c("green","blue", "red"))

###Gaussion process###
library(GPfit)
GPmodel = GP_fit(x,y);

par(ps = 16, cex = 1, cex.main = 1)
plot(GPmodel, range=c(0, 6), resolution=50, colors=c('black', 'blue', 'red'), xlab =
 "Eruption time (min)", ylab = "Waiting time to next eruption (min)")
legend(1.3, 40, c( "Model Prediction: y^(x) ", "Uncertanity Bounds: y^(x) ± 2 × s(x) "),
 lty=c( 1, 1), lwd=c( 2, 2), col=c("blue", "red"))
```

R code for the Internet traffic dataset

```
MyData <- read.csv(file="European_hour.csv", head=TRUE, sep=",")
x <- MyData$Time
y <- MyData$Intensity

###Kernal regression###
# Default bandwidth of 0.5:
internet.reg.default <- ksmooth(x, y, kernel="normal")
# A smaller bandwidth of 0.1
internet.reg.smallbw <- ksmooth(x, y, kernel="normal", bandwidth=0.1)
# A larger bandwidth of 2
internet.reg.largebw <- ksmooth(x, y, kernel="normal", bandwidth=2);
# An extreme bandwidth of 6:
internet.reg.extremebw <- ksmooth(x, y, kernel="normal", bandwidth=6);

# Plotting the estimated curve on top of the data:
par(ps = 16, cex = 1, cex.main = 1)
plot(x,y, xlab = "Time (hour)", ylab = "Internet traffic (bit)")
lines(internet.reg.default, col="red", lwd=2.5)
legend(980, 1.06e+11, "h = 0.5 ", lty=1, lwd=2.5,col= "red")

plot(x,y, xlab = "Time (hour)", ylab = "Internet traffic (bit)")
lines(internet.reg.smallbw, col="blue", lwd=2.5)
legend(980, 1.06e+11, "h = 0.1 ", lty=1, lwd=2.5,col= "blue")

plot(x,y, xlab = "Time (hour)", ylab = "Internet traffic (bit)")
lines(internet.reg.largebw, col="green", lwd=2.5)
legend(980, 1.06e+11, "h = 2 ", lty=1, lwd=2.5,col= "green")

plot(x,y, xlab = "Time (hour)", ylab = "Internet traffic (bit)")
lines(internet.reg.extremebw, col="purple", lwd=2.5)
legend(980, 1.06e+11, "h = 6 ", lty=1, lwd=2.5,col= "purple")

###Splines###
library(splines)
# Specifying 50 equally spaced knots:
smallnumber.knots <- 50
spacings
 <seq(from=min(x),to=max(x),length=smallnumber.knots+2)[2:(smallnumber.knots+1)]
smallregr.spline <- lm(y ~ bs(x, df = NULL, knots=spacings, degree = 3, intercept=T))
# Specifying 500 equally spaced knots:
largenumber.knots <- 500
spacings <-
```

```
 seq(from=min(x),to=max(x),length=largenumber.knots+2)[2:(largenumber.knots+1)]
largeregr.spline <- lm(y ~ bs(x, df = NULL, knots=spacings, degree = 3, intercept=T))

# plotting the data with the regression spline overlain:
x.values <- seq(from=min(x), to=max(x), length=200)
par(ps = 16, cex = 1, cex.main = 1)
plot(x,y, xlab = "Time (hour)", ylab = "Internet traffic (bit)")
lines(x.values, predict(smallregr.spline, data.frame(x=x.values)), col="blue", lwd=2.5)
lines(x.values, predict(largeregr.spline, data.frame(x=x.values)),col="red", lwd=2.5 )
legend(1000, 1.04e+11, c("50 knots", "500 knots"), lty=c(1, 1), lwd=c(2.5,    2.5),
 col=c("blue", "red"))

###(Cubic) Smoothing Splines ###
# The smooth.spline() function in R computes (cubic) smoothing splines
smoothspline.reg <- smooth.spline(x, y)
# By default, the value of the smoothing parameter is
smoothspline.reg$spar # The default choice is 0.
# Specify a larger smoothing parameter of 0.2:
smoothspline.reg.large <- smooth.spline(x, y, spar = 0.2)
# Specify a smoothing parameter of 0.4:
smoothspline.reg.small <- smooth.spline(x, y, spar = 0.4)

# plotting the data with the smoothing spline overlain:
par(ps = 16, cex = 1, cex.main = 1)
plot(x,y, xlab = "Time (hour)", ylab = "Internet traffic (bit)")
lines(smoothspline.reg, col="blue", lwd=2.5)
lines(smoothspline.reg.small, col="green", lwd=2.5)
lines(smoothspline.reg.large, col="red", lwd=2.5)
legend(980,1.05e+11, c("smoothing para 0", "smoothing para 0.2", "smoothing para
 0.4"), lty=c(1, 1, 1), lwd=c(2.5, 2.5, 2.5), col=c("blue","green", "red"))

###Gaussian process###
library(GPfit)
GPmodel2 = GP_fit(x,y);
plot.GP(GPmodel)
par(ps = 16, cex = 1, cex.main = 1)
plot(GPmodel2, range=c(0, 1200), resolution=50, colors=c('black', 'blue', 'red'))
legend(860, 1.05e+11, c( "Model Prediction: y^(x) ", "Uncertanity Bounds: y^(x) ± 2 ×
 s(x) "), lty=c( 1, 1), lwd=c( 2, 2), col=c("blue", "red"))
```

References

1. David A. Freedman (2005). Statistical Models: Theory and Practice. Cambridge University Press.
2. Wolfgang Hardle (1994). Applied Nonparametric Regression. Cambridge.
3. Cameron, A. and P. Trivedi (2003). Microeconometrics: Methods and Applications. Cambridge University Press.
4. Yatchew, A (2003). Semiparametic Regression for the Applied Econometrician. Cambridge University Press.
5. Green, P. J. and Silverman, B. W. (1994). Nonparametric Regression and Generalized Linear Models. CRC Press.
6. Nadaraya, E. A (1964). "On Estimating Regression". Theory of Probability and its Applications 9 (1): 141–2.
7. Simonoff, Jeffrey S. (1996). Smoothing Methods in Statistics. Springer.
8. Ruppert, David; Wand, M. P. and Carroll, R. J. (2003). Semiparametric Regression. Cambridge University Press.
9. De Boor, C. (2001). A Practical Guide to Splines (Revised Edition). Springer.
10. Wahba, G (1990). Spline Models for Observational Data. SIAM. Philadelphia.
11. C. E. Rasmussen and C. K. I. Williams (2006), Gaussian Processes for Machine Learning, MIT Press, 2006.
12. Rasmussen, C. E. (2004). "Gaussian Processes in Machine Learning". Advanced Lectures on Machine Learning. Lecture Notes in Computer Science 3176.
13. Seeger, Matthias (2004). "Gaussian Processes for Machine Learning". International Journal of Neural Systems 14 (2): 69–104.
14. Rasmussen, C.E.; Williams, C.K.I (2006). Gaussian Processes for Machine Learning. MIT Press.
15. Azzalini, A. and Bowman, A. W. (1990). A look at some data on the Old Faithful geyser. Applied Statistics 39, 357-365.

16. MacDonald, B., Ranjan, P., & Chipman, H. (2013). GPfit: an R package for Gaussian process model fitting using a new optimization algorithm. arXiv preprint arXiv:1305.0759.
17. Liu, Z., Wu, L., & Hauskrecht, M. (2013). Modeling clinical time series using Gaussian process sequences. In SIAM International Conference on Data Mining (SDM) (pp. 623-631).
18. Turner, R. D. (2012). Gaussian processes for state space models and change point detection (Doctoral dissertation, University of Cambridge).