Handout on Nonparametric Regression

Richard L. Smith

April 23, 2019

Like the other handouts I have produced for STOR 556, this is intended to supplement the corresponding mateiral in the course text [2], but also to introduce at least one complex example of my own. The relevant part of [2] is Chapter 14.

1 Introduction

The idea is illustrated by Figure 1. The Old Faithful dataset is concerned with waiting times and length of eruptions of the Old Faithful geyser in Yellowstone National Park: specifically, using the length of an eruption as a predictor of the waiting time until the next eruptions (both measured in minutes). The straight line regression (in red) at first seems a reasonable predictor, but closer look at the data shows there are two distinct clusters of observations and a nonlinear fit, such as the one in blue, could well be a better predictor. This chapter discusses various methods of constructing such a nonlinear regression curve and the issues that go into such constructions.

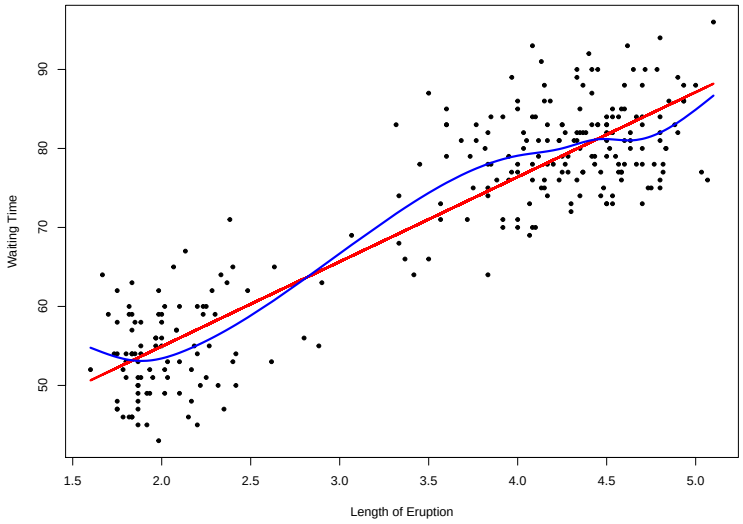

Figure 1: Scatterplot of Old Faithful data with linear and nonlinear regression lines

2 Nadaraya-Watson Estimator

The basic idea is to smooth the data using a kernel by defining

$$f_\lambda(x) = \frac{\sum_{i=1}^n w_i y_i}{\sum_{i=1}^n w_i}, \quad w_i = \frac{1}{\lambda} K\left(\frac{x - x_i}{\lambda}\right). \tag{1}$$

Here K is known as the *kernel function* and could in principle be any non-negative function that integrates to 1; a common choice is the standard normal density function; an alternative which for both computational and statistical efficiency reasons is often thought better is the *Epanechnikov kernel* defined by $K(x) = \frac{3}{4}(1 - x^2)$ for $|x| < 1$ and $K(x) = 0$ for $|x| > 1$. A more critical parameter in practice is the *bandwidth* parameter λ, which is at the choice of the analyst: a small λ tends to produce a very jagged plot with too many spurious changes of slope, but too large a λ will produce an oversmoothed plot that does not pick up the nonlinear features of the data.

This method may be implemented with the R function `ksmooth` where the user defines the bandwidth `bw`. An alternative is to choose the bandwidth automatically by *cross-validation*: one way to implement this is through the function `sm.regression` in the Bowman-Azzalini package `sm`.

3 Splines

Two forms considered:

3.1 Smoothing Splines

Basic idea: choose a function f to minimize a sum of squares criterion subject to a *penalty function* designed to enforce smoothness. This leads to the criterion of choosing f to minimize:

$$\frac{1}{n}\sum (y_i - f(x_i))^2 + \lambda \int (f''(x))^2 dx \tag{2}$$

where λ plays the role of a smoothing parameter here (larger λ means a bigger penalty hence a smoother estimated $\hat{f}$).

It's also possible to use alternatives to least squares, generalizing (2) to:

$$\frac{1}{n}\sum \rho(y_i - f(x_i)) + \lambda \int (f''(x))^2 dx \tag{3}$$

for some function ρ. For example, $\rho(x) = |x|$ in (3) would be a more robust choice than (2).

It is known from mathematical theory that the solution $\hat{f}$ of (2) is a piecewise polynomial in which $\hat{f}$, $\hat{f}'$, $\hat{f}''$ are all continuous functions. Given this, it's possible to define $\hat{f}$ in terms of its coefficients and thereby to reduce (2) to a function of its polynomial coefficients, which is therefore a finite-dimensional optimization problem. (To the best of my knowledge, there's no equivalent simplification of (3), but [2] doesn't discuss any implementation of this method.)

The R function `smooth.spline` illustrates this for the Old Faithful data. However, the text also illustrates and example (`exb` dataset) where the cross-validation smoother makes no sense.

Moral of the story: Cross-validation is useful technique for suggesting a stable smoothing parameter and thereby avoiding the subjectivity of choosing λ only after looking at the plots. However, it doesn't always work and, in the end, it's always up to the analyst to choose the smoothing parameter she wants to use.

3.2 Regression Splines

Regression splines are also based on piecewise polynomial functions, but instead of putting a *knot* (where the coefficients change) at every datapoint, it only uses a subset of knots at pre-selected positions. The number of knots then controls the smoothness of the fit: fewer knots means a smoother fit.

As an example, here is the R code by which I drew Figure 1.

```
library(faraway)
data(faithful)
par(mfrow=c(1,1),cex=1.1)
plot(waiting~eruptions,pch=20,ylab='Waiting Time',xlab='Length of Eruption',faithful)
# initial straight line fit
lines(faithful$eruptions,lm(waiting~eruptions,faithful)$fitted,lw=3,col='red')
# regression splines fit with 8 knots
library(splines)
erupspl=ns(faithful$eruptions,8)
lines(lm(waiting[order(eruptions)]~erupspl[order(eruptions),])$fitted~sort(eruptions),
faithful,lw=3,col='blue')
```

Note the rather complicated format of the last `lines` command: because the eruptions are not sorted into increasing or decreasing numerical order, we must order them internally before plotting the nonlinear curve. An alternative would be to sort `eruptions` into increasing or decreasing order before drawing the plot: this is illustrated in the posted R code.

In this example I used the R function `ns` to define *natural splines*. An alternative is to use `bs` which gives *B-splines*. Both functions are piecewise cubic polynomials and they tend to perform rather similarly in practice, though most people think `ns` is more flexible in non-standard situations, for example if the x points are very unequally spaced or if the function behaves differently near the endpoints. Note that you must load `library(splines)` before you can use `bs` or `ns`. Faraway [2] also discusses *piecewise linear splines* which are easier to understand and to implement mathematically, but they lead to rather jagged plots that don't look so good. I recommend using `bs` or `ns` which give good results in practice, though you still have to think about the number of knots. (Note that if you dig down into the details, it's also possible for the user to specify the locations of the knots themselves, not just the number of knots, but most users in practice just specify the number of knots.)

4 Local Polynomials Method

This is known as the *locally weighted sum of squares* method (abbreviated to `lowess` or `loess`), originally proposed by Cleveland [1]. The basic idea is this: whereas (1) smoothes the data by

calculating weighted averages with weights w_i, `loess` performs a weighted regression of y_i on x_i using the same weights w_i. Since the weights are dependent on the point x for which the prediction is desired, this local regression must be repeated for every x value. That is what `loess` does conceptually: in practice, the whole algorithm runs very fast using efficient methods for solving the regression equations. An illustration is as follows:

```
with(faithful,{
plot(waiting~eruptions,col='green',pch=20)
f=loess(waiting~eruptions,span=0.25)
i=order(eruptions)
lines(f$x[i],f$fitted[i],lw=3,col='blue')
})
```

Note two things here: (a) line 4 to sort the eruptions before plotting (the same as we did for Figure 1), and (b) the `span=0.25` specification within `loess`. This is another form of smoothing parameter: `span` reflects the proportion of the data included in the regression at any point x. In this case, I think `span=0.25` undersmoothes the data, but you should try different values of `span` to see how they work. The default value of `span` (in other words, what R uses if you omit this parameter) is 0.75: however, this is a lot cruder rule than cross-validation, and I don't recommend it as an all-purpose choice. Faraway [2] suggested "the default choice is satisfactory" for the Old Faithful data but he criticizes it as too large for one of his other examples.

5 Confidence Intervals

We would like to be able to supplement these plots with some indication of the uncertainty of the estimated curve. Therefore, calculating and plotting 95% confidence intervals is a useful additional technique. Note that there are two potential types of confidence intervals we could use: *pointwise confidence intervals* are designed to have coverage probability 95% at each individual point x, whereas *simultaneous confidence intervals* are designed to provide uniform coverage across all x (in other words, so that the probability that the confidence band excludes the true function value *anywhere in the range of the plot* should be no more than 0.05). You may have seen a method for constructing simultaneous confidence banks for a simple y versus x regression line: however, the examples discussed here are only for pointwise confidence bands.

The text describes two methods, one as a direct extension of the `loess` method and the other using the `mgcv` package for splines. Here are implementations of both methods for the Old Faithful data:

```
ggplot(faithful,aes(x=eruptions,y=waiting))+geom_point(alpha=0.25)+geom_smooth(
method='loess',span=0.5)
library(mgcv)
ggplot(faithful,aes(x=eruptions,y=waiting))+geom_point(alpha=0.25)+geom_smooth(
method='gam',formula=y~s(x))
```

Note: [2] discusses an additional `k=20` parameter associated with the `gam` method of the second example. I was able to reproduce Faraway's plot for the `exa` dataset that he uses on page 308, but varying the `k` parameter doesn't seem to make any difference for the Old Faithful data. I do not know the reason for this.

5.1 Direct Calculation of Pointwise Confidence Limits

The automated methods just shown are very quick and easy to use when they can be used, but it's also useful to know how to calculate these confidence limits directly.

The model being fitted through a regression spline may be written in the form

$$y_i \quad = \quad \beta_0 + \sum_{j=1}^{k} \beta_j s_{ij} + \epsilon_i, \quad \epsilon_i \quad \sim \quad N[0, \sigma^2] \tag{4}$$

where s_{ij} is the value of j'th spline basis function at x_i, β_0 is the intercept, and $\beta_1, ..., \beta_k$ are the regression coefficients associated with the k spline basis functions. In standard regression notation, equation (4) may be written in the form $Y \sim N[X\beta, \sigma^2 I_n]$ where X is the matrix of covariates (with ones in the leading column corresponding to the intercept term of the model), β is the vector of regression coefficients and I_n is the $n \times n$ identity matrix. In this notation, it is well known that the least squares estimates of β are $\hat{\beta} = (X^T X)^{-1} X^T Y$ with covariance matrix $(X^T X)^{-1}\sigma^2$, which is standardly estimated by $(X^T X)^{-1}\hat{\sigma}^2$ where $\hat{\sigma}^2$ is the usual unbiased estimator of σ^2.

In particular, this shows that the variance of $\hat{\beta}_0 + \sum_{j=1}^{k} \hat{\beta}_j s_{ij}$, which is the estimated value of the regression response at $x = x_i$, is σ^2 multiplied by the i'th diagonal entry of the matrix $S(X^T X)^{-1} S^T$ where S is the $n \times (k+1)$ matrix

$$S \quad = \quad \begin{pmatrix} 1 & s_{11} & s_{12} & \dots & s_{1k} \\ 1 & s_{21} & s_{22} & \dots & s_{2k} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & s_{n1} & s_{n2} & \dots & s_{nk} \end{pmatrix}.$$

Therefore, the *standard errors of the estimated regression responses*, evaluated at the data points $x_1, ..., x_n$, are the square roots of the diagonal entries of the matrix $\hat{\sigma}^2 S(X^T X)^{-1} S^T$.

This suggests the following procedure for adding pointwise 95% confidence intervals to Figure 1. Note that `vcov(lm1)` is the estimated covariance matrix of the parameters in the model `lm1`, that is, $\hat{\sigma}^2 S(X^T X)^{-1} S^T$.

```
erupspl=ns(faithful$eruptions,8)
lm1=lm(waiting~erupspl,faithful)
# add ones to erupspl to allow for intercept term
erups2=cbind(rep(1,272),erupspl)
# compute pointwise standard deviations at all values of eruptions
sd_points=sqrt(diag(erups2 %*% vcov(lm1) %*% t(erups2)))
# t statistic for 95% confidence intervals (adjust confidence limit if desired)
ts=qt(0.975,df.residual(lm1))
# redraw plot with confidence lines
i=order(faithful$eruptions)
par(mfrow=c(1,1),cex=1.1)
with(faithful,{
plot(waiting~eruptions,pch=20,ylab='Waiting Time',xlab='Length of Eruption')
lines(eruptions[i],lm1$fitted[i],lw=5,col='blue')
```

```
lines(eruptions[i],lm1$fitted[i]+ts*sd_points[i],lw=3,col='green')
lines(eruptions[i],lm1$fitted[i]-ts*sd_points[i],lw=3,col='green')
})
```

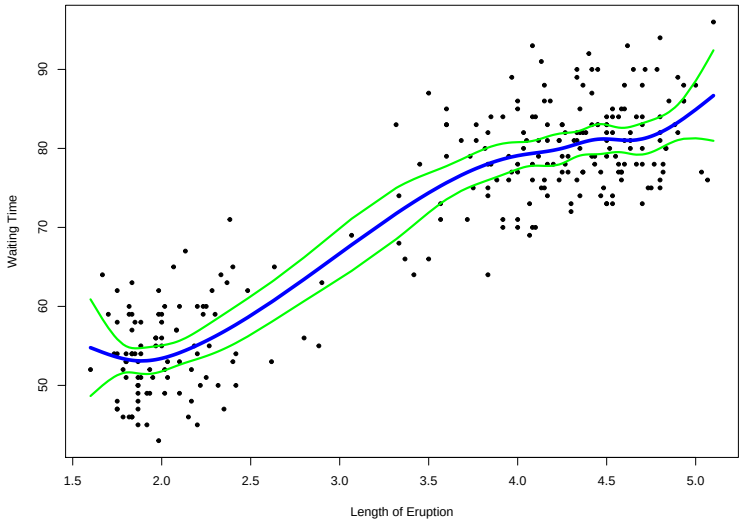

Figure 2: Old Faithful data with regression splines and pointwise 95% confidence limits

Note the broadening of the confidence bands near the ends of the data. This is, to some extent, a problem with all the methods we have discussed.

Note added after my April 23 class. As I was going through this in class, I realized there is a much quicker way to do this example: simply use the `predict` command that can be applied to any linear model object. So, replace the above code by

```
erupspl=ns(faithful$eruptions,8)
lm1=lm(waiting~erupspl,faithful)
pr1=predict(lm1,se.fit=T,interval='confidence')
i=order(faithful$eruptions)
par(mfrow=c(1,1),cex=1.1)
with(faithful,{
plot(waiting~eruptions,pch=20,ylab='Waiting Time',xlab='Length of Eruption')
lines(eruptions[i],pr1$fit[i,1],lw=5,col='blue')
lines(eruptions[i],pr1$fit[i,2],lw=3,col='green')
lines(eruptions[i],pr1$fit[i,3],lw=3,col='green')
})
```

and you get exactly the same picture.

6 Combining Nonparametric Regression with Random Effects

This would be problematic for most of the nonparametric methods we have discussed, but for one, it is entirely natural: the regression splines method. The reason is that, after deciding on the number and possibly the locations of the knots, regression splines reduce to a standard multiple regression, and are therefore amenable to the `lmer` routine.

6.1 Boston Marathon Example

Over a number of months, I assembled a dataset consisting of finish times, together with age and gender, of runners in the Boston Marathon. The objective was to see how a "typical" runner's time varies with age and gender. Most studies of age and gender effects in running focus on world or national record times, but a world or national record holder is hardly representative of most runners who enter road races. I should point out that the Boston Marathon is not quite representative, either, since the race requires qualifying times and many potential entrants might be unable to run because they could not qualify. On the other hand, there is a core of runners who run the race regularly, for whom it is therefore possible to track their variations of performance with age.

The dataset consists of 7,219 individual performances achieved by 806 different runners between years 2001 and 2013. The criterion to be included in the study was six finishes for a male runner and five for a female runner. However, since my method of searching the online results pages was very far from complete, it is certain that there are many more runners who could have been included under these criteria. Each row of the data consists of an identity number of the runner, year of performance, sex (1 for female, 2 for male), age and finish time in minutes. One of the included years was 2013, the year of the Boston bombings, when many runners were unable to finish. However, a previous study [3] estimated the finish times of all those runners and I have included those estimates in the analysis.

A plausible model for the performance of runner i in year j is

$$y_{ij} \quad = \quad \alpha_i + \gamma_j + s(a_{ij};\beta) + \epsilon_{ij} \tag{5}$$

where α_i is a zero-mean random effect for runner i; γ_j is a zero-mean random effect for year j; a_{ij} is the age of runner i in year j; ϵ_{ij} is a $N[0,\sigma^2]$ random error; and $s(a_{ij};\beta) = \beta_0 + \sum_{k=1}^{K} \beta_k s_k(a_{ij})$ where $s_k(a)$ is the k'th regression spline basis function evaluated at age a and $\beta_0, ..., \beta_K$ is the vector of fixed-effect regression coefficients. Note that we included the intercept β_0 as a fixed effect so that the eventually fitted function $s(a_{ij};\beta)$ would be correctly centered around some mean performance time. The reason for including a random effect for year was that some years (notably, 2004 and 2012) were very hot and this has an overall effect of slowing down runners' times. The model (5) was fitted separately for men and women; a possible refinement would be to further separate the runners into different ability groups to examine the extent to which $s(a_{ij};\beta)$ varies between fast and slow runners.

The results of this exercise are shown in Figure 3. The results look generally realistic but there are some peculiarities. Note, in particular, the apparent crossover of male and female running times beyond the age of 70; this is almost certainly an artifact of the fact that there are very few runners of either gender in this age range.

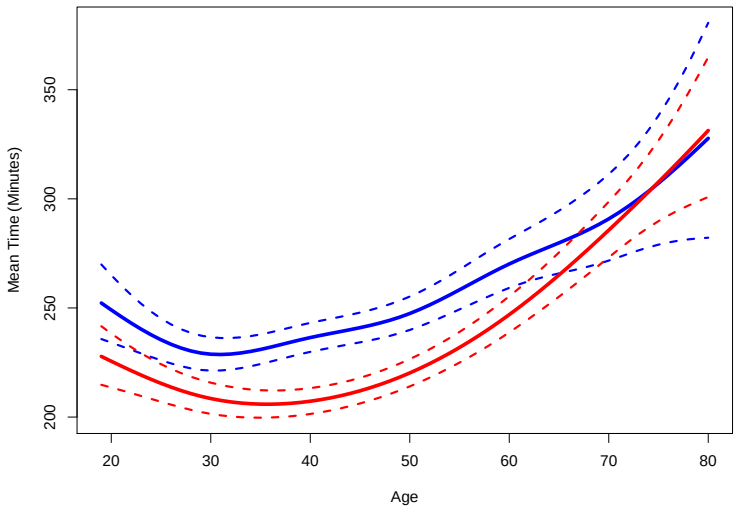

Figure 3: Nonparametric regression curves for men (red curve) and women (blue curve), together with 95% pointwise confidence limits (dashed curves).

References

[1] W. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.

[2] J.J. Faraway. *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models. Second Edition.* Chapman and Hall/CRC Press, Boca Raton, Florida, 2016.

[3] D. Hammerling, M. Cefalu, J. Cisewski, F. Dominici, G. Parmigiani, C. Paulson, and R.L. Smith. Completing the results of the 2013 Boston Marathon. *PLoS ONE*, 9(4):e93800, 2014.