Smoothing and Non-Parametric Regression

Germán Rodríguez
grodri@princeton.edu

Spring, 2001

Objective: to estimate the effects of covariates X on a response y *non-parametrically*, letting the data suggest the appropriate functional form.

1 Scatterplot Smoothers

Consider first a linear model with one predictor

$$y = f(x) + \epsilon.$$

We want to estimate f, the *trend* or *smooth*. Assume the data are ordered so $x_1 < x_2 < \ldots < x_n$. If we have multiple observations at a given x_i we introduce a weight w_i.

1.1 Running Mean

We estimate the smooth at x_i by averaging the y's corresponding to x's in a neighborhood of x_i:

$$S(x_i) = \sum_{j \in N(x_i)} (y_j)/n_i,$$

for a neighborhood $N(x_i)$ with n_i observations. A common choice is to take a symmetric neighborhood consisting of the nearest $2k+1$ points:

$$N(x_i) = \{\max(i-k,1), \ldots, i-1, i, i+1, \ldots, \min(i+k,n)\}.$$

Problems: it's wiggly, bad near the endpoints (bias). Use only for equally spaced points.

1.2 Running Line

One way to reduce the bias is by fitting a *local* line:

$$S(x_i) = \hat{\alpha}_i + \hat{\beta}_i x_i,$$

where $\hat{\alpha}_i$ and $\hat{\beta}_i$ are OLS estimates based on points in a neighborhood $N(x_i)$ of x_i. This is actually easy to do thanks to well-known regression updating formulas. Extension to weighted data is obvious. Much better than running means.

1.3 Kernel Smoothers

An alternative approach is to use a *weighted* running mean, with weights that decline as one moves away from the target value. To calculate $S(x_i)$, the j-th point receives weight

$$w_{ij} = \frac{c_i}{\lambda} d(\frac{|x_i - x_j|}{\lambda}),$$

where $d(.)$ is an *even* function, λ is a tunning constant called the window width or bandwidth, and c_i is a normalizing constant so the weights add up to one for each x_i. Popular choices of function $d(.)$ are

- Gaussian density,
- Epanechnikov: $d(t) = \frac{3}{4}(1 - t^2), t^2 < 1$, 0 otherwise,
- Minimum var: $d(t) = \frac{3}{8}(3 - 5t^2), t^2 < 1$, 0 otherwise.

One difficulty is that a kernel smoother still exhibits bias at the end points. Solution? Combine the last two approaches: use kernel weights to estimate a running line.

1.4 Loess/Lowess

One such approach is *loess*, a locally weighted running line smoother due to Cleveland and implemented in S and R. To calculate $S(x_i)$ you basically

- find a symmetric nearest neighborhood of x_i,
- find the distance from x_i to the furthest neighbor and use this as λ,
- use a tri-cube weight function $d(t) = (1 - t^3)^3, 0 \leq t \leq 1$, 0 otherwise,
- estimate a local line using these weights, take the fitted value at x_i as $S(x_i)$.

A variant uses robust regression in each neighborhood.

1.5 Other Approaches

Splines are a popular family of smoothers. We will study splines in the next section.

All of the methods discussed so far are *linear* smoothers, we can always write

$$S(x) = Ay$$

where S and y are n-vectors and A is an $n \times n$ matrix that depends on the x's.

There are also *non-linear* smoothers. These are usually based on running medians, followed by enhacements such as Hanning, splitting, and twicing. Popularized by Tukey, two of the best known smoothers are called 3RSSH and 4253H, where the nomenclature indicates the span of the running medians (4,2,5,3) and the enhacements (H for Hanning). We will not discuss these.

2 Splines

A *spline* is a piece-wise polynomial with pieces defined by a sequence of *knots*

$$\xi_1 < \xi_2 < \ldots < \xi_k$$

such that the pieces join smoothly at the knots.

The simplest case is a linear spline.

For a spline of degree m one usually requires the polynomials and their first $m-1$ derivatives to agree at the knots, so that $m-1$ derivatives are continuous.

A spline of degree m can be represented as a power series:

$$S(x) = \sum_{j=0}^{m} \beta_j X^j + \sum_{j=1}^{k} \lambda_j (x - \xi_j)_+^m$$

where the notation

$$(x - \xi_j)_+ = \begin{cases} x - \xi_j, x > \xi_j \\ 0, \text{otherwise} \end{cases}$$

Example: Here is a linear spline, with one knot:

$$S(x) = \beta_0 + \beta_1 x + \gamma(x - \xi)_+.$$

The most popular splines are cubic splines:

$$S(x) = \beta_o + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{j=1}^{k} \gamma_j (x - \xi_j)_+^3.$$

We will focus on these.

2.1 Interpolating Splines

Supose we know the values of a function at k points $x_1 < \ldots < x_k$ and would like to interpolate for other x's.

If we were to use a spline of degree m with knots at the observed x's, we would have $m + 1 + k$ parameters to estimate with only k observations. Obviously we need some restrictions.

2.1.1 Natural Splines

A spline of odd degree $m = 2\nu - 1$ is called a *natural* spline if it is a polynomial of degree $\nu - 1$ outside the range of the knots (i.e. below ξ_1 or above ξ_k).

A natural *cubic* spline is linear outside the range of the data. For a natural spline

$$\begin{array}{ll} \beta_j = 0 & \text{for } j = \nu, \ldots, 2\nu - 1 \\ \sum_{i=1}^{k} \gamma_i \xi_i^j = 0 & \text{for } j = 0, 1, \ldots, \nu - 1. \end{array}$$

This imposes exactly $m + 1$ restrictions, so we have k parameters left. Note that a natural cubic spline has the form

$$S(x) = \beta_0 + \beta_1 x + \sum_{j=1}^{k} \gamma_j (x - \xi_j)_+^3,$$

subject to the restrictions

$$\sum \gamma_j = 0 \qquad \text{and} \qquad \sum \gamma_j \xi_j = 0$$

so we end up with k parameters.

2.1.2 Restricted Splines

An alternative is to introduce other boundary restrictions. For example one could consider points $a < x_1$ and $b > x_n$ and fix the value of the spline and the value of some of its derivatives at a and b.

For a cubic spline we need 4 restrictions, so we could fix the values and the first two derivatives:

$$S(a), S'(a), S(b), S'(b),$$

usually to things like 0. (For example one could assume fertility at ages 12 and 50 is zero and is not changing. See McNeil et al in the readings for just such an example.)

Estimation of the parameters of either a natural or restricted interpolating spline amounts to solving a simple system of linear equations.

2.2 Spline Regression

Consider now the problem of *smoothing* a scatterplot, as opposed to interpolating.

One approach is to select s suitable set of knots with $k << n$ (that means k substantially less than n), and then fit a spline by OLS (or WLS, or maximum likelihood).

For a cubic spline, this amounts to regressing y on $k + 4$ predictors, namely

$$1, x, x^2, x^3, (x - \xi_1)^3_+, (x - \xi_2)^3_+, \ldots, (x - \xi_k)^3_+$$

For a natural cubic spline we would drop x^2 and x^3 and impose the additional constraints

$$\sum \gamma = \sum \gamma\xi = 0.$$

Actually, these constraints can be eliminated by suitable re-parametrization. For example a natural cubic spline with two interior knots plus one knot at each extreme of the data can be fit by regressing y on three covariates, x, z_1 and z_2, where

$$z_1 = (x - \xi_1)^3_+ - \frac{(\xi_1 - \xi_4)}{\xi_3 - \xi_4}(x - \xi_3)^3_+$$

and

$$z_2 = (x - \xi_2)^3_+ - \frac{(\xi_2 - \xi_4)}{\xi_3 - \xi_4}(x - \xi_3)^3_+.$$

The proof is left as an exercise.

2.2.1 B-Splines

The power series representation is useful for understanding splines but is not well suited for computation because successive terms tend to be highly correlated. It is, however, very easy to use. Be judicious.

A much better representation of splines for computation is as linear combinations of a set of basis splines called *B-splines*. These are numerically more stable, among other reasons because each B-spline is non-zero over a limited range of knots. They are not so easy to calculate, but fortunately R and S have functions for calculating a basis, see `bs` for B-splines and `ns` for natural B-splines.

Regression splines are very popular (particularly with me :-) because they are easy to use, and can be incorporated without difficulty as part of other estimation procedures.

The main problem is where to place the knots. Often they are placed at selected quantiles (i.e. the terciles, or quartiles, or quintiles, depending on how many knots you want). A smarter strategy would place more knots in regions where $f(x)$ is changing more rapidly. Knot placement is an arcane art form, and the first disadvantage cited by detractors of regression splines.

2.3 Smoothing Splines

A more formal approach to the problem is to consider fitting a spline with knots at every data point, so potentially it could fit perfectly, but estimate its parameters by minimizing the usual sum of squares plus *a roughness penalty*.

A suitable penalty is to integrate the squared second derivative, leading to the following criterion, known as the penalized sum of squares:

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

where integration is over the range of x and λ is a tuning parameter. As $\lambda \to 0$ we impose no penalty and end up with a very close fit, but the resulting curve could be very noisy as it follows every detail in the data. As $\lambda \to \infty$ the penalty dominates and the solution converges to the OLS line, which is as smooth as you can get (the second derivative is always 0), but may be a very poor fit.

Amazingly, it can be shown that minimizing the PSS for a fixed λ over the space of all continuous differentiable functions leads to a unique solution, and this solution is a natural cubic spline with knots at the data points.

More generally, penalizing the squared ν-th derivative leads to a natural spline of degree $2v-1$. For a proof see Reinsch (1967).

2.3.1 Computation

Implicit in the work of Reinsch is the fact that the penalty may be written as a quadratic form

$$\int (S''(x))^2 dx = \mu' K \mu$$

where $\mu = S(x_i)$ is the fit, K is an $n \times n$ matrix given by $K = \Delta' W^{-1} \Delta$, Δ is an $(n-2) \times n$ matrix of second differences, with elements

$$\Delta_{ii} = \frac{1}{h_i}, \Delta_{i,i+1} = -\frac{1}{h_i} - \frac{1}{h_{i+1}}, \Delta_{i,i+2} = \frac{1}{h_{i+1}}$$

W is a symmetric tridiagonal matrix of order $n-2$ with elements

$$W_{i-1,i} = W_{i,i-1} = \frac{h_i}{6}, w_{ii} = \frac{h_i + h_{i+1}}{3}$$

and $h_i = \xi_{i+1} - \xi_i$, the distance between successive knots (or x values).

This implies that we can compute a smoothing spline as

$$\hat{S}(x) = (I + \lambda K)^{-1} y.$$

Proof: Write the penalized sum of squares as

$$\text{PSS} = (y-\mu)'(y-\mu) + \lambda \mu' K \mu$$

Taking derivatives

$$\frac{\partial \text{PSS}}{\partial \mu} = -2(y-\mu) + 2\lambda K \mu$$

Setting this to zero gives

$$y = \hat{\mu} + \lambda K \hat{\mu} = (I + \lambda K)\hat{\mu},$$

and premultiplying both sides by $(I + \lambda K)^{-1}$ completes the proof.

Any serious implementation of smoothing splines would take advantage of the banded nature of these matrices to economize storage and compute the required inverses, but these formulas can be used as they are for small or moderate n.

2.3.2 Extensions to WLS and GLMs

The result extends easily to WLS, where we minimize the *weighted* PSS

$$\text{WPSS} = (y - \mu)'W(y - \mu) + \lambda\mu' K\mu.$$

The same steps we followed before lead to

$$\hat{S}(x) = (W + \lambda K)^{-1} W y.$$

This strategy can be extended to GLMS with link η, where we would smooth the linear predictor or transformed mean

$$\eta(\mu_i) = S(x_i)$$

by maximizing a penalized log-likelihood, or equivalently, solving a series of penalized weighted least squares problems.

(The situation is exactly analogous to maximizing the usual log-likelihood via iteratively re-weighted least squares.)

2.4 Cross-Validation

We have solved the problem of knot placement, but now we have to pick an appropriate value for λ. Some claim this is easier, because we are left with a single number to worry about.

Wabba and others have suggested a technique know as *cross-validation*. Let $\hat{S}_\lambda^{(-i)}$ denote the spline fit with tuning parameter λ while omitting the i-th observation. We can compare this fit with the observed value y_i, and we can summarize these differences by computing a sum of squares

$$\text{CVSS} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{S}_\lambda^{(-i)}(x_i))^2$$

which depends on λ. The idea is to pick λ to minimize the CVSS.

This sounds like a lot of work but in fact it isn't, thanks again to regression updating formulas, which can be used to show that

$$\text{CVSS} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{S}_\lambda^{(-i)}(x_i)}{1 - A_{ii}}\right)^2$$

where A_{ii} is a diagonal element of $A = (I - \lambda K)^{-1}$. This extends easily to WLS.

An alternative criterion is to replace the A_{ii} by their average, which is $\text{tr}(A)/n$. This leads to a *generalized* CVSS that has been found to work well in practice.

2.4.1 Effective DF

A much simpler strategy advocated by Hastie and Tibshirani is to decide in advance how many d.f one is willing to spend on a fit. If a smoothing spline (or in fact any linear smoother) has the form

$$\hat{S} = Ay$$

then we define the degrees of freedom as $\text{tr}(A)$, the sum of the eigenvalues. Two other popular definitions are

$$n - \text{tr}(2A - AA') \text{ and } \text{tr}(AA')$$

All three are motivated by analogy with linear models, where the role of A is played by the 'hat' matrix $H = X(X'X)^{-1}X'$. The general idea is to search for a value of λ that produces a matrix A with trace equal to the desired degrees of freedom.

2.5 P-Splines

There is an intermediate solution betwen regression and smoothing splines, proposed more recently by Eilers and Marx.

The idea is to start with a B-spline basis with somewhat more knots than you would use if you were doing regression splines (but not as many as one per observation), and then introduce a roughness penalty as you do with smoothing splines.

The novelty is that these authors propose using symmetric B-splines and penalize them not on the second derivatives, but instead on the differences between the coefficients of adjacent splines, a criterion that is easy to implement, yet turns out to be closely related to the usual penalty.

Suppose we pick a set of k splines to start with. Let $B_j(x)$ denote the j-th B-spline evaluated at x and let α_j denote it's coefficient. Then the idea behind P-splines is to minimize

$$\text{PSS} = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{k}\alpha_j B_i(x_i))^2 + \lambda \sum_{j=p+1}^{k}(\Delta^p \alpha_j)^2$$

where Δ is the difference operator and p is the order of the penalty.

The main advantage of P-splines over smoothing splines is that the calculations are simpler because fewer knots are used. Moreover, the calculations can be easily implemented using a data augmentation algorithm. We will return to this topic in the context of survival analysis.

3 Semi-Parametric Models

So far we have considered models with only one x. Let us take a small step forward by considering several x's where *one* (henceforth denoted t) is treated non-parametrically, and the others (denoted X) are handled the old-fashioned way. We could consider a linear model, where

$$y = X\beta + S(t) + \epsilon.$$

or a generalized linear model, where

$$E(y) = \mu \qquad \text{and} \qquad \eta = g(\mu) = X\beta + S(t)$$

Green and Yandell discuss estimation of this model by maximizing the penalized log-likelihood

$$\text{PLL} = \log L(\beta, \gamma) - \frac{1}{2}\lambda\gamma' K\gamma,$$

where $\gamma = S(t)$ is the smooth and $\gamma' K\gamma$ is the matrix form of the penalty.

This log-likelihood can be maximized using relatively straightforward extensions of the IRLS algorithm.

An even simpler alternative is to use regression splines to estimate $S(t)$, in which case we are back in the usual framework of fully parametric models.

In the end the decision amounts again to a choice between fixing the knots or fixing the tuning parameter. P-splines may well represent a sensible compromise between these two alternatives.

Each approach has its advocates and sometimes the discussions get quite lively. You may enjoy the discussions that follow the papers by Ramsay, on monotone splines, and Eilers and Marx, on P-splines, both in *Statistical Science*.

4 Non-Parametric Models

Let us now consider the case of *several* predictors that we would like to treat non-parametrically.

4.1 The Curse of Dimensionality

One might consider multidimensional smoothers aimed at estimating S in the model

$$y = S(x_1, x_2, \ldots, x_p) + \epsilon$$

Running means are easily generalized if one can define a neighborhood in p-dimensional space. These are typically spherical, but it is not clear whether one should measure distance in the original scale, or after standardizing the variables, or indeed after taking into account their correlation structure, using for example, Mahalanobis distance as the metric.

A related problem is that, as the number of dimensions increases, the neighborhoods become increasingly less local. With uniform data in a cube we get 10% of the points by moving 0.1 in one dimension, but in 10 dimensions we need to move 0.8 to get 10%, so the neighborhood is hardly local any more.

Very similar comments apply to running lines.

Kernel smoothers can be extended too, a popular choice of kernel in two dimensions is the bivariate normal distribution.

Splines have been generalized as well. An example in two-dimensions is the so-called thin-plate spline, which penalizes all three second derivatives $\partial^2 S/\partial x_1^2, \partial^2 S/\partial x_2^2$, and $\partial^2 S/\partial x_1 \partial x_2$.

All of these techniques are useful for small p, say $p = 2$, and can be computationally quite intensive.

4.2 Additive Models

Hastie and Tibshirani have proposed using additive models with separate smoothers for each coordinate, so the model becomes

$$y = \alpha + S_1(x_1) + S_2(x_2) + \ldots + S_p(x_p) + \epsilon$$

The choice of smoother is left open, one could use a smoothing spline for x_1, a simple linear term for x_2, a running line for x_3, and so on.

Note the lack of interactions. The hope is that a suitable transformation of each coordinate will make the additive structure appropriate, hence the name *additive* models.

There are three main approaches to fitting this type of model, in increasing order of complexity:

- Multiple linear regression: easy but rigid.
- Regression splines: pick a basis for each x_i, ideally B-splines, and fit the model by regression or maximum likelihood. Convenient and quite flexible, but requires knot placement.
- General smoothers: use a smoothing spline, or in fact any scatterlot smoother, for each x_i. This approach is computationally intensive but very flexible.

Hastie and Tibshirani proposed a simple estimation procedure that works well for the more complex case with arbitrary smoothers, known as the *backfitting algorithm*:

- Provide initial values for the constant α and all S_i. A sensible choice is $\alpha^0 = \bar{y}$, and S_i^0 given by the linear regression of y on x_i.
- Cycle through the predictors:
  - calculate $y - \alpha^j - \sum_{i \neq 1} S_i^j$ and smooth against x_1
  - calculate $y - \alpha^j - \sum_{i \neq 2} S_i^j$ and smooth against x_2
  - ...
  - calculate $y - \alpha^j - \sum_{i \neq p} S_i^j$ and smooth against x_p
- Repeat the cycling until the S's do not change.

Surprinsingly, the algorithm works and converges to the correct solution. You may easily try this for a multiple regression with two predictors. It will eventually converge to the OLS solution if by 'smooth' you read 'run a simple linear regression'. In fact the method is related to the Gauss-Seidel algorithm for solving systems of linear equations.

The algorithm seems to work well in practice with a variety of smoothers, although proofs of convergence are available only for special cases, most notably smoothing splines.

The same idea can be extended to generalized linear models. We now have

$$E(y) = \mu \qquad \text{and} \qquad \eta = g(\mu) = \alpha + \sum_{j=1}^{p} S_j(x_j).$$

It turns out that the backfitting algorithm can be adapted to fit this model as well. (This should be no surprise by now considering the close connection between maximum likleihood and IRLS.) The resulting algorithm is called *local scoring*.

The local scoring algorithm is implemented is the S function `gam`, which regretably is not available in R (yet).