Details on R's `smooth.spline()`

Martin Maechler

June 8, 2022

Smoothing splines penalized regression

Given observations (our data), (x_i, Y_i) $(i = 1, \ldots, n)$, a quite general model for such data is

$$Y_i = m(x_i) + \varepsilon_i, \tag{1}$$

where $\varepsilon_1, \ldots, \varepsilon_n$ i.i.d. with $\mathbb{E}[\varepsilon_i] = 0$ and $m : \mathbb{R} \to \mathbb{R}$ is an "arbitrary" function. The function $m(\cdot)$ is called the nonparametric regression function and it satisfies $m(x) = \mathbb{E}[Y|x]$ and should fulfill some kind of smoothness conditions.

One fruitful approach to estimate such a "smooth" function $m()$ is via so called "smoothing splines" (or their generalization, "penalized regression splines").

Penalized sum of squares

Consider the following problem: among all functions m with continuous second derivative, find the one which minimizes the penalized residual sum of squares

$$L_\lambda(m) \quad := \quad \sum_{i=1}^{n} (Y_i - m(x_i))^2 + \lambda \int m''(t)^2 \, dt, \tag{2}$$

where $\lambda > 0$ is a smoothing parameter. The first term measures closeness to the data and the second term penalizes curvature ("roughness") of the function. The two extreme cases are:

- $\lambda = 0$: As any function m interpolating the data gives $L_0(m) = 0$, hence (2) does require $\lambda > 0$. In the limit, $\lambda \to 0$, however, $\hat{m}_\lambda \to$ the well defined interpolating natural cubic spline). [1]
- $\lambda = \infty$: any linear function fulfills $m''(x) \equiv 0$, and the minimizer of (2) is the least squares regression line.

The smoothing spline solution

Remarkably, the minimizer of (2) is *finite*-dimensional, although the criterion to be minimized is over the infinite-dimensional Sobolev space of functions for which the integral $\int m''^2$ is finite.

Let us assume for now that the data has x values sorted and unique,

$$x_1 < x_2 < \ldots < x_n.$$

[1] We will see that taking the limit $\lambda \to 0$ is problematic directly numerically and in practice you should rather use `spline()` for spline *interpolation* instead of smoothing.

The solution $\hat{m}_\lambda(\cdot)$ (i.e., the unique minimizer of (2)) is a natural **cubic spline** with knots $t_1, t_2, \ldots, t_{n_k}$ which are the sorted unique values of $\{x_1, x_2, \ldots, x_n\}$. That is, $\hat{m}$ is a piecewise cubic polynomial in each interval $[t_j, t_{j+1})$ such that $\hat{m}_\lambda^{(k)}$ $(k = 0, 1, 2)$ is continuous everywhere and has "natural" boundary conditions $\hat{m}''(t_1) = \hat{m}''(t_{n_k}) = 0$. For the $n_k - 1$ cubic polynomials, we'd need $(n_k - 1) \cdot 4$ coefficients. Since there are $(n_k - 2) \cdot 3$ continuity conditions (at every "inner knot", $j = 2, \ldots, n_k - 1$) plus the 2 "natural" conditions, this leaves $4(n_k - 1) - [3(n_k - 2) + 2] = n_k$ free parameters (the β_j's below). Knowing that the solution is a cubic spline, it can be obtained by linear algebra. We represent

$$m_\lambda(x) = \sum_{j=1}^{n_k} \beta_j B_j(x), \tag{3}$$

where the $B_j(\cdot)$'s are basis functions for natural splines. The unknown coefficients can then be estimated from least squares in linear regression under side constraints. The criterion in (2) for $\hat{m}_\lambda$ as in (3) then becomes

$$\tilde{L}_\lambda(\boldsymbol{\beta}) := L_\lambda(m) \quad = \quad \|\mathbf{Y} - X\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^\intercal \Omega \boldsymbol{\beta},$$

respectively, when not all weights w_i are 1,

$$\tilde{L}_\lambda(\boldsymbol{\beta}) \quad = \quad (\mathbf{Y} - X\boldsymbol{\beta})^\intercal W (\mathbf{Y} - X\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^\intercal \Omega \boldsymbol{\beta}, \tag{4}$$

where the design matrix X has jth column $(B_j(x_1), \ldots, B_j(x_n))^\intercal$, i.e.,

$$\begin{aligned} X_{ij} &= B_j(x_i) \quad \text{for } i = 1, \ldots, n, \\ W &= \texttt{diag(w)}, \text{ i.e., } W_{ij} = \mathbf{1}_{[i=j]} \cdot w_i, \qquad \text{and} \\ \Omega_{jk} &= \int B_j''(t) B_k''(t)\, dt, \quad \text{for } j, k = 1, \ldots, n_k. \end{aligned}$$

The solution, $\widehat{\boldsymbol{\beta}} = \arg\min_\beta \tilde{L}_\lambda(\boldsymbol{\beta})$ can then be derived by setting the gradient $\frac{\partial}{\partial \boldsymbol{\beta}} \tilde{L}_\lambda(\boldsymbol{\beta})$ to zero: $\mathbf{0} = -2(X^\intercal W \mathbf{Y})^\intercal \boldsymbol{\beta} + 2(X^\intercal W X + \lambda \Omega)\boldsymbol{\beta}$, and hence

$$\widehat{\boldsymbol{\beta}} = (X^\intercal W X + \lambda \Omega)^{-1} X^\intercal W \mathbf{Y}. \tag{5}$$

When B-splines are used as basis function B_j, both X and Ω are *banded* matrices, i.e., zero apart from a "band", i.e., few central diagonals. As,

$$\hat{m}_\lambda(x) = \sum_{j=1}^{n_k} \hat{\beta}_j B_j(x),$$

the fitted values are $\hat{\mathbf{Y}} = X\widehat{\boldsymbol{\beta}}$, where $\hat{Y}_i = \hat{m}_\lambda(x_i)$ $(i = 1, \ldots, n)$, and

$$\hat{\mathbf{Y}} = X\widehat{\boldsymbol{\beta}} = \mathcal{S}_\lambda \mathbf{Y}, \quad \text{where} \quad \mathcal{S}_\lambda = X(X^\intercal W X + \lambda \Omega)^{-1} X^\intercal W. \tag{6}$$

The hat matrix $\mathcal{S}_\lambda = \mathcal{S}_\lambda{}^\intercal$ is symmetric which implies elegant mathematical properties (real-valued eigen-decomposition).

....

Notes

1.
2.