NONPARAMETRIC SMOOTHING APTS

Assessment R. J. Samworth/Sep 2010

Comments and corrections to r.samworth@statslab.cam.ac.uk

1. Recall that the Epanechnikov kernel is a second-order kernel defined by

$$K_E(x) = \frac{3}{4\sqrt{5}}\left(1 - \frac{x^2}{5}\right)\mathbb{1}_{\{|x| \leq \sqrt{5}\}},$$

and that $\mu_2(K_E) = 1$. Let K_0 be another non-negative second-order kernel with $\mu_2(K_0) = 1$. By considering $e(x) = K_0(x) - K_E(x)$, and noting that $\int_{-\infty}^{\infty} e(x)\,dx = \int_{-\infty}^{\infty} x^2 e(x)\,dx = 0$ and $e(x) \geq 0$ for $|x| \geq \sqrt{5}$, show that $R(K_0) \geq R(K_E)$.

[*This shows that the Epanechnikov kernel is the optimal second-order kernel in the sense of minimising the AMISE.*]

2. [*In this question you may assume any required regularity conditions are satisfied.*]

A natural kernel estimator of the rth derivative, $f^{(r)}(x)$ of a density $f(x)$ is

$$\hat{f}_h^{(r)}(x) = \frac{1}{nh^{r+1}} \sum_{i=1}^{n} K^{(r)}\left(\frac{x - X_i}{h}\right).$$

By integrating by parts r times, show that $K_h^{(r)} * f = K_h * f^{(r)}$, and hence that

$$\mathrm{MSE}\{\hat{f}_h^{(r)}(x)\} = \frac{1}{nh^{2r+1}} R(K^{(r)}) f(x) + \frac{1}{4}\mu_2(K)^2 f^{(r+2)}(x)^2 h^4 + o\left(\frac{1}{nh^{2r+1}} + h^4\right).$$

Deduce that the optimal MSE is of order $n^{-4/(2r+5)}$ and argue informally that the optimal MISE is of the same order.

3. In the homoscedastic random design nonparametric regression model with regression function m and marginal density f for the design points, it can be shown that under the conditions given in class,

$$\hat{s}_{r,h}(x) \equiv \frac{1}{n}\sum_{i=1}^{n}(X_i - x)^r K_h(X_i - x) = \begin{cases} h^r \mu_r(K) f(x) + o_P(h^r) & \text{if } r \text{ is even} \\ h^{r+1}\mu_{r+1}(K) f'(x) + o_P(h^{r+1}) & \text{if } r \text{ is odd.} \end{cases}$$

Use this to find an asymptotic expression for the conditional bias $\mathbb{E}\{\hat{m}_h(x;0)|X_1, \ldots, X_n\} - m(x)$ of the Nadaraya–Watson estimator at $x \in (0, 1)$.

4. Plot the Lloyds Bank share price data, and add the natural cubic spline estimate using `sm.spline` in the `pspline` package with penalty parameter chosen using ordinary (not generalised) cross-valdation.

Now add in red the spline estimate obtained using the truncated cubic spline basis and 50 equally spaced knots between 3 and 250 (you should write the code for the design matrix yourself). What do you notice? Finally, add in green the penalised version of this estimate using the `smooth.spline` function (which uses the B-spline basis for computations), again with 50 knots, and with the regularisation parameter chosen by ordinary cross-validation.