# Chapter 1. Nonparametric Curve Estimation
# Part 4

January 31, 2007

## 1 Other bandwidth selection methods

Since the bandwidth plays an essential role, some other methods have also been proposed

### 1.1 Generalized Cross-validation methods

Recall that

$$\hat{m}(x) = \sum_{i=1}^{n} K_h(X_i - x)Y_i / \sum_{i=1}^{n} K_h(X_i - x) = \ell_n(x)^\top Y$$

where $Y = (Y_1, \cdots, Y_n)^\top$ and

$$\ell_n(x) = \{\sum_{i=1}^{n} K_h(X_i - x)\}^{-1}(K_h(X_1 - x), \cdots, K_h(X_n - x))^\top$$

Let

$$S_n = \begin{pmatrix} \ell_n(X_1) \\ \ell_n(X_2) \\ \vdots \\ \ell_n(X_n) \end{pmatrix}$$

we have

$$\begin{pmatrix} \hat{m}(X_1) \\ \hat{m}(X_2) \\ \cdots \\ \hat{m}(X_n) \end{pmatrix} = S_n Y$$

Then, one can prove that

$$CV(h) = n^{-1} \sum_{i=1}^{n} \frac{(Y_i - \hat{m}(X_i))^2}{(1 - S_n(i,i))^2}$$

Based on this, Craven and Wahba (1979) proposed to consider the so called generalized cross-validation

$$GCV(h) = \frac{n^{-1} \sum_{i=1}^{n} (Y_i - \hat{m}(X_i))^2}{\{1 - tr(S_n)/n\}^2}$$

The bandwidth selected by GCV is

$$\hat{h} = \arg\min_h GCV(h)$$

**Example 1.1 (simulation)** *100 observations from*

$$Y = cos(2\pi X) + 0.2\varepsilon$$

*where* $X \sim uniform(0,1)$ *and* $\varepsilon \sim N(0,1)$

*the estimated regression function is shown in Fig. 1*

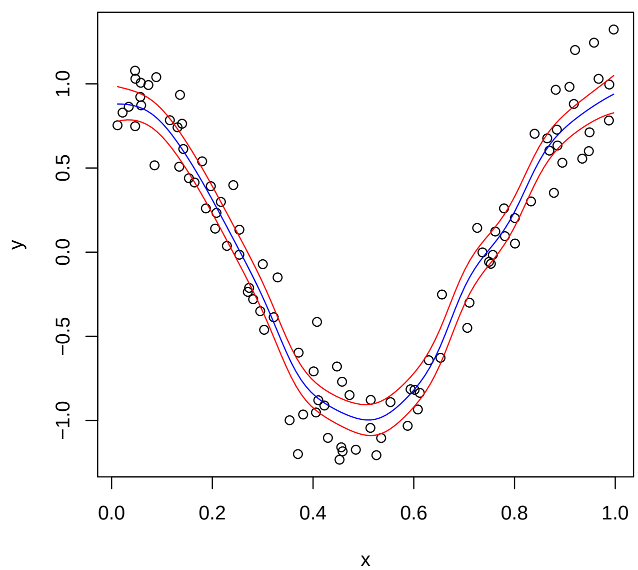


Figure 1: calculation results for Example 1.1 **(Gcvh)**, **(ks)**, **(code)**

**Example 1.2 (motorcycle)** **(data)** *the bandwidth is chosen to be 1.38 by GCV as well as CV. the estimated regression function is shown in Fig. 2*

## 1.2 plug-in method

Recall the optimal bandwidth is

$$h_{opt}(x) = \left\{ \frac{d_0 \sigma^2}{4f(x)c_2^2\{\frac{1}{2}m''(x) + f^{-1}(x)m'(x)f'(x)\}^2} \right\}^{1/5} n^{-1/5}.$$

Select an initial bandwidth $h$, say the one below. The estimator of $m(x)$ is

$$\hat{m}(x) = \sum_{i=1}^{n} K_h(X_i - x)Y_i / \sum_{i=1}^{n} K_h(X_i - x)$$

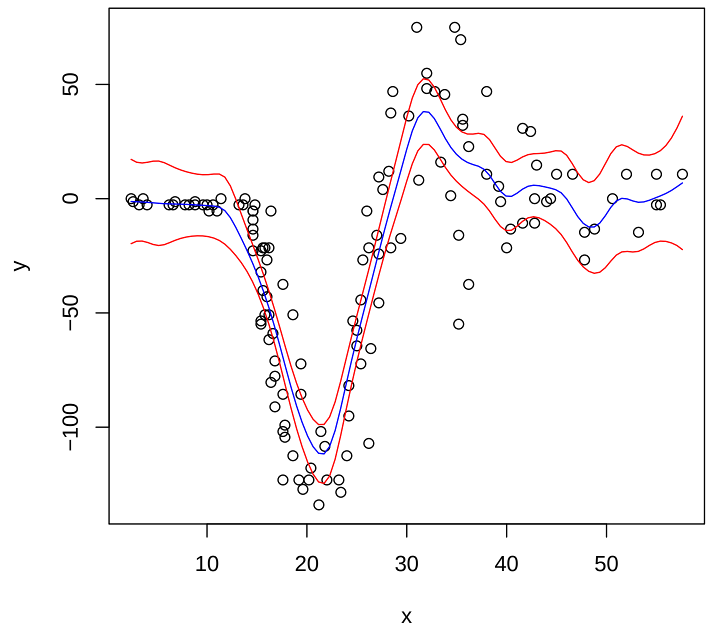


Figure 2: calculation results for Example 1.2 **(code)**

Therefore, we have

$$\hat{m}'(x) = d[\sum_{i=1}^{n} K_h(X_i - x)Y_i / \sum_{i=1}^{n} K_h(X_i - x)]/dx$$

and

$$\hat{m}''(x) = d^2[\sum_{i=1}^{n} K_h(X_i - x)Y_i / \sum_{i=1}^{n} K_h(X_i - x)]/dx^2$$

For the density

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

and

$$\hat{f}'(x) = n^{-1} \sum_{i=1}^{n} \frac{dK_h(X_i - x)}{dx}.$$

We can then estimate the bandwidth $h_{opt}(x)$ based on these functions.

### 1.3 The bandwidth for density estimation

Consider the estimation of density function of $X$. Suppose $X_1, \cdots, X_n$ are samples. Bickel and Doksum (1977) and Silverman (1986) proved that if the true density of $X$ is normal, then the optimal bandwidth is

$$\begin{aligned} \text{Gaussian kernel}: &\qquad h = 1.06 s_x n^{-1/5} \\ \text{Epanechnikov}: &\qquad h = 2.34 s_x n^{-1/5}. \end{aligned}$$

where $s_x = (n^{-1}\sum_{i=1}^{n}(X_i - \bar{X})^2)^{1/2}$.

For the estimation regression function, we can also use it after we standardize $Y$.

## 2 Local linear kernel smoothing

Again, consider the conditional expectation of $Y$ given $X = x$. Suppose that $(X_i, Y_i), i = 1, \cdots, n$ are samples.

$$Y_i = m(X_i) + \varepsilon_i$$

For any given point $x$ and any $X_i$, if $X_i$ is close to $x$ we consider a local linear approximation

$$m(X_i) \approx m(x) + m'(x)(X_i - x).$$

Thus the model is

$$Y_i \approx m(x) + m'(X_i - x) + \varepsilon_i$$

or

$$\begin{aligned} Y_1 &\approx m(x) + m'(x)(X_i - x) + \varepsilon_1 \\ Y_2 &\approx m(x) + m'(x)(X_i - x) + \varepsilon_2 \\ &\vdots \\ Y_n &\approx m(x) + m'(x)(X_i - x) + \varepsilon_n \end{aligned}$$

This is a linear regression model with parameters $m(x)$ and $m'(x)$. It is easy to see that we care the approximation at $x$. Therefore, we give higher weight to those points close to $x$. The weight can be defined as

$$K_h(X_i - x) = h^{-1} K\Big(\frac{X_i - x}{h}\Big)$$

We use the following weighted least squares problem to estimate the value $m(x)$ and $m'(x)$.

$$\sum_{i=1}^{n}\{Y_i - m(x) - m'(x)(X_i - x)\}^2 K_h(X_i - x).$$

The minimizer to the above value is

$$\begin{pmatrix} \hat{m}(x) \\ \hat{m}'(x) \end{pmatrix} = \Big\{ \sum_{i=1}^{n} K_h(X_i - x) \begin{pmatrix} 1 \\ X_i - x \end{pmatrix} \begin{pmatrix} 1 \\ X_i - x \end{pmatrix}^{\top} \Big\}^{-1} \times \sum_{i=1}^{n} K_h(X_i - x) \begin{pmatrix} 1 \\ X_i - x \end{pmatrix} Y_i$$

We can write it as

$$\hat{m}(x) = \frac{n^{-1} \sum_{i=1}^{n} \{s_{n,2}(x) K_h(X_i - x) - s_{n,1}(x) K_h(X_i - x)\} Y_i}{s_{n,2}(x) s_{n,0}(x) - s_{n,1}^2(x)}$$

where

$$s_{n,k}(x) = n^{-1} \sum_{i=1}^{n} K_h(X_i - x) \Big(\frac{X_i - x}{h}\Big)^k, \quad k = 0, 1, 2$$

Let

$$\mathbf{X} = \begin{pmatrix} 1 & X_1 - x \\ 1 & X_2 - x \\ \cdots & \\ 1 & X_n - x \end{pmatrix}$$

and $\mathbf{W}$ be the diagonal matrix of weights

$$W = diag\{K_h(X_i - x)\}.$$

and $\beta = (m(x), m'(x))^{\top}$. Then the least squares problem can be written as

$$(Y - \mathbf{X}\beta)^{\top} \mathbf{W} (\mathbf{Y} - \mathbf{X}\beta)$$

The minimizer to the above problem is

$$\hat{\beta} = \begin{pmatrix} \hat{m}(x) \\ \hat{m}'(x) \end{pmatrix} = \{\mathbf{X}^{\top} \mathbf{W} \mathbf{X}\}^{-1} \mathbf{X}^{\top} \mathbf{W} \mathbf{Y}$$

**Example 2.1 (simulation)** *100 observations from*

$$Y = cos(2\pi X) + 0.2\varepsilon$$

*where $X \sim uniform(0,1)$ and $\varepsilon \sim N(0,1)$. with $h$ = 0.05, we have the following simulations; see Fig 3*

*From this simulation, we can see that local linear kernel smoothing estimator has better performance at the boundary points than NW local constant estimator.*

**Example 2.2 (air pollution in Hong Kong)** (data)*; Ozone is a second polluatnt, i.e. it is generated by chemical reaction of other polluatnts such as $SO_2$ and $NO_2$ with sunlight.*

*Apply local linear kernel smoothing method, we find the relation as shown Fig 4*

*From this simulation, we can see that local linear kernel smoothing estimator has better performance at the boundary points than NW local constant estimator.*

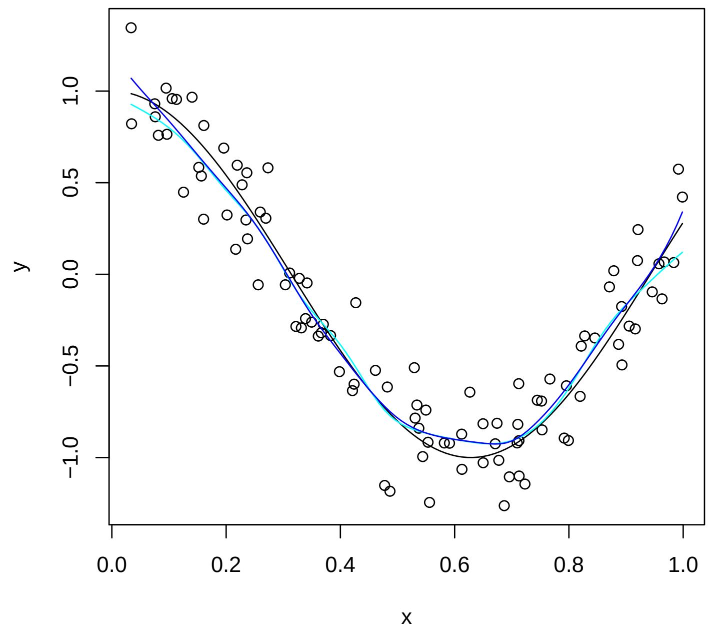


Figure 3: calculation results for Example 2.1. black: true function; cyan: NW estimator; blue: LL kernel estimator. **(ksLL) (code)**

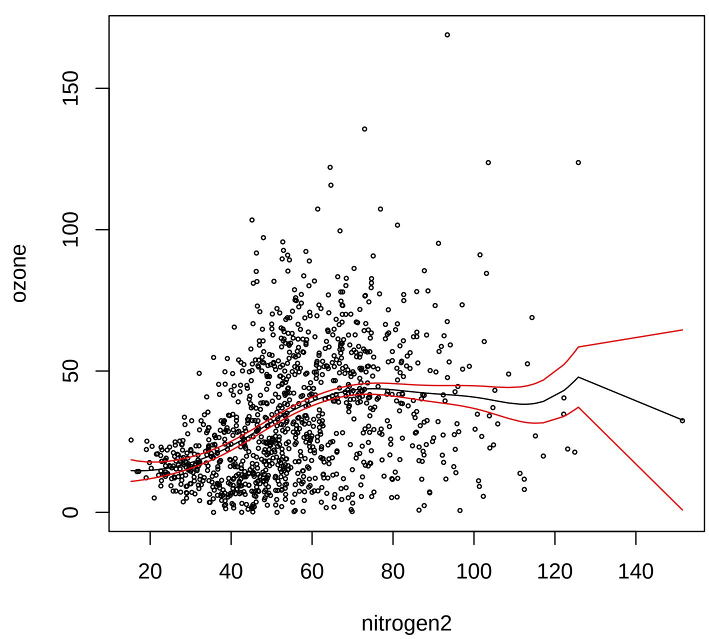


Figure 4: calculation results for Example 2.2. black: true function; cyan: NW estimator; blue: LL kernel estimator. **(ksLL) (code)**