1-Dimensional Kernel and Local Regression Smoothers

Stephen Vardeman
Analytics Iowa LLC
ISU Statistics and IMSE

Basic idea and 1-D kernel smoothing

The basic idea here is that $\hat{f}(\mathbf{x}_0)$ might weight points in a training set according to how close they are to $\mathbf{x}_0$, do some kind of fitting around $\mathbf{x}_0$, and ultimately read off the value of the fit at $\mathbf{x}_0$.

Suppose that x takes values in $[0, 1]$. Make weights for points in the training set based on a (usually, symmetric about 0) non-negative, real-valued function $D(t)$ that is non-increasing for $t \geq 0$ and non-decreasing for $t \leq 0$. Often $D(t)$ has value 0 unless $|t| \leq 1$. Then, a (smoothing) kernel function is

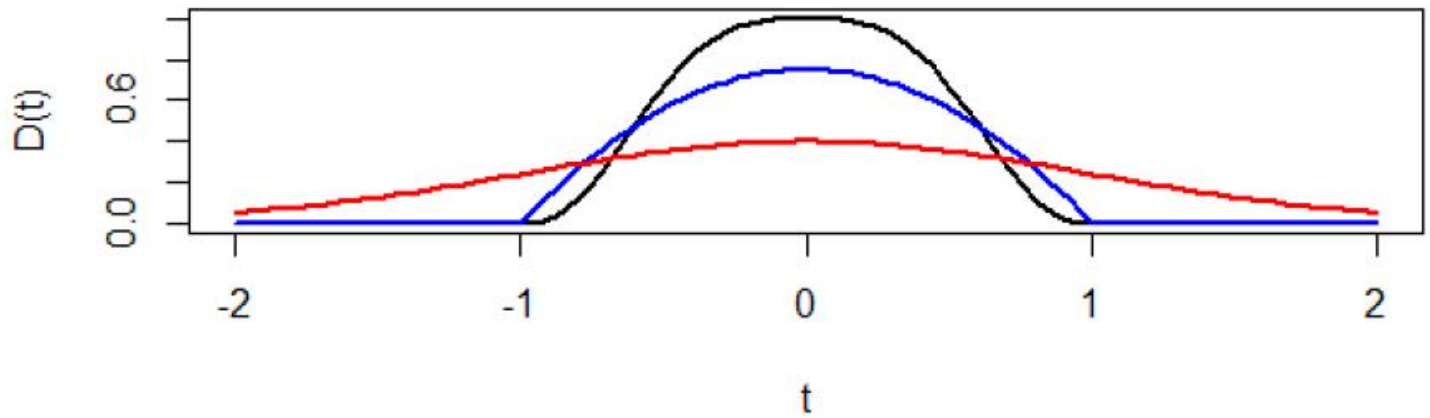
$$K_\lambda(x, x_0) = D\left(\frac{x - x_0}{\lambda}\right) \tag{1}$$

λ is a "bandwidth" parameter that controls the rate at which weights drop off as one moves away from x_0 (and in the case that $D(t) = 0$ for $|t| > 1$, how far one moves away from x_0 before *no weight* is assigned).

Common 1-d smoothing kernels

Common choices for 1-dimensional smoothing kernels D are

1. the Epanechnikov quadratic kernel (blue below), $D(t) = \frac{3}{4}\left(1 - t^2\right) I\left[|t| \leq 1\right]$,
2. the "tri-cube" kernel (black), $D(t) = \left(1 - |t|^3\right)^3 I\left[|t| \leq 1\right]$, and
3. the standard normal density (red), $D(t) = \phi(t)$.



The Nadaraya-Watson kernel smoother

Using weights (1) to make a weighted average of training responses, the Nadaraya-Watson kernel-weighted prediction at x_0 is

$$\hat{f}_\lambda(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i)\, y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)} \tag{2}$$

This typically smooths training outputs, y_i, in a more pleasing way than does a k-nearest neighbor average. But it has obvious problems at the ends of the interval $[0, 1]$ and at places in the interior of the interval where training data are dense to one side of x_0 and sparse to the other, if the target $\mathrm{E}[y|x = z]$ has non-zero derivative at $z = x_0$. For example, at $x_0 = 1$ only $x_i \leq 1$ get weight, and if $\mathrm{E}[y|x = z]$ is decreasing at $z = x_0 = 1$, $\hat{f}_\lambda(1)$ will be positively biased. That is, with usual symmetric kernels, (2) will fail to adequately follow an obvious trend at 0 or 1 (or at any point between where there is a sharp change in the density of input values in the training set).

Locally weighted regression

A way to fix this problem with the Nadaraya-Watson predictor is to replace the locally-fitted constant with a locally-fitted line. That is, at x_0 one might choose $\alpha(x_0)$ and $\beta(x_0)$ to solve the optimization problem

$$\underset{\alpha \text{ and } \beta}{\text{minimize}} \sum_{i=1}^{N} K_\lambda(x_0, x_i)\left(y_i - (\alpha + \beta x_i)\right)^2 \tag{3}$$

and then employ the prediction

$$\hat{f}_\lambda(x_0) = \alpha(x_0) + \beta(x_0)\, x_0 \tag{4}$$

Explicit form for locally weighted regression

The weighted least squares problem (3) has an explicit solution. Let

$$\underset{N\times 2}{\mathbf{B}} = \begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_N \end{pmatrix}$$

and take

$$\underset{N\times N}{\mathbf{W}}(x_0) = \operatorname{diag}\left(K_\lambda(x_0, x_1), \ldots, K_\lambda(x_0, x_N)\right)$$

then (4) is

$$\hat{f}_\lambda(x_0) = (1, x_0)\left(\mathbf{B}'\mathbf{W}(x_0)\mathbf{B}\right)^{-1}\mathbf{B}'\mathbf{W}(x_0)\mathbf{Y} = \mathbf{l}'(x_0)\mathbf{Y} \qquad (5)$$

for the $1 \times N$ vector $\mathbf{l}'(x_0) = (1, x_0)\left(\mathbf{B}'\mathbf{W}(x_0)\mathbf{B}\right)^{-1}\mathbf{B}'\mathbf{W}(x_0)$. It is thus obvious that locally weighted linear regression is (an albeit x_0-dependent) *linear* operation on the vector of outputs.

Kernel smoothing df

Recall that for smoothing splines, smoothed values are

$$\widehat{\mathbf{Y}}_\lambda = \mathbf{S}_\lambda \mathbf{Y}$$

where the parameter λ is the penalty weight, and

$$\mathrm{df}(\lambda) = \mathrm{tr}(\mathbf{S}_\lambda)$$

We may do something parallel in the present context. We may take

$$\underset{N \times N}{\mathbf{L}_\lambda} = \begin{pmatrix} \mathbf{l}'(x_1) \\ \vdots \\ \mathbf{l}'(x_N) \end{pmatrix}$$

where now the parameter λ is the bandwidth, write

$$\widehat{\mathbf{Y}}_\lambda = \mathbf{L}_\lambda \mathbf{Y}$$

and define

$$\mathrm{df}(\lambda) = \mathrm{tr}(\mathbf{L}_\lambda)$$

Kernel smoothers and N-W smoothing

The weights in $\mathbf{l}'(x_0)$ combine the original kernel values and the least squares fitting operation to produce a kind of "equivalent kernel" (for a Nadaraya-Watson-type weighted average). HTF suggest that matching degrees of freedom for a smoothing spline and a kernel smoother produces very similar equivalent kernels, smoothers, and predictions.

There is a famous theorem of Silverman that adds technical credence to this notion. It says that in some appropriate probabilistic sense the smoother matrix for cubic spline smoothing has entries like those that would come from a corresponding kernel smoothing.

Silverman theorem

Roughly, the theorem says that for large N, if in the case $p = 1$ the inputs $x_1, x_2, \ldots, x_N$ are iid with density $p(x)$ on $[a, b]$, λ is neither too big nor too small,

$$D_S(u) = \frac{1}{2} \exp\left(-\frac{|u|}{\sqrt{2}}\right) \sin\left(\frac{|u|}{\sqrt{2}} + \frac{\pi}{4}\right)$$

$$\gamma(x) = \left(\frac{\lambda}{Np(x)}\right)^{1/4}$$

and

$$G_\lambda(z, x) = \frac{1}{\gamma(x)\, p(x)} D_S\left(\frac{z - x}{\gamma(x)}\right)$$

then for x_i not too close to either a or b,

$$(\mathbf{S}_\lambda)_{ij} \approx \frac{1}{N} G_\lambda(x_i, x_j)$$

in some appropriate probabilistic sense.

Adaptive choice of bandwidth

Many (most?) implementations of locally weighted mean smoothing (Nadaraya-Watson kernel smoothing) and locally weighted linear regression do not employ a single bandwidth λ across all x_0, but rather make it depend upon x_0. The idea is to use a large bandwidth where x_i in the training set are sparse and small one where they are dense. One way of doing this is for a variable "*span*" between 0 and 1 to set

$$\lambda(x_0) = \text{minimum value of } \lambda \text{ such that at least a fraction } \textit{span} \text{ of the training cases have } x_i \text{ within } \lambda \text{ of } x_0$$

(In this context the *span* serves as a complexity parameter, typically chosen via cross-validation.)