Basis Expansions, Regularizations, and Kernel Smoothing Methods

Thomas Lonon

Division of Financial Engineering
Stevens Institute of Technology

September 27, 2017

Linear Basis Expansion

Denote by $h_m(X) : \mathbb{R}^p \mapsto \mathbb{R}$, the m^{th} transformation of X.

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$$

1. $h_m(X) = X_m, m = 1, \ldots, p$ recovers the original linear model
2. $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ allows us to augment with polynomial terms of higher order
3. $h_m(X) = \log(X_j), \sqrt{X_j}, \ldots$ permits other nonlinear transformations
4. $h_m(X) = \mathbb{I}_{\{L_m \leq X_k \leq U_m\}}$ indicators of region X_k which allows models with piecewise contributions

[2]

Methods

- Restriction Methods: decide before-hand to limit the class of functions. For example if we assume our model has the form

$$f(X) = \sum_{j=1}^{p} f_j(X_j) = \sum_{j=1}^{p} \sum_{m=1}^{M_j} \beta_{jm} h_{jm}(X_j)$$

  the size of the model is limited by the basis functions M_j
- Selection Methods: Adaptively scan the dictionary and include only those basis functions h_m that contribute significantly to the fit of the model
- Regularization Methods: Use the entire dictionary but restrict the coefficients

From Wikipedia: "In mathematics, a spline is a numeric function that is piecewise-defined by polynomial functions, and which possesses a high degree of smoothness at the places where the polynomial pieces connect (which are known as knots)"

A simple basis for representing a function would be using indicator functions:

$$h_1(X) = \mathbb{I}_{\{X<\xi_1\}}$$
$$h_2(X) = \mathbb{I}_{\{\xi_1 \leq X<\xi_2\}}$$
$$h_3(X) = \mathbb{I}_{\{X \geq \xi_2\}}$$

The least squares estimate of the model

$$f(X) = \sum_{m=1}^{3} \beta_m h_m(X)$$

amount to $\hat{\beta}_m = \bar{Y}_m$

To improve upon this model we could allow for more versatility by including the basis functions:

$$h_{m+3} = h_m(X)X, m = 1, \ldots, 3$$

If we wanted this piecewise function to be continuous at the knots, we can include constraints such as

$$f(\xi_1^-) = f(\xi_1^+)$$

which implies

$$\beta_1 + \xi_1\beta_4 = \beta_2 + \xi_1\beta_5$$

We can also incorporate the constraints into the basis, giving us at this point:

$$h_1(X) = 1$$
$$h_2(X) = X$$
$$h_3(X) = (X - \xi_1)_+$$
$$h_4(X) = (X - \xi_2)_+$$

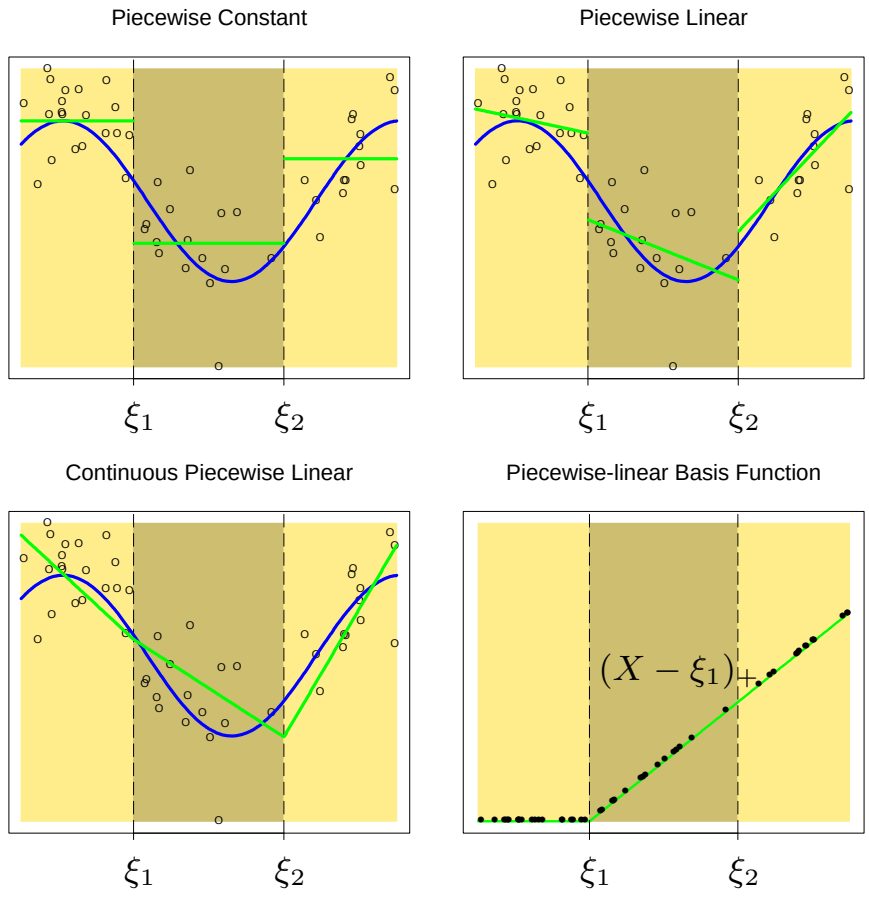


FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were generated with Gaussian noise. The remaining two panels show piecewise linear functions fit to the same data—the top right unrestricted, and the lower left restricted to be continuous at the knots. The lower right panel shows a piecewise-*

To smooth out these splines, we can include higher order terms. To improve the fit at the knots we also look at higher order terms there. This leads us to the basis for a **cubic spline** as

$$
\begin{aligned}
h_1(X) &= 1 \\
h_2(X) &= X \\
h_3(X) &= X^2 \\
h_4(X) &= X^3 \\
h_5(X) &= (X - \xi_1)_+^3 \\
h_6(X) &= (X - \xi_2)_+^3
\end{aligned}
$$

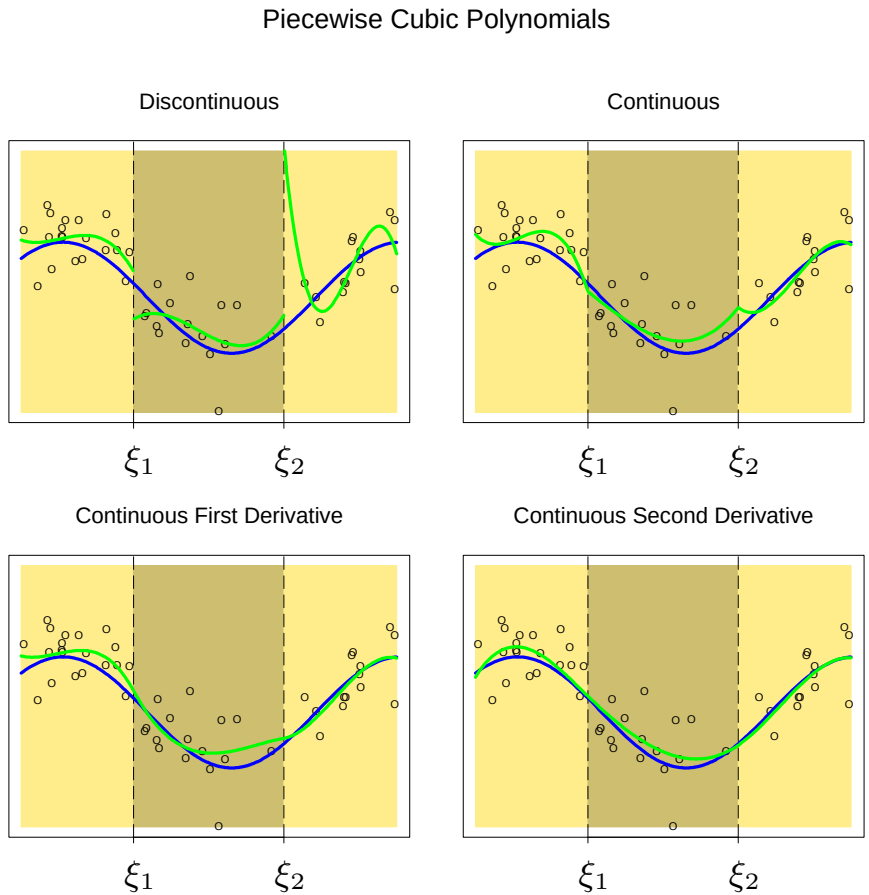


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

We can express this generally for an M-order spline with knots $\xi_j, j = 1, \ldots, K$ (which will have continuous derivatives up to order $M - 2$) as:

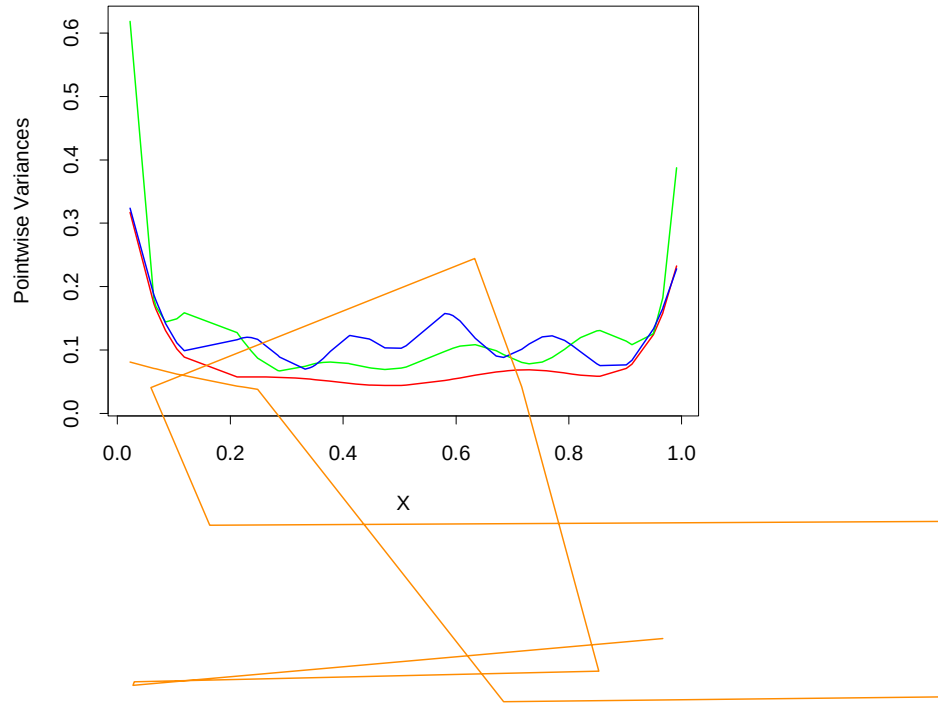
$$h_j(X) = X^{j-1}, j = 1, \ldots, M$$
$$h_{M+\ell} = (X - \xi_\ell)_+^{M-1}, \ell = 1, \ldots, K$$

a cubic spline has $M = 4$. These fixed knot splines are also known as **regression splines**[2]

These splines cause the fitting of the polynomials in the area near the boundary to be very erratic.

A **natural cubic spline** adds additional constraints that the function is linear beyond the boundary knots.

The tradeoff of this assumption is in the bias near the boundaries


Pointwise Variances
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.0
0.2
0.4
0.6
0.8
1.0
X

A natural cubic spline with K knots is represented by K basis functions. Starting with the basis for cubic splines and reducing the basis by imposing the boundary constraints gives us:

$$\begin{aligned} N_1(X) &= 1 \\ N_2(X) &= X \\ N_{k+2}(X) &= d_k(X) - d_{K-1}(X) \end{aligned}$$

where

$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$$

Smoothing Splines

An approach that avoids the knot selection problem (by using the maximal set). Among all functions f that have two continuous derivatives, find the one that minimizes

$$RSS(f, \lambda) = \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int (f''(t))^2 dt$$

where λ is a fixed **smoothing parameter**

$\lambda = 0$: f can be any function that interpolates the data
$\lambda = \infty$: simple least squares fit (no second order derivative tolerated)

It can be shown that this representation of $RSS(f, \lambda)$ has an explicit, finite-dimensional unique minimizer which is a natural cubic spline with knots at $x_i, i = 1, \ldots, N$

As a natural spline, we can express it as

$$f(x) = \sum_{j=1}^{N} N_j(x)\theta_j$$

where the $N_j(x)$ are an N-dimensional set of basis functions

The criterion reduces to:

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\Omega_N\theta$$

where $\{\mathbf{N}\}_{ij} = N_j(x_i)$ and $\{\dot{}_N\}_{jk} = \int N_j''(t)N_k''(t)dt$. The solution is:

$$\hat{\theta} = (\mathbf{N}^T\mathbf{N} + \lambda\Omega_N)^{-1}\mathbf{N}^T\mathbf{y}$$

The fitted smoothing spline is:

$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x)\hat{\theta}_j$$

A smoothing spline with a prechosen λ is called a **linear smoother**. Denote $\hat{\mathbf{f}}$ the N-vector of fitted values $\hat{f}(x_i)$ at the predictors x_i:

$$\hat{\mathbf{f}} = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\Omega_N)^{-1}\mathbf{N}^T\mathbf{y} = \mathbf{S}_\lambda\mathbf{y}$$

The finite linear operator $\mathbf{S}_\lambda$ is known as the **smoother matrix** (and only depends on λ and x_i)

Fixing the Degrees of Freedom

The **effective degrees of freedom** is given by:

$$df_\lambda = \text{trace}(\mathbf{S}_\lambda)$$

If we fix this degrees of freedom, we can specify a λ

In R, we can specify the amount of smoothing through the degrees of freedom such as in commands like: smooth.spline(x,y,df=6)

Bias-Variance Tradeoff

Since $\hat{\mathbf{f}} = \mathbf{S}_\lambda \mathbf{y}$,

$$\begin{aligned} Cov(\hat{\mathbf{f}}) &= \mathbf{S}_\lambda Cov(\mathbf{y}) \mathbf{S}_\lambda^T \\ &= \mathbf{S}_\lambda \mathbf{S}_\lambda^T \end{aligned}$$

The diagonal contains the pointwise variances at the training x_i. The bias is given by:

$$\begin{aligned} Bias(\hat{\mathbf{f}}) &= \mathbf{f} - \mathbb{E}[\hat{\mathbf{f}}] \\ &= \mathbf{f} - \mathbf{S}_\lambda \mathbf{f} \end{aligned}$$

where $\mathbf{f}$ is the vector of evaluations of the true f at the training X's

The integrated squared prediction error (EPE) combines both bias and variance in a single summary:

$$\begin{aligned} EPE(\hat{f}_\lambda) &= \mathbb{E}[Y - \hat{f}_\lambda(X)]^2 \\ &= \mathbb{V}(Y) + \mathbb{E}[Bias^2(\hat{f}_\lambda(X)) + \mathbb{V}(\hat{f}_\lambda(X))] \\ &= \sigma^2 + MSE(\hat{f}_\lambda) \end{aligned}$$

In reality we don't know the real function and so can't use the EPE. Instead we could look at some of our alternate methods, such as the LOOCV (N-fold):

$$CV(\hat{f}_\lambda) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_\lambda^{(-i)}(x_i))^2$$
$$= \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_\lambda(i,i)}\right)^2$$

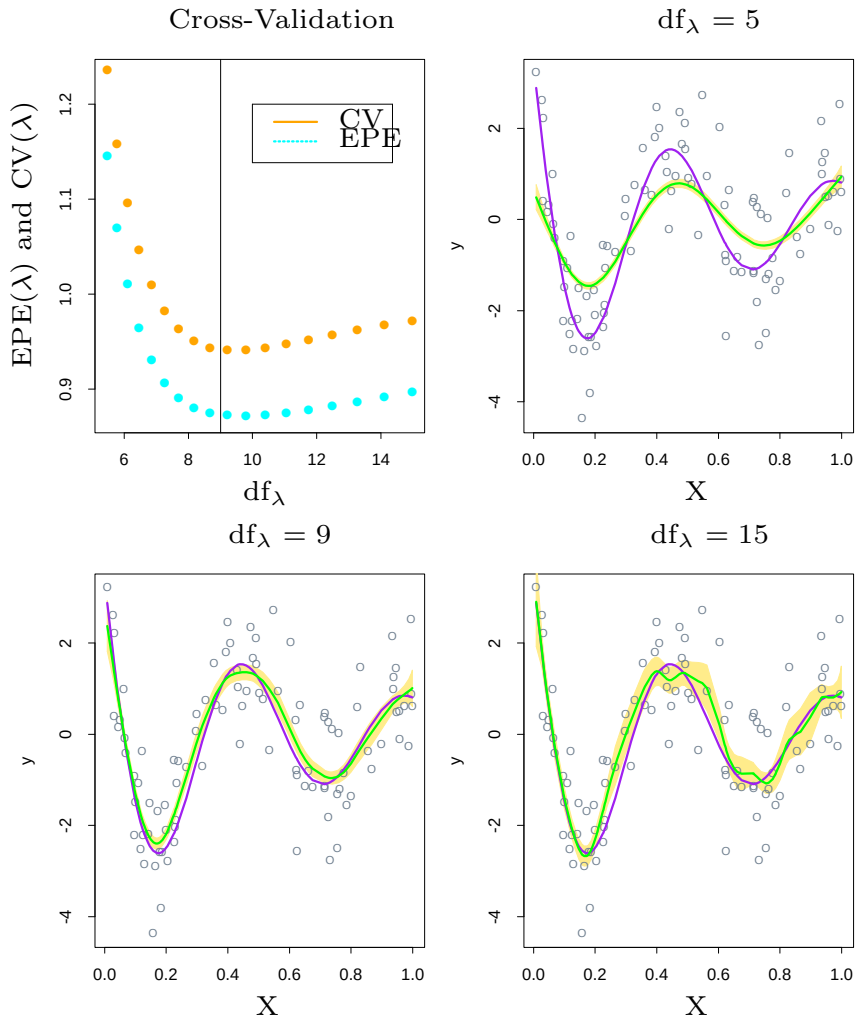


FIGURE 5.9. *The top left panel shows the* EPE(λ) *and* CV(λ) *curves for a realization from a nonlinear additive error model (5.22). The remaining panels show the data, the true functions (in purple), and the fitted curves (in green) with yellow shaded* $\pm 2\times$ *standard error bands, for three different values of* df_λ.

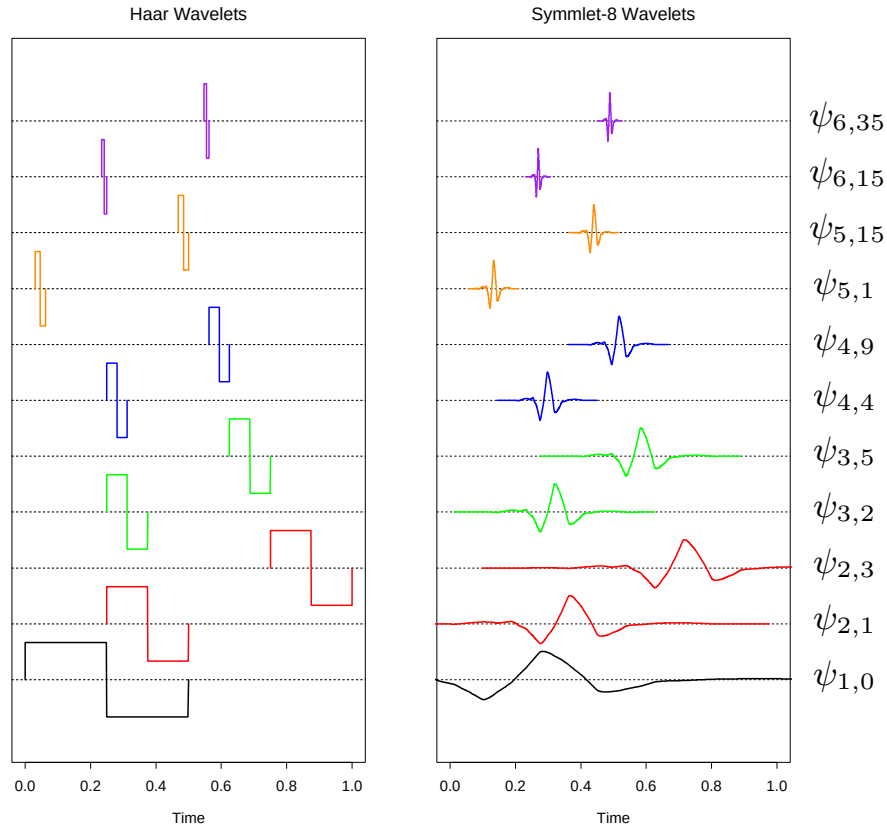


FIGURE 5.16. *Some selected wavelets at different translations and dilations for the Haar and symmlet families. The functions have been scaled to suit the display.*

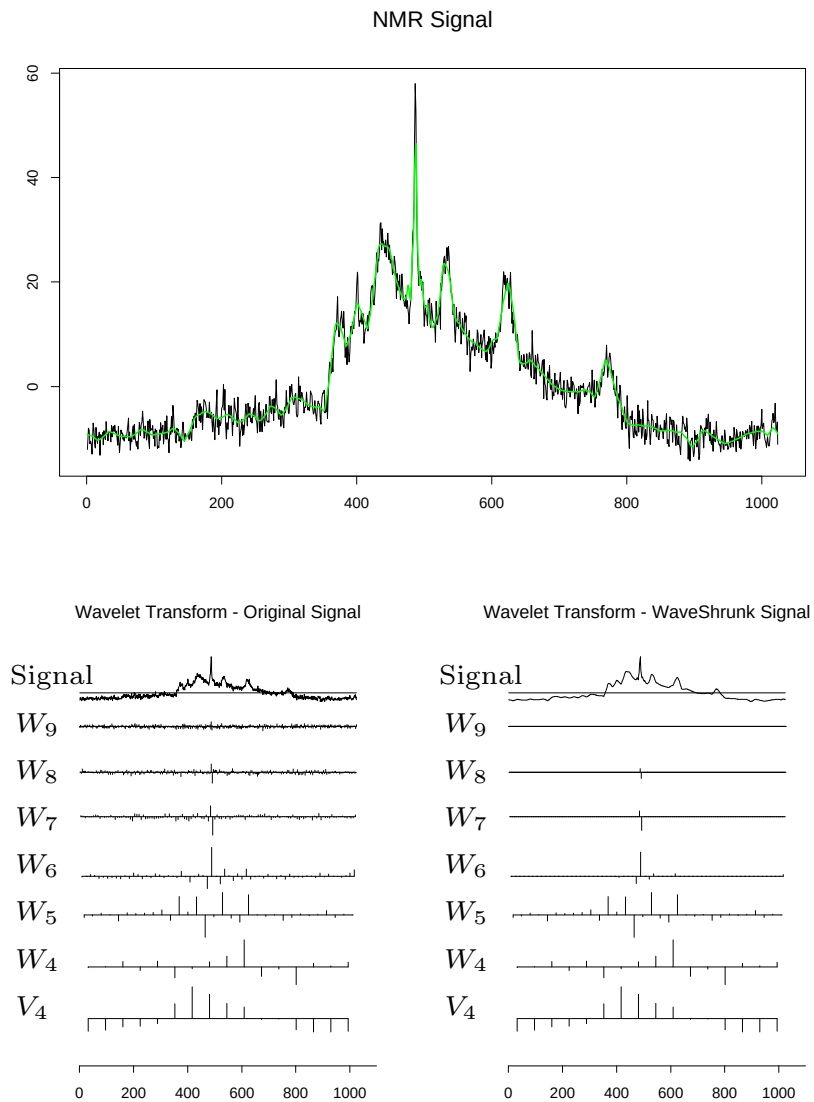


FIGURE 5.17. *The top panel shows an NMR signal, with the wavelet-shrunk version superimposed in green. The lower left panel represents the wavelet trans-*

K-Nearest Neighbor Average

The k-nearest neighbor average is given by:

$$\hat{f}(x) = \text{Ave}(y_i | x_i \in N_k(x))$$

which is an estimator for $\mathbb{E}[Y|X = x]$ where $N_k(x)$ is the set of k points closest to x. This leads to a discontinuous estimator.

The continuous version is the Nadaraya-Watson kernel-weighted average

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

with the *Epanechnikov* quadratic kernel

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

with

$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2), & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

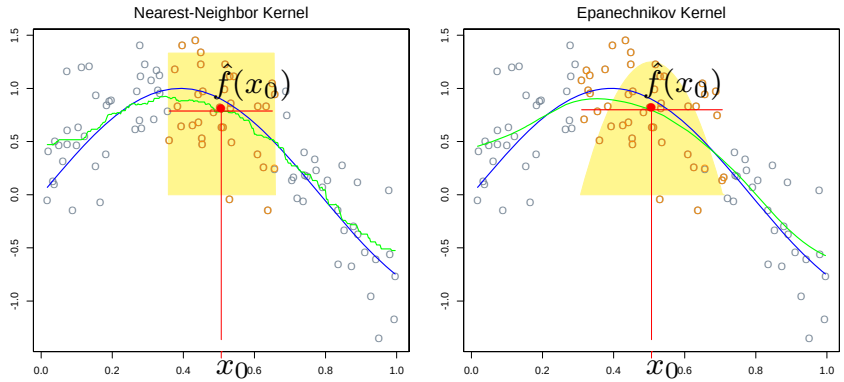


FIGURE 6.1. *In each panel* 100 *pairs* x_i, y_i *are generated at random from the blue curve with Gaussian errors:* $Y = \sin(4X) + \varepsilon$, $X \sim U[0,1]$, $\varepsilon \sim N(0, 1/3)$. *In the left panel the green curve is the result of a* 30*-nearest-neighbor running-mean smoother. The red point is the fitted constant* $\hat{f}(x_0)$, *and the red circles indicate those observations contributing to the fit at* x_0. *The solid yellow region indicates the weights assigned to observations. In the right panel, the green curve is the kernel-weighted average, using an Epanechnikov kernel with (half) window width* $\lambda = 0.2$.

Local Linear Regression

Locally weighted regression solves a separate problem at each x_0:

$$\min_{\alpha(x_0),\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i)[y_i - \alpha(x_0) - \beta(x_0)x_i]^2$$

which will give the estimate $\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$.

Algorithm 7.1: Local Regression at $X = x_0$

1. Gather the fraction $s = \frac{k}{n}$ of training points whose x_i are closest to x_0.
2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest as the highest weight. All but these k nearest neighbors get weight zero.
3. Fit a *weighted least squares regression* of the y_i on the x_i using aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize
$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2$$
4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$

[1]

Local Polynomial Regression

We can fit local polynomials of degree d using:

$$\min_{\alpha(x_0),\beta_j(x_0),j=1,\ldots,d} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \left[y_i - \alpha(x_0) - \sum_{j=1}^{d} \beta_j(x_0) x_i^j \right]^2$$

with local solutions:

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \sum_{j=1}^{d} \hat{\beta}_j(x_0) x_0^j$$

In each kernel, K_λ, λ is a parameter that controls the width:

- For the Epanechinikov kernel with metric width, λ is the radius of the support region
- For the Gaussian kernel, λ is the standard deviation
- λ is the number k of nearest neighbors in k-nearest neighborhoods often expressed as a fraction or *span* k/N of the total training sample

[2]

The width of the window influences the bias-variance tradeoff (most explicit for local averages)

- If the window is narrow, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0, and its variance will be relatively large-close to that of an individual y_i. The bias will tend to be small, again because each of the $\mathbb{E}[y_i] = f(x_i)$ should be close to $f(x_0)$.
- If the window is wide, the variance of $\hat{f}(x_0)$ will be small relative to the variance of any y_i, because of the effects of averaging. The bias will be higher, because we are now using observations x_i further from x_0, and there is no guarantee that $f(x_i)$ will be close to $f(x_0)$.

[2]

[1] Trevor Hastie Gareth James, Daniela Witten and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Number v. 6. Springer, 2013.

[2] Robert Tibshirani Trevor Hastie and Jerome Friedman. *The Elements of Stastical Learning: Data Mining, Inference, and Prediction*. Number v.2 in Springer Series in Statistics. Springer, 2009.