Lec 16: Connections and generalization

Weiping Zhang

December 17, 2020

Linear Smoother

- Variance reduction and influence
- Degrees of freedom and unbiased risk estimation
  - Leave-one-out and generalized cross-validation
- Inference with linear smoothers

Smoothing and Penalized Least Square

Locally adaptive estimators

- Wavelets
- The strengths of wavelets, the limitations of linear smoothers
- Locally adaptive regression splines

Linear Smoother

- Literally every estimator we have discussed so far, trained on $(x_i, y_i) \in \mathcal{R}^d \times \mathcal{R}$, $i = 1, \ldots, n$, produces fitted values $\hat{\mathbf{f}} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))$ of the form

$$\hat{\mathbf{f}} = Sy$$

  for some matrix $S \in \mathcal{R}^{n \times n}$ depending on the inputs $x_1, \ldots, x_n$—and also possibly on a tuning parameter such as h in kernel smoothing, or λ in smoothing spline–but not on y.

- For some of the smoothers we have defined we can define a weight sequence for any x and define

$$\hat{f}(x) = \sum_{i=1}^{n} W_i(x) y_i.$$

- Recall that such estimators are called **linear smoothers**
- Often, in a predictive setting, we want to compare (estimates of) test error between several methods (e.g., linear regression, k-nearest-neighbors, kernel smoothing, spline smoothing) in order choose between them. **How can we characterize the amount of smoothing being performed?**
- The smoothing parameters provide a characterization, but it is not ideal because it does not permit us to compare between different smoothers and for smoothers like loess it does not take into account the shape of the weight function nor the degree of the polynomial being fit.

Variance reduction and influence

- The variance of the interpolation estimate is $Var(y_1) = \sigma^2$. The variance of our smooth estimate is

$$Var(\hat{f}(x)) = \sigma^2 \sum_{i=1}^{n} W_i^2(x)$$

  so we define $\sum_{i=1}^{n} W_i^2(x)$ as **the variance reduction**. Under mild conditions one can show that this is less than 1.

- Because

$$\sum_{i=1}^{n} Var(\hat{f}(x_i)) = \sigma^2 tr(SS')$$

  the total variance reduction from $\sum_{i=1}^{n} Var(y_i)$ is $tr(SS')/n$.

- The sensitivity of the fitted value, say $\hat{f}(x_i)$, to the data point y_i can be measured by $W_i(x_i)/\sum_{j=1}^{n} W_i(x_j)$ or S_{ii} (remember the denominator is usually 1).

- The total influence or sensitivity is $\sum_{i=1}^{n} W_i(x_i) = tr(S)$.

Degrees of freedom and unbiased risk estimation

- The notion of degrees of freedom gives us a way of precisely making this comparison. Roughly speaking, the degrees of freedom of a fitting procedure (like kernel regression with $h = 1.5$, or k-nearest-neighbors with $k = 10$) describes the effective number of parameters used by this procedure, and hence provides a quantitive measure of estimator complexity.
- Keeping track of degrees of freedom therefore saves us from unsuspectingly comparing a procedure that uses say, 10 effective parameters to another that uses 100.

- Even though the concept it represents is quite broad, degrees of freedom has a rigorous definition. Suppose that we observe

$$y_i = f(x_i) + e_i$$

where the errors $e_i, i = 1, \ldots, n$ are uncorrelated with common variance σ^2 (note: this is weaker than assuming $e_i \sim N(0, \sigma^2)$, i.i.d. for $i = 1, \ldots, n$). Here we will treat the predictor measurements $x_i, i = 1, \ldots, n$ as fixed (equivalently: consider conditioning on the values of the random predictors).

- Now consider the fitted values $\hat{y}_i = \hat{f}(x_i), i = 1, \ldots, n$ from a regression estimator $\hat{f}$. We define the degrees of freedom of $\hat{y}$ (i.e., the degrees of freedom of $\hat{f}$) as

$$df(\hat{y}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} Cov(\hat{y}_i, y_i).$$

- The above definition of degrees of freedom looks at the amount of covariance between each point y_i and its corresponding fitted values $\hat{y}_i$. We add these up over $i = 1, \ldots, n$, and divide the result by σ^2(dividing by σ^2 gets rid of the dependence of the sum on the marginal error variance).
- It is going to be helpful for some purposes to rewrite the definition of degrees of freedom in matrix notation. This is

$$df(\hat{y}) = \frac{1}{\sigma^2} tr(Cov(\hat{y}, y)).$$

- In linear regression the variance reduction is related to the degrees of freedom, or number of parameters. For linear regression, $\sum_{i=1}^{n} Var(\hat{f}(x_i)) = p\sigma^2$. The degrees of freedom is

$$df(\hat{\mathbf{f}}) = \frac{1}{\sigma^2} tr(Cov(\hat{y}, y)) = tr(SS') = tr(S) = p$$

  where $S = X(X'X)^{-1}X'$.
- One widely used definition of degrees of freedoms for linear smoothers $\hat{\mathbf{f}} = S\mathbf{y}$ is

$$df(\hat{\mathbf{f}}) = tr(S)$$

- Finally we notice that

$$E[(\mathbf{y} - \hat{\mathbf{f}})'(\mathbf{y} - \hat{\mathbf{f}})] = \sigma^2(n - 2tr(S) + tr(SS'))$$

  In the linear regression case this is $(n-p)\sigma^2$. We therefore denote $n - 2tr(S) + tr(SS')$ as the residual degrees of freedom.
- A third definition of degrees of freedom of a smoother is then $2tr(S) - tr(SS')$.

Example

- For a regression spline estimator, of polynomial degree k, with knots at the locations $t_1, \ldots, t_p$, recall that $\hat{\mathbf{f}} = G(G'G)^{-1}G'y$ for $G \in \mathcal{R}^{n \times (p+k+1)}$ the degree k spline basis matrix over the knots $t_1, \ldots, t_p$. Therefore

$$df(\hat{\mathbf{f}}) = tr(G(G'G)^{-1}G') = tr(G'G(G'G)^{-1}) = p + k + 1,$$

- The degrees of freedom of a regression spline is

$$the\ number\ of\ knots + polynomial\ degree + 1$$

  The same calculation shows that the degrees of freedom of a regression natural spline is simply the number of knots (independent of the polynomial order)

Example

- For a smoothing spline estimator, recall that we were able to write the fitted values as $\hat{\mathbf{f}} = (I + \lambda K)^{-1} y$, i.e., as

$$\hat{\mathbf{f}} = U(1 + \lambda D)^{-1} U' y,$$

where UDU' is the eigendecomposition of the Reinsch matrix $K = (N')^{-1} \Omega N^{-1}$(and here K depends only on the input points $x_1, \ldots, x_n$ and the polynomial order k). The smoothing spline hence has degrees of freedom

$$df(\hat{\mathbf{f}}) = tr(U(1 + \lambda D)^{-1} U') = \sum_{i=1}^{n} \frac{1}{1 + \lambda d_j},$$

where $D = diag(d_1, \ldots, d_n)$. This is monotone decreasing in λ, with $df(\hat{\mathbf{f}}) = n$ when $\lambda = 0$, and $df(\hat{\mathbf{f}}) \to (k+1)/2$ when $\lambda \to \infty$, the number of zero eigenvalues among $d_1, \ldots, d_n$.

- Degrees of freedom is generally a useful concept since it allows us to put two different estimators on equal footing. E.g., suppose we wanted to compare kernel smoothing versus smoothing splines; we could tune them to match their degrees of freedom, and then compare their performance.
- A second more concrete motivation for considering degrees of freedom: it allows us to form an unbiased estimate of the error, or risk. Let $y_i = f(x_i) + e_i, i = 1, \ldots, n$, $Ee_i = 0$, $Var(e_i) = \sigma^2$. $\mathbf{f} = (f(x_1), \ldots, f(x_n)^T \in \mathcal{R}^n$ be the vector given by evaluating the underlying regression function at the inputs, i.e., $E\mathbf{y} = \mathbf{f}$. Then

$$\widehat{Err} = \frac{1}{n}\|\mathbf{y} - \hat{\mathbf{f}}\|_2^2 - \sigma^2 + \frac{2\sigma^2}{n} df(\hat{\mathbf{f}})$$

serves as an unbiased estimate of the error $Err = E\|\mathbf{f} - \hat{\mathbf{f}}\|_2^2/n$. This is simply

$$\widehat{Err} = \frac{1}{n}\|y - Sy\|_2^2 - \sigma^2 + \frac{2\sigma^2}{n} tr(S)$$

- Meanwhile, if

$$y_i' = f(x_i) + \epsilon_i', \quad i = 1, \ldots n$$

is an independent test sample (important: note here that the predictors measurements x_i, $i = 1, \ldots n$ are the same-i.e., we considering these fixed) from the same distribution as our training sample, then

$$\mathbb{E}\|y' - \hat{y}\|^2/n$$

is the expected test error

- Interestingly, it turns out that (in this simple setup, with $x_i, i = 1, \ldots n$ fixed) we have the relationship

$$\mathbb{E}\|y' - \hat{y}\|^2/n = \mathbb{E}\|y - \hat{y}\|^2/n + \frac{2\sigma^2}{n} df(\hat{\mathbf{f}})$$

- Suppose our linear smoother of interest depends on a tuning parameter α (e.g., h for kernel smoothing, λ for smoothing splines, or λ for Mercer kernels), and express this as $\hat{\mu}_\alpha = S_\alpha y$. Then we could choose the tuning parameter α to minimize the estimated test error, as in

$$\hat{\alpha} = \arg\min_\alpha \frac{1}{n}\|y - S_\alpha y\|_2^2 + \frac{2\sigma^2}{n} tr(S_\alpha).$$

- This is just like the C_p criterion, or AIC, in ordinary linear regression (we could also replace the factor of 2 above with $logn$ to obtain something like BIC)

Leave-one-out and generalized cross-validation

- Of course, cross-validation gives us another way to perform error estimation and model selection. For linear smoothers $\hat{\mathbf{f}} = (\hat{f}(x_1), \ldots, \hat{f}(x_n)) = Sy$, leave-one-out cross-validation can be particularly appealing because in many cases we have the seemingly magical reduction

$$CV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}^{-i}(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right)^2, \quad (1)$$

where $\hat{f}^{-i}$ denotes the estimated regression function that was trained on all but the ith pair (x_i, y_i). This leads to a big computational savings since it shows us that, to compute leaveone-out cross-validation error, we don't have to actually ever compute $\hat{f}^{-i}, i = 1, \ldots, n$

- Why does (1) hold, and for which linear smoothers $\hat{\mathbf{f}} = Sy$? Just rearranging (1) perhaps demystifies this seemingly magical relationship and helps to answer these questions. Suppose we knew that $\hat{f}$ had the property

$$\hat{f}^{-i}(x_i) = \frac{1}{1 - S_{ii}}(\hat{f}(x_i) - S_{ii}y_i). \tag{2}$$

  That is, to obtain the estimate at x_i under the function $\hat{f}^{-i}$ fit on all but (x_i, y_i), we take the sum of the linear weights (from our original fitted function $\hat{f}$) across all but the ith point, $\hat{f}(x_i) - S_{ii}y_i = \sum_{j \neq i} S_{ij}y_j$, and then renormalize so that these weights sum to 1.

- This is not an unreasonable property; e.g., we can immediately convince ourselves that it holds for kernel smoothing. A little calculation shows that it also holds for smoothing splines (using the Sherman-Morrison update formula). How about for k-nearest-neighbors?

- From the special property (2), it is easy to show the leave-one-out formula (1). We have

$$y_i - \hat{f}^{-i}(x_i) = \frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}$$

  and and then squaring both sides and summing over n gives (1).
- Finally, generalized cross-validation is a small twist on the right-hand side in (1) that gives an approximation to leave-one-out cross-validation error. It is defined as by replacing the appearences of diagonal terms S_{ii} with the average diagonal term $tr(S)/n$,

$$GCV(\hat{f}) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x_i)}{1 - tr(S)/n}\right)^2.$$

- This can be of computational advantage in some cases where $tr(S)$ is easier to compute that individual elements S_{ii}, and is also closely tied to the unbiased test error estimate in $\widehat{Err}$, seen by making the approximation $1/(1-x)^2 \approx 1 + 2x$

Inference with linear smoothers

- Now we will learn the inference tools—pointwise confidence intervals, and F tests between fitted models—for general linear smoothers, beyond linear regression. We will assume a model

$$y_i = f(x_i) + e_i$$

  where $x_i, i = 1, \ldots, n$ are considered fixed. Because our estimator $\hat{f}$ is a linear smoother, we can write the fit as $\hat{f}(x_0) = w(x_0)^T y$ at an arbitrary point x_0, and $\hat{y} = Sy$ for the vector of fitted values across $x_1, \ldots, x_n$.

- To preface, there are certainly other ways to construct confidence intervals and significance tests than the "direct" ones we describe below, e.g., Bootstrap. But the direct tools are more computationally efficient, have a close tie to those from linear regression, and are already implemented in R software, so they're worth knowing.

- The tools that we will describe below, just like those for linear regression, assume that the inputs $x_1, \ldots, x_n$ are fixed. The standard pairs bootstrap, on the other hand, treats the inputs as random (since we resample pairs (x_i, y_i)). To use the bootstrap and respect the fixed input setup, we'd have to use the residual bootstrap.
- This is not to say that one route is generally less correct than the other, but rather, that these differences should be kept in mind when comparing the results produced by different tools.

Pointwise confidence intervals for the regression function

- Just as in the linear regression case, at an arbitrary point x_0, the variance of the fit $\hat{f}(x_0) = w(x_0)^T y$ is

$$Var(\hat{f}(x_0)) = \sigma^2 w(x_0)^T w(x_0)$$

- How to estimate σ? We can now use the estimate

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - d}$$

where $d = df(\hat{y}) = tr(S)$, the degrees of freedom of the fit $\hat{y}$. Note: this replaces p in the usual expression for the estimated error variance in linear regression, so it should make intuitive sense to you from what you know about degrees of freedom. Now, $(n-d)\hat{\sigma}^2/\sigma^2 \sim \chi^2_{n-d}$, but this is only an approximation in the case of general linear smoothers, and not exact like it was for linear regression. It is a good approximation nonetheless.

- This yields the estimated variance of $\hat{f}(x_0)$

$$\hat{s}^2(\hat{f}(x_0)) = \hat{\sigma}^2 w(x_0)^T w(x_0),$$

and from the same arguments as before, an approximate $(1-\alpha)$ confidence interval for $f(x_0)$, the underlying regression function at a point x_0, is

$$[\hat{f}(x_0) - q_2 \hat{s}(\hat{f}(x_0)), \hat{f}(x_0) - q_1 \hat{s}(\hat{f}(x_0))],$$

where q_1, q_2 are the $\alpha/2, (1-\alpha/2)$ quantiles of t_{n-d}, respectively.

- For confidence intervals of the regression function at the observed inputs $x_i, i = 1, \ldots, n$, the same story holds; an approximate confidence interval for $f(x_i)$ is $[\hat{y}_i - q_2 \hat{s}(\hat{y}), \hat{y} - q_1 \hat{s}(\hat{y})]$. Now

$$\hat{s}^2(\hat{y}_i) = \hat{\sigma}^2 w(x_i)^T w(x_i),$$

or another way of writing this is to use the fact that

$$Var(\hat{y}) = Var(Sy) = \sigma^2 SS^T$$

so $Var(\hat{y}_i) = \sigma^2 (SS^T)_{ii}$, and the estimated variance is $\hat{s}^2(\hat{y}) = \hat{\sigma}^2 (SS^T)_{ii}$.

Learning to love the bias

- It is important to take a step back and think about the bias. Note that the confidence intervals in the last section utilize the t_{n-d} distribution for the calculation of quantiles q_1, q_2. As in the linear regression case, this stems from claiming that

$$\frac{\hat{f}(x_0) - f(x_0)}{\hat{s}(\hat{f}(x_0))} \sim t_{n-d}$$

- Remember that this is now an approximate result, because the denominator is only approximately χ^2_{n-d}(times constant factors). But there is something else going on too: in modeling this statistic as t_{n-d}, we are assuming that its numerator has mean zero, i.e.,

$$E\hat{f}(x_0) = f(x_0)$$

or at least approximately so.

- Note that this is the same as saying that $\hat{f}(x_0)$ has zero bias or at least small bias. When this is true, i.e., when the bias is small, then we are more or less justified in saying that (2) holds, so that our confidence interval provides appropriate coverage for $f(x_0)$
- But when this is not true, i.e., when $\hat{f}(x_0)$ is badly biased, then we nevertheless have that

$$\frac{\hat{f}(x_0) - E\hat{f}(x_0)}{\hat{s}(\hat{f}(x_0))} \sim t_{n-d}$$

(or again, at least approximately so) and therefore the confidence interval that we construct $[\hat{f}(x_0) - q_2\hat{s}(\hat{f}(x_0)), \hat{f}(x_0) - q_1\hat{s}(\hat{f}(x_0))]$ is actually a confidence interval for $E\hat{f}(x_0)$, rather than $f(x_0)$.

- So what is $E\hat{f}(x_0)$? Well

$$E\hat{f}(x_0) = w(x_0)^T f(x_0)$$

which is a smoothed version of $f(x_0)$. In terms of the fitted values $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_n)$, we have

$$E\hat{y} = Sf$$

where $f = (f(x_1), \ldots, f(x_n))^T$, and again we can think of $E[\hat{y}]$ as a smoothed version of the true regression function values.

- Hence, in the presence of nonnegligible bias, we have to keep it in mind that our confidence intervals are really for $E[\hat{f}(x_0)]$ or $E[\hat{y}_i]$, which are smoothed versions of the true regression function values $f(x_0)$ and $f(x_i)$, and not the true values themselves.

Significance tests between fitted models

- Here we present an analog of the F test in linear regression. Suppose that we are comparing two estimates $\hat{f}_1$ and $\hat{f}_2$, and the model class for $\hat{f}_1$ is nested within that of $\hat{f}_2$. Write

$$\hat{y}^{(1)} = S_1 y, \quad \hat{y}^{(2)} = S_2 y,$$

for the fitted values from $\hat{f}_1$ and $\hat{f}_2$ respectively,

$$d_1 = tr(S_1), \quad d_2 = tr(S_2)$$

for their respective degrees of freedom, and also

$$RSS_1 = \sum_{i=1}^{n} (y_i - \hat{y}^{(1)})^2, \quad RSS_2 = \sum_{i=1}^{n} (y_i - \hat{y}^{(2)})^2$$

for their respective residual sums of squares

- A standard example is when $\hat{f}_1$ is a linear fit and $\hat{f}_2$ is a more flexible fit coming from (say) a smoothing spline. Expressing the true regression function as $f(x) = \beta_0 + \beta_1 x + \delta(x)$, we wish to test the hypothesis

$$H_0 : \delta(x) = 0 \leftrightarrow H_1 : \delta(x) \neq 0$$

- In general, we must assume that $\hat{y}_i^{(2)} = \hat{f}_2(x_i)$ is approximately unbiased for $f(x_i), i = 1, \ldots, n$, and that $\hat{y}_i^{(1)} = \hat{f}_1(x_i)$ is approximately unbiased for $f(x_i), i = 1, \ldots, n$ under the null hypothesis. Then the F statistic for testing the significance of the fit $\hat{y}^{(2)}$ over $\hat{y}^{(1)}$ is

$$\frac{(RSS_1 - RSS_2)/(d_2 - d_1)}{RSS_2/(n - d_2)},$$

- and the null hypothesis, it holds that, approximately,

$$\frac{(RSS_1 - RSS_2)/(d_2 - d_1)}{RSS_2/(n - d_2)} \sim F_{d_2-d_1,n-d_2}.$$

  As before, we reject when this statistic exceeds q, the $(1 - \alpha)$ quantile of $F_{d_2-d_1,n-d_2}$.

Smoothing and Penalized Least Square

- In last lecture, we saw that the smoothing spline solution to a penalized least squares is a linear smoother.
- We can write the penalized criterion

$$(\mathbf{y} - B\theta)'(\mathbf{y} - B\theta) + \lambda\theta'\Omega\theta \tag{3}$$

- Setting derivatives with respect to θ equal to 0 gives the estimating equation

$$(B'B + \lambda\Omega)\theta = B'\mathbf{y}$$

the $\hat{\theta}$ that solves this equation will give us the estimate $\hat{g} = B\hat{\theta}$.

- Write

$$\hat{g} = B\hat{\theta} = B(B'B + \lambda\Omega)^{-1}B'\mathbf{y} = (I + \lambda K)^{-1}\mathbf{y}$$

  where $K = B'^{-1}\Omega B^{-1}$

- Notice we can write the penalized criterion as

$$(\mathbf{y} - g)'(\mathbf{y} - g) + \lambda g' K g$$

- If we plot the rows of this linear smoother we will see that it is like a kernel smoother.
- Notice that for any linear smoother with a symmetric and nonnegative definite S, i.e. there S^{-1} exists, then we can argue in reverse: $\hat{\mathbf{f}} = S\mathbf{y}$ is the value that minimizes the penalized least squares criteria of the form

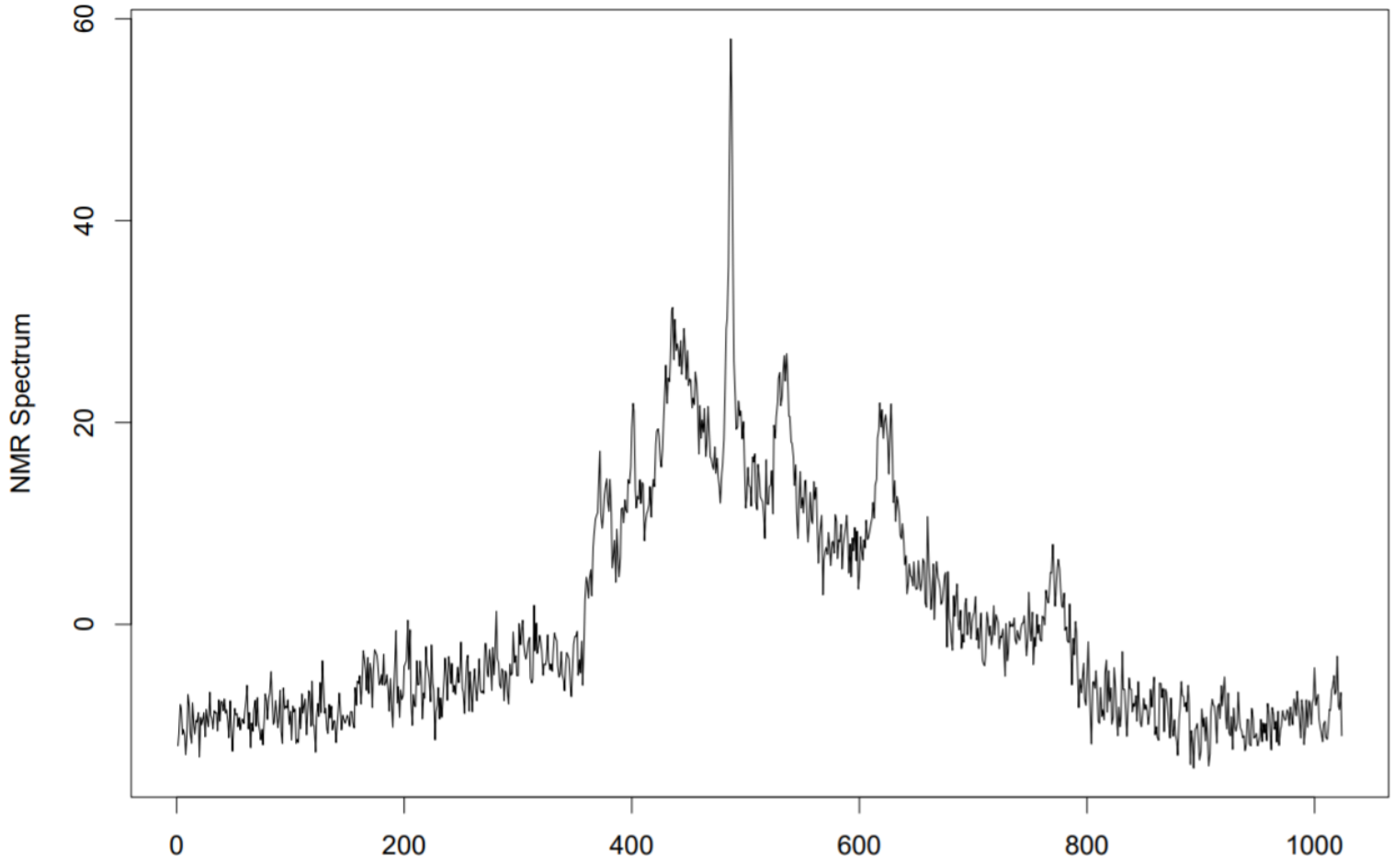
$$(\mathbf{y} - \mathbf{f})'(\mathbf{y} - \mathbf{f}) + \mathbf{f}'(S^{-1} - I)\mathbf{f}$$

Wavelets

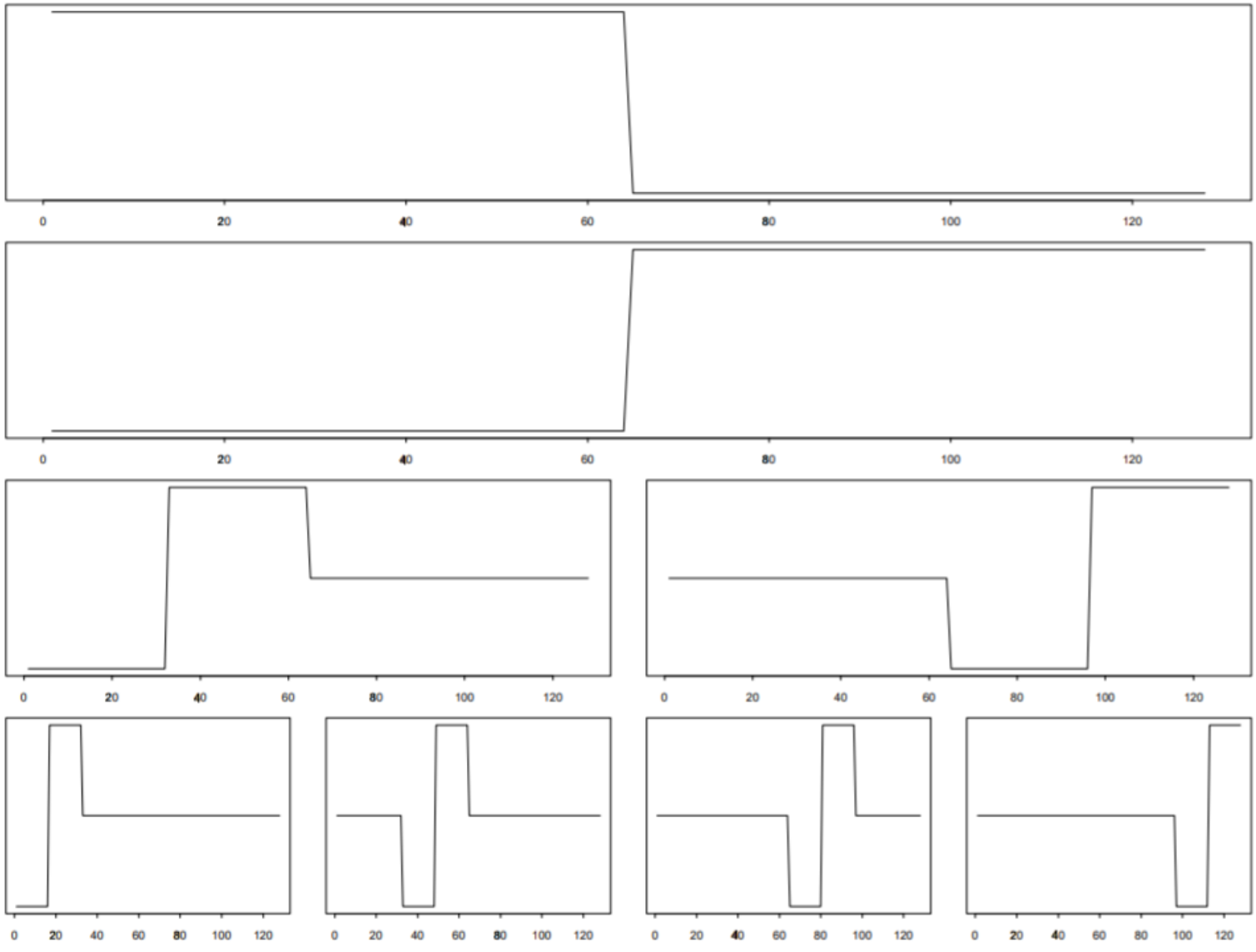
- Not every nonparametric regression estimate needs to be a linear smoother (though this does seem to be very common), and wavelet smoothing is one of the leading nonlinear tools for nonparametric estimation. The theory of wavelets is elegant and we only give a brief introduction here; see Mallat (2008) for an excellent reference.
- You can think of wavelets as defining an orthonormal function basis, with the basis functions exhibiting a highly varied level of smoothness. Importantly, these basis functions also display spatially localized smoothness at different locations in the input domain. There are actually many different choices for wavelets bases (Haar wavelets, symmlets, etc.), but these are details that we will not go into.

Reference: `http://staff.washington.edu/dbp/s530/`

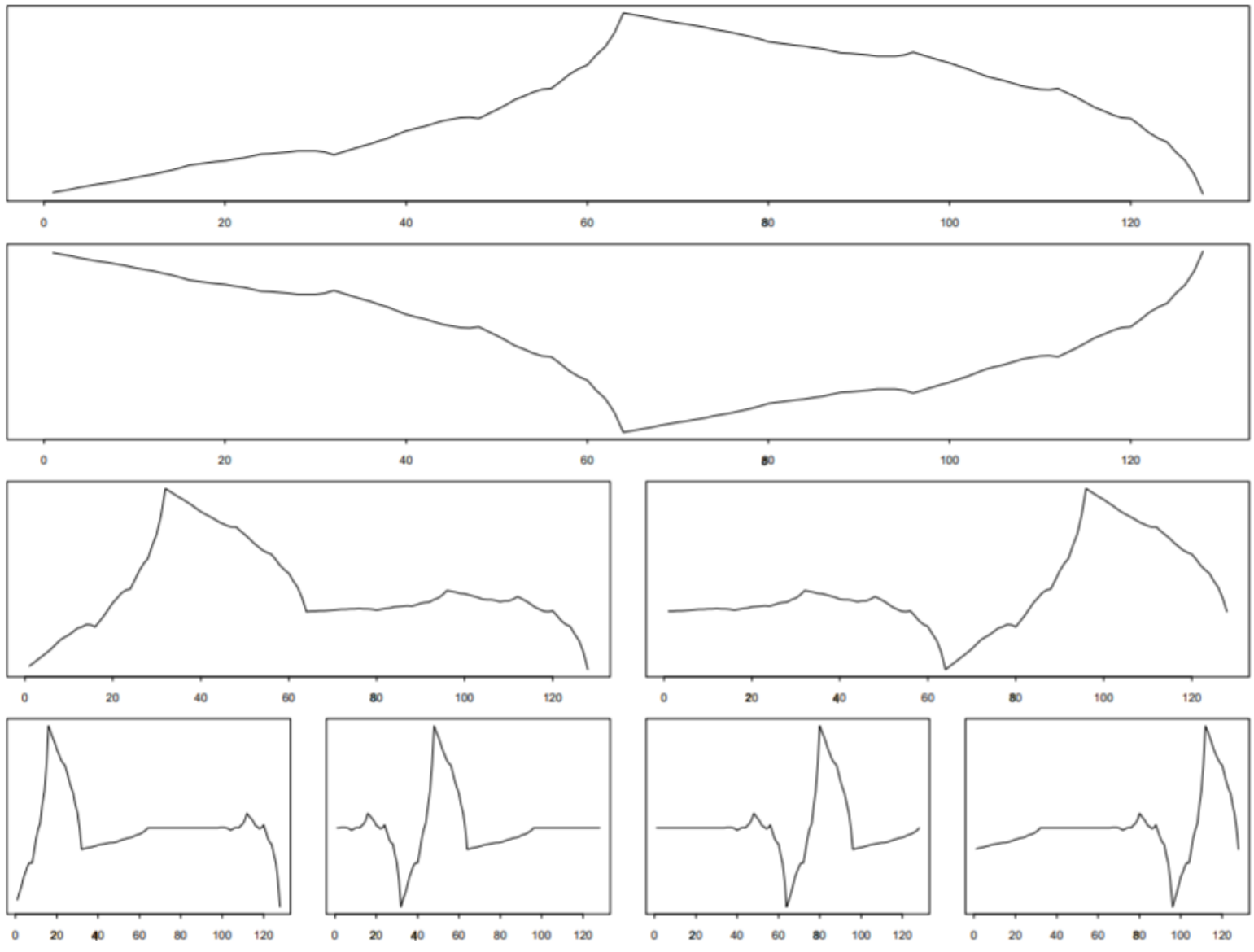
The following plot show a nuclear magnetic resonance (NMR) signal.



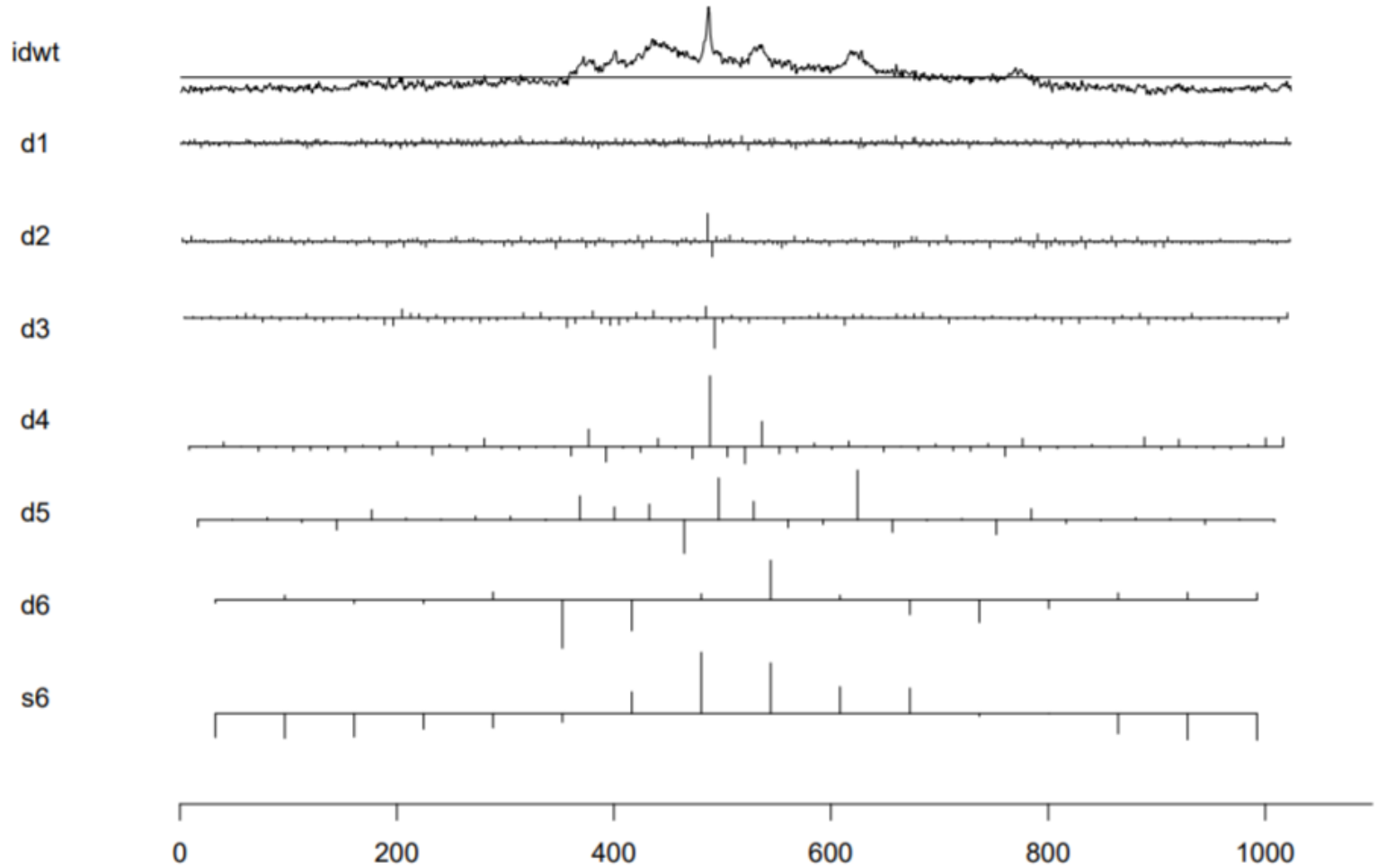
- The signal does appear to have some added noise so we could use $y_i = f(x_i) + e_i$ to model the process. However, $f(x)$ appears to have a peak at around $x = 500$ making it not very smooth at that point.
- Situations like these are where wavelets analyses is especially useful for "smoothing". Now a more appropriate word is "de-noising".
- The Discrete Wavelet Transform defines an orthogonal basis just like the DFT and DCT. However the columns of DWT are locally smooth. This means that the coefficients can be interpreted as local smoothness of the signal for different locations.
- Here are the columns of the Haar DWT, the simplest wavelet.


0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120

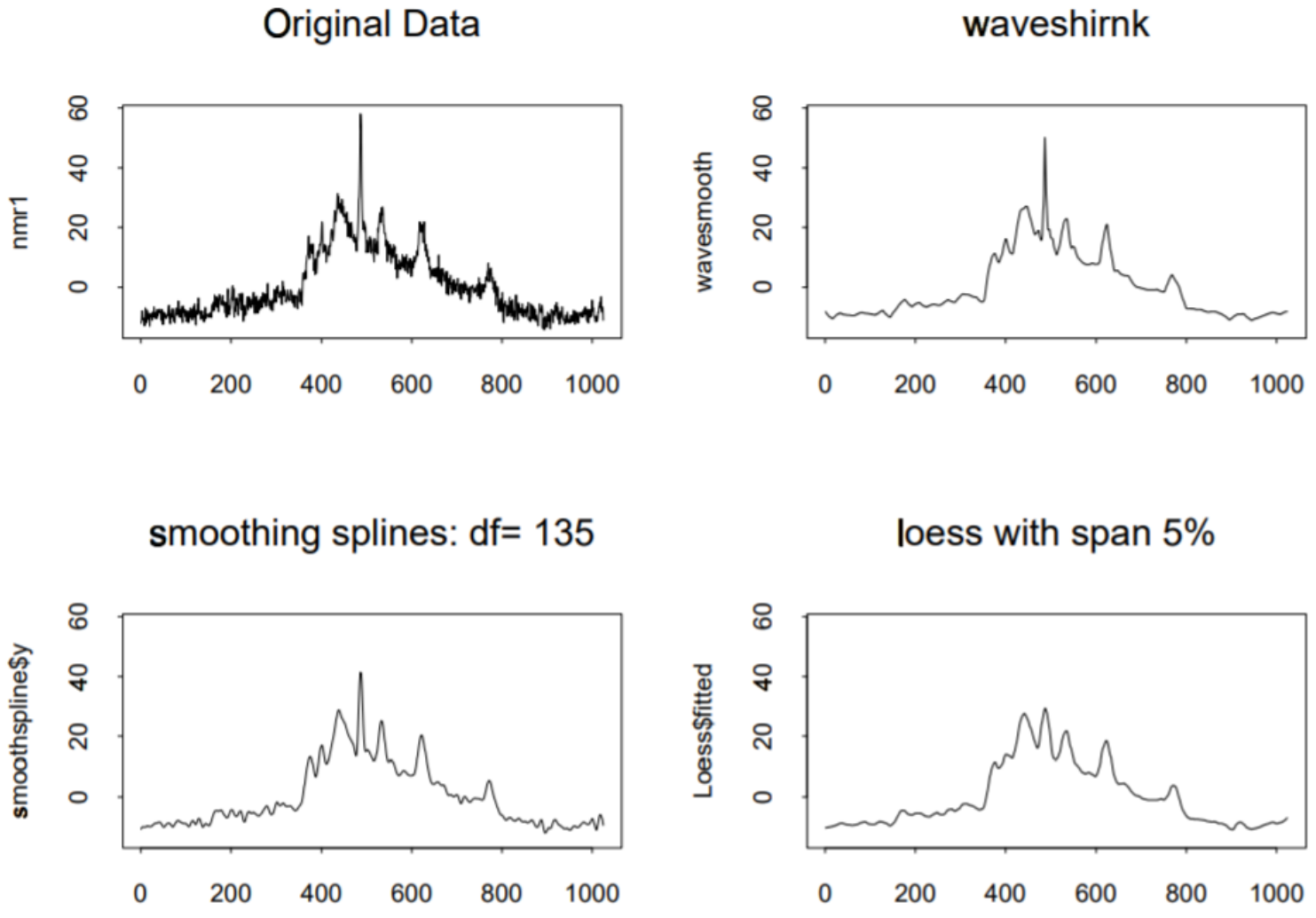
- Notice that these are step function. However, there are ways (they involve complicated math and no closed forms) to create "smoother" wavelets. The following are the columns of DWT using the Daubechies wavelets


0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120
0 20 40 60 80 100 120

The following plot shows the coefficients of the DWT by smoothness level and by location:



Using wavelet with shrinkage seems to perform better at de-noising than smoothing splines and loess as shown by the following figure.



Wavelet smoother

- We assume $d = 1$. Local adaptivity in higher dimensions is not nearly as settled as it is with smoothing splines or (especially) kernels (multivariate extensions of wavelets are possible, i.e., ridgelets and curvelets, but are complex)
- Consider basis functions, $\phi_1, \ldots, \phi_n$, evaluated over n equally spaced inputs over $[0, 1]$:

$$x_i = i/n, i = 1, \ldots, n.$$

The assumption of evenly spaced inputs is crucial for fast computations; we also typically assume with wavelets that n is a power of 2. We now form a wavelet basis matrix $W \in \mathbb{R}^{n \times n}$, defined by

$$W_{ij} = \phi_j(x_i), i, j = 1, \ldots, n.$$

- The goal, given outputs $y = (y_1, \ldots, y_n)$ over the evenly spaced input points, is to represent y as a sparse combination of the wavelet basis functions. To do so, we first perform a wavelet transform (multiply by W^T):

$$\tilde{\theta} = W^T y$$

we threshold the coefficients θ (the threshold function T_λ to be defined shortly):

$$\hat{\theta} = T_\lambda(\tilde{\theta}),$$

and then perform an inverse wavelet transform (multiply by W):

$$\hat{\mu} = W\hat{\theta}$$

- The wavelet and inverse wavelet transforms (multiplication by W^T and W) each require $O(n)$ operations, and are practically extremely fast due do clever pyramidal multiplication schemes that exploit the special structure of wavelets
- The threshold function T_λ is usually taken to be hard-thresholding, i.e.,

$$[T_\lambda^{hard}(z)]_i = z_i \cdot 1\{|z_i| \geq \lambda\}, i = 1, \ldots, n,$$

or soft-thresholding, i.e.,

$$[T_\lambda^{soft}(z)]_i = (z_i - sign(z_i)\lambda) \cdot 1\{|z_i| \geq \lambda\}, i = 1, \ldots, n,$$

These thresholding functions are both also $O(n)$, and computationally trivial, making wavelet smoothing very fast overall

- We can write the wavelet smoothing estimate in a more familiar form, following our previous discussions on basis functions and regularization. For hard-thresholding, we solve

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^n} \|y - W\theta\|_2^2 + \lambda^2 \|\theta\|_0,$$

and then the wavelet smoothing fitted values are $\hat{\mu} = W\hat{\theta}$. Here $\|\theta\|_0 = \sum_{i=1}^n 1\{\theta_i \neq 0\}$, the number of nonzero components of θ, called the "ℓ_0 norm". For soft-thresholding, we solve

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^n} \|y - W\theta\|_2^2 + 2\lambda \|\theta\|_1,$$

and then the wavelet smoothing fitted values are $\hat{\mu} = W\hat{\theta}$. Here $\|\theta\|_1 = \sum_{i=1}^n |\theta_i|$, the ℓ_1 norm.

The strengths of wavelets, the limitations of linear smoothers

- Apart from its computational efficiency, an important strength of wavelet smoothing is that it can represent a signal that has a spatially heterogeneous degree of smoothness, i.e., it can be both smooth and wiggly at different regions of the input domain. The reason that wavelet smoothing can achieve such local adaptivity is because it selects a sparse number of wavelet basis functions, by thresholding the coefficients from a basis regression
- We can make this more precise by considering convergence rates over an appropriate function class.

- In particular, we define the total variation class $F(k, C)$, for an integer $k \geq 0$ and $C > 0$, to contain all k times (weakly) differentiable functions whose kth derivative satisfies

$$TV(f^{(k)}) = \sup_{0=z_1<z_2<\cdots<z_N<z_{N+1}=1} \sum_{j=1}^{N} |f^{(k)}(z_{i+1}) - f^{(k)}(z_i)| \leq C$$

(Note that if f has $k+1$ continuous derivatives, then $TV(f^{(k)}) = \int_0^1 |f^{(k+1)}(x)|dx$.)

- Define the Sobolev class of functions $W_1(m, C)$, for an integer $m \geq 0$ and $C > 0$, to contain all m times differentiable functions $f : \mathbb{R} \to \mathbb{R}$ such that

$$\int \left(f^{(m)}(x)\right)^2 dx \leq C^2$$

(The Sobolev class $W_d(m, C)$ in d dimensions can be defined similarly, where we sum over all partial derivatives of order m.)

- Now let us define the Holder class of functions $H_d(k + \gamma, L)$, for an integer $k \geq 0, 0 < \gamma \leq 1$ and $L > 0$, to contain all k times differentiable functions $f : \mathbb{R}^d \to \mathbb{R}$ such that

$$\left| \frac{\partial^k f(x)}{\partial x_1^{\alpha_1} x_2^{\alpha_2} \cdots \partial x_d^{\alpha_d}} - \frac{\partial^k f(z)}{\partial x_1^{\alpha_1} x_2^{\alpha_2} \cdots \partial x_d^{\alpha_d}} \right| \leq L \|x - z\|_2^{\gamma}, \quad \text{for all } x, z,$$

and $\alpha_1 + \ldots + \alpha_d = k$.
Note that $H_d(1, L)$ is the space of all L -Lipschitz functions, and $H_d(k + 1, L)$ is the space of all functions whose k th-order partial derivatives are L -Lipschitz.

- Assuming $f_0 \in W_1(m, C)$ for the underlying regression function, where $C > 0$ is a constant, the smoothing spline estimator $\hat{f}$ in (9) of polynomial order $k = 2m - 1$ with tuning parameter $\lambda \asymp n^{1/(2m+1)} \asymp n^{1/(k+2)}$ satisfics

$$\left\| \hat{f} - f_0 \right\|_n^2 \lesssim n^{-2m/(2m+1)} \text{ in probability.}$$

The proof of this result uses much more fancy techniques from empirical process theory (entropy numbers) than the proofs for kernel smoothing. See Chapter 10.1 of van de Geer (2000)

- This rate is seen to be minimax optimal over $W_1(m, C)$ (c.g., Nussbaum (1985)) . Also, it is worth noting that the Sobolev $W_1(m, C)$ and Holder $H_1(m, L)$ classes are equivalent in the following sense: given $W_1(m, C)$ for a constant $C > 0$, there are $L_0, L_1 > 0$ such that

$$H_1(m, L_0) \subseteq W_1(m, C) \subseteq H_1(m, L_1)$$

The first containment is easy to show; the second is far more subtle, and is a consequence of the Sobolev embedding theorem. (The same equivalences hold for the d -dimensional versions of the Sobolev and Holder spaces.)

For the wavelet smoothing estimator, denoted by $\hat{f}^{wav}$, Donoho & Johnstone (1998) provide a seminal analysis. Assuming that $f_0 \in F(k, C)$ for a constant $C > 0$ (and further conditions on the setup), they show that (for an appropriate scaling of the smoothing parameter λ),

$$E\|\hat{f}^{wav} - f_0\|_2^2 \lesssim n^{-(2k+2)/(2k+3)},$$
$$\inf_{\hat{f}} \sup_{f_0 \in F(k,C)} E\|\hat{f} - f_0\|_2^2 \gtrsim n^{-(2k+2)/(2k+3)}$$

Thus wavelet smoothing attains the minimax optimal rate over the function class $F(k, C)$. (For a translation of this result to the notation of the current setting, see Tibshirani (2014).)

Some important questions:
(i) just how big is the function class $F(k, C)$? And
(ii) can a linear smoother also be minimax optimal over $F(k, C)$?

It is not hard to check $F(k, C) \supseteq W_1(k+1, C')$, the (univariate) Sobolev space of order $k+1$, for some other constant $C' > 0$. We know from the previously mentioned theory on Sobolev spaces that the minimax rate over $W_1(k+1, C')$ is again $n^{-(2k+2)/(2k+3)}$. This suggests that these two function spaces might actually be somewhat close in size.

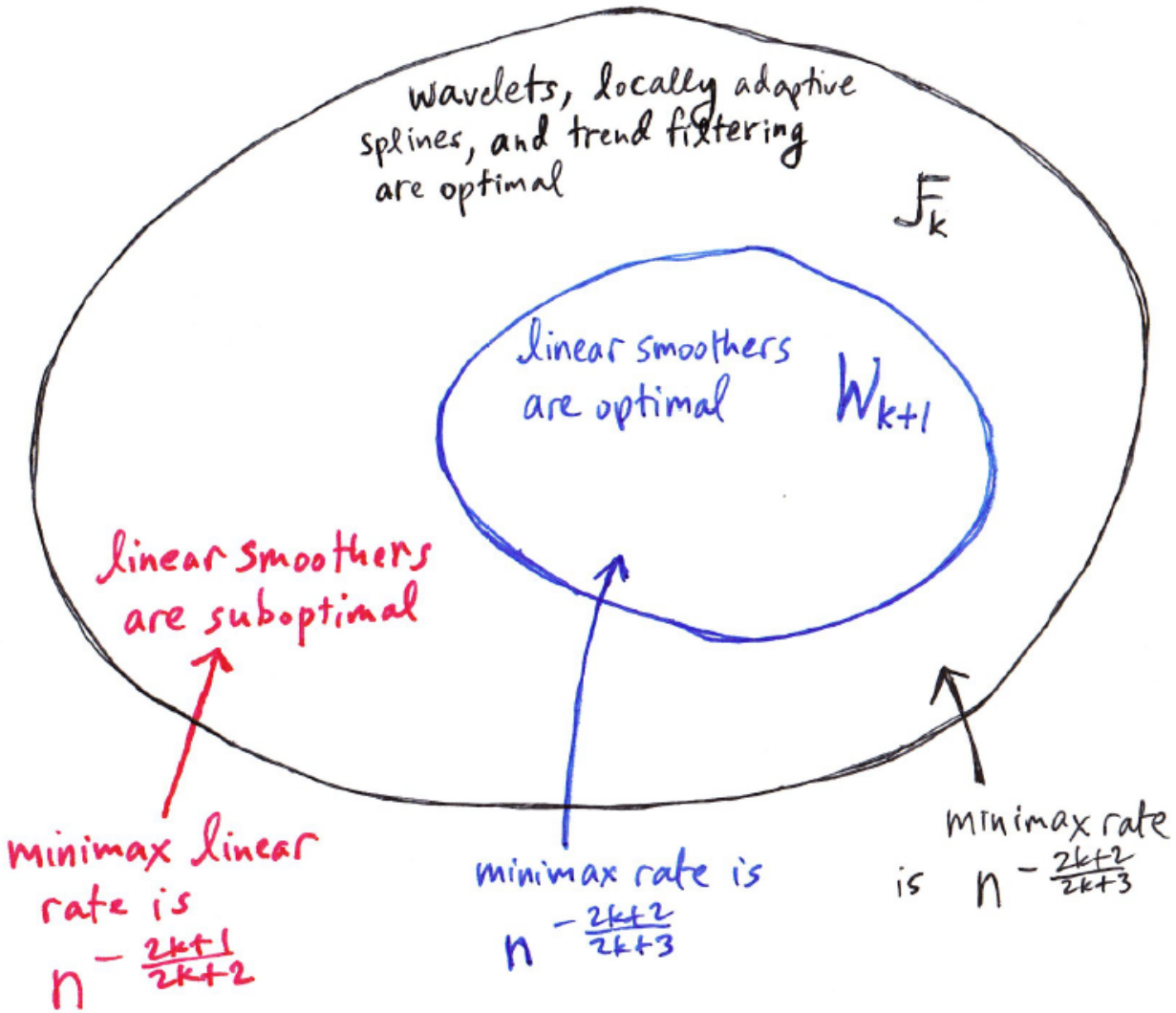
- But in fact, the overall minimax rates here are sort of misleading, and we will see from the behavior of linear smoothers that the function classes are actually quite different. Donoho & Johnstone (1998) showed that the minimax error over $F(k, C)$, restricted to linear smoothers, satisfies

$$\inf_{\hat{f} linear} \sup_{f_0 \in F(k,C)} E\|\hat{f} - f_0\|_2^2 \gtrsim n^{-(2k+1)/(2k+2)}$$

  (See again Tibshirani (2014) for a translation to the notation of the current setting.)

- Hence the answers to our questions are: (ii) linear smoothers cannot cope with the heterogeneity of functions in $F(k, C)$, and are bounded away from optimality, which means (i) we can interpret $F(k, C)$ as being much larger than $W_1(k + 1, C')$, because linear smoothers can be optimal over the latter class but not over the former. See the following figure for a diagram.

A diagram of the minimax rates over $F(k, C)$ (denoted $\mathcal{F}_k$ in the picture) and $\mathcal{W}_1(k+1, C)$ (denoted $\mathcal{W}_{k+1}$ in the picture)



- Practically, the differences between wavelets and linear smoothers in problems with spatially heterogeneous smoothness can be striking as well. However, you should keep in mind that wavelets are not perfect: a shortcoming is that they require a highly restrictive setup: recall that they require evenly spaced inputs, and n to be power of 2, and there are often further assumptions made about the behavior of the fitted function at the boundaries of the input domain
- Also, though you might say they marked the beginning of the story, wavelets are not the end of the story when it comes to local adaptivity. The natural thing to do, it might seem, is to make (say) kernel smoothing or smoothing splines more locally adaptive by allowing for a local bandwidth parameter or a local penalty parameter. People have tried this, but it is both difficult theoretically and practically to get right. A cleaner approach is to redesign the kind of penalization used in constructing smoothing splines directly, which we discuss next

Locally adaptive regression splines

- Locally adaptive regression splines (Mammen & van de Geer 1997), as their name suggests, can be viewed as variant of smoothing splines that exhibit better local adaptivity. For a given integer order $k > 0$, the estimate is defined as

$$\hat{f} = \arg\min_f \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda TV(f^{(k)}). \tag{4}$$

  The minimization domain is infinite-dimensional, the space of all functions for which the criterion is finite

- Another remarkable variational result, similar to that for smoothing splines, shows that (4) has a kth order spline as a solution (Mammen & van de Geer 1997). This almost turns the minimization into a finite-dimensional one, but there is one catch: the knots of this kth-order spline are generally not known, i.e., they need not coincide with the inputs $x, \ldots, x_n$. (When $k = 0, 1$, they do, but in general, they do not)

- To deal with this issue, we can redefine the locally adaptive regression spline estimator to be

$$\hat{f} = \arg\min_{f \in \mathcal{G}_k} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda TV(f^{(k)}). \tag{5}$$

  i.e., we restrict the domain of minimization to be $\mathcal{G}_k$, the space of kth-order spline functions with knots in T_k, where T_k is a subset of $x_1, \ldots, x_n$ of size $n - k - 1$. The precise definition of T_k is not important; it is just given by trimming away $k + 1$ boundary points from the inputs

- As we already know, the space $\mathcal{G}_k$ of kth-order splines with knots in T_k has dimension $|T_k| + k + 1 = n$. Therefore we can choose a basis $g_1, \ldots, g_n$ for the functions in $\mathcal{G}_k$, and the problem in (5) becomes one of finding the coefficients in this basis expansion

$$\hat{\beta} = \arg\min_{\beta} \sum_{i=1}^{n} \Big(y_i - \sum_{j=1}^{n} \beta_j g_j(x_i) \Big)^2 + \lambda TV\Big\{ \Big(\sum_{j=1}^{n} \beta_j g_j(x_i) \Big)^{(k)} \Big\}. \tag{6}$$

  and then we have $\hat{f}(x) = \sum_{j=1}^{n} \hat{\beta}_j g_j(x)$.
- Now define the basis matrix $G \in \mathbb{R}^{n \times n}$ by

$$G_{ij} = g_j(x_i), i = 1, \ldots, n.$$

- Suppose we choose $g_1, \ldots, g_n$ to be the truncated power basis. Denoting $T_k = \{t_1, \ldots, t_{n-k-1}\}$, we compute

$$\Big(\sum_{j=1}^{n} \beta_j g_j(x_i)\Big)^{(k)} = k! + k! \sum_{j=k+2}^{n} \beta_j 1\{x \geq t_{j-k+1}\},$$

and so

$$TV\Big\{\Big(\sum_{j=1}^{n} \beta_j g_j(x_i)\Big)^{(k)}\Big\} = k! \sum_{j=k+2}^{n} |\beta_j|.$$

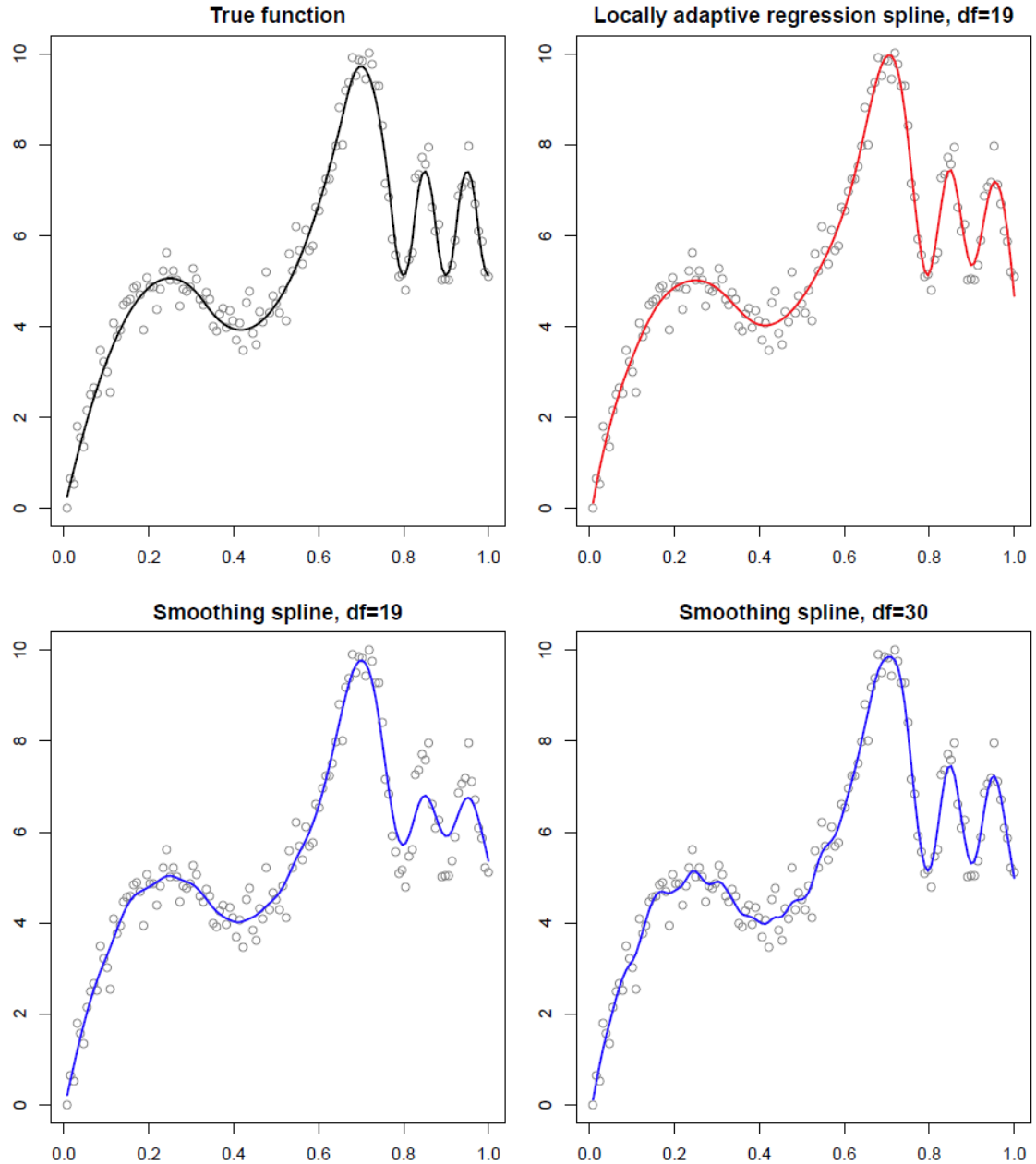
Hence the locally adaptive regression spline problem (6) can be expressed as

$$\hat{\beta} = \arg\min_{\beta \in \mathbb{R}^n} \|y - G\beta\|_2^2 + \lambda k! \sum_{i=k+2}^{n} |\beta_i|. \tag{7}$$

This is a lasso regression problem on the truncated power basis matrix G, with the first $k+1$ coefficients (those corresponding to the pure polynomial functions, in the basis expansion) left unpenalized

- This reveals a key difference between the locally adaptive regression splines (7) (originally, problem (5)) and the smoothing splines (3) (originally, problem (6) in Lec 15). In the first problem, the total variation penalty is translated into an ℓ_1 penalty on the coefficients of the truncated power basis, and hence this acts a knot selector for the estimated function. That is, at the solution in (7), the estimated spline has knots at a subset of T_k (at a subset of the input points $x_1, \ldots, x_n$), with fewer knots when λ is larger. In contrast, recall, at the smoothing spline solution in (3), the estimated function has knots at each of the inputs $x_1, \ldots, x_n$. This is a major difference between the ℓ_1 and ℓ_2 penalties

- From a computational perspective, the locally adaptive regression spline problem in (7) is actually a lot harder than the smoothing spline problem in (3). Recall that the latter reduces to solving a single banded linear system, which takes $O(n)$ operations. On the other hand, fitting locally adaptive regression splines in (7) requires solving a lasso problem with a dense $n \times n$ regression matrix G; this takes something like $O(n^3)$ operations. So when $n = 10,000$, there is a big difference between the two
- There is a tradeoff here, as with extra computation comes much improved local adaptivity of the fits. See the following figure for an example.


True function
Locally adaptive regression spline, df=19
Smoothing spline, df=19
Smoothing spline, df=30
0
2
4
6
8
10
0.0
0.2
0.4
0.6
0.8
1.0

- The top left plot shows a simulated true regression function, which has inhomogeneous smoothness: smoother towards the left part of the domain, wigglier towards the right.
- The top right plot shows the locally adaptive regression spline estimate with 19 degrees of freedom; notice that it picks up the right level of smoothness throughout. The bottom left plot shows the smoothing spline estimate with the same degrees of freedom; it picks up the right level of smoothness on the left, but is undersmoothed on the right. The bottom right panel shows the smoothing spline estimate with 33 degrees of freedom; now it is appropriately wiggly on the right, but oversmoothed on the left.
- Smoothing splines cannot simultaneously represent different levels of smoothness at different regions in the domain; the same is true of any linear smoother

- Theoretically, when $f_0 \in F(k, C)$ for a constant $C > 0$, Mammen & van de Geer (1997) show the locally adaptive regression spline estimator, denoted $\hat{f}^{lrs}$, with $\lambda \asymp n^{1/(2k+3)}$, satisfies

$$\|\hat{f}^{lrs} - f_0\|_n^2 \lesssim n^{-(2k+2)/(2k+3)} \quad \textit{in probability}$$

  so (like wavelets) it achieves the minimax optimal rate $n^{-(2k+2)/(2k+3)}$ over $F(k, C)$. In this regard,as we discussed previously, they actually have a big advantage over any linear smoother (not just smoothing splines)


Donoho, D. L. & Johnstone, I. (1998), Minimax estimation via wavelet shrinkage, Annals of Statistics 26(8), 879-921.


Tibshirani, R. J. (2014), Adaptive piecewise polynomial estimation via trend filtering , Annals of Statistics 42(1), 285-323.


Mallat, S. (2008), A wavelet tour of signal processing, Academic Press. Third edition


Mammen, E. & van de Geer, S. (1997), Locally apadtive regression splines, Annals of Statistics 25(1), 387-413.