Nonparametric Regression

Statistical Machine Learning, Spring 2015

Ryan Tibshirani (with Larry Wasserman)

1 Introduction, and k-nearest-neighbors

1.1 Basic setup, random inputs

- Given a random pair $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$, recall that the function

$$f_0(x) = \mathbb{E}(Y|X = x)$$

  is called the regression function (of Y on X). The basic goal in nonparametric regression is to construct an estimate $\hat{f}$ of f_0, from i.i.d. samples $(x_1, y_1), \ldots (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ that have the same joint distribution as (X, Y). We often call X the input, predictor, feature, etc., and Y the output, outcome, response, etc. Importantly, in nonparametric regression we do not assume a certain parametric form for f_0

- Note for i.i.d. samples $(x_1, y_1), \ldots (x_n, y_n)$, we can always write

$$y_i = f_0(x_i) + \epsilon_i, \quad i = 1, \ldots n,$$

  where $\epsilon_1, \ldots \epsilon_n$ are i.i.d. random errors, with mean zero. Therefore we can think about the sampling distribution as follows: $(x_1, \epsilon_1), \ldots (x_n, \epsilon_n)$ are i.i.d. draws from some common joint distribution, where $\mathbb{E}(\epsilon_i) = 0$, and then $y_1, \ldots y_n$ are generated from the above model

- In addition, we will assume that each ϵ_i is independent of x_i. As discussed before, this is actually quite a strong assumption, and you should think about it skeptically. We make this assumption for the sake of simplicity, and it should be noted that a good portion of theory that we cover (or at least, similar theory) also holds without the assumption of independence between the errors and the inputs

1.2 Basic setup, fixed inputs

- Another common setup in nonparametric regression is to directly assume a model

$$y_i = f_0(x_i) + \epsilon_i, \quad i = 1, \ldots n,$$

  where now $x_1, \ldots x_n$ are *fixed* inputs, and $\epsilon_1, \ldots \epsilon$ are still i.i.d. random errors with $\mathbb{E}(\epsilon_i) = 0$

- For arbitrary points $x_1, \ldots x_n$, this is really just the same as starting with the random input model, and conditioning on the particular values of $x_1, \ldots x_n$. (But note: once we condition on the inputs, the errors are only i.i.d. because we have assumed that the errors and inputs were independent in the first place)

- Generally speaking, nonparametric regression estimators are not defined with the random or fixed setups specifically in mind, i.e., there is no real distinction made here. A caveat: some estimators (like wavelets) do in fact assume evenly spaced fixed inputs, as in

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

for evenly spaced inputs in the univariate case

- It is also important to mention that the theory is not completely the same between the random and fixed worlds, and some theoretical statements are sharper when assuming fixed input points, especially evenly spaced input points

1.3 What we cover here

- We won't be very precise about which setup we assume—random or fixed inputs—because, as mentioned before, it doesn't really matter when defining nonparametric regression estimators and discussing basic properties

- When it comes to theory, we will mix and match. The goal is to give you a flavor of some interesting results over a variety of methods, and under different assumptions. A few topics we will cover into more depth than others, but overall, this will not be a complete coverage. There are some excellent texts out there that you can consult for more details, proofs, etc., and some are listed below. There are surely others too, and you can always come ask one of us if you are looking for something in particular

  - Kernel smoothing, local polynomials
    - Tsybakov (2009)
  - Regression splines, smoothing splines
    - de Boor (1978)
    - Green & Silverman (1994)
    - Wahba (1990)
  - Reproducing kernel Hilbert spaces
    - Scholkopf & Smola (2002)
    - Wahba (1990)
  - Wavelets
    - Johnstone (2011)
    - Mallat (2008)
  - General references, more theoretical
    - Gyorfi, Kohler, Krzyzak & Walk (2002)
    - Wasserman (2006)
  - General references, more applied
    - Simonoff (1996)
    - Hastie, Tibshirani & Friedman (2009)

- Throughout, our discussion will mostly center around the univariate case, $d = 1$. Mutlivariate extensions, $d > 1$, will be mentioned, but as we will see, these typically become statistically and computationally inefficient as d grows. A fix is to use structured regression models in high dimensions, which use the univariate (or low-dimensional) estimators as building blocks, and we will study these near the end

- Finally, a lot the discussed methods can be extended from nonparametric regression to nonparametric classification, as we'll see at the end

1.4 k-nearest-neighbors regression

- Here's a basic method to start us off: *k-nearest-neighbors* regression. We fix an integer $k \geq 1$ and define

$$\hat{f}(x) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(x)} y_i, \tag{1}$$

  where $\mathcal{N}_k(x)$ contains the indices of the k closest points of $x_1, \ldots x_n$ to x

- This is not at all a bad estimator, and you will find it used in lots of applications, in many cases probably because of its simplicity. By varying the number of neighbors k, we can achieve a wide range of flexibility in the estimated function $\hat{f}$, with small k corresponding to a more flexible fit, and large k less flexible

- But it does have its limitations, an apparent one being that the fitted function $\hat{f}$ essentially always looks jagged, especially for small or moderate k. Why is this? It helps to write

$$\hat{f}(x) = \sum_{i=1}^{n} w_i(x) y_i, \tag{2}$$

  where the weights $w_i(x)$, $i = 1, \ldots n$ are defined as

$$w_i(x) = \begin{cases} 1/k & \text{if } x_i \text{ is one of the } k \text{ nearest points to } x \\ 0 & \text{else.} \end{cases}$$

  Note that $w_i(x)$ is discontinuous as a function of x, and therefore so if $\hat{f}(x)$

- The representation (2) also reveals that the k-nearest-neighbors estimate is in a class of estimates we call *linear smoothers*, i.e., writing $y = (y_1, \ldots y_n) \in \mathbb{R}^n$, the vector of fitted values

$$\hat{y} = \big(\hat{f}(x_1), \ldots \hat{f}(x_n)\big) \in \mathbb{R}^n$$

  can simply be expressed as $\hat{y} = Sy$. (To be clear, this means that for fixed inputs $x_1, \ldots x_n$, the vector of fitted values $\hat{y}$ is a linear function of y; it does not mean that $\hat{f}(x)$ need behave linearly as a function of x!) This class is quite large, and contains many popular estimators, as we'll see in the coming sections

- The k-nearest-neighbors estimator is consistent, under the random input model, provided we take $k = k_n$ such that $k_n \to \infty$ and $k_n/n \to 0$; e.g., $k = \sqrt{n}$ will do. See Section 6.2 of Gyorfi et al. (2002)

- Furthermore, assuming that the underlying regression function f_0 is Lipschitz continuous, the k-nearest-neighbors estimate with $k = \Theta(n^{2/(2+d)})$ satisfies

$$\mathbb{E}\big(\hat{f}(X) - f(X)\big)^2 = O(n^{-2/(2+d)}).$$

  See Section 6.3 of Gyorfi et al. (2002)

- Note that the above rate exhibits a very poor dependence on the dimension d. For a small ϵ, think about how large we need to make n to ensure that $(1/n)^{2/(2+d)} \leq \epsilon$; rearranged, this says $n \geq (1/\epsilon)^{(2+d)/2}$. That is, as we increase d, we require exponentially more samples n to achieve an MSE bound of ϵ

- In fact, this phenomenon is not specific to k-nearest-neighbors. It is a reflection of the *curse of dimensionality*, the principle that estimation becomes exponentially harder as the number of dimensions increases. To circumvent this, we usually turn to structured regression models in high dimensions, covered near the end

2 Kernel smoothing, local polynomials

2.1 Kernel smoothing

- Assume for now that $d = 1$, for simplicity. As in kernel density estimation, *kernel regression* or *kernel smoothing* begins with a kernel function $K : \mathbb{R} \to \mathbb{R}$, satisfying

$$\int K(x)\,dx = 1, \quad \int xK(x)\,dx = 0, \quad 0 < \int x^2 K(x)\,dx < \infty.$$

  Two common examples are the Gaussian kernel:

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2),$$

  and the Epanechnikov kernel:

$$K(x) = \begin{cases} 3/4(1-x^2) & \text{if } |x| \le 1 \\ 0 & \text{else} \end{cases}$$

- Given a bandwidth $h > 0$, the (Nadaraya-Watson) kernel regression estimate is defined as

$$\hat{f}(x) = \frac{\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) y_i}{\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)}. \tag{3}$$

  Hence kernel smoothing is also a linear smoother (2), with choice of weights $w_i(x) = K((x - x_i)/h) / \sum_{j=1}^n K((x - x_j)/h)$

- In comparison to the k-nearest-neighbors estimator in (1), which can be thought of as a raw (discontinuous) moving average of nearby outputs, the kernel estimator in (3) is a smooth moving average of outputs. See Figure 1

- A shortcoming: the kernel regression suffers from poor bias at the boundaries of the domain of the inputs $x_1, \ldots x_n$. This happens because of the asymmetry of the kernel weights in such regions. See Figure 2

2.2 Local polynomials

- We can alleviate this boundary bias issue by moving from a local constant fit to a local linear fit, or a local higher-order fit

- To build intuition, another way to view the kernel estimator in (3) is the following: at each input x, it employs the estimate $\hat{f}(x) = \hat{\theta}$, where $\hat{\theta}$ is the minimizer of

$$\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)(y_i - \theta)^2.$$

  Instead we could consider forming the local estimate $\hat{f}(x) = \hat{\alpha} + \hat{\beta}x$, where $\hat{\alpha}, \hat{\beta}$ minimize

$$\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)(y_i - \alpha - \beta x_i)^2.$$

  This is called *local linear regression*

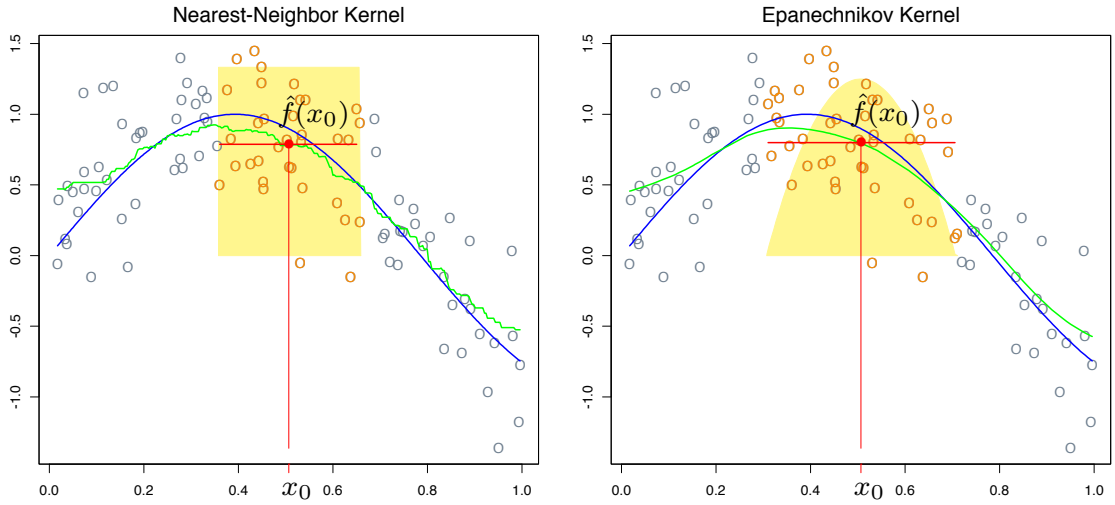

Figure 1: *Comparing k-nearest-neighbor and Epanechnikov kernels. From Chapter 6 of Hastie et al. (2009)*

- We can rewrite the local linear regression estimate $\hat{f}(x)$. This is just given by a weighted least squares fit, so
$$\hat{f}(x) = b(x)^T (B^T \Omega B)^{-1} B^T \Omega y,$$
where $b(x) = (1, x_i)$, B in an $n \times 2$ matrix with ith row $b(x_i) = (1, x)$, and Ω is an $n \times n$ diagonal matrix with ith diagonal element $K((x - x_i)/h)$. We can express this more concisely as $\hat{f}(x) = w(x)^T y$ for $w(x) = \Omega B(B^T \Omega B)^{-1} b(x)$, and so local linear regression is a linear smoother too
- The vector of fitted values $\hat{y} = (\hat{f}(x_1), \ldots \hat{f}(x_n))$ can be expressed as
$$\hat{y} = \begin{pmatrix} w_1(x)^T y \\ \vdots \\ w_n(x)^T y \end{pmatrix} = B(B^T \Omega B)^{-1} B^T \Omega y,$$
which should look familiar to you from weighted least squares
- Now we'll sketch how the local linear fit reduces the bias. Compute at a fixed point x,
$$\mathbb{E}[\hat{f}(x)] = \sum_{i=1}^n w_i(x) f_0(x_i).$$
Using a Taylor expansion about x,
$$\mathbb{E}[\hat{f}(x)] = f_0(x) \sum_{i=1}^n w_i(x) + f_0'(x) \sum_{i=1}^n (x_i - x) w_i(x) + R,$$
where the remainder term R contains quadratic and higher-order terms, and under regularity conditions, is small. One can check that in fact for the local linear regression estimator $\hat{f}$,
$$\sum_{i=1}^n w_i(x) = 1 \quad \text{and} \quad \sum_{i=1}^n (x_i - x) w_i(x) = 0,$$
and so $\mathbb{E}[\hat{f}(x)] = f_0(x) + R$, which means that $\hat{f}$ is unbiased to first order

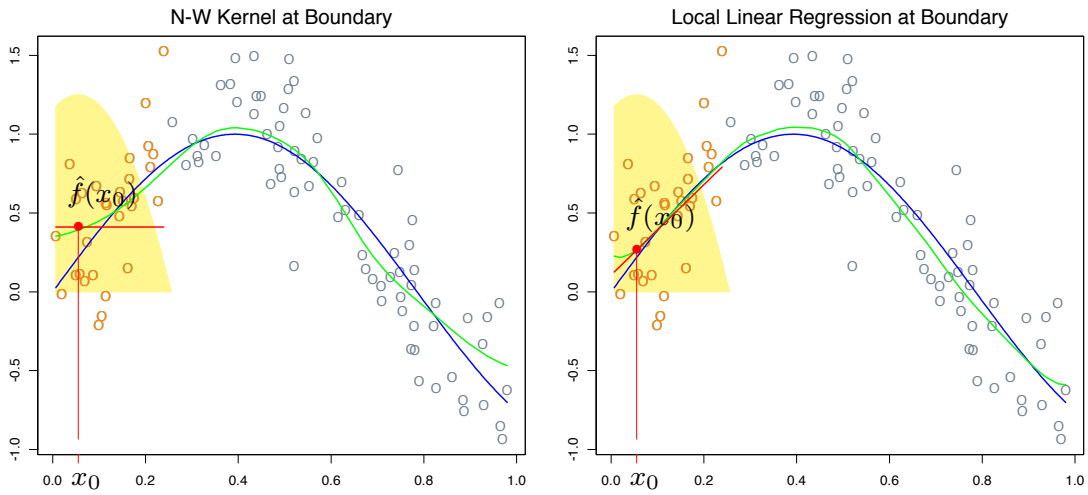

Figure 2: *Comparing (Nadaraya-Watson) kernel smoothing to local linear regression; the former is biased at the boundary, the latter is unbiased (to first order). From Chapter 6 of Hastie et al. (2009)*

- We don't have to stop with a local linear fit, we can more generally fit $\hat{f}(x) = \hat{\beta}_0 + \sum_{j=1}^p \hat{\beta}_j x^j$, where $\hat{\beta}_0, \ldots \hat{\beta}_p$ minimize

$$\sum_{i=1}^n K\left(\frac{x - x_i}{h}\right)\left(y_i - \beta_0 - \sum_{j=1}^p \beta_j x_i^j\right)^2.$$

  This is called *local polynomial regression*

- Again we can express

$$\hat{f}(x) = b(x)(B^T \Omega B)^{-1} B^T \Omega y = w(x)^T y,$$

  where $b(x) = (1, x, \ldots x^p)$, B is an $n \times (p+1)$ matrix with ith row $b(x_i) = (1, x_i, \ldots x_i^p)$, and Ω is as before. Hence again, local polynomial regression is a linear smoother

- The basic message is that a higher degree in the local polynomial fit can help reduce the bias, even in the interior of the domain of inputs, but (not surprisingly) this comes at the expense of an increase in variance

2.3 Error bounds

- Consider the Holder class of functions $\Sigma(k, L)$, for an integer $k \geq 1$ and constant $L > 0$, which contains the set of all $k-1$ times differentiable functions $f : \mathbb{R} \to \mathbb{R}$ whose $(k-1)$st derivative is Lipschitz continuous:

$$|f^{(k-1)}(x) - f^{(k-1)}(z)| \leq L|x - z|, \quad \text{for any } x, z.$$

  (Actually, the function class $\Sigma(\gamma, L)$ can also be defined for non-integral γ, but this is not really important.)

- Consider the fixed inputs model, with appropriate conditions on the spacings of the inputs $x_1, \ldots x_n \in [0, 1]$ (and the error distribution and choice of kernel K). Assuming that the true

function f_0 is in the Holder class $\Sigma(k, L)$, a Taylor expansion shows that the local polynomial estimator $\hat{f}$ of order $k-1$ has bounded bias and variance,

$$\big|\mathbb{E}[\hat{f}(x)] - f_0(x)\big| \leq C_1 h^k, \quad \mathrm{Var}\big(\hat{f}(x)\big) \leq \frac{C_2}{nh},$$

over all $x \in [0, 1]$. Hence taking $h = \Theta(n^{-1/(2k+1)})$, the L_2 error of $\hat{f}$ has convergence rate

$$\mathbb{E}\|\hat{f} - f_0\|_{L_2}^2 = O(n^{-2k/(2k+1)}).$$

(Here $\|f\|_{L_2}^2 = \int_0^1 f(x)^2 dx$.) See Section 1.6.1 of Tsybakov (2009)

- How fast is this convergence rate? In fact, we can't broadly do better over the function class $\Sigma(k, L)$. If we assume fixed inputs evenly spaced over $[0, 1]$, $x_i = i/n$ for $i = 1, \ldots n$, and a mild condition on the error distribution, the minimax risk is

$$\min_{\hat{f}} \max_{f_0 \in \Sigma(k,L)} \mathbb{E}\|\hat{f} - f_0\|_{L_2}^2 = \Omega(n^{-2k/(2k+1)}),$$

where the minimum above is over all estimators $\hat{f}$. See Section 2.6.2 of Tsybakov (2009)

- Is this the end of the story? Not at all. We'll see that by widening our the scope of functions that we consider, local polynomials are far from optimal

- As an aside, why did we study the Holder class $\Sigma(k, L)$ anyway? Because it was pretty natural to assume that $f_0^{(k)}$ is Lipschitz after performing a kth order Taylor expansion to compute the bias and variance

2.4 Multivariate kernels

- Kernel smoothing and local polynomials extend very naturally to higher dimensions, using, e.g., $K(\|x - x_i\|_2/h)$ as the kernel weight between points $x, x_i \in \mathbb{R}^d$. The corresponds to an isotropic kernel, since it spherically invariant (depends only on the distance between x and x_i)

- Computational efficiency and statisical efficiency are both very real concerns when d grows large, or even just moderately large (say, larger than 10). Boundary effects are greatly exaggerated as d increases, since the fraction of data points near the boundary grows rapidly. Again, in high dimensions we usually rely on something less flexible, like an additive model, which we cover near the end

3 Regression splines, smoothing splines

3.1 Splines

- Regression splines and smoothing splines are motivated from a different perspective than kernels and local polynomials; in the latter case, we started off with a special kind of local averaging, and moved our way up to a higher-order local models. With regression splines and smoothing splines, we build up our estimate globally, from a set of select basis functions

- These basis functions, as you might guess, are *splines*. Let's assume that $d = 1$, for simplicity. A kth order spline is a piecewise polynomial function of degree k, that is continuous and has continuous derivatives of orders $1, \ldots k-1$, at its knot points

- Formally, a function $f : \mathbb{R} \to \mathbb{R}$ is a kth order spline with knot points at $t_1 < \ldots < t_m$, if f is a polynomial of degree k on each of the intervals $(-\infty, t_1], [t_1, t_2], \ldots [t_m, \infty)$, and $f^{(j)}$ is continuous at $t_1, \ldots t_m$, for each $j = 0, 1, \ldots k-1$

- Splines have some very special properties are have been a topic of interest among statisticians and mathematicians for a long time. See de Boor (1978) for an in-depth coverage. Roughly speaking, a spline is a lot smoother than a piecewise polynomial, and so modeling with splines can serve as a way of reducing the variance of fitted estimators. See Figure 3

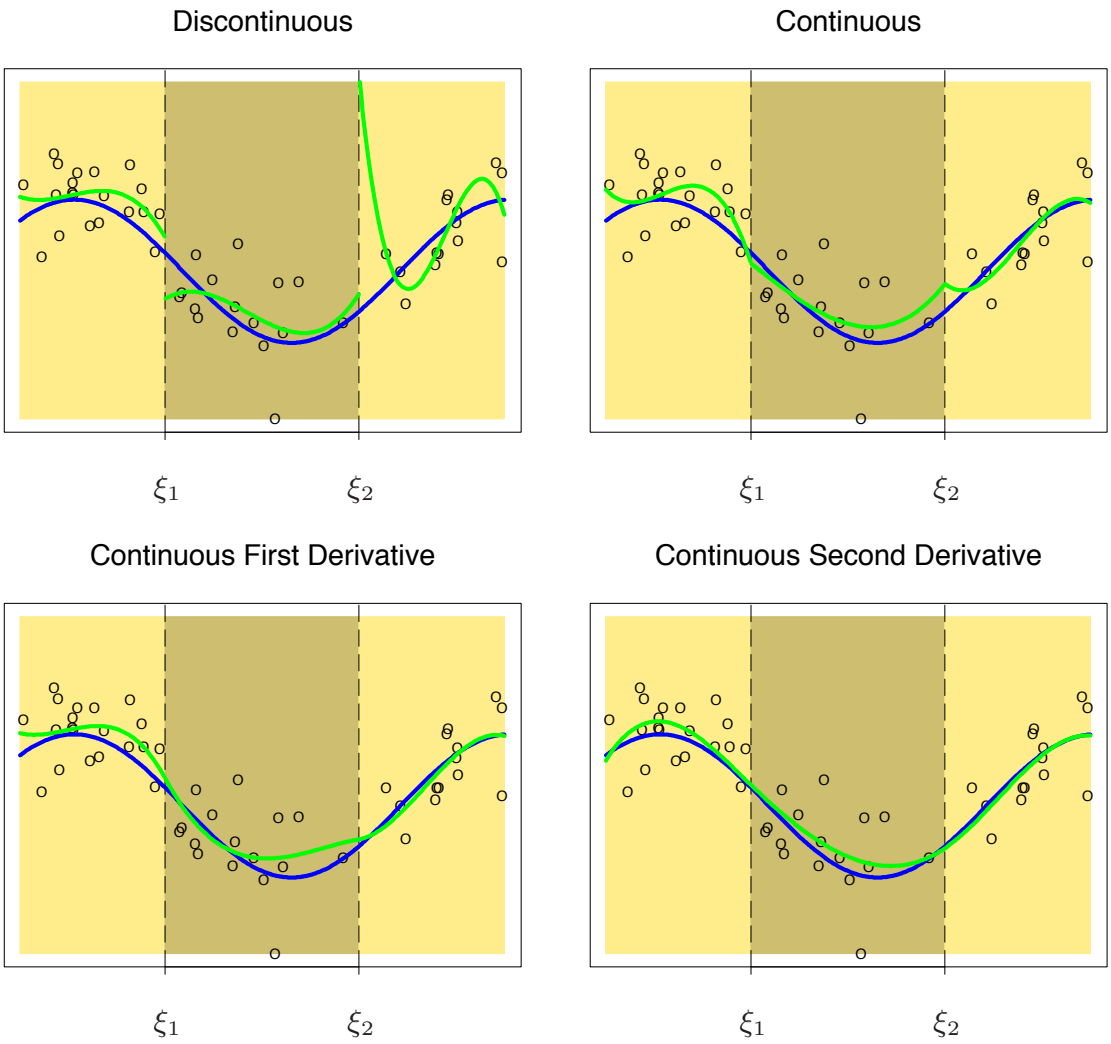

Figure 3: *Illustration of the effects of enforcing continuity at the knots, across various derivative orders, for a cubic piecewise polynomial. From Chapter 5 of Hastie et al. (2009)*

- A bit of statistical folklore: it is said that a cubic spline is so smooth, that one cannot detect the locations of its knots by eye!

- How can we parametrize the set of a splines with knots at $t_1, \dots t_m$? The most natural way is to use the *truncated power basis*, $g_1, \dots g_{m+k+1}$, defined as

$$\begin{aligned} g_1(x) = 1,\ g_2(x) = x,\ \dots\ g_{k+1}(x) = x^k, \\ g_{k+1+j}(x) = (x - t_j)_+^k, \quad j = 1, \dots m. \end{aligned} \tag{4}$$

(Here x_+ denotes the positive part of x, i.e., $x_+ = \max\{x, 0\}$.) From this we can see that the space of kth order splines with knots at $t_1, \ldots t_m$ has dimension $m + k + 1$

- While these basis functions are natural, a much better computational choice, both for speed and numerical accuracy, is the *B-spline* basis. This was a major development in spline theory and is now pretty much the standard in software; see de Boor (1978) for details

3.2 Regression splines

- A first idea: let's perform regression on a spline basis. In other words, given inputs $x_1, \ldots x_n$ and outputs $y_1, \ldots y_n$, we consider fitting functions f that are kth order splines with knots at some chosen locations $t_1, \ldots t_m$. This means expressing f as

$$f(x) = \sum_{j=1}^{m+k+1} \beta_j g_j(x),$$

  where $\beta_1, \ldots \beta_{m+k+1}$ are coefficients and $g_1, \ldots g_{m+k+1}$, are basis functions for order k splines over the knots $t_1, \ldots t_m$ (e.g., the truncated power basis or B-spline basis)

- Letting $y = (y_1, \ldots y_n) \in \mathbb{R}^n$, and defining the basis matrix $G \in \mathbb{R}^{n \times (m+k+1)}$ by

$$G_{ij} = g_j(x_i), \quad i = 1, \ldots n, \; j = 1, \ldots m + k + 1,$$

  we can just use linear regression to determine the optimal coefficients $\hat{\beta} = (\hat{\beta}_1, \ldots \hat{\beta}_{m+k+1})$,

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^{m+k+1}}{\operatorname{argmin}} \; \|y - G\beta\|_2^2,$$

  which then leaves us with the fitted *regression spline* $\hat{f}(x) = \sum_{j=1}^{m+k+1} \hat{\beta}_j g_j(x)$

- Of course we know that $\hat{\beta} = (G^T G)^{-1} G^T y$, so the fitted values $\hat{y} = (\hat{f}(x_1), \ldots \hat{f}(x_n))$ are

$$\hat{y} = G(G^T G)^{-1} G^T y,$$

  and regression splines are linear smoothers

- This is a classic method, and can work well provided we choose good knots $t_1, \ldots t_m$; but in general choosing knots is a tricky business. There is a large literature on knot selection for regression splines via greedy methods like recursive partitioning

3.3 Natural splines

- A problem with regression splines is that the estimates tend to display erractic behavior, i.e., they have high variance, at the boundaries of the input domain. (This is the opposite problem to that with kernel smoothing, which had poor bias at the boundaries.) This only gets worse as the polynomial order k gets larger

- A way to remedy this problem is to force the piecewise polynomial function to have a lower degree to the left of the leftmost knot, and to the right of the rightmost knot—this is exactly what *natural splines* do. A natural spline of order k, with knots at $t_1 < \ldots < t_m$, is a piecewise polynomial function f such that

  - f is a polynomial of degree k on each of $[t_1, t_2], \ldots [t_{m-1}, t_m]$,
  - f is a polynomial of degree $(k-1)/2$ on $(-\infty, t_1]$ and $[t_m, \infty)$,

– f is continuous and has continuous derivatives of orders $1, \ldots k-1$ at its knots $t_1, \ldots t_m$.

It is implicit here that natural splines are only defined for odd orders k

- What is the dimension of the span of kth order natural splines with knots at $t_1, \ldots t_m$? Recall for splines, this was $m + k + 1$ (the number of truncated power basis functions). For natural splines, we can compute this dimension by counting:

$$\underbrace{(k+1) \cdot (m-1)}_{a} + \underbrace{\Big(\frac{(k-1)}{2} + 1\Big) \cdot 2}_{b} - \underbrace{k \cdot m}_{c} \;=\; m.$$

Above, a is the number of free parameters in the interior intervals $[t_1, t_2], \ldots [t_{m-1}, t_m]$, b is the number of free parameters in the exterior intervals $(-\infty, t_1], [t_m, \infty)$, and c is the number of constraints at the knots $t_1, \ldots t_m$. The fact that the total dimension is m is amazing; this is independent of k!

- Note that there is a variant of the truncated power basis for natural splines, and a variant of the B-spline basis for natural splines. Again, B-splines are the preferred parametrization for computational speed and stability

- Natural splines of cubic order is the most common special case: these are smooth piecewise cubic functions, that are simply linear beyond the leftmost and rightmost knots

3.4 Smoothing splines

- Smoothing splines are an interesting creature: at the end of the day, these estimators perform a regularized regression over the natural spline basis, placing knots at all inputs $x_1, \ldots x_n$. Smoothing splines circumvent the problem of knot selection (as they just use the inputs as knots), and simultaneously, they control for overfitting by shrinking the coefficients of the estimated function (in its basis expansion)

- What makes them even more interesting is that they can be alternatively motivated directly from a functional minimization perspective. With inputs $x_1, \ldots x_n$ contained in an interval $[a, b]$, the *smoothing spline* estimate $\hat{f}$ of a given order k is defined as

$$\hat{f} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \int_a^b \big(f^{((k+1)/2)}(x)\big)^2 \, dx. \tag{5}$$

This is an infinite-dimensional optimization problem over all functions f for the which the criterion is finite. This criterion trades off the least squares error of f over the observed pairs (x_i, y_i), $i = 1, \ldots n$, with a penalty term that is large when the $((k+1)/2)$nd derivative of f is wiggly. The tuning parameter $\lambda \geq 0$ governs the strength of each term in the minimization

- By far the most commonly considered case is $k = 3$, i.e., cubic smoothing splines, which are defined as

$$\hat{f} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \int_a^b \big(f''(x)\big)^2 \, dx \tag{6}$$

- Remarkably, it so happens that the minimizer in the general smoothing spline problem (5) is unique, and is a natural kth order spline with knots at the input points $x_1, \ldots x_n$! Here we follow a proof for the cubic case, $k = 3$, from Green & Silverman (1994) (see also Exercise 5.7 in Hastie et al. (2009))

The key result can be stated as follows: if $\tilde{f}$ is any twice differentiable function on $[a, b]$, and $x_1, \ldots x_n \in [a, b]$, then there exists a natural cubic spline f with knots at $x_1, \ldots x_n$ such that $f(x_i) = \tilde{f}(x_i)$, $i = 1, \ldots n$ and

$$\int_a^b f''(x)^2 \, dx \leq \int_a^b \tilde{f}''(x)^2 \, dx.$$

Note that this would in fact prove that we can restrict our attention in (6) to natural splines with knots at $x_1, \ldots x_n$

Proof: the natural spline basis with knots at $x_1, \ldots x_n$ is n-dimensional, so given any n points $z_i = \tilde{f}(x_i)$, $i = 1, \ldots n$, we can always find a natural spline f with knots at $x_1, \ldots x_n$ that satisfies $f(x_i) = z_i$, $i = 1, \ldots n$. Now define

$$h(x) = \tilde{f}(x) - f(x).$$

Consider

$$\begin{aligned}
\int_a^b f''(x)h''(x) \, dx &= f''(x)h'(x)\Big|_a^b - \int_a^b f'''(x)h'(x) \; dx \\
&= -\int_{x_1}^{x_n} f'''(x)h'(x) \; dx \\
&= -\sum_{j=1}^{n-1} f'''(x)h(x)\Big|_{x_j}^{x_{j+1}} + \int_{x_1}^{x_n} f^{(4)}(x)h'(x) \; dx \\
&= -\sum_{j=1}^{n-1} f'''(x_j^+)\big(h(x_{j+1}) - h(x_j)\big),
\end{aligned}$$

where in the first line we used integration by parts; in the second we used the that $f''(a) = f''(b) = 0$, and $f'''(x) = 0$ for $x \leq x_1$ and $x \geq x_n$, as f is a natural spline; in the third we used integration by parts again; in the fourth line we used the fact that f''' is constant on any open interval (x_j, x_{j+1}), $j = 1, \ldots n-1$, and that $f^{(4)} = 0$, again because f is a natural spline. (In the above, we use $f'''(u^+)$ to denote $\lim_{x \downarrow u} f'''(x)$.) Finally, since $h(x_j) = 0$ for all $j = 1, \ldots n$, we have

$$\int_a^b f''(x)h''(x) \, dx = 0.$$

From this, it follows that

$$\begin{aligned}
\int_a^b \tilde{f}''(x)^2 \, dx &= \int_a^b \big(f''(x) + h''(x)\big)^2 \, dx \\
&= \int_a^b f''(x)^2 \, dx + \int_a^b h''(x)^2 \, dx + 2\int_a^b f''(x)h''(x) \, dx \\
&= \int_a^b f''(x)^2 \, dx + \int_a^b h''(x)^2 \, dx,
\end{aligned}$$

and therefore

$$\int_a^b f''(x)^2 \, dx \leq \int_a^b \tilde{f}''(x)^2 \, dx, \tag{7}$$

with equality if and only if $h''(x) = 0$ for all $x \in [a, b]$. Note that $h'' = 0$ implies that h must be linear, and since we already know that $h(x_j) = 0$ for all $j = 1, \ldots n$, this is equivalent to $h = 0$. In other words, the inequality (7) holds strictly except when $\tilde{f} = f$, so the solution in (6) is uniquely a natural spline with knots at the inputs

3.5 Finite-dimensional form

- The key result presented above tells us that we can choose a basis $\eta_1, \dots \eta_n$ for the set of kth order natural splines with knots over $x_1, \dots x_n$, and reparametrize the problem (5) as

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \sum_{i=1}^n \Big(y_i - \sum_{j=1}^n \beta_j \eta_j(x_i) \Big)^2 + \lambda \int_a^b \Big(\sum_{j=1}^n \beta_j \eta_j^{((k+1)/2)}(x) \Big)^2 dx. \tag{8}$$

  This is already a finite-dimensional problem, and once we solve for the coefficients $\hat{\beta} \in \mathbb{R}^n$, we know that the smoothing spline estimate is simply $\hat{f}(x) = \sum_{j=1}^n \hat{\beta}_j \eta_j(x)$

- Defining the basis matrix and penalty matrices $N, \Omega \in \mathbb{R}^{n \times n}$ by

$$N_{ij} = \eta_j(x_i) \quad \text{and} \quad \Omega_{ij} = \int_0^1 \eta_i^{((k+1)/2)}(t) \eta_j^{((k+1)/2)}(t)\, dt \quad \text{for } i, j = 1, \dots n,$$

  the problem in (8) can be written more succintly as

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - N\beta\|_2^2 + \lambda \beta \Omega \beta, \tag{9}$$

  which shows the smoothing spline problem to be a type of generalized ridge regression problem. In fact, the solution in (9) has the explicit form

$$\hat{\beta} = (N^T N + \lambda \Omega)^{-1} N^T y,$$

  and therefore the fitted values $\hat{y} = (\hat{f}(x_1), \dots \hat{f}(x_n))$ are

$$\hat{y} = N(N^T N + \lambda \Omega)^{-1} N^T y. \tag{10}$$

  Therefore, once again, smoothing splines are a kind of linear smoother

- A special property of smoothing splines: the fitted values in (10) can be computed in $O(n)$ operations. This is achieved by forming N from the B-spline basis (for natural splines), and in this case the matrix $N^T N + \Omega I$ ends up being banded (with a bandwidth that only depends on the polynomial order k). In practice, smoothing spline computations are extremely fast

3.6 Reinsch form

- It is informative to rewrite the fitted values in (10) is what is called Reinsch form,

$$\begin{aligned} \hat{y} &= N(N^T N + \lambda \Omega)^{-1} N^T y \\ &= N\Big(N^T \big(I + \lambda (N^T)^{-1} \Omega N^{-1} \big) N \Big)^{-1} N^T y \\ &= (I + \lambda K)^{-1} y, \end{aligned} \tag{11}$$

  where $K = (N^T)^{-1} \Omega N^{-1}$

- Note that this matrix K does not depend on λ. If we compute an eigendecomposition $K = UDU^T$, then the eigendecomposition of $S = N(N^T N + \lambda \Omega)^{-1} = (I + \lambda K)^{-1}$ is

$$S = \sum_{j=1}^n \frac{1}{1 + \lambda d_j} u_j u_j^T,$$

  where $D = \operatorname{diag}(d_1, \dots d_n)$

- Therefore the smoothing spline fitted values are $\hat{y} = Sy$, i.e.,

$$\hat{y} = \sum_{j=1}^{n} \frac{u_j^T y}{1 + \lambda d_j} u_j. \tag{12}$$

  An interpretation: smoothing splines perform a regression on the orthonormal basis $u_1, \ldots u_n \in \mathbb{R}^n$, yet they shrink the coefficients in this regression, with more shrinkage assigned to eigenvectors u_j that correspond to large eigenvalues d_j

- So what exactly are these basis vectors $u_1, \ldots u_n$? These are known as the *Demmler-Reinsch basis*, and a lot of their properties can be worked out analytically (Demmler & Reinsch 1975). Basically: the eigenvectors u_j that correspond to smaller eigenvalues d_j are smoother, and so with smoothing splines, we shrink less in their direction. Said differently, by increasing λ in the smoothing spline estimator, we are tuning out the more wiggly components. See Figure 4

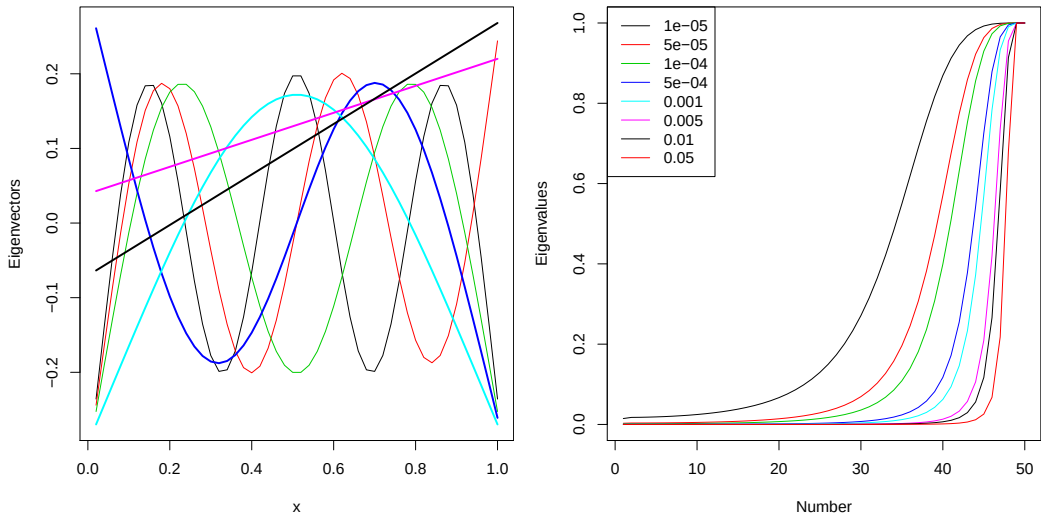

Figure 4: *For a problem with $n = 50$ evenly spaced inputs over $[0, 1]$, the left plot shows the bottom 7 eigenvectors of the Reinsch operator K. We can see that the smaller the eigenvalue, the "smoother" the eigenvector. The right plot shows the shrinkage weights $w_j = 1/(1 + \lambda d_j)$, $j = 1, \ldots n$ that are implicitly used by the smoothing spline estimator* (12)*, over 8 values of λ. We can see that when λ is larger, the weights decay faster, and hence the smoothing spline estimator places less weight on the "nonsmooth" eigenvectors*

3.7 Kernel smoothing equivalence

- It turns out that the cubic smoothing spline estimator is in some sense asymptotically equivalent to a kernel regression estimator, with an unusual choice of kernel. Recall that both are linear smoothers; this equivalence is achieved by showing that under some conditions the smoothing spline weights converge to kernel weights, under the kernel

$$K(x) = \frac{1}{2} \exp(-|x|/\sqrt{2}) \sin(|x|/\sqrt{2} + \pi/4), \tag{13}$$

  and a local choice of bandwidth $h(x) = \lambda^{1/4} f(x)^{-1/4}$, where $f(x)$ is the density of the input points. That is, the bandwidth adapts to the local distribution of inputs. This result is due to Silverman (1984), and hence (13) is sometimes called the "Silverman kernel". See Figure 5

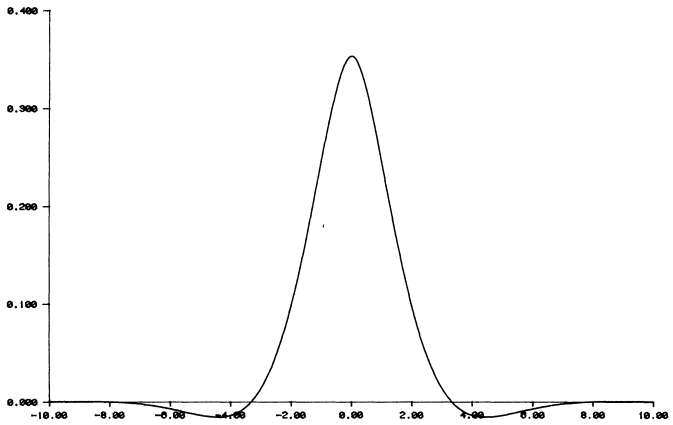

Figure 5: *The Silverman kernel in* (13)*, which is the (asymptotically) equivalent implicit used by smoothing splines. Note that it can be negative. From Silverman (1984)*

3.8 Multivariate splines

- Splines can be extended to multiple dimensions via the *thin-plate spline* construction. This extension is highly nontrivial, especially compared to the (conceptually) simple extension of kernels to higher dimensions. Another option is to use the *tensor-product spline* formulation in higher dimensions. Both of these concepts are discussed in Chapter 7 of Green & Silverman (1994)

- As per our usual comment, spline smoothing is both statistically and computationally challenging in multiple dimensions. Additive models built from univariate smoothing splines (or bivariate thin-plate splines) are a commonly used tool in high dimensions, and we'll discuss these near the end

4 Mercer kernels, RKHS

- Smoothing splines are just one example of an estimator of the form

$$\hat{f} = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda J(f), \tag{14}$$

  where $\mathcal{H}$ is a space of functions, and J is a penalty functional

- Another important subclass of this problem form: we choose the function space $\mathcal{H} = \mathcal{H}_K$ to be what is called a *reproducing kernel Hilbert space*, or RKHS, associated with a particular kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. To avoid confusion: this is not the same thing as a smoothing kernel! We'll adopt the convention of calling this second kind of kernel, i.e., the kind used in RKHS theory, a *Mercer kernel*, to differentiate the two

- There is an immense literature on the RKHS framework; here we follow the RKHS treatment in Chapter 5 of Hastie et al. (2009). Suppose that K is a positive definite kernel; examples include the polynomial kernel:

$$K(x, z) = (x^T z + 1)^k,$$

and the Gaussian radial basis kernel:

$$K(x, z) = \exp\left(-\delta \|x - z\|_2^2\right).$$

Mercer's theorem tells us that for any positive definite kernel function K, we have an eigenexpansion of the form

$$K(x, z) = \sum_{i=1}^{\infty} \gamma_i \phi_i(x) \phi_i(z),$$

for eigenfunctions $\phi_i(x)$, $i = 1, 2, \ldots$ and eigenvalues $\gamma_i \geq 0$, $i = 1, 2, \ldots$, satisfying $\sum_{i=1}^{\infty} \gamma_i^2 < \infty$. We then define $\mathcal{H}_K$, the RKHS, as the space of functions generated by $K(\cdot, z)$, $z \in \mathbb{R}^d$, i.e., elements in $\mathcal{H}_K$ are of the form

$$f(x) = \sum_{m \in M} \alpha_m K(x, z_m),$$

for a (possibly infinite) set M

- The above eigenexpansion of K implies that elements $f \in \mathcal{H}_K$ can be represented as

$$f(x) = \sum_{i=1}^{\infty} c_i \phi_i(x),$$

subject to the constraint that we must have $\sum_{i=1}^{\infty} c_i^2/\gamma_i < \infty$. In fact, this representation is used to define a norm $\|\cdot\|_{\mathcal{H}_K}$ on $\mathcal{H}_K$: we define

$$\|f\|_{\mathcal{H}_K}^2 = \sum_{i=1}^{\infty} c_i^2/\gamma_i.$$

- The natural choice now is to take the penalty functional in (14) as this squared RKHS norm, $J(f) = \|f\|_{\mathcal{H}_K}^2$. This yields the RKHS problem

$$\hat{f} = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \|f\|_{\mathcal{H}_K}^2. \tag{15}$$

A remarkable achievement of RKHS theory is that the infinite-dimensional problem (15) can be reduced to a finite-dimensional one (as was the case with smoothing splines). This is called the *representer theorem* and is attributed to Kimeldorf & Wahba (1970). In particular, this result tells us that the minimum in (15) is uniquely attained by a function of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i),$$

or in other words, a function f lying in the span of the functions $K(\cdot, x_i)$, $i = 1, \ldots n$. Furthermore, we can rewrite the problem (15) in finite-dimensional form, as

$$\hat{\alpha} = \underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - K\alpha\|_2^2 + \lambda \alpha^T K \alpha, \tag{16}$$

where $K \in \mathbb{R}^{n \times n}$ is a symmetric matrix defined by $K_{ij} = K(x_i, x_j)$ for $i, j = 1, \ldots n$. Once we have computed the optimal coefficients $\hat{\alpha}$ in (16), the estimated function $\hat{f}$ in (15) is given by

$$\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(x, x_i)$$

- The solution in (16) is
$$\hat{\alpha} = (K + \lambda I)^{-1} y,$$
so the fitted values $\hat{y} = (\hat{f}(x_1), \ldots \hat{f}(x_n))$ are
$$\hat{y} = K(K + \lambda I)^{-1} y = (I + \lambda K^{-1})^{-1} y,$$
showing that the RKHS estimator is yet again a linear smoother

- There is something subtle but extremely important about the finite-dimensional problem in (16): to express a flexible nonparametric function, in multiple dimensions, note that we need not write down an explicit basis, but need only to define a "kernelized" inner product between any two input points, i.e., define the entries of the kernel matrix $K_{ij} = K(x_i, x_j)$. This encodes a notion of similarity between x_i, x_j, or equivalently,
$$K(x_i, x_i) + K(x_j, x_j) - 2K(x_i, x_j)$$
encodes a notion of distance between x_i, x_j

- It can sometimes be much easier to define an appropriate kernel than to define explicit basis functions. Think about, e.g., the case when the input points are images, or strings, or some other weird objects—the kernel measure is defined entirely in terms of pairwise relationships between input objects, which can be done even in exotic input spaces

- Given the kernel matrix K, the kernel regression problem (16) is completely specified, and the solution is implicitly fit to lie in the span of the (infinite-dimensional) RKHS generated by the chosen kernel. This is a pretty unique way of fitting flexible nonparametric regression estimates. Note: this idea isn't specific to regression: kernel classification, kernel PCA, etc., are built in the analogous way

5 Linear smoothers

5.1 Degrees of freedom and unbiased risk estimation

- Literally every estimator we have discussed so far, when trained on (x_i, y_i), $i = 1, \ldots n$, produces fitted values $\hat{y} = (\hat{f}(x_1), \ldots \hat{f}(x_n))$ of the form
$$\hat{y} = Sy$$
for some matrix $S \in \mathbb{R}^{n \times n}$ 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 splines—but not on y. Recall that such estimators are called *linear smoothers*

- Consider the input points as fixed, and assume that y has i.i.d. components with mean 0 and variance σ^2. Recall that in this setting, we defined the degrees of freedom of an estimator $\hat{y}$
$$\mathrm{df}(\hat{y}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}(\hat{y}_i, y_i).$$
In particular, recall that for linear smoothers $\hat{y} = Sy$, the degrees of freedom is
$$\mathrm{df}(\hat{y}) = \sum_{i=1}^{n} S_{ii} = \mathrm{tr}(S),$$
the trace of the smooth matrix S

- Example: for a regression spline estimator, of polynomial order k, with knots at the locations $t_1, \ldots t_m$, recall that $\hat{y} = G(G^T G^{-1})G^T y$ for $G \in \mathbb{R}^{n \times (m+k+1)}$ the order k spline basis matrix over the knots $t_1, \ldots t_m$. Therefore

$$\mathrm{df}(\hat{y}) = \mathrm{tr}\big(G(G^T G)^{-1} G^T\big) = \mathrm{tr}\big(G^T G (G^T G)^{-1}\big) = m + k + 1,$$

  i.e., the degrees of freedom of a regression spline estimator is the number of knots + the polynomial order + 1. The same calculation shows that the degrees of freedom of a regression natural spline estimator is simply the number of knots (independent of the polynomial order)

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

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

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

$$\mathrm{df}(\hat{y}) = \mathrm{tr}\big(U(1 + \lambda D)^{-1} U^T\big) = \sum_{i=1}^{n} \frac{1}{1 + \lambda d_j},$$

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

- Degrees of freedom is generally a useful concept because it allows us to put two different procedures 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 performances

- A second more concrete motivation for degrees of freedom, recall, comes from the perspective of unbiased risk estimation, or unbiased test error estimation. In particular,

$$\hat{T} = \frac{1}{n} \|y - \hat{y}\|_2^2 + \frac{2\sigma^2}{n} \mathrm{df}(\hat{y})$$

  serves as an unbiased estimate of the test error $\mathbb{E}\|y' - \hat{y}\|_2^2 / n$, where y' is an i.i.d. copy of y. For linear smoothers, this is simply

$$\hat{T} = \frac{1}{n} \|y - Sy\|_2^2 + \frac{2\sigma^2}{n} \mathrm{tr}(S) \tag{17}$$

- Suppose our linear smoother of interest depends on a tuning parameter $\lambda \in \Lambda$ (representing, e.g., the bandwidth for kernel smoothing, or the penalty parameter for smoothing splines and Mercer kernels), and express this by $\hat{y}_\lambda = S_\lambda y$. Then we could choose the tuning parameter λ to minimize the estimated risk, as in

$$\hat{\lambda} = \underset{\lambda \in \Lambda}{\mathrm{argmin}} \ \frac{1}{n} \|y - S_\lambda y\|_2^2 + \frac{2\sigma^2}{n} \mathrm{tr}(S_\lambda).$$

  This is just like the C_p criterion, or AIC, in ordinary regression; we could replace the factor of 2 above with $\log n$ to obtain something like BIC

5.2 Leave-one-out and generalized cross-validation

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

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}^{-i}(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{f}(x_i)}{1 - S_{ii}}\right)^2, \tag{18}$$

  where $\hat{f}^{-i}$ denotes the estimated function trained on all but the ith pair (x_i, y_i). This provides an enormous computational savings because it shows that to compute leave-one-out cross-validation error, we don't have to actually ever compute $\hat{f}^{-i}$, $i = 1, \ldots n$

- Why does (18) hold, and for which linear smoothers $\hat{y} = Sy$? Just rearranging (18) perhaps demystifies this seemingly magical relationship and helps to answer these questions. Suppose we knew that $\hat{f}$ had the property

$$\hat{f}^{-i}(x_i) = \frac{1}{1 - S_{ii}}\big(\hat{f}(x_i) - S_{ii}y_i\big). \tag{19}$$

  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 across all but the ith point, $\hat{f}(x_i) - S_{ii}y_i = \sum_{i \neq j} S_{ij}y_j$, 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 (19), it is easy to show the leave-one-out formula (18). We have

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

  and then squaring both sides and summing over n gives (18)

- Finally, *generalized cross-validation* is a small twist on the right-hand side in (18) 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 $\mathrm{tr}(S)/n$,

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

  This can be of computational advantage in some cases where $\mathrm{tr}(S)$ is easier to compute that individual elements S_{ii}, and is also closely tied to the unbiased test error estimate in (17), seen by making the approximation $1/(1-x)^2 \approx 1 + 2x$

6 Locally adaptive estimators

6.1 Wavelet smoothing

- Not every nonparametric regression estimate needs to be a linear smoother (though this does seem to be incredibly common), and *wavelet smoothing* is one of the leading nonlinear tools for nonparametric estimation. The theory of wavelets is quite elegant and we only give a very terse 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

- Assume that $d = 1$. Local adaptivity in higher dimensions is still largely an unexplored topic (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 in $[0, 1]$:

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

  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

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

- 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):

  $$\theta = W^T y,$$

  we threshold the coefficients θ:

  $$\hat{\theta} = \mathcal{T}_\lambda(\theta),$$

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

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

- 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 $\mathcal{T}_\lambda$ is usually taken to be hard-thresholding, i.e.,

  $$[\mathcal{T}_\lambda(z)]_i = z_i \cdot 1\{|z_i| \geq \lambda\}, \quad i = 1, \ldots n,$$

  or soft-thresholding, i.e.,

  $$[\mathcal{T}_\lambda(z)]_i = \big(z_i - \text{sign}(z_i)\lambda\big) \cdot 1\{|z_i| \geq \lambda\}, \quad i = 1, \ldots n.$$

  These thresholding functions are both also $O(n)$, and computationally trivial, making wavelet smoothing very fast overall. It should be emphasized that wavelet smoothing is not a linear smoother, i.e., there is no matrix S here such that $\hat{y} = Sy$ for all y

- 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

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

  and then the wavelet smoothing fitted values are $\hat{y} = W\hat{\theta}$. Here $\|\theta\|_0 = \sum_{i=1}^n 1\{\theta_i \neq 0\}$, the number of nonzero components of θ. For soft-thresholding, the corresponding optimization problem is

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

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

6.2 The strengths of wavelets, the limitations of linear smoothers

- Apart from its sheer computational speed, 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

- Can a linear smoother do the same? We can appeal to convergence theory to give a precise answer to this question. We consider the function class $F(k, L)$ of all k times (weakly) differentiable functions, whose kth derivative satisfies $\mathrm{TV}(f^{(k)}) \leq L$, with $\mathrm{TV}(\cdot)$ being the total variation operator. A seminal result of Donoho & Johnstone (1998) shows that, assuming $f_0 \in F(k, L)$ (and further conditions on the problem setup), the wavelet smoothing estimator with an appropriately tuned parameter λ converges at the rate $n^{-(2k+2)/(2k+3)}$. These authors also show that this is the minimax rate of convergence over $F(k, L)$

- Now, it can be shown that $F(k, L) \supseteq \Sigma(k+1, L-1)$, where $\Sigma(k+1, L-1)$ is the order $k+1$ Holder class that we considered previously. From what we know about kernel smoothing, this estimator converges at the rate

$$n^{-(2(k+1))/(2(k+1)+1)} = n^{-(2k+2)/(2k+3)},$$

  over $\Sigma(k+1, L-1)$. But how about the larger class $F(k, L)$? Can kernel smoothing achieve the same (minimax) rate over this larger class? How about linear smoothers in general?

  A remarkable result of Donoho & Johnstone (1998) shows that no linear smoother can attain the minimax rate over $F(k, L)$, and that a lower bound on the convergence rate achieved by linear smoothers is $n^{-(2k+1)/(2k+2)}$. There is actually a big difference in these rates:

$$\frac{n^{-(2k+1)/(2k+2)}}{n^{-(2k+2)/(2k+3)}} = n^{(k+1)/((2k+2)(2k+3))} \to \infty \quad \text{as } n \to \infty.$$

  Practically, the performance of 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 major 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

- Wavelets are not the end of the story when it comes to local adaptivity. E.g., both kernel smoothing and smoothing splines can be made to be more locally adaptive by allowing for a local bandwidth parameter or a local penalty parameter; but this can be difficult to implement in practice. Next we discuss two different locally adaptive nonparametric estimators. These only use a single smoothing parameter, and still achieve the minimax rate of convergence over $F(k, L)$

6.3 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 order k, the estimate is defined as

$$\hat{f} = \operatorname*{argmin}_f \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \mathrm{TV}(f^{(k)}), \tag{20}$$

where $\mathrm{TV}(\cdot)$ is the total variation operator. 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 (20) 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 $x_1, \ldots x_n$. (When $k = 0, 1$, they do, but in general, they do not)

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

$$\hat{f} = \underset{f \in \mathcal{G}_k}{\operatorname{argmin}} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \mathrm{TV}(f^{(k)}), \tag{21}$$

  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 (21) becomes one of finding the coefficients in this basis expansion,

$$\hat{\beta} = \underset{f \in \mathcal{G}_k}{\operatorname{argmin}} \sum_{i=1}^{n} \Big(y_i - \sum_{j=1}^{n} \beta_j g_j(x_i)\Big)^2 + \lambda \mathrm{TV}\Big\{\Big(\sum_{j=1}^{n} \beta_j g_j(x_i)\Big)^{(k)}\Big\}, \tag{22}$$

  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

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

  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

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

  Hence the locally adaptive regression spline problem (22) can be expressed as

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - G\beta\|_2^2 + \lambda k! \sum_{i=k+2}^{n} |\beta_i|. \tag{23}$$

  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 (23) (originally, problem (21)) and the smoothing splines (9) (originally, problem (5)). 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 (23), the estimated spline will have knots at a subset of T_k (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 (9), the estimated function will have 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 (23) is actually a lot harder than the smoothing spline problem in (9). 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 (23) 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 real tradeoff here, because with extra computation comes much improved local adaptivity of the fits. See Figure 6 for an example. Theoretically, locally adaptive regression splines (like wavelets) achieve the minimax rate $n^{-(2k+2)/(2k+3)}$ of convergence over $F(k, L)$ (Mammen & van de Geer 1997). In this regard, as we discussed previously, they have a big advantage over any linear smoother

6.4 Trend filtering

- At a high level, you can think of trend filtering as computationally efficient version of locally adaptive regression splines, though their original construction (Steidl et al. 2006, Kim et al. 2009) comes from a fairly different perspective. We will begin by describing their connection to locally adaptive regression splines, following Tibshirani (2014)

- Revisit the formulation of locally adaptive regression splines in (21), where the minimization domain is $\mathcal{G}_k = \text{span}\{g_1, \ldots g_n\}$, and $g_1, \ldots g_n$ are the kth order truncated power basis in (4) having knots in a set $T_k \subseteq \{x_1, \ldots x_n\}$ with size $|T_k| = n - k - 1$. The *trend filtering* problem is given by replacing $\mathcal{G}_k$ with a different function space,

$$\hat{f} = \underset{f \in \mathcal{H}_k}{\text{argmin}} \sum_{i=1}^{n} \big(y_i - f(x_i)\big)^2 + \lambda \text{TV}(f^{(k)}), \tag{24}$$

where the new domain is $\mathcal{H}_k = \text{span}\{h_1, \ldots h_n\}$. Assuming that the input points are ordered, $x_1 < \ldots < x_n$, the functions $h_1, \ldots h_n$ are defined by

$$\begin{aligned} h_j(x) &= \prod_{\ell=1}^{j-1} (x - x_\ell), \quad j = 1, \ldots k+1, \\ h_{k+1+j}(x) &= \prod_{\ell=1}^{k} (x - x_{j+\ell}) \cdot 1\{x \geq x_{j+k}\}, \quad j = 1, \ldots n-k-1. \end{aligned} \tag{25}$$

(Our convention is to take the empty product to be 1, so that $h_1(x) = 1$.) These are dubbed the *falling factorial basis*, and are piecewise polynomial functions, taking an analogous form to the truncated power basis functions in (4). Loosely speaking, they are given by replacing an rth order power function in the truncated power basis with an appropriate r-term product, e.g., replacing x^2 with $(x - x_2)(x - x_1)$, and $(x - t_j)^k$ with $(x - x_{j+k})(x - x_{j+k-1}) \cdot \ldots (x - x_{j+1})$

- Defining the falling factorial basis matrix

$$H_{ij} = h_j(x_i), \quad i, j = 1, \ldots n,$$

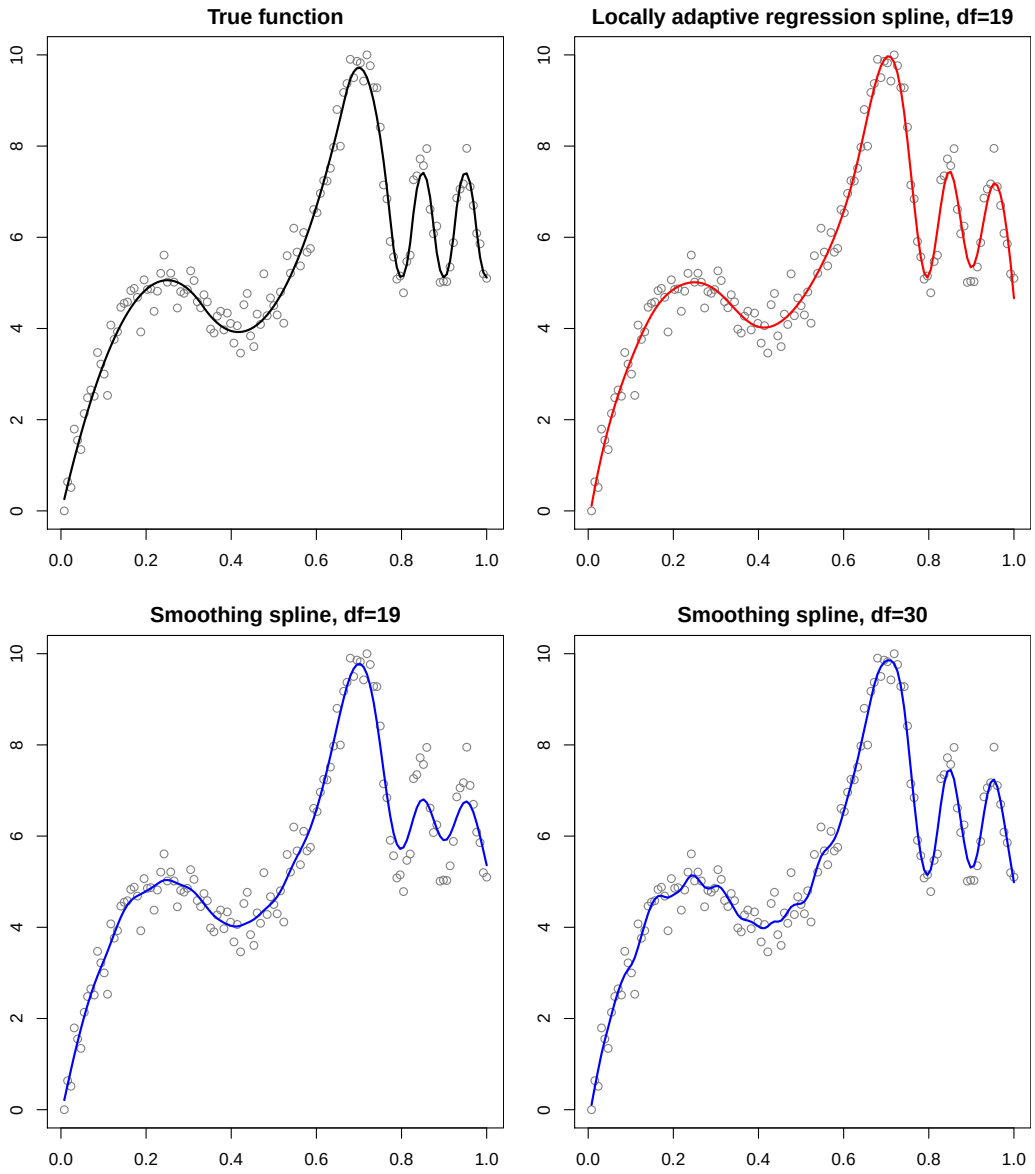

Figure 6: *The top left plot shows the true regression function, which has inhomogeneous smoothness—smoother towards the left part of the domain, wigglier towards the right. The top right plot shows the locally adaptive regression spline estimate with 19 degrees of freedom; notice that it picks up the right level of smoothness throughout. The bottom left plot shows the smoothing spline fit 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 fit 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*

it is now straightforward to check that the proposed problem of study, trend filtering in (24), is equivalent to

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - H\beta\|_2^2 + \lambda k! \sum_{i=k+2}^{n} |\beta_i|. \tag{26}$$

This is still a lasso problem, but now in the falling factorial basis matrix H. Compared to the locally adaptive regression spline problem (23), there may not seem to be much of a difference here—like G, the matrix H is dense, and solving (26) would be slow. So why did we go to all the trouble of defining trend filtering, i.e., introducing the somewhat odd basis $h_1, \dots h_n$ in (25)?

- The utility of trend filtering (26) is seen after reparametrizing the problem, by inverting H. Let $\theta = H\beta$, and rewrite the trend filtering problem as

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^n}{\operatorname{argmin}} \ \|y - \theta\|_2^2 + \lambda \|D\theta\|_1, \tag{27}$$

where $D \in \mathbb{R}^{(n-k-1)\times n}$ denotes the last $n-k-1$ rows of $k! \cdot H^{-1}$. Explicit calculation shows that D is a banded matrix (Tibshirani 2014, Wang et al. 2014). For simplicity of exposition, consider the case when $x_i = i$, $i = 1, \dots n$. Then

$$D = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & & & & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix} \quad \text{when } k = 0,$$

$$D = \begin{bmatrix} 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ 0 & 0 & 1 & -2 & \dots & 0 \\ \vdots & & & & & \end{bmatrix} \quad \text{when } k = 1,$$

$$D = \begin{bmatrix} -1 & 3 & -3 & 1 & \dots & 0 \\ 0 & -1 & 3 & -3 & \dots & 0 \\ 0 & 0 & -1 & 3 & \dots & 0 \\ \vdots & & & & & \end{bmatrix} \quad \text{when } k = 2, \text{ etc.}$$

One can therefore interpret D as a kind of discrete derivative operator, of order $k+1$. This also suggests an intuitive interpretation of trend filtering (27) as a discrete approximation to the original locally adaptive regression spline problem in (20)

- The bandedness of D means that the trend filtering problem (27) can be solved in nearly linear time (complexity $O(n^{1.5})$ in the worst case). Hence trend filtering estimates are much easier to fit than locally adaptive regression splines

- But what of their statistical relevancy? Did switching over to the falling factorial basis in (25) wreck the strong local adaptivity properties that we cared about in the first place? Fortunately, the answer is no, and in fact, trend filtering and locally adaptive regression spline estimates are extremely hard to distinguish in practice. See Figure 7

- Furthermore, one can prove a strong coupling between the truncated power basis (4) and the falling factorial basis (25), and use this to establish that the locally adaptive regression and

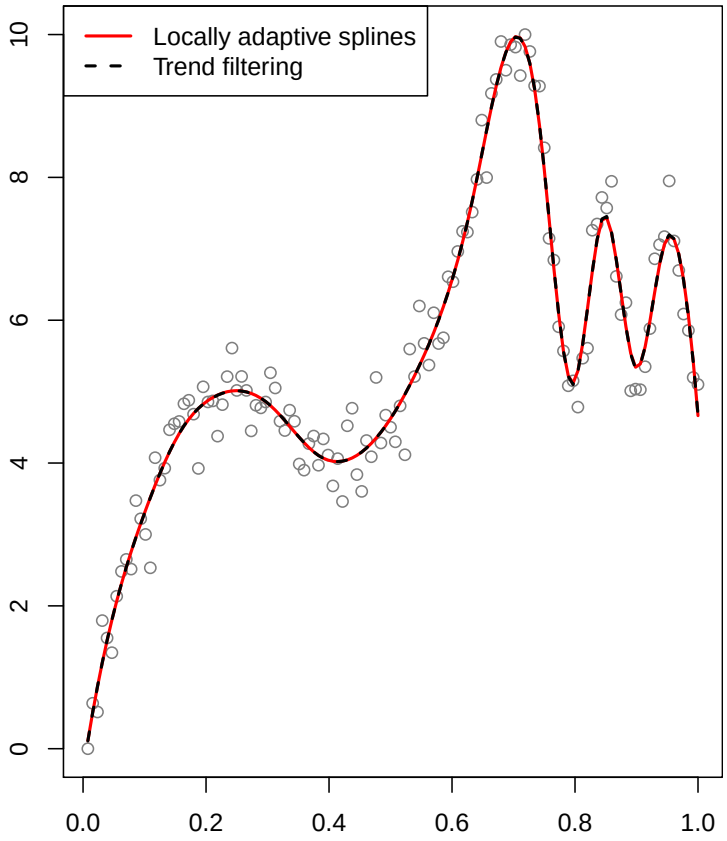

Figure 7: *Trend filtering and locally adaptive regression spline estimates, fit on the same data set as in Figure 6. The two are tuned at the same level, and the estimates are visually indistinguishable*

trend filtering estimates are close. The two are close enough that trend filtering inherits the minimax optimality of locally adaptive regression splines, i.e., it too converges at the rate $n^{-(2k+2)/(2k+3)}$ over $F(k, L)$ (Tibshirani 2014, Wang et al. 2014). This means that trend filtering offers significant improvements over linear smoothers, and yet its computational cost is not too much steeper than a single banded linear system solve

7 The multivariate case

7.1 The curse of dimensionality

- So far, we've looked mostly at the univariate case, $d = 1$. But as we've briefly mentioned, most everything we've discussed so far has a multivariate counterpart; e.g., recall, kernel smoothing very naturally extends to multiple dimensions; univariate splines can be generalized via thin-plate splines or tensor-product splines; and Mercer kernels were defined directly in multiple dimensions. All of these multivariate extensions are capable or producing rich nonparametric fits in low to moderate dimensions

- But in high dimensions the story is very different; if d is large compared to n, then "true" multivariate extensions like these ones are problematic, and suffer from poor variance. The

curse of dimensionality, in rough terms, says that estimation gets exponentially harder as the number of dimensions increases. (This term usually is attributed to Bellman (1962), who encountered an analogous issue but in a separate context—dynamic programming)

- This curse of dimensionality is echoed by the role of d in the minimax rates of convergence of nonparametric regression estimators, across various setups. E.g., recall the Holder class of functions $\Sigma(k, L)$ that we defined, in the univariate case, of functions whose $(k-1)$st derivative is L-Lipschitz. In higher dimensions, the natural extension of this is the space $\Sigma_d(k, L)$ of functions on $\mathbb{R}^d$ whose kth order partial derivatives are all L-Lipschitz. It can be shown that

$$\min_{\hat{f}} \max_{f_0 \in \Sigma_d(k,L)} \mathbb{E}\big(\hat{f}(X) - f_0(X)\big)^2 = \Omega(n^{-2k/(2k+d)}),$$

for the random input model, under mild conditions. See Section 3.2. of Gyorfi et al. (2002). What does this rate mean? Flipped around, this implies that we need $n \geq (1/\epsilon)^{(2k+d)/(2k)}$ to achieve an MSE bound of ϵ, i.e., the sample complexity grows exponentially in d

7.2 Additive models

- *Additive models* finesse the curse of dimensionality by fitting an estimate that decomposes as a sum of univariate functions across dimensions. Instead of considering a full d-dimensional function of the form

$$f(x) = f(x_{\cdot 1}, \ldots x_{\cdot d}), \tag{28}$$

we restrict our attention to functions of the form

$$f(x) = f_1(x_{\cdot 1}) + \ldots f_d(x_{\cdot d}). \tag{29}$$

(Here the notation $x_{\cdot j}$ denotes the jth component of $x \in \mathbb{R}^d$, slightly unusual notation used so as not to confuse with the labeling of the d-dimensional inputs $x_1, \ldots x_n$). As each function f_j, $j = 1, \ldots d$ is univariate, fitting an estimate of the form (29) is certainly less ambitious than fitting one of the form (28). On the other hand, the additive framework is still flexible enough to capture interesting (marginal) behavior in high dimensions

- The choice of modeler (29) need not be regarded as an assumption we make about the true function f_0, just like we don't always assume that the true model is linear when using linear regression. In many cases, we fit an additive model because we think it may provide a useful approximation to the truth, and is able to scale well with the number of dimensions d

- A beautiful result by Stone (1985) encapsulates this idea precisely. This author shows that, while it may be difficult to estimate an arbitrary regression function f_0 in high dimensions, we can still estimate its *best additive approximation* $\bar{f}_0$ well. If we assume that each component function $\bar{f}_{0,j}$, $j = 1, \ldots d$ lies in Holder class $\Sigma(k, L)$, and we estimate component function $\hat{f}_j$, $j = 1, \ldots d$ using a kth degree spline, then

$$\mathbb{E}\|\hat{f}_j - \bar{f}_{0,j}\|_{L_2} = O(n^{-2k/(2k+1)}), \quad j = 1, \ldots d.$$

Thus each component of the best additive approximation $\bar{f}_0$ to f_0 can be estimated at the optimal rate. In other words, though we cannot hope to recover f_0 arbitrarily, we can recover its major structure along the coordinate axes

- Estimation with additive models is actually very simple; we can just choose our favorite univariate smoother (i.e., nonparametric procedure), and cycle through estimating each function f_j, $j = 1, \ldots d$ individually (like a block coordinate descent algorithm). Denote our choice of

univariate smoother by $\mathcal{S}$, so that the fitted function from smoothing $y = (y_1, \ldots y_n) \in \mathbb{R}^n$ over the inputs $z = (z_1, \ldots z_n) \in \mathbb{R}^n$ can be written as

$$\hat{f} = \mathcal{S}(z, y).$$

E.g., it is common to choose $\mathcal{S}$ to be the cubic smoothing spline smoother, where the tuning parameter λ selected by generalized cross-validation

- Once $\mathcal{S}$ has been chosen, we initialize $\hat{f}_1, \ldots \hat{f}_d$ (say, to all to zero), and repeat the following steps for $j = 1, \ldots d, 1, \ldots d, \ldots$:
  - define $r_i = y_i - \sum_{\ell \neq j} \hat{f}_\ell(x_{i\ell})$, $i = 1, \ldots n$;
  - smooth $\hat{f}_j = \mathcal{S}(x_{\cdot j}, r)$;
  - center $\hat{f}_j = \hat{f}_j - \frac{1}{n} \sum_{i=1}^n \hat{f}_j(x_{ij})$.

  This algorithm is known as *backfitting*. The last line in the update above is used to remove the mean from each function $\hat{f}_j$, $j = 1, \ldots d$, otherwise the model would not be identifiable. Our final estimate therefore takes the form

$$\hat{f}(x) = \bar{y} + \hat{f}_1(x_{\cdot 1}) + \ldots \hat{f}_d(x_{\cdot d}),$$

where $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$. Hastie & Tibshirani (1990) provide a very nice exposition on the some of the more practical aspects of backfitting and additive models

7.3 Sparse additive models

- In truly high dimensions, instead of estimating a full set of d additive component functions, we might choose a smaller number of dimensions and estimate component functions along these directions only. That is, our estimate would take the form

$$\hat{f}(x) = \bar{y} + \sum_{j \in A} \hat{f}_j(x_{\cdot j}),$$

where $A \subseteq \{1, \ldots d\}$ is an active set of dimensions. This is a natural idea, but it is a research topic still very much in development. Some recent works are Lin & Zhang (2006), Ravikumar et al. (2009), Raskutti et al. (2012)

8 Nonparametric classification

- Lastly, we discuss nonparametric estimators for classification. Indeed, it is pretty easy to build nonparametric classifiers given our regression tools. For $(X, Y) \in \mathbb{R}^d \times \{0, 1\}$, consider the regression function

$$f_0(x) = \mathbb{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x),$$

which is now the conditional probability of observing class 1, given $X = x$. Given samples $(x_1, y_1), \ldots (x_n, y_n) \in \mathbb{R}^d \times \{0, 1\}$, we can use any nonparametric regression method to form an estimate $\hat{f}$ of f_0, and then define a "plug-in" classifier via

$$\hat{C}(x) = \begin{cases} 0 & \text{if } \hat{f}(x) \leq 1/2 \\ 1 & \text{if } \hat{f}(x) < 1/2 \end{cases}.$$

This of course estimates the optimal classification rule, sometimes called the *Bayes classifier*,

$$C_0(x) = \begin{cases} 0 & \text{if } f_0(x) \leq 1/2 \\ 1 & \text{if } f_0(x) < 1/2 \end{cases}$$

- What does this look like, e.g., for k-nearest-neighbors? Recall that here the regression estimate is $\hat{f}(x) = \frac{1}{k}\sum_{i \in \mathcal{N}_k(x)} y_i$, and so $\hat{C}(x) = 1$ if and only if more than half of the neighbors $\mathcal{N}_k(x)$ of x are of class 1. This is precisely the k-nearest-neighbors classifier that you have likely already learned, which "votes" based on class memberships of neighboring points to decide the predicted class

- In general it may seem a bit funny to estimate f_0 using a standard nonparametric regression tool, since these tools are usually defined by a squared error loss criterion, and f_0 always lies between 0 and 1. A somewhat more principle approach would be to change the loss function so that it is appropriate for the classification setting. E.g., we could instead directly estimate the conditional log oggs,

$$f_0(x) = \log\Big(\frac{\mathbb{P}(Y = 1|X = x)}{\mathbb{P}(Y = 1|X = x)}\Big),$$

  and this leads to a nonparametric logistic regression model. Taking, say, the smoothing spline approach, we could define our estimate by

$$\hat{f} = \underset{f}{\operatorname{argmin}} \sum_{i=1}^{n} \big(- y_i f(x_i) + \log(1 + e^{-f(x_i)})\big) + \lambda \int_a^b \big(f^{((k+1)/2)}(x)\big)^2 \, dx.$$

  Let $\eta_1, \ldots \eta_n$ denote the natural kth order spline basis with knots over the inputs $x_1, \ldots x_n$, and define the basis matrix $N \in \mathbb{R}^{n \times n}$ and penalty matrix $\Omega \in \mathbb{R}^{n \times n}$ just as we did for smoothing splines in regression,

$$N_{ij} = \eta_j(x_i) \quad \text{and} \quad \Omega_{ij} = \int_0^1 \eta_i^{((k+1)/2)}(t) \eta_j^{((k+1)/2)}(t) \, dt \quad \text{for } i, j = 1, \ldots n.$$

  Then we can reformulate the above problem as

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^n}{\operatorname{argmin}} \; -y^T N\beta + \sum_{i=1}^{n} \log(1 + e^{-(N\beta)_i}) + \lambda \beta^T \Omega \beta,$$

  which is a logistic regression problem with a generalized ridge penalty. Our final classifier $\hat{C}(x)$ now outputs class 1 when $\hat{f}(x) = \sum_{j=1}^{n} \hat{\beta}_j \eta_j(x) > 0$, and outputs 0 otherwise

- An approach like the one above extends seamlessly to the additive model setting, and also to any loss derived from an exponential family distribution. This lies at the heart of *generalized additive models*, a framework for producing flexible nonparametric estimates in multiple dimensions, whether predicting a continuous response (regression), or binary response (logistic regression), or counts (Poisson regression), etc. See Hastie & Tibshirani (1990)

References

Bellman, R. (1962), *Adaptive Control Processes*, Princeton University Press, Princeton.

de Boor, C. (1978), *A Practical Guide to Splines*, Springer, New York.

Demmler, A. & Reinsch, C. (1975), 'Oscillation matrices with spline smoothing', *Numerische Mathematik* **24**(5), 375–382.

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

Green, P. & Silverman, B. (1994), *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*, Chapman & Hall/CRC Press, Boca Raton.

Gyorfi, L., Kohler, M., Krzyzak, A. & Walk, H. (2002), *A Distribution-Free Theory of Nonparametric Regression*, Springer, New York.

Hastie, T. & Tibshirani, R. (1990), *Generalized Additive Models*, Chapman and Hall, London.

Hastie, T., Tibshirani, R. & Friedman, J. (2009), *The Elements of Statistical Learning; Data Mining, Inference and Prediction*, Springer, New York. Second edition.

Johnstone, I. (2011), *Gaussian estimation: Sequence and wavelet models*, Under contract to Cambridge University Press. Online version at `http://www-stat.stanford.edu/~imj`.

Kim, S.-J., Koh, K., Boyd, S. & Gorinevsky, D. (2009), 'ℓ_1 trend filtering', *SIAM Review* **51**(2), 339–360.

Kimeldorf, G. & Wahba, G. (1970), 'A correspondence between bayesian estimation on stochastic processes and smoothing by splines', *Annals of Mathematical Statistics* **41**(2), 495–502.

Lin, Y. & Zhang, H. H. (2006), 'Component selection and smoothing in multivariate nonparametric regression', *Annals of Statistics* **34**(5), 2272–2297.

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

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

Raskutti, G., Wainwright, M. & Yu, B. (2012), 'Minimax-optimal rates for sparse additive models over kernel classes via convex programming', *Journal of Machine Learning Research* **13**, 389–427.

Ravikumar, P., Liu, H., Lafferty, J. & Wasserman, L. (2009), 'Sparse additive models', *Journal of the Royal Statistical Society: Series B* **75**(1), 1009–1030.

Scholkopf, B. & Smola, A. (2002), 'Learning with kernels'.

Silverman, B. (1984), 'Spline smoothing: the equivalent variable kernel method', **12**(3), 898–916.

Simonoff, J. (1996), *Smoothing Methods in Statistics*, Springer, New York.

Steidl, G., Didas, S. & Neumann, J. (2006), 'Splines in higher order TV regularization', *International Journal of Computer Vision* **70**(3), 214–255.

Stone, C. (1985), 'Additive regression models and other nonparametric models', *Annals of Statistics* **13**(2), 689–705.

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

Tsybakov, A. (2009), *Introduction to Nonparametric Estimation*, Springer, New York.

Wahba, G. (1990), *Spline Models for Observational Data*, Society for Industrial and Applied Mathematics, Philadelphia.

Wang, Y., Smola, A. & Tibshirani, R. J. (2014), 'The falling factorial basis and its statistical properties', *International Conference on Machine Learning* **31**.

Wasserman, L. (2006), *All of Nonparametric Statistics*, Springer, New York.