Smoothing Method

Guoyu Zhang, Xiaopu Wang

University of Science and Technology of China

December 7, 2022

Content

Table of Contents

Motivation

- When facing regression problem

$$E[Y|X] = f(X)\,,$$

  it is extremely unlikely that the true function $f(X)$ is actually linear in X as we assume in linear regression model.
- Unlike the generalized linear model (GLM), here we assume that f is a smooth function of X, the continuity of which would play an important role on borrowing the information of the neighbourhood of independent sample point.

Polynomials

- Famous approximation theory tells us that any continuous function could be approximated by a polynomial with any accuracy. But this is unpractical in real world application.

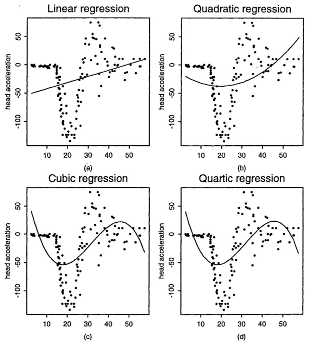


Figure: Polynomial fitting.

Some thoughts for Smoothing

- To learn the nonlinear smooth pattern, we prefer to use other efficient ways instead of using the polynomial fitting.
- High dimensional fitting (maybe infinite dimensional) with regularization is necessary.
- Instead of interpretation, prediction accuracy is the main guideline for the model selection of the amount of smoothness.

Table of Contents

Motivation

The core idea is to assume $f(X)$ falls in the linear function space (maybe infinite finite-dimensional) feature space of X, e.g.,

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

for some $h_m(X)$, the reason of doing this involves:

- It' easy to connect the least square model.
- Have nice geometrical interpretation.
- It actually can specify a broad range of models given we have discrete data.

Examples

Some simple and widely used examples of the h_m are the following:

- $h_m(X) = X_m, m = 1, \dots, p$ recovers the original linear model
- $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ allows us to augment the inputs with polynomial terms to achieve higher-order Taylor expansions.
- $h_m(X) = I(L_m \leq X_k < U_m)$, an indicator for a region of X_k. By breaking the range of X_k up into M_k such non-overlapping regions results in a model with a piece-wise constant contribution for X_k

But the choice of the basis function is critical, some included irrelevant functions would only increase the variance of estimation, do no help correct the bias.

Reasons for Splines

- Naturally, we can always use a high order polynomial to approximate f if we don't know anything, i.e.

$$f(X) = \sum_{m=0}^{M} \beta_m X^m$$

- However, the famous Runge's phenomenon shows that high-degree polynomial interpolation can be troublesome. The polynomial fitting is also not sufficient to capture the local signal.
- The brilliant idea of using piece-wise polynomials, which is called splines, comes out and shows its excellent properties.

Splines

- We assume X to be one-dimension if not specifying from now on.
- A piece-wise polynomial function is obtained by dividing the domain of X into contiguous intervals, and representing f by a separate polynomial in each interval.

We show some pictures of piece-wise polynomials.

Piece-wise Polynomials

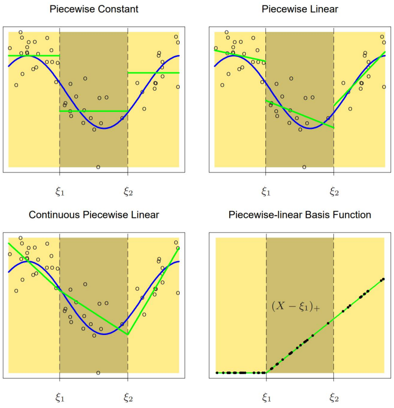


Figure: Piece-wise polynomials.

Piecewise-linear Basis Function

- Except in special cases, we would typically prefer the third panel, which is also piece-wise linear, but restricted to be continuous at the two knots. These continuity restrictions lead to linear constraints on the parameters. Since there are two restrictions, we expect to get back two parameters, leaving four free parameters[3].
- A direct way to proceed in this case is to use a basis that incorporates the constraints:

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

where t_+ denotes the positive part of t.

Piece-wise Cubic Polynomials

- We often prefer smoother functions, and these can be achieved by increasing the order of the local polynomial
- We now show s a series of piece-wise cubic polynomials fit to the same data, with increasing orders of continuity at the knots
- The function in the lower right panel is continuous, and has continuous first and second derivatives at the knots. It is known as a cubic spline

Piece-wise Cubic Polynomials

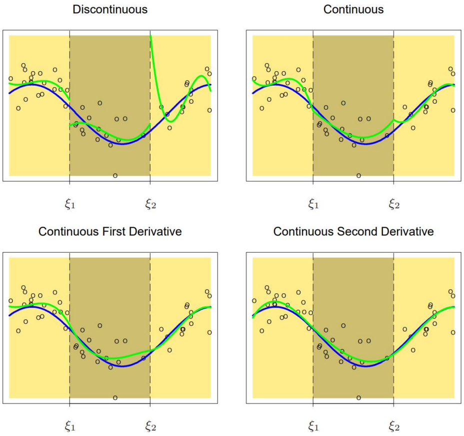


Figure: Piece-wise cubic polynomials.

Basis of cubic spline

- It is claimed that cubic splines are the lowest-order spline for which the knot-discontinuity is not visible to the human eye, so there is seldom any good reason to go beyond cubic-splines.
- It is not hard to show that the following basis represents a cubic spline with knots at ξ_1 and ξ_2 :

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

- Note that enforcing one more order of continuity would lead to a global cubic polynomial

Order-M Spline

- More generally, an order-M spline with knots $\xi_j, j = 1, \ldots, K$ is a piece-wise polynomial of order M, and has continuous derivatives up to order $M - 2$.
- For example, a cubic spline has $M = 4$, the piece-wise constant function is an order-1 spline, while the continuous piece-wise linear function is an order-2 spline.
- The general form for the truncated-power basis set would be

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

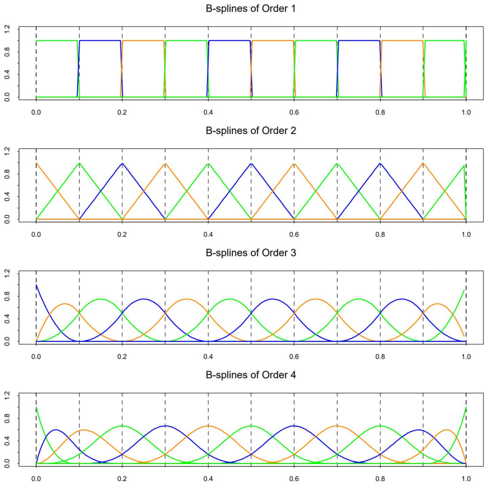
Knot Selection

- The location of the knot denotes the high order discontinuity of the raw function.
- Also, the increase of knots would enhance the model complexity of the learning space.
- In practice, we may uniformly distribute sufficient knots over an interval, but it's maybe more reasonable to select the knots by data.

B-Spline

- While the truncated power basis is conceptually simple, it is not too attractive numerically: its computation for coefficients uses algorithm like least-square.
- The B-spline basis, on the other hand, allows for efficient computations even when the number of knots K is large
- The following pictures of B -plines might give you a first expression of what B splines looks like and why they are attractive numerically.

Pictures of B-Splines



Construction of B-Splines

- Before we can get started, we need to augment the knot sequence above.
- Let $\xi_0 < \xi_1$ and $\xi_K < \xi_{K+1}$ be two boundary knots, we now define the augmented knot sequence τ such that

$$\tau_1 \le \tau_2 \le \cdots \le \tau_M \le \xi_0;$$

$$\tau_{j+M} = \xi_j, \; j = 1, \cdots, K;$$

$$\xi_{K+1} \le \tau_{K+M+1} \le \tau_{K+M+2} \le \cdots \le \tau_{K+2M}$$

- The actual values of these additional knots beyond the boundary are arbitrary, and it is customary to make them all the same and equal to ξ_0 and ξ_{K+1}, respectively.

Construction of B-Splines

Denote by $B_{i,m}(x)$ the i th B-spline basis function of order m for the knot-sequence τ, $m \leq M$. They are defined recursively in terms of divided differences as follows:

$$B_{i,1}(x) = \begin{cases} 1 & \text{if } \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, K + 2M - 1$.
These are also known as Haar basis functions.

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x)$$

$$\text{for } i = 1, \ldots, K + 2M - m.$$

Properties of B-Splines

- With $M = 4, B_{i,4},\ \ i = 1, \cdots, K+4$ are the $K+4$ cubic B-spline basis functions for the knot sequence ξ. This recursion can be continued and will generate the B-spline basis for any order spline.
- If we adopt the convention that $B_{i,1} = 0$ if $\tau_i = \tau_{i+1}$, then by induction $B_{i,m} = 0$ if $\tau_i = \tau_{i+1} = \ldots = \tau_{i+m}$.
- Note also that in the construction above, only the subset $B_{i,m}, i = M - m + 1, \ldots, M + K$ are required for the B-spline basis.
- The local support of B-splines has important computational implications, especially when the number of knots K is large.

Penalized Least Square

- The above content about the splines shows great properties of it, but when we actually use this, choosing knots will be a big problem.
- To overcome this, we try to the add the penalty term to the likelihood to make it clear how to choose knots.

Consider a penalized problem like this:

$$\min_{f\in\mathcal{H}} \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda \|Lf\|_{\mathcal{H}}^2$$

where $\mathcal{H}$ is a Hilbert space with norm $\|\cdot\|_{\mathcal{H}}$ and L is a linear operator of it. Under certain condition, the above solution would uniquely exist.

Penalized Least Square

The penalty problem can be shown to remarkably reducing bias if we use the right penalty[4]: For the penalized problem with solution $\hat{f}$, we can show

$$\left\{ \int \mathbb{E}\hat{f}(t) - f^*(t) \, \mathrm{d}t \right\}^2 \leq \lambda L | f|^2_{\mathcal{H}},$$

where f^* is the real function of the model

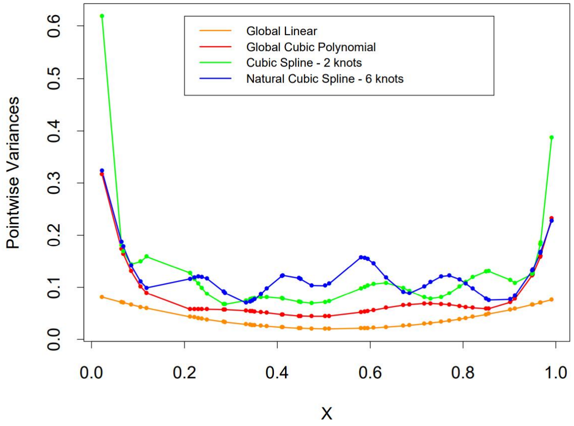
$$Y = f^*(X) + \epsilon.$$

With this, we can introduce the natural splines.

Natural Splines

- The behavior of polynomials fit to data tends to be erratic near the boundaries, and extrapolation can be dangerous.
- These problems are exacerbated with splines. The polynomials fit beyond the boundary knots behave even more wildly than the corresponding global polynomials in that region.
- We give a picture of the point-wise variances for a variety of different models. The explosion of the variance near the boundaries is clear, and inevitably is worst for cubic splines.

Pictures of point-wise variance curves



Natural Splines

- A natural cubic spline adds additional constraints, namely that the function is linear beyond the boundary knots.
- This frees up four degrees of freedom (two constraints each in both boundary regions), which can be spent more profitably by sprinkling more knots in the interior region.
- There will be a price paid in bias near the boundaries, but assuming the function is linear near the boundaries (where we have less information anyway) is often considered reasonable.

Basis of Natural Cubic Splines

- A natural cubic spline with K knots is represented by K basis functions.
- One can start from a basis for cubic splines, and derive the reduced basis by imposing the boundary constraints.
- For example, starting from the truncated power series basis described above, we arrive at

$$N_1(X) = 1,\ N_2(X) = X,\ N_{k+2}(X) = d_k(X) - d_{K-1}(X),$$

where

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

- Each of these basis functions can be seen to have zero second and third derivative for $X \geq \xi_K$.

Smoothing Splines

- Here we discuss a spline basis method that avoids the knot selection problem completely by using a maximal set of knots.
- Consider the following problem: among all functions $f(x)$ with two continuous derivatives, find one that minimizes

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

  where λ is a fixed smoothing parameter
- Notice that this criterion is defined on an infinite-dimensional function space in fact, a Sobolev space of functions for which the second term is defined.
- Remarkably, it can be shown that this has an explicit, finite-dimensional, unique minimizer which is a natural cubic spline with knots at the unique values of the x_i, $i = 1, \dots, N$.

Estimation of Penalized Splines

- Since the solution is a natural spline, we can write it as

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

  where the $N_j(x)$ are an N-dimensional set of basis functions for representing this family of natural splines.

- The penalized criterion thus reduces to

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

  where $\{\mathbf{N}\}_{ij} = N_j(x_i)$ and $\{\mathbf{\Omega}_N\}_{jk} = \int N_j''(t)N_k''(t)dt$.

Estimation of Penalized Splines

- The solution is easily seen to be

$$\hat{\theta} = \left(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N\right)^{-1}\mathbf{N}^T\mathbf{y},$$

  a generalized ridge regression
- The fitted smoothing spline is given by

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

P-spline

- The integral of the square of the second derivative of a fitted function has become common as a smoothness penalty.
- However, there is nothing special about the second derivative; in fact, lower or higher orders might be used as well which makes the reduction of the bias hopeless[1].
- A new spline, called the penalty B-spline, basing the penalty on (higher-order) finite differences of the coefficients of adjacent B-splines has been proposed:

$$\sum_{i=1}^{m}\left\{y_i - \sum_{j=1}^{n} a_j B_j(x_i)\right\}^2 + \lambda \sum_{j=k+1}^{n}\left(\Delta^k a_j\right)^2.$$

where Δ is a difference operator.

P-spline

- In practice, we proposed to use a relatively large number of knots.
- To prevent overfitting, a penalty on the second derivative restricts the flexibility of the fitted curve, which is achieved via the difference penalty on the coefficients themselves of adjacent B-splines.

Selection of the Smoothing Parameters

- To judge the goodness of our fit, we give an integrated squared prediction error (EPE) which combines both bias and variance in a single summary:

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

- Since we don't know the true function, we do not have access to EPE, and need an estimate.Techniques such as K-fold cross-validation, GCV and C_p are all in common use.

Selection of the Smoothing Parameters

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

which can (remarkably) be computed for each value of λ from the original fitted values and the diagonal elements $S_\lambda(i,i)$ of $\mathbf{S}_\lambda$, which is defined as

$$
\begin{aligned}
\hat{\mathbf{f}} &= \mathbf{N}\left(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N\right)^{-1}\mathbf{N}^T\mathbf{y} \\
&= \mathbf{S}_\lambda\mathbf{y}
\end{aligned}
$$

Selection of the Smoothing Parameters

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

is called generalized cross-validation and is another way for us to choose the optimal tuning parameter.

- In practice, we can use CV, GCV and others. GCV is usually preferable since it smooths the trace of projection operator simultaneously.
- All of these ways can be applied to any generalized penalty least square problem with the fitting value of y is a linear transformation of y.

Multidimensional Splines

- So far we have focused on one-dimensional spline models. Each of the approaches have multidimensional analogs.
- Suppose $X \in \mathbb{R}^2$, and we have a basis of functions $h_{1k}(X_1)$, $k = 1, \dots, M_1$ for representing functions of coordinate X_1, and likewise a set of M_2 functions $h_{2k}(X_2)$ for coordinate X_2. Then the $M_1 \times M_2$ dimensional tensor product basis defined by

  $$g_{jk}(X) = h_{1j}(X_1)\, h_{2k}(X_2)\,, j = 1, \dots, M_1, k = 1, \dots, M_2$$

  can be used for representing a two-dimensional function:

  $$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk} g_{jk}(X)$$

Thin-plate splines

- The above coefficients can be fit by least squares, as before. This can be generalized to d dimensions, but note that the dimension of the basis grows exponentially fast.
- Another way for multidimensional splines comes from this: one-dimensional smoothing splines (via regularization) generalize to higher dimensions as well. Suppose we have pairs y_i, x_i with $x_i \in \mathbb{R}^d$, and we seek a d-dimensional regression function $f(x)$. The idea is to set up the problem

$$\min_f \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda J[f],$$

where J is an appropriate penalty functional for stabilizing a function f in $\mathbb{R}^d$.

Thin-plate splines

- A natural generalization of the one-dimensional roughness penalty for functions on $\mathbb{R}^2$ is

$$J[f] = \iint_{\mathbb{R}^2} \left[\left(\frac{\partial^2 f(x)}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(x)}{\partial x_2^2} \right)^2 \right] dx_1 dx_2.$$

- Optimizing this penalty

$$\min_f \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda J[f],$$

leads to a smooth two-dimensional surface, known as a thin-plate spline

Thin-plate splines

- The solution has the form

$$f(x) = \beta_0 + \beta^T x + \sum_{j=1}^{N} \alpha_j h_j(x)$$

  where $h_j(x) = \|x - x_j\|^2 \log \|x - x_j\|$.
- The coefficients are found by plugging this into penalty, which reduces to a finite-dimensional penalized least squares problem.
- For the penalty to be finite, the coefficients α_i have to satisfy a set of linear constraints.
- Thin-plate splines are defined more generally for arbitrary dimension d, for which an appropriately more general J is used.

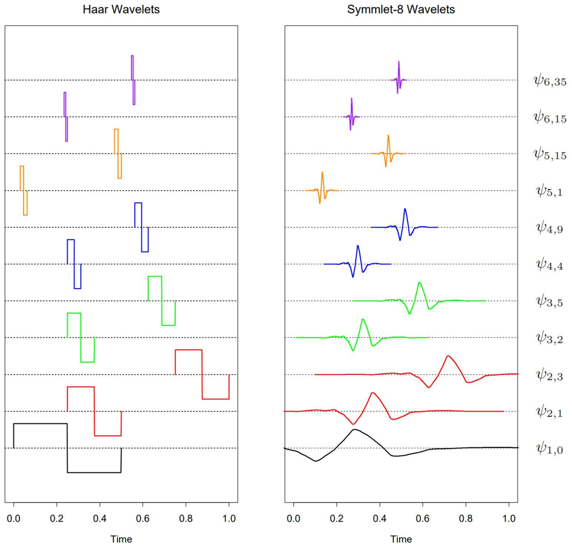
Wavelet Smoothing

We have seen two different modes of operation with dictionaries of basis functions

- With regression splines, we select a subset of the bases, using either subject-matter knowledge, or else automatically
- With smoothing splines, we use a complete basis, but then shrink the coefficients toward smoothness

Wavelets, on the other hand, typically use a complete orthonormal basis to represent functions, but then shrink and select the coefficients toward a sparse representation.
Before we give details, let's look at the Haar wavelets in the left panel of following pictures to get an intuitive idea of how wavelet smoothing works.

Pictures of Wavelet



Wavelet Smoothing

- 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
- Just as a smooth function can be represented by a few spline basis functions, a mostly flat function with a few isolated bumps can be represented with a few (bumpy) basis functions
- Wavelets bases are very popular in signal processing and compression, since they are able to represent both smooth and/or locally bumpy functions in an efficient way—a phenomenon dubbed time and frequency localization

Wavelet Smoothing

- Wavelet smoothing fits the coefficients for this basis by least squares, and then thresholds (discards, filters) the smaller coefficients
- Since there are many basis functions at each scale, it can use bases where it needs them and discard the ones it does not need, to achieve time and frequency localization
- The Haar wavelets are simple to understand, but not smooth enough for most purposes. The symmlet wavelets in the right panel of picture above have the same orthonormal properties, but are smoother.

Wavelet Smoother

- We assume $d = 1$. 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

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

Wavelet Smoother

- 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) :

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

Wavelet Smoother

- 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.

Table of Contents

Introduction

- In this chapter we describe a class of regression techniques that achieve flexibility in estimating the regression function by fitting a model separately at each point x_0.
- By using those observations close to the target point x_0 to fit the model, the resulting estimated function $\hat{f}(X)$is smooth in $\mathbb{R}^p$.
- This localization is achieved via a weighting function or kernel $K_\lambda(x_0, x_i)$, which assigns a weight to x_i based on its distance from x_0.

Introduction

Typically we are interested in the regression function

$$f(x) = E[Y|X = x].$$

In addition we may be willing to impose some smoothness constraints on this unknown $f(x)$ and an error model for the n pairs, $Y_i = f(x_i) + \epsilon_i, i = 1, ..., n$.
The conventional approach is to treat the ϵ_i as independent and identically distributed.

K–Nearest-Neighbor

We first introduce a method that does not work so well. We may use the k–nearest-neighbor average to estimate $f(x)$.

- Denote $N_k(x)$ as the k points nearest to x in squared distance.
- The idea is to relax the definition of conditional expectation, we compute an average in a neighborhood of the target point.
- The estimation is then defined as $\hat{f}(x) = \mathrm{mean}(y_i | x_i \in N_k(x))$.
- From the definition its clear to see that the estimated function is not smooth, since $N_k(x)$ only varies when encountering different x_i.

K–Nearest-Neighbor

In this chapter, we always take $Y = \sin(4X) + \epsilon$, $X \sim \mathrm{U}[0, 1]$, and $\epsilon \sim N(0, 1/3)$ as a example. The green curve is the result of a 30-nearest-neighbor mean smoother.

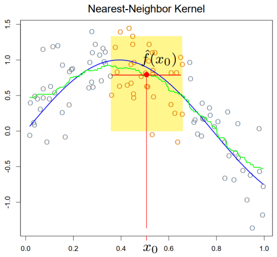


Figure: K–Nearest-Neighbor.

Indeed,the simulation shows that the estimated curve is not smooth.

Nadaraya-Wayson Kernel Smoothing

- We expect the curve to be smooth. A way to improve this method is to add a "fade-in,fade-out" feature for the points. Assigning lower weights to points that are about to exit or just entered the set can improve the smoothness.
- Function K is called a kernel if $K : R \to R$ is an integrable function satisfying $\int K(u)du = 1$. Examples:
  - $K(u) = \sqrt{2\pi} exp(u^2/2)$ (Gaussian kernel).
  - $K(u) = 3/4(1 - u^2)I(|u| \leq 1)$ (Epanechnikov kernel).
  - For a kernel K, define $K_\lambda(x_0, x) = K(\frac{|x-x_0|}{\lambda})$. Parameter λ is called window width.$K_\lambda(x_0, x)$ is a weight function centered at x_0.

Concepts

Already having a kernel, a reasonable way to add the data up according to the weight kernel is the following so called Nadaraya-Watson kernel-weighted average:

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

It is actually the weighted average.

Kernel method

Now we estimate $f(x)$ using NW average and Epanechnikov kernel, taking 0.2 for λ.

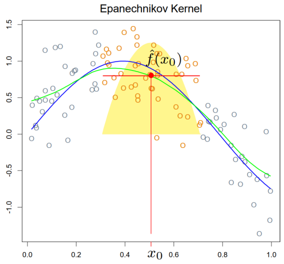


Figure: Epanechnikov kernel

Kernel method

Still, the kernel method does not perfect well near the boundary. This is because the kernel extends to outer area where no data exist, the kernel becomes asymmetric.

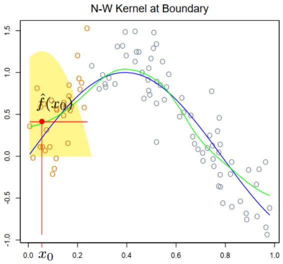


Figure: The flipping at boundaries.

Local linear method

The idea here is to use the Taylor expansion of f:

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0).$$

The problem, at x_0, written in formula is

$$(\alpha(x_0), \beta(x_0)) = \text{argmin}_{\alpha,\beta} \Sigma_{i=1}^{N} K_\lambda(x_i, x_0)[y_i - \alpha - \beta(x_i - x_0)]^2.$$

Obviously, $\alpha(x_0), \beta(x_0)$ are the coefficients of our linear estimation. Thus, our estimation can be written as $\hat{f}(x_0) = \alpha(x_0)$.

Local linear method

It's also equivalent that

$$(\alpha(x_0), \beta(x_0)) = \text{argmin}_{\alpha,\beta} \Sigma_{i=1}^{N} K_\lambda(x_i, x_0)[y_i - \alpha - \beta x_i]^2.$$

Now, our estimation can be written as $\hat{f}(x_0) = \alpha(x_0) + \beta(x_0)x_0$.

Local linear method

Using basic algebra, we can solve the minimizing problem. Define the vector-valued function $b(x)^T = (1, x)$.
Let B be the $N \times 2$ regression matrix with ith row $b(x_i)^T$, and $W(x_0)$ the $N \times N$ diagonal matrix with ith diagonal element $K_\lambda(x_0, x_i)$ and write the explicit expression for the local linear regression as

$$\hat{f}(x_0) = b(x_0)^T (B^T W(x_0) B)^{-1} B^T W(x_0) y = \Sigma_{i=1}^N l_i(x_0) y_i,$$

where l_i is the weight applied on y_i, and is referred as to the equivalent kernel.

Local linear method

A comparison of Local linear with NW

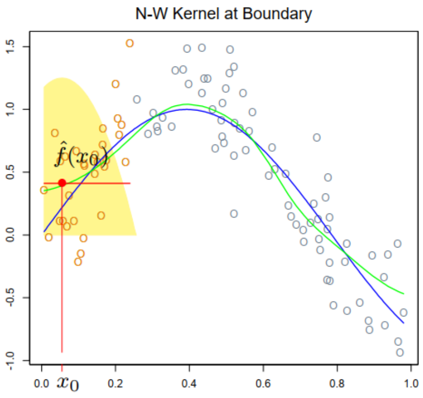


Figure: NW.

Local linear method

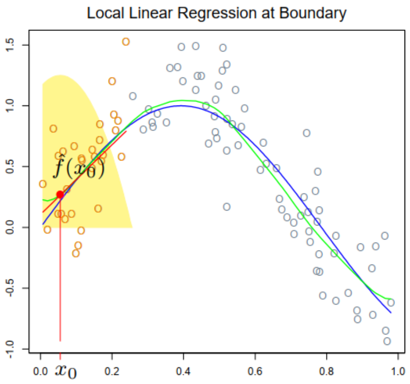


Figure: Local linear smoother.

Local linear method

Direct computation with series expansion indicates that

$$\begin{aligned} E\hat{f}(x_0) &= \Sigma_{i=1}^N l_i(x_0) f(x_i) \\ &= f(x_0)\Sigma_{i=1}^N l_i(x_0) + f'(x_0)\Sigma_{i=1}^N (x_i - x_0) l_i(x_0) + R, \end{aligned}$$

where R involves second and higher-order derivatives of f, and is typically small under suitable smoothness assumptions. For local linear regression, $\Sigma_{i=1}^N l_i(x_0) = 1$ and $\Sigma_{i=1}^N (x_i - x_0) l_i(x_0) = 0$. Hence, $E\hat{f}(x_0) - f(x_0) = R$, if f has no high order derivative, R is 0.

Local linear method

Local linear fits tend to be biased in regions of curvature of the true function, since we expect best performance on first degree estimation, there has to be compromises. This phenomenon referred to as 'trimming the hills and filling the valleys'. More over we can use polynomial to any degree for regression.

Local polynomial method

Now we introduce our Local Polynomial Regression. Similarly, our regression is:

$$\beta(x_0) = \text{argmin}_{\beta} \Sigma_{i=1}^{N} K_{\lambda}(x_0, x_i)[y_i - \Sigma_{j=0}^{P} \beta_j (x_i - x_0)^j]^2.$$

The estimation is: $\hat{f}(x_0) = \beta_0(x_0)$. As an example, we know that curvature is a second degree property of a curve, we can do regression to second degree to improve curvature fitting.

Local polynomial method

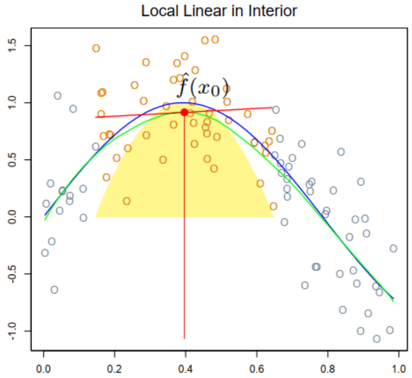


Figure: Local linear.

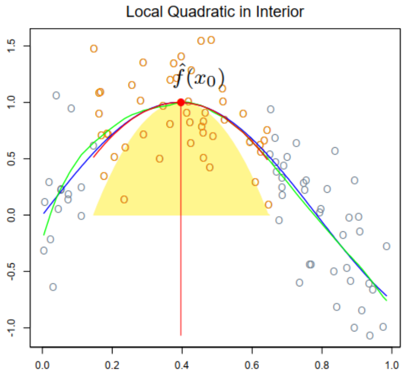


Figure: Quadratic.

Local polynomial method

There is a price to be paid for this bias reduction, and that is increased variance. Assuming the model

$$y_i = f(x_i) + \epsilon_i,$$

with ϵ_i independent and identically distributed with mean zero and variance σ^2, $Var(\hat{f}(x_0)) = \sigma^2 ||l(x_0)||^2$, where $l(x_0)$ is the vector of equivalent kernel weights at x_0. It can be shown that $||l(x_0)||$ increases with d, and so there is a bias–variance trade-off in selecting the polynomial degree.

Local polynomial method

- What if we try local linear fits at the boundary while using local quadratic fits in the interior? We do not recommend such strategies. Usually the application will dictate the degree of the fit.
- For example, if we are interested in extrapolation, then the boundary is of more interest, and local linear fits are probably more reliable. And for interpolation, we might choose the local polynomial method.

Summary

- Local linear fits can help bias dramatically at the boundaries at a modest cost in variance. Local quadratic fits do little at the boundaries for bias, but increase the variance a lot.
- Local quadratic fits tend to be most helpful in reducing bias due to curvature in the interior of the domain.
- Asymptotic analysis suggest that local polynomials of odd degree dominate those of even degree. This is largely due to the fact that asymptotically the MSE is dominated by boundary effects.

Bandwidth

Now we shall discuss about selecting the width of the kernel.
Examples:

- $K_\lambda(x_0, x) = K(\frac{|x-x_0|}{\lambda})$ (General formula).
- $K_\lambda(x_0, x) = 3/4(1 - \frac{|x-x_0|}{\lambda}^2)I(\frac{|x-x_0|}{\lambda} \leq 1)$ (Epanechnikov kernel). For Epanechnikov kernel , λ is the radius of the support region.
- $K_\lambda(x_0, x) = \sqrt{2\pi} exp((\frac{|x-x_0|}{\lambda})^2/2)$ (Gaussian kernel)
  For Gaussian kernel, λ is the standard deviation.

Bias-variance trade-off

- There is a natural bias–variance tradeoff as we change the width of the averaging window.
- For local averages: If the window is narrow, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0, and its variance will be relatively large, close to that of an individual y_i. The bias will tend to be small, again because each of the $E(y_i) = f(x_i)$ should be close to $f(x_0)$.
- If the window is wide, the variance of $\hat{f}(x_0)$ will be small, because of the effects of averaging. Meanwhile, the bias will be higher, because we are now using observations x_i further from x_0, and there is no guarantee that $\hat{f}(x_i)$ will be close to $f(x_0)$.

Bias-variance trade-off

- Similar arguments apply to local regression estimates, as the width goes to zero, the estimates approach a piecewise-linear function that interpolates the training data;
- As the width gets infinitely large, the fit approaches the global linear least-squares fit to the data.

CV method

Now we want to minimize

$$\text{MISE}(\hat{m}_p|x_1,...,x_n) = E[\int [\hat{m}_p(u) - m(u)]^2 f_X(u)du|x_1,...,x_n]$$

We can see that $\hat{m}_p(u)$ is a function of Kernel, window width λ, and the estimator $\hat{m}_p$ with polynomial order p. Leaving the Kernel alone, we want to find a window width λ minimize the problem. A discrete approximation to MISE is MASE,which is defined as follows,

$$\text{MASE} = E[\frac{\Sigma_i[\hat{m}_p(x_i) - m(x_i)]^2}{n}],$$

[5] where x_i is the sample point.

CV method

- Cross-validation proceeds directly to estimate MASE, using the idea: leave-one-out' prediction. It has the form

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{m}_p^{(i)}(x_i)]^2$$

  where $\hat{m}_p^{(i)}(x_i)$ is the estimate base on the data with x_i removed and evaluated at x_i.
- By minimizing the function, we have our chosen λ.

Asymptotic result

Another method to select the bandwidth is on basis of the asymptotic theory. Here [2]

$$\lambda = \left[\frac{(p+1)(p!)^2 R\left(K_{(p)}\right) \sigma^2}{2n\mu_{p+1}\left(K_{(p)}\right)^2 \int m^{(p+1)}(u)^2 f_X(u)du} \right]^{1/(2p+3)}$$

minimizes the MISE. But

- Unsuitable bandwidth would lead to over-fitting, or under-fitting, and poor convergence rate for estimated function.
- The calculation of λ depends on the unknown identity of data itself.
- λ decreases as $\sigma^2 \downarrow$, and $n \uparrow$.

Plug-in estimator

- Instead, we can first pick a appropriate λ.
- Plugging into our problem we have a estimate $\hat{m}_p$, then we swap the original $\hat{m}_p$ with the new one, and compute a minimal h for the new problem based on the asymptotic result, which gives a new λ.
- By plugging in our new λ we start a new iteration.
- As the algorithm converges, we have our chosen λ.

Higher dimensions

Now we extend the local regression method to higher dimensions.

- Let $b(X)$ be a vector of polynomial terms in $X \in \mathbb{R}^p$ of maximum degree d.
- To explain this, given $d = 1$ and $p = 2$, we get $b(X) = (1, X_1, X_2)$.
- Given $d = 2$ we get $b(X) = (1, X_1, X_2, X_{12}, X_{22}, X_1X_2)$;
- And trivially with $d = 0$, we get $b(X) = 1$.

Higher dimensions

- At each $x_0 \in \mathbb{R}^p$ solve

$$\min_{\beta(x_0)} \sum_{i=1}^{N} K_\lambda(x_0, x_i) \left(y_i - b(x_i)^T \beta(x_0) \right)^2$$

  to produce the fit $\hat{f}(x_0) = b(x_0)^T \hat{\beta}(x_0)$.
- Typically the kernel will be

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

  where $\|\cdot\|$ is the Euclidean norm.

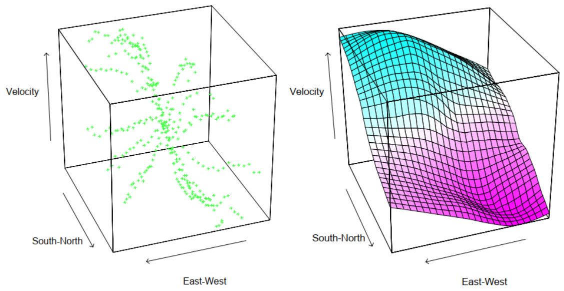


Figure: A picture of some velocity data

Local regression becomes less useful in dimensions much higher than two or three, since the area of the boundary would become larger and larger. Also, the smoothness condition would not be useful constraint for high dimensional function space.

Higher dimensions

- It is impossible to simultaneously maintain localness which leads to low bias and a sizable sample in the neighborhood which leads to low variance as the dimension increases, without the total sample size increasing exponentially in p.
- Visualization of $\hat{f}(X)$ also becomes difficult in higher dimensions, and this is often one of the primary goals of smoothing. Figure shows an analysis of some environmental data with three predictors.

References I

[1] Paul HC Eilers and Brian D Marx. “Flexible smoothing with B-splines and penalties”. In: *Statistical science* 11.2 (1996), pp. 89–121.

[2] J Fan. and I. Gijbels. *Local polynomial modelling and its applications*. Chapman Hall, 1996.

[3] Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[4] Tailen Hsing and Randall Eubank. *Theoretical foundations of functional data analysis, with an introduction to linear operators*. Vol. 997. John Wiley & Sons, 2015.

[5] Jeffrey S. s=Simonoff. *Smoothing Methods in Statistics*. Springer, 1996.