STATS 205: Introduction to Nonparametric Statistics

Lecturer: Tengyu Ma
Scribe: Paula Gablenz & Zheng Yan

Lecture # 3
October 11th, 2019

1 Review and Motivation

1.1 Linear Smoothers

As a review, let's take another look at **linear smoothers**. A linear smoother is any estimator $\hat{r}(x)$ that, for any x in our set of inputs, can be expressed in the form:

$$\hat{r}(x) = \sum_{i=1}^{n} l_i(x) Y_i \tag{1}$$

Intuitively, each x_i corresponds with a vector $l(x_i)$, which can depend on x_i and represents the weights given to each output $[y_1, y_2, ..y_n]$ when our estimator is "de-noising" y_i. All effective local linear smoothers do some variation of smooth out any particular output through taking some "pieces" from the other outputs. It's finding the function l(x), or how big of a piece to take from each output, that's the tricky part.

Our final de-noised outputs can be represented as LY, where Y is a vector containing our noisy outputs $[y_1, y_2, ..y_n]$, and the matrix L contains the columns $[l(x_1), l(x_2)..l(x_n)]$. L is known as our **smoothing matrix**.

We've covered these linear smoothers in the previous lectures:

1. **Regressogram**
2. **Local averaging kernel**
3. **Local linear regression/polynomial regression**

1.2 The Bias Problem and Motivations

So far, local linear/polynomial regression have been the best methods for approximating more complex functions, through having relatively low theoretical bias.

Recall that the bias of local linear regression can be written as follows:

$$Bias = \frac{h_n^4}{2}\left(\int x^2 K(x)dx\right)^2 \int \left(r''(x)\right)^2 dx \tag{2}$$

Notably, let's we look at the term:

$$\int r''(x)^2 dx$$

Intuitively, this term is high if our **true, de-noised functions grows or shrinks erratically or non-linearly**. This is because the second derivative of r(x) represents how "quickly" the slope changes. In our bias equation, we square and take its integral, which sums up the magnitude of the true function's slope change throughout the interval we care about.

Given a dataset produced by r(x) as a noise-less linear function, our local linear regression would be able to approximate it perfectly. However, if our $r(x)$ was, say, x^5, we would have significant error regardless of noisiness. The less constant our slope is, the worse our estimator is.

With the above considerations, we know we will **only use local linear regression for when we believe the true function $r(x)$ is relatively smooth**. Note that we were previously implicitly assuming that $r''(x)$ is small, since the success of the method depends on having a (small) second derivative. This lecture discusses a direct approach to find functions with small second order derivatives.

2 Penalized Regression: Overview

The goal is to find a smooth function that fits the data well. For anyone who has done any sort of AI/ML, we know that we can often smooth out estimators and avoid overfitting by adding a **regularization term**. We define the objective as follows:

$$L_\lambda(\hat{r}) \triangleq \min_{\hat{r}} \sum_{i=1}^{N} (Y_i - \hat{r}(x_i))^2 + \lambda J(\hat{r}), \tag{3}$$

where $J(\hat{r}) = \int (\hat{r}''(x))^2 dx$. To minimize the complexity of our estimator, we use the metric we established above for "something that goes up when functions are less smooth".

To parse this better, we consider two extreme cases.

1. $\lambda = 0$: If $\lambda = 0$, then the solution is the interpolating function. We output $\hat{r}$ such that $\hat{r}(x_i) = Y_i$. Thus, we are not denoising the data and likely to overfit. Note that this is similar to having bandwidth $h = 0$.
2. $\lambda \to \infty$: If $\lambda \to \infty$, then $\hat{r}$ converges to the least squares line. This is because $\lambda \to \infty$ implies $J(\hat{r}) \to 0$. From there we see that $\hat{r}''(x) \to 0\ \forall x$. This implies that $\hat{r}$ converges to a linear function. Note that this extreme case is similar to having a bandwidth $h = \infty$. Recall that for local linear regression with boxcar kernel we obtained $\sum_{i=1}^{N} w_i(Y_i - ax_i + b)$. When we just fit the standard MSE loss with linear regression, all $w_i = 1$.

Thus, we see that there is a trade-off with regard to the regularization parameter λ.

Equation (3) sounds like a very hard optimization problems, because we are optimizing over all functions. However, the following theorem finds that the problem actually does have some structure so that we can simplify the optimization in $L_\lambda(\hat{r})$.

Theorem 1. *The function $\hat{r}$ that solves the optimization problem $L_\lambda(\hat{r})$ is a natural cubic spline with knots at the data points. This estimator $\hat{r}$ is called a smoothing spline.*

As we will see, we can prove that the theoretically best solution/estimator for this loss function, for any set inputs/outputs, must be a **natural cubic spline**. By proving that our solution is a certain type of function, we can simplify our optimization process to look for only these functions instead.

Before we can proceed to proof the theorem in section 5 and to show how to find the optimal spline in section 6, we need some background on the definitions (section 3) and properties of splines (section 4).

3 Definitions of Splines

Definition 2 (**Knots**). *Let $\xi_1 < \xi_2 < ... < \xi_k$ be a set of points called knots. These points are contained in some interval (a, b).*

Definition 3 (**Cubic Splines**). *A cubic spline q is a continuous function such that:*

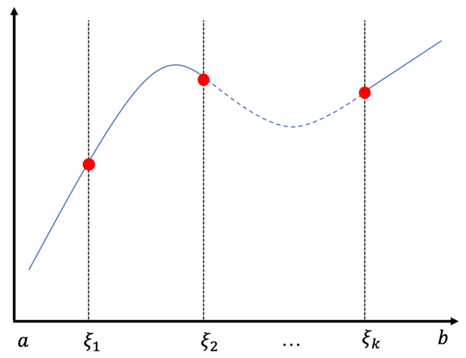
1. *q is a cubic polynomial over $(a, \xi_1), (\xi_1, \xi_2), ..., (\xi_{k-1}, \xi_k), (\xi_k, b)$.*
2. *q has continuous first and second order derivatives.*

More generally, an ***M^{th}-order spline*** *is a piecewise $M - 1$ degree polynomial with $M - 2$ continuous derivatives at the knots.*

Definition 4 (**Natural Splines**). *A natural spline is a spline that is linear beyond the boundary knots, i.e. in (a, ξ_1) and (ξ_k, b).*

Proposition 5 (**Representation of Cubic Splines**). *A cubic spline $q(x)$ can be represented as $q(x) = a_{3,i}x^3 + a_{2,i}x^2 + a_{1,i}x + a_{0,i}$, which holds $\forall x \in [\xi_i, \xi_{i+1}]$ with $i = \{0, ..., k\}$, where $\xi_0 = a$ and $\xi_{k+1} = b$.*

Figure 1 below shows a sketch of cubic splines, using the same notation as defined above.



Cubic polynomials between each of the knots ξ_i, $i \in \{1, ..., k\}$.

Figure 1: Sketch of Splines

Following from definition 3, cubic splines correspond to $M = 4$. As a remark, this whole lecture uses cubic splines.

4 Properties of Splines

In this section, we will consider the following Lemma:

Lemma 6. *All cubic splines with fixed knots $\xi_1, ..., \xi_k$ form a $(k+4)$ dimensional subspace of functions.*

More specifically, we will find a $(k+4)$-dimensional basis for the space of all cubic splines (in Lemma 6). Before this, we need some preliminary definitions for subspaces, basis and dimensionality in section 4.1. Lemma 6 will be verified in section 4.2.

4.1 Subspaces and Dimensionality

Definition 7 (Subspace and Basis of Vectors). *Consider a set S such that $v \in S$, $w \in S \implies \lambda_1 v + \lambda_2 w \in S$, $\forall\ \lambda_1, \lambda_2 \in \mathbb{R}$. If S is a subspace, then $\exists$ a **basis** $b_1, ..., b_m$ such that $\forall v \in S$, $v = \sum_{i=1}^{m} \beta_i b_i$ with $i = \{1, ..., m\}$ (i.e. every vector is a linear combination of the basis).*

Intuitively, we can also think of functions as vectors. For example, we can have a function $f : [0,1] \to \mathbb{R}$. Then, we might discretize the interval $[0,1]$ into $0, \frac{1}{n}, \frac{2}{n}, ..., \frac{n}{n}$ and put all the values of the function into an n-dimensional vector G.

$$G = \begin{bmatrix} f(0) \\ f(\frac{1}{n}) \\ \vdots \\ f(\frac{n}{n}) \end{bmatrix}$$

Now, we can define a subspace of functions.

Definition 8 (Subspace of Functions). *A set of functions $\mathscr{F}$ is a subspace if $\forall\ f, g \in \mathscr{F}$, $\forall\ \lambda_1, \lambda_2 \in \mathbb{R}$, $\lambda_1 f + \lambda_2 g \in \mathscr{F}$.*

Definition 9 (Dimensionality). *A subspace of functions $\mathscr{F}$ has dimensionality of at most k if there exists a basis $f_1, ..., f_k \in \mathscr{F}$ such that $\forall f \in \mathscr{F}$, f can be represented as $f = \beta_1 f_1 + \beta_2 f_2 + ... + \beta_k f_k$.*

The dimensionality of $\mathscr{F}$ is the minimum k such that there exists a basis satisfying $\forall f \in \mathscr{F}$, $f = \sum_i \beta_i f_i$.

As an example for definition 9, consider $\mathscr{F} = \{f : f(x) = ax^2 + bx + c; a, b, c \in \mathbb{R}\}$. We see that the bases $f_1(x) = x^2$, $f_2(x) = x$ and $f_3(x) = 1$ span the entire subspace. Thus, $\forall f \in \mathscr{F}$, $f = af_2 + bf_1 + cf_3$. Since we have three functions that can span the entire subspace, we get that $dim(\mathscr{F}) \leq 3$.

Now that we have defined subspaces, basis and dimensionality, we can proceed to verify Lemma 6.

4.2 Finding a Basis for Cubic Splines

In order to verify lemma 6 we will check that the set of cubic splines is a subspace in section 4.2.1 and find the actual basis for the subspace in section 4.2.2.

4.2.1 Check that the set of cubic splines is a subspace

We want to check that the set of cubic splines with knots $\xi_1, ..., \xi_k$ is a subspace. Suppose we have a cubic spline f and another cubic spline g. Then, $f + g$ will also be a cubic spline. This is because both f and g, and therefore also $f + g$, are degree-3 polynomials.

4.2.2 Check dimensionality and find a basis

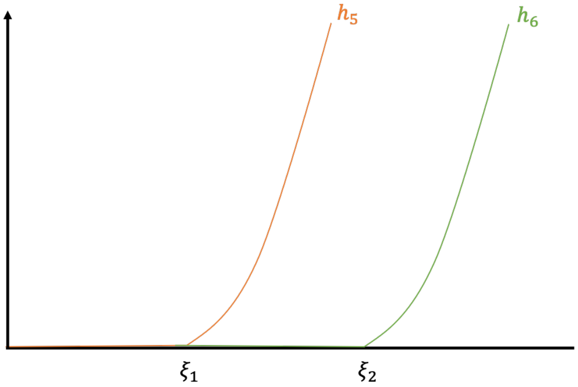
Next, we need to check the dimensionality. The dimensionality needs to be at most $k + 4$. Suppose there exists a set of functions $h_1, ..., h_{k+4}$ such that for all cubic splines f, $f = \sum_{i=1}^{k+4} \beta_i h_i$.

Definition 10. *Let this set of functions $h_1, ..., h_{k+4}$ be defined as follows:*

$$\begin{gathered} h_1(x) = 1 \\ h_2(x) = x \\ h_3(x) = x^2 \\ h_4(x) = x^3 \\ h_{i+4}(x) = (x - \xi_i)_+^3 \textit{ with } i = 1, ..., k \end{gathered}$$

Here, the notation with the plus-sign in the subscript means $[x]_+ = \max\{x, 0\}$.

As an example, figure 2 below shows a sketch of $h_5(x - \xi_1)_+^3$ and $h_6(x - \xi_2)_+^3$.



$h_5(x-\xi_1)^3_+ = 0 \ \forall x \leq \xi_1$ and $h_6(x-\xi_2)^3_+ = 0 \ \forall x \leq \xi_2$.

Figure 2: Sketch of $h_5(x-\xi_1)^3_+$ and $h_6(x-\xi_2)^3_+$

Claim 11 (**Representation of Cubic Splines**)**.** *Any cubic spline $f(x)$ can be represented as a linear combination of the functions $h_1(x)$ to $h_{i+4}(x)$, with $i = \{1, ..., k\}$ such that $f(x) = \sum_{i=1}^{k+4} \beta_i h_i(x)$.*

Proof of claim 11. The proof is conducted in two steps.

First, all h_i are splines and h_i is in the set of cubic splines, $\forall i = \{1, ..., k\}$ by construction.

Second, we proceed via induction on k.

Base Case $k = 0$**:** Since there are no knots, it is just a cubic polynomial. Any cubic polynomial is the sum of 4 functions: $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. Thus, the claim is valid for $k = 0$.

Now, consider the different base case $k = 1$.

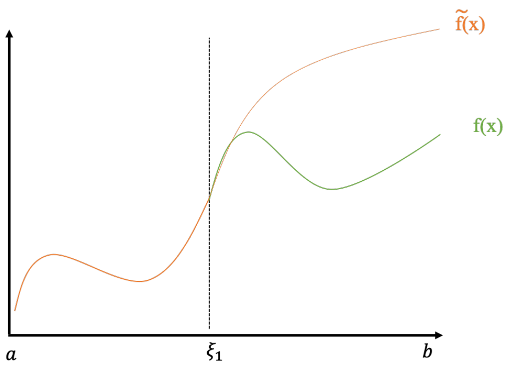
Base Case $k = 1$**:** Let $f(x)$ be a cubic spline with knot ξ_1. Then, there exist a_3, a_2, a_1, a_0 such that $\forall x, x \leq \xi_1$, $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0 = a_3h_4 + a_2h_3 + a_1h_2 + a_0h_1$.

To show this, consider the function

$$\widetilde{f}(x) = a_3x^3 + a_2x^2 + a_1x + a_0, \text{where } x \in [a, b] \tag{4}$$

This function $\widetilde{f}(x)$ is a degree-3 polynomial. We define it in a way such that it agrees with $f(x)$ on the left hand side of ξ_1 such that $f(x) = \widetilde{f}(x)$ when $x \leq \xi_1$.

Figure 3 below shows a sketch of what $\widetilde{f}(x)$ and $f(x)$ could look like.



$\widetilde{f}(x)$ and $f(x)$ agree before ξ_1 and differ after.

Figure 3: Sketch of $\widetilde{f}(x)$ and $f(x)$

We now consider the difference between $f(x)$ and $\widetilde{f}(x)$. This difference has the following properties:

1. $f(x) - \widetilde{f}(x) = 0$ on $[a, \xi_1]$
2. $f(x) - \widetilde{f}(x)$ is a cubic polynomial on $[\xi_1, b]$

Let's examine $f(x) - \widetilde{f}(x)$ by first looking at the interval $[\xi_1, b]$ and then trying to find a global extension.

1. Thus, first, let's restrict the attention to the interval $[\xi_1, b]$. The difference between $f(x)$ and $\widetilde{f}(x)$ and thus be written as

   $$f(x) - \widetilde{f}(x) = b_3(x - \xi_1)^3 + b_2(x - \xi_1)^2 + b_1(x - \xi_1) + b_0 \tag{5}$$

   Note that this is permissible since it is intuitively just a shift in the coordinate system. To further clarify why this holds, suppose g is a cubic polynomial. Then $g(x + \xi_1) = b_3x^3 + b_2x^2 + b_1x + b_0$ is also a cubic polynomial. Rearranging yields that $g(x) = b_3(x - \xi_1)^3 + b_2(x - \xi_1)^2 + b_1(x - \xi_1) + b_0$.

2. Second, let's find a global extension since we are not only interested in the interval $[\xi_1, b]$. We can fix this by letting

   $$f(x) - \widetilde{f}(x) = b_3(x - \xi_1)_+^3 + b_2(x - \xi_1)_+^2 + b_1(x - \xi_1)_+ + b_0, \forall x \in (a, b) \tag{6}$$

   When then consider the following two cases:

   (a) If $x \geq \xi_1$, then by definition of the $[x]_+ = \max\{x, 0\}$ notation, the $[x]_+$ subscripts in above formula disappear.

   (b) If $x \leq \xi_1$, we get $f(x) - \widetilde{f}(x) = 0$ on the left hand side of ξ_1 by definition of $\widetilde{f}(x)$. On the ride hand side, due to the $[x]_+$ notation, we get $f(x) - \widetilde{f}(x) = b_0$.

Next, note that we can rewrite equation 6 as follows:

$$f(x) = \widetilde{f}(x) + b_3(x-\xi_1)_+^3 + b_2(x-\xi_1)_+^2 + b_1(x-\xi_1)_+ + b_0,\ \forall x \in [a,b] \tag{7}$$

We now need to show that $b_2 = b_1 = b_0 = 0$. Note that we haven't used the continuity at the knot yet. This continuity assumption is going to be the main idea in showing that $b_2 = b_1 = b_0 = 0$.

Thus, let's consider the derivative at the left and right hand sides of the knot. Since $f(x)$ and $\widetilde{f}'(x)$ agree on the left hand side, we get $f(x) - \widetilde{f}(x) = 0$ if $x \leq \xi_1$. Thus, we get that $f'(\xi_1) - \widetilde{f}'(\xi_1) = 0$ and also that $f''(\xi_1) - \widetilde{f''}(\xi_1) = 0$. Now, since the derivative is continuous at the knot, we also get $f'(\xi_1)_+ - \widetilde{f}'(x)_+ = 0$ and therefore $f''(\xi_1)_+ - \widetilde{f''}(x)_+ = 0$.

Now, we can use the above argument to get that $b_2 = b_1 = b_0 = 0$.

First, taking first derivatives we get that $b_1 = 0$ since

$$\begin{aligned}
\left(b_3(x-\xi_1)_+^3 + b_2(x-\xi_1)_+^2 + b_1(x-\xi_1)_+ + b_0\right)'\Big|_{+,\xi_1} &= 0 \\
\implies \left(3b_3(x-\xi_1)^2 + 2b_2(x-\xi_1) + b_1\right)\Big|_{\xi_1} &= 0 \\
\implies b_1 &= 0
\end{aligned}$$

Next, using the second derivatives we get that $b_2 = 0$.

$$\begin{aligned}
\left(b_3(x-\xi_1)_+^3 + b_2(x-\xi_1)_+^2 + b_1(x-\xi_1)_+ + b_0\right)''\Big|_{+,\xi_1} &= 0 \\
\implies \left(6b_3(x-\xi_1) + 2b_2\right)\Big|_{\xi_1} &= 0 \\
\implies b_2 &= 0
\end{aligned}$$

Then, since we know that it is continuous, we get

$$\begin{aligned}
\lim_{x\to\xi_1} f(x) - \widetilde{f}(x) &= 0 \\
\lim_{x\to+,\ \xi_1} f(x) - \widetilde{f}(x) &= 0 \\
\implies b_0 &= 0
\end{aligned}$$

Therefore, we have shown that $b_1 = b_2 = b_0 = 0$. Thus, from equation 7 with $b_1 = b_2 = b_0 = 0$ we get that

$$f(x) = \widetilde{f}(x) + b_3(x-\xi_1)_+^3 \tag{8}$$

Plugging in the definition of $\widetilde{f}(x)$ from equation 4, we get

$$\begin{aligned} f(x) &= a_3x^3 + a_2x^2 + a_1x + a_0 + b_3(x-\xi_1)_+^3 \\ &= a_3h_4(x) + a_2h_3(x) + a_1h_2(x) + a_0h_1(x) + b_3h_5(x) \end{aligned} \tag{9}$$

where the second line comes from definition 10.

Thus, we have shown that $f(x)$ is a linear combination of 5 functions, as it should be for $k = 1$. The result can be similarly proven for all other k.

Therefore, we conclude that every cubic spline $f(x)$ can be written as $f(x) = \sum_{i=1}^{k+4} \beta_i h_i(x)$.

□

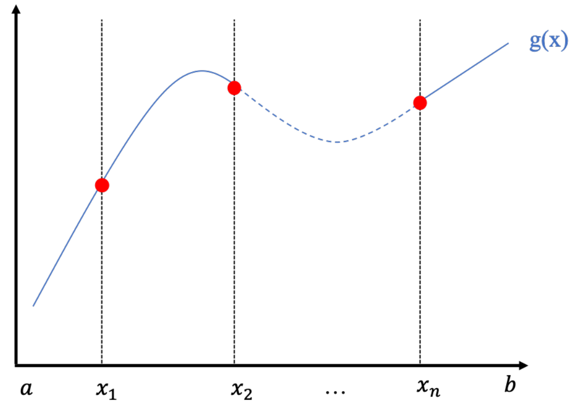
As a short remark, from definition 4 we know that a natural cubic spline extrapolates linearly at the boundaries. We can therefore also write it as a linear combination of functions. Thus, there exists $h_1, ..., h_k$ such that for every natural cubic spline $f(x)$, $f(x) = \sum_{i=1}^{k} \beta_i h_i(x)$.

Using the previous results, we can now proof theorem 1.

5 Proof of Theorem 1

The main idea of the proof is to proceed via contradiction. We will suppose that $g(x)$ is a minimizer and then argue that $g(x)$ has to be a cubic polynomial in each piece on $[\xi_i, \xi_{i+1}]$, where the knots actually are at the datapoints $x_1, ..., x_n$. Thus, it is a natural cubic spline with knots at the datapoints.

Figure 4 below shows a sketch of what $g(x)$ could look like.



$g(x)$ is a natural cubic spline with knots at $x_1, ..., x_n$.

Figure 4: Sketch of $g(x)$

Next, we proceed with the proof of theorem 1.

Proof. Suppose function g is a minimizer and is <u>not</u> a spline. Then, we can find some $\widetilde{g}$ such that equations 10, 11 and 12 are satisfied.

$$\widetilde{g}(x_i) = g(x) \ \forall i \in \{1, ..., n\} \tag{10}$$

$$J(\widetilde{g}) = \int (\widetilde{g}''(x))^2 dx < J(g) = \int (g''(x))^2 dx \tag{11}$$

Then, 10 and 11 imply

$$L_\lambda(\widetilde{g}) < L_\lambda(g), \tag{12}$$

which is a contradiction.

Now, we actually need to find such a $\widetilde{g}$ that satisfies the properties above to get the desired contradiction and to proof theorem 1.

Definition 12. *Let $\widetilde{g}$ be the natural cubic spline satisfying $\widetilde{g}(x_1) = g(x_1), ..., \widetilde{g}(x_n) = g(x_n)$. Such function exists since we have n degrees of freedom so we can satisfy n equations.*

Note that by construction of $\widetilde{g}$, $\widetilde{g}$ satisfies equation 10. We then need to show that it satisfies equation 11. To show this, define h as the difference between g and $\widetilde{g}$.

Definition 13. *Let $h = g - \widetilde{g}$. Rearranging yields $g = \widetilde{g} + h$.*

To show that condition 11 holds, let's focus on $J(g) = \int (g''(x))^2 dx$.

$$\begin{aligned} J(g) &= \int (g''(x))^2 dx \\ &= \int \Big((\widetilde{g} + h)''(x) \Big)^2 dx \\ &= \int (\widetilde{g}''(x))^2 dx + 2 \int \widetilde{g}''(x) h''(x) dx + \int (h''(x))^2 dx \end{aligned} \tag{13}$$

First, consider the term $\int (h''(x))^2 dx$. Note that $h'' \equiv 0 \implies h \equiv 0$. Thus, $g = \widetilde{g}$ and g is a cubic spline, which is a contradiction since we assumed that it is <u>not</u> a spline. Thus, unless $h'' = 0$, $\int (h''(x))^2 dx > 0$.

To proceed with the proof, let's consider the term $\int \widetilde{g}''(x) h''(x) dx$. We need to show that $\int \widetilde{g}''(x) h''(x) dx = 0$.

$$
\begin{aligned}
\int_a^b \widetilde{g}''(x)h''(x)dx &= \widetilde{g}''(x)h'(x)\Big|_a^b - \int_a^b h'(x)\widetilde{g}'''(x)dx && \text{by integration by parts} \\
&= -\int_a^b h'(x)\widetilde{g}'''(x)dx && \widetilde{g}''(a) = \widetilde{g}''(b) = 0 \text{ since it is a natural cubic spline} \\
&= -\sum_{i=0}^{k} \int_{\xi_i}^{\xi_{i+1}} h'(x)\widetilde{g}'''(x)dx && \widetilde{g}'''(x) \text{ is piecewise constant} \\
&= -\sum_{i=0}^{k} \widetilde{g}'''(\xi_i) \int_{\xi_i}^{\xi_{i+1}} h'(x)dx && \text{pull out constants, knots at datapoints} \\
&= -\sum_{i=0}^{k} \widetilde{g}'''(\xi_i)(h(\xi_{i+1}) - h(\xi_i)) && \text{by evaluation} \\
&= -\sum_{i=0}^{k} \widetilde{g}'''(\xi_i)(h(x_{i+1}) - h(x_i)) && \text{since knots are at datapoints} \\
&= -\sum_{i=0}^{k} \widetilde{g}'''(\xi_i)(0 - 0) && h(x_i) = g(x_i) - \widetilde{g}(x_i) = 0 \text{ by definitions 10 and 13} \\
&= 0 && (14)
\end{aligned}
$$

Thus, we see that that $\int \widetilde{g}''(x)h''(x)dx = 0$.

Now, going back to equation 13, we can plug the results above in to get

$$
\begin{aligned}
J(g) &= \int (g''(x))^2 dx \\
&= \int (\widetilde{g}''(x))^2 dx + 2 \times 0 + \int (h''(x))^2 dx \\
&= \int (\widetilde{g}''(x))^2 dx + \int (h''(x))^2 dx \\
&> \int (\widetilde{g}''(x))^2 dx.
\end{aligned} \tag{15}
$$

Then, following the argument above, the fulfillment of 10 and 11 implies 12, which is a contradiction.

□

6 Finding the Optimal Spline

In the previous section we saw that the function $\hat{r}$ that solves the optimization problem $L_\lambda(\hat{r})$ is a natural cubic spline with knots at the data points. Thus, we can do a simpler search

$$\arg\min_{\hat{r}} \sum_{i=1}^{N} (Y_i - \hat{r}(x_i))^2 + \lambda J(\hat{r}), \tag{16}$$

where $\hat{r}$ is a natural cubic spline.

This section first simplifies equation 16 in section 6.1 and then solves the optimization problem in section 6.2.

6.1 Simplifying Equation 16

Recall from section 4.2.2 that we can write $\hat{r}(x) = \sum_{i=1}^{n+4} \beta_i h_i(x)$. We are interested in the coefficients β_i. Let's re-write equation 16 as follows:

$$\arg\min_{\beta_1,\ldots,\beta_{n+4}} \sum_{i=1}^{N} (Y_i - \sum_{j=1}^{n+4} \beta_j h_j(x))^2 + \lambda \int \left((\sum_{j=1}^{n+4} \beta_j h_j(x))'' \right)^2 dx, \tag{17}$$

The catch here is that we actually have a quadratic function in β, so we can get an analytic solution.

First, let's simplify the term $\sum_{i=1}^{N} (Y_i - \sum_{j=1}^{n+4} \beta_j h_j(x))^2$.

Definition 14 (**Matrix Notation for Loss**)**.** *Let*

$$B = \begin{bmatrix} h_1(x_1) & \ldots & h_{n+4}(x_1) \\ \vdots & \ddots & \vdots \\ h_1(x_n) & \ldots & h_{n+4}(x_n) \end{bmatrix} \in \mathbb{R}^{n\times(n+4)},$$

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_{n+4} \end{bmatrix} \in \mathbb{R}^{(n+4)},$$

and

$$Y = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix} \in \mathbb{R}^{n}.$$

We can then write

$$Y - B\beta = \begin{bmatrix} Y_1 - \Big(\beta_1 h_1(x_1) + \beta_2 h_2(x_2) + \cdots + \beta_{n+4} h_{n+4}(x_1)\Big) \\ \vdots \\ Y_i - \Big(\beta_1 h_1(x_i) + \beta_2 h_2(x_i) + \cdots + \beta_{n+4} h_{n+4}(x_i)\Big) \\ \vdots \\ Y_n - \Big(\beta_1 h_1(x_n) + \beta_2 h_2(x_n) + \cdots + \beta_{n+4} h_{n+4}(x_n)\Big) \end{bmatrix} \in \mathbb{R}^n.$$

Now, note that

$$||Y - B\beta||_2^2 = \sum_{i=1}^{N} (Y_i - \sum_{j=1}^{n+4} \beta_j h_j(x))^2. \tag{18}$$

Next, let's simplify the term $\int \Big((\sum_{j=1}^{n+4} \beta_j h_j(x))'' \Big)^2 dx$.

$$\begin{aligned} \int \Big((\sum_{j=1}^{n+4} \beta_j h_j(x))'' \Big)^2 dx &= \int \Big((\sum_{j=1}^{n+4} \beta_j h_j''(x)) \Big)^2 dx \\ &= \int \Big(\sum_{j,\ k=1}^{n+4} \beta_j \beta_k h_j''(x) h_k''(x) \Big) dx \\ &= \sum_{j=1,\ k=1}^{n+4} \beta_j \beta_k \Big(\int h_j''(x) h_k''(x) dx \Big) \\ &= \sum_{j=1,\ k=1}^{n+4} \beta_j \beta_k \Omega_{j,k} \\ &= \beta^T \Omega \beta, \end{aligned} \tag{19}$$

where Ω is a matrix with elements $\Omega_{j,k}$. Note that $\Omega_{j,k}$ only depends on the basis and thus behaves like a constant and can be computed in advance.

Combining results from equations 18 and 19, we can re-state the objective in 17 as follows:

$$L_\lambda(\beta) \triangleq \arg\min_\beta ||Y - B\beta||_2^2 - \lambda \beta^T \Omega \beta \tag{20}$$

Using this simplified notation, we can go on to solve the problem.

6.2 Solving Problem 20

From 20, we see that we need to solve

$$\nabla L_\lambda(\beta) = 0. \tag{21}$$

Thus, taking the derivative and setting it to zero we get

$$\begin{aligned} -2B^T(Y - B\beta) + 2\lambda\Omega\beta &= 0 \\ (B^TB + \lambda\Omega)\beta &= B^TY \\ \beta &= (B^TB + \lambda\Omega)^{-1}B^TY. \end{aligned} \tag{22}$$

Now that we have have β, we can evaluate $\hat{r}$ at every point. To de-noise, we are particularly interested in evaluating it at $x_1, ..., x_n$.

$$\hat{r} = \begin{bmatrix} \hat{r}(x_1) \\ \vdots \\ \hat{r}(x_n) \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{n+4} \beta_j h_j(x_1) \\ \vdots \\ \sum_{j=1}^{n+4} \beta_j h_j(x_i) \\ \vdots \\ \sum_{j=1}^{n+4} \beta_j h_j(x_n) \end{bmatrix} = B\beta = B(B^TB + \lambda\Omega)^{-1}B^TY \tag{23}$$

Equation 23 shows that the de-noised version is actually linear because every entry is a linear combination of Y. Thus, following the notation of the smoother matrix L as in previous lectures, we get

$$\hat{r} = LY, \tag{24}$$

because $\hat{r}(x_i) = \sum_{j=1}^{n} L_{ij}Y_j$. We see that the theory that works for linear smoothers also works here, which is a nice property to have.

7 Remark: Background for Homework 2

Recall from the previous lectures on kernel estimation that

$$\hat{r}(x) = \frac{\sum_{j=1}^{n} w_j Y_j}{\sum_{j=1}^{n} w_j} \tag{25}$$

where $w_j = K(\frac{x_j - x}{h})$.

Smoothing splines can approximately be described as:

$$\hat{r}(x) \approx \frac{\sum_{j=1}^{n} w_j Y_j}{\sum_{j=1}^{n} w_j} \tag{26}$$

with $w_j = \frac{n^{\frac{1}{4}}}{\lambda^{\frac{1}{4}} f(x_j)^{\frac{3}{4}}} K\left(\frac{x_j - x}{(\frac{\lambda}{nf(x_j)})^{\frac{1}{4}}}\right)$ where $f(x)$ is the density of the covariate. Note that here the bandwidth is data-dependent! Thus, it is not fixed anymore as before and we can adjust the bandwidth depending on how clustered the datapoints are around x_j. For example, we might want a smaller bandwidth if the datapoints are highly clustered.