MATH 7339 - Machine Learning and Statistical Learning Theory 2 –He Wang

Section. Splines and Regularization.

1. **Basis Expansions**
2. **Splines**
3. **Natural Cubic Splines**
4. **Degree of Freedom**
5. **Spline for regression and classification**
6. **Regularization**
7. **Multi-dimensional Splines**

➢ Smoothing Procedures

There are four main views on smoothing mechanisms:

- **Piecewise smoothing** cuts the domain into regions and models each region separately.
- **Basis smoothing** attempts to write the data as a weighted sum of basis functions. This will be a unifying theme throughout this lecture.
- **Kernel smoothing** passes a window over over the data, taking a weighted average of data values near a point. Note: this usage of kernel is not the same as in the kernel method from before.
- **Dimensional reduction/expansion** attempts to smooth data by passing through a compression or dimensional reduction filter and then projects it back upwards.

We will assume unless otherwise stated that x is one dimensional first.

Recall Linear Models:

Consider a test vector $\vec{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$

We want to predict a random variable $Y \in \mathbb{R}$ or $\{0,1\}$ from knowing $\vec{x}$.

We often use a linear predictor function

$$f(\vec{x}) = \vec{x}^T \vec{\theta} = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$$

For example, assume the model is linear

$E\left(Y \mid \vec{X}\right) = f(\vec{x})$ for linear regression

Or $\log \frac{P(Y = 1 | \vec{X} = \vec{x})}{P(Y = 0 | \vec{X} = \vec{x})} = f(\vec{x})$ for logistics regression.

Or LDA, SVM, etc.

Beyond Linear Models:

Linear machinery quite appealing! It is convenient, and sometimes a necessary, approximation.

linear model is easy to interpret, and is the first-order Taylor approximation to $h(X)$. Sometimes necessary, because with data size $\boldsymbol{n}$ small and/or number of features d large, a linear model might be all we are able to fit to the data without overfitting.

However, in regression problems, $f(\vec{X}) = E(Y|\vec{X})$ will typically be nonlinear and nonadditive in $\vec{X}$. Then a linear model often too crude.

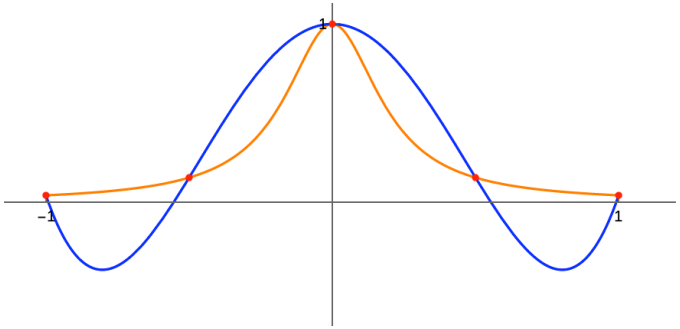
Build more general models: Impose linearity on **functions** of the original inputs.

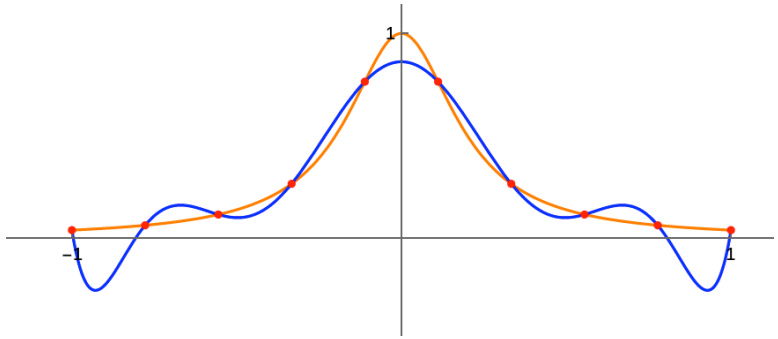
Get richer space of possible functions, and can still use familiar algorithms.

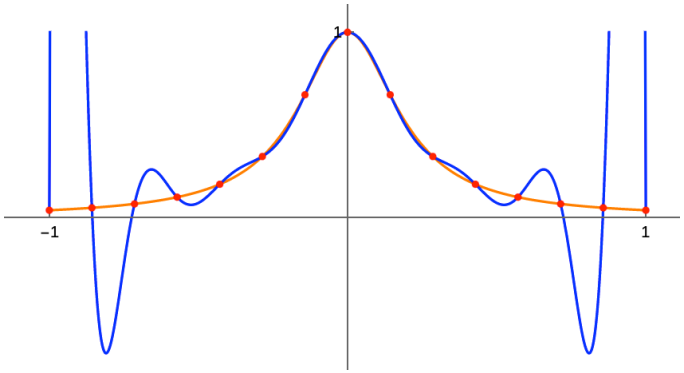
Related to more general transformations of original variables x_i

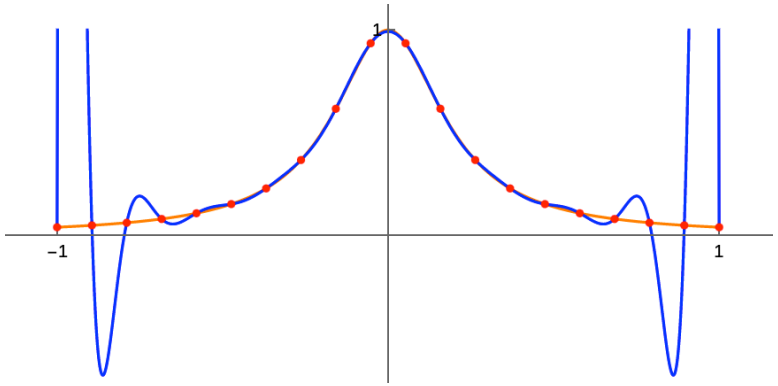
Example: The Runge function $f(x) = \dfrac{1}{1+25x^2}$

Higher order **polynomial interpolation is a bad idea**. (N=5,10,15,20 equal spaced sample points .)



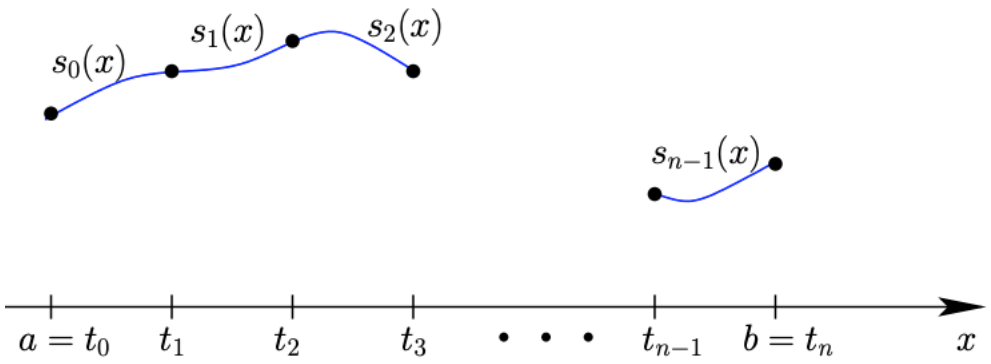






https://demonstrations.wolfram.com/RungesPhenomenon/

A better strategy – piece-wise polynomial or spline interpolation (overview)



Cubic Example:

$$S(x) = \begin{cases} s_0(x) = a_0x^3 + b_0x^2 + c_0x + d_0 & \text{for } t_0 \le x \le t_1 \\ \vdots & \\ s_{n-1}(x) = a_{n-1}x^3 + b_{n-1}x^2 + c_{n-1}x + d_{n-1} & \text{for } t_{n-1} \le x \le t_n \end{cases}$$

We want $S(x)$ to be **continuous**, i.e., $s_{i-1}(x_i) = s_i(x_i)$ for all $i = 1,2,\dots,n-1$

continuous 1st derivative, i.e., $s'_{i-1}(x_i) = s'_i(x_i)$ for all $i = 1,2,\dots,n-1$

continuous 2ed derivative, i.e., $s''_{i-1}(x_i) = s''_i(x_i)$ for all $i = 1,2,\dots,n-1$

Another notation for piece-wise polynomial is

$$S(x) = s_0(x)\,\mathbb{I}(t_0 \le x \le t_1) + s_1(x)\mathbb{I}(t_1 \le x \le t_2) + \ldots + s_{n-1}(x)\mathbb{I}(t_{n-1} \le x \le t_n)$$

$$= (a_0x^3 + b_0x^2 + c_0x + d_0)\,\mathbb{I}(t_0 \le x \le t_1)$$

$$+(a_1x^3 + b_1x^2 + c_1x + d_1)\mathbb{I}(t_1 \le x \le t_2) + \cdots$$

$$+ (a_{n-1}x^3 + b_{n-1}x^2 + c_{n-1}x + d_{n-1})\mathbb{I}(t_{n-1} \le x \le t_n)$$

Here, $\mathbb{I}(z)$ is the **indicator function.**

$$\mathbb{I}(z) = \begin{cases} 1 & \text{If z is true.} \\ 0 & \text{If z is false.} \end{cases}$$

Furthermore, each piece $\tilde{s}_j(x) = a_j(x - t_j)^3 + b_j\big(x - t_j\big)^2 + c_j(x - t_j) + d_j$ is widely used for computation. In this case, $d_j = \tilde{s}_j\big(t_j\big)$

➢ Basis function methods.

Instead of assuming that $f(\vec{x}) = \sum_{i=1}^{d} \theta_i \cdot x_i$

Replace each coordinate x_i by a more general function of $\vec{x}$ (i.e., not just directly projecting $\vec{x}$ onto a coordinate x_i).

Assume a linear basis expansion in $\vec{x}$

$$\hat{y} = f(\vec{x}) = \sum_{m=1}^{M} \beta_m \cdot h_m(\vec{x})$$

where $h_m(\vec{x})$: $\mathbb{R}^d \rightarrow \mathbb{R}$ are (fixed) basis functions. Note here the β_m are pre-determined and do not depend on $\vec{x}$.

Some questions about basis functions:

Question 1: What is the choice of basis $h_m(\vec{x})$?

For example, here the h_m's might be:

- $h_m(\vec{x}) = x_m, m = 1, \ldots, p$ recovers the original linear model.
- $h_m(\vec{x}) = x_j^2 \; or \; h_m(\vec{x}) = x_j x_k$ gives the polynomial model.
- $h_m(\vec{x}) = \log(\vec{x})$, or $\sqrt{\vec{x}}$, or $\sin m\pi x$, or …
- $h_m(\vec{x}) = \mathbb{I}(L_m \leq x_k \leq U_m)$, where $\mathbb{I}$ is the indicator function.

More generally, $h_m(\vec{x})$ can be piecewise polynomials/splines, reproducing kernel functions, wavelets, Fourier bases.)

Question 2: How "rich" should this class be?

Use lots of different functions $h_m(\vec{x})$

Question 3: Which functions $h_m(\vec{x})$ to be used and which skipped (feature selection)?

May need to **restrict** choices of $h_m(\vec{x})$, or may want "**complexity penalization**" for using too many different functions $h_m(\vec{x})$. (risk of overfitting)]

Polynomials are natural choices for $h_m(\vec{x})$ but can be too rigid - may need more flexibility. Also, for higher order polynomials we have excessive oscillations.

Question 4: What is the selection of the basis elements; how do we estimate the β_m?

Similar as linear regression. E.g., polynomial regression.

➢ Review some concepts of vector spaces of functions

- Vector spaces of functions
- Linear combination
- Span
- Linearly Independent
- Basis

Piecewise Polynomials

Assume in one-dimension only, with $\boldsymbol{x} \in [\boldsymbol{a}, \boldsymbol{b}]$, a fixed interval.

Assume the model is $f(x) = \sum_{i=1}^{p} \theta_i \cdot h_i(x)$

Example 1. piecewise constant

$$h_1(x) = \mathbb{I}(x < \xi_1),$$

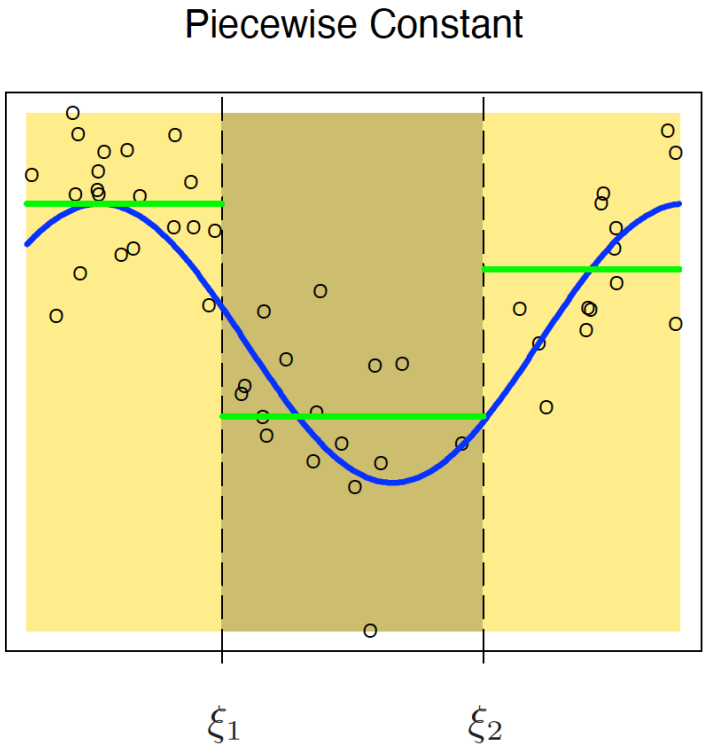
$$h_2(x) = \mathbb{I}(\xi_1 \leq x < \xi_2),$$

$$h_3(x) = \mathbb{I}(\xi_2 \leq x)$$

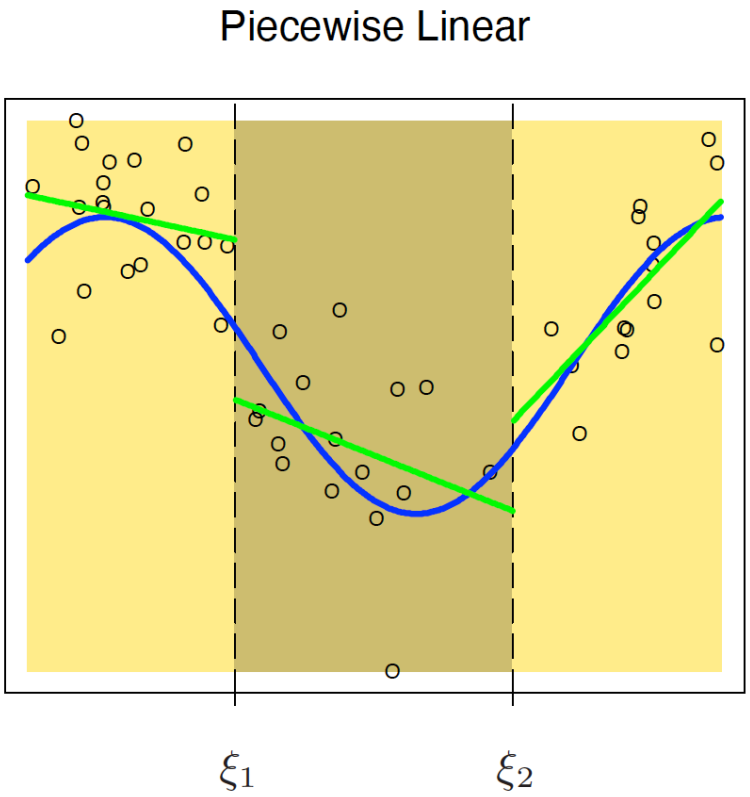
where $\mathbb{I}$ is the indicator function.

There are three **basis** functions here.

Model: $f(x) = \theta_1 h_1(x) + \theta_2 h_2(x) + \theta_3 h_3(x)$



The blue curve represents the true function, from which the data were generated with Gaussian noise.



Example 2. Piecewise linear

$$h_4(x) = \mathbb{I}(x < \xi_1) \cdot x,$$

$$h_5(x) = \mathbb{I}(\xi_1 \leq x < \xi_2) \cdot x,$$

$$h_6(x) = \mathbb{I}(\xi_2 \leq x) \cdot x$$

three more basis functions.
(6 functions in total now.)

Model: $f(x) = \theta_1 h_1(x) + \theta_2 h_2(x) + \cdots \theta_6 h_6(x)$

Example 3. Continuous Piecewise Linear

Piecewise linear, but restricted to be continuous at the two knots.

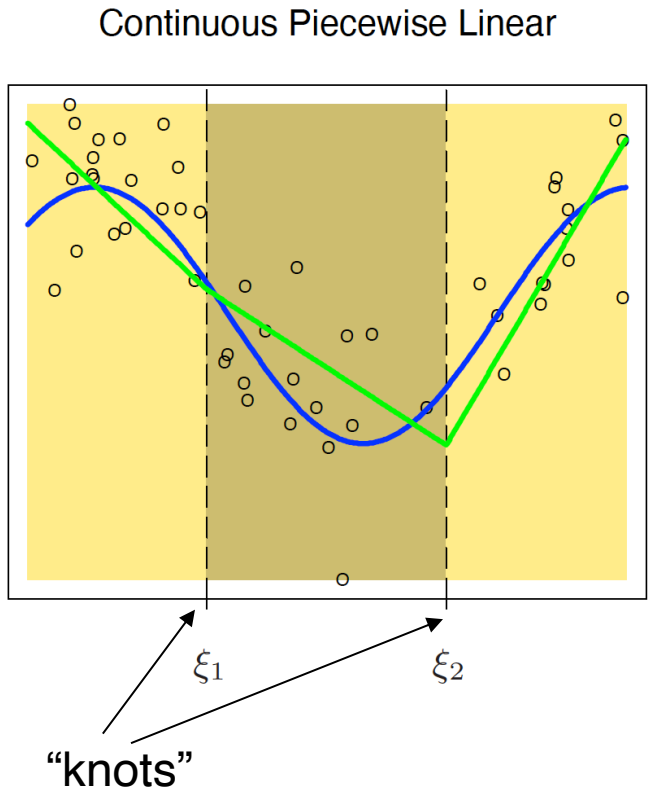
$$f(\xi_1^-) = f(\xi_1^+)$$

$$f(\xi_2^-) = f(\xi_2^+)$$

These continuity restrictions lead to linear constraints on the parameters.

$$\theta_1 + \xi_1\theta_4 = \theta_2 + \xi_1\theta_5$$

$$\theta_2 + \xi_1\theta_5 = \theta_3 + \xi_1\theta_6$$



Number of free parameters $= \cdots = 4$

(3 regions) X (2 parameters per region)– (2 knots X 1 constraint per knot)

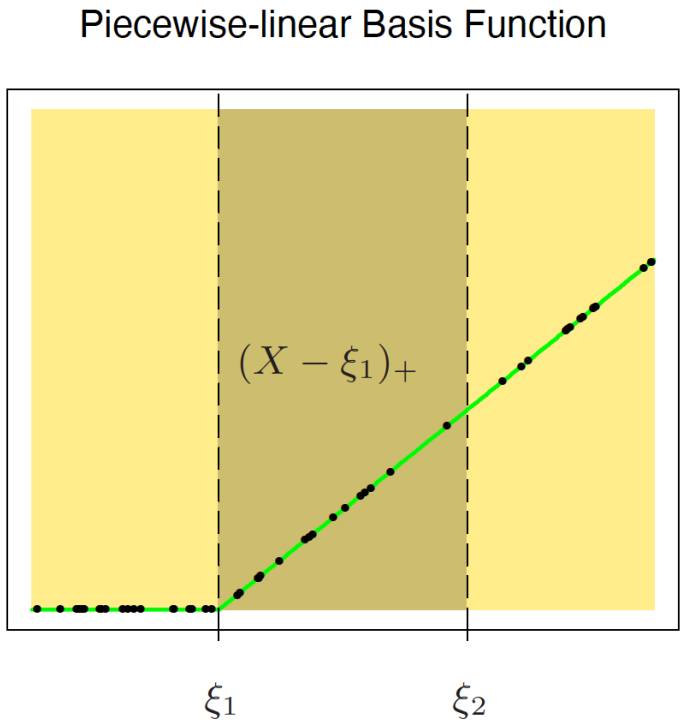
A more direct way to use a **basis**

$$h_1(x) = 1$$

$$h_2(x) = x$$

$$h_3(x) = (x - \xi_1)_+$$

$$h_4(x) = (x - \xi_2)_+$$



Here, $(z)_+ = \max(z, 0)$ denotes the positive part of z. Same as the ReLU function

Fitting Piecewise Linear Basis (algorithm)

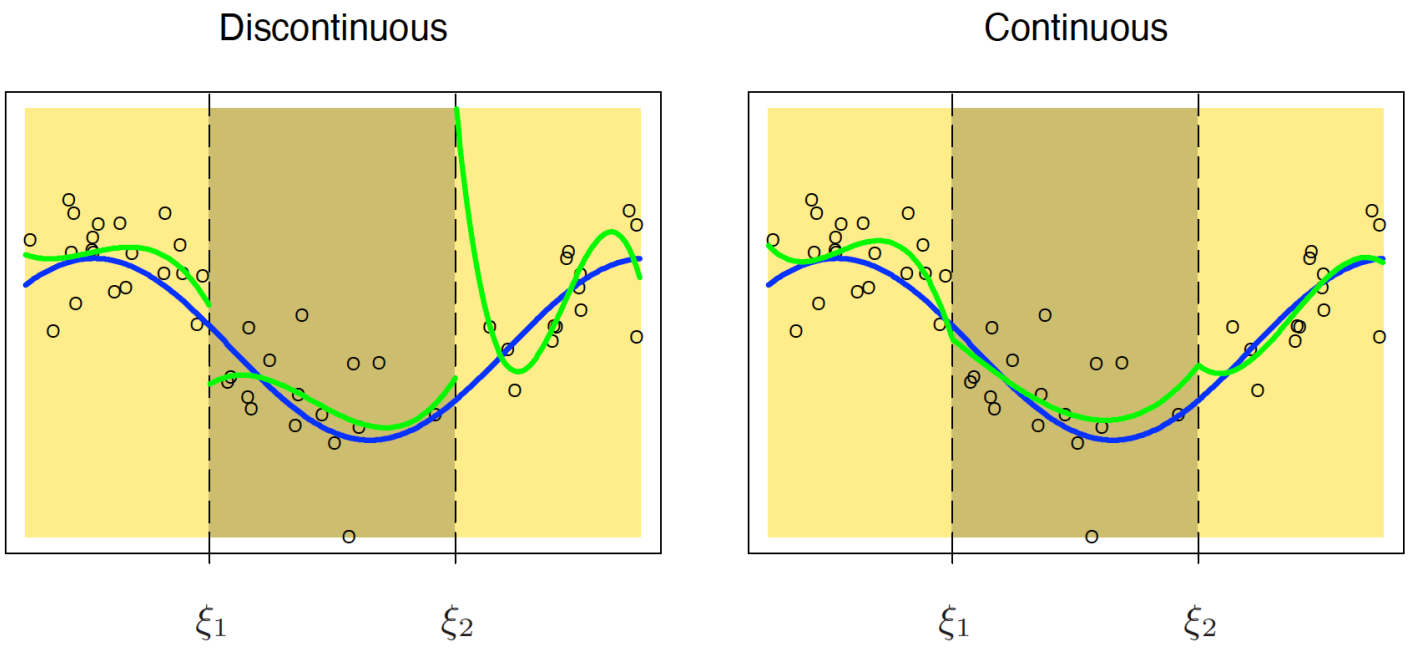
To fit a piecewise linear basis:

- Fit the linear function $\theta_0 + \theta_1 x$ to the whole dataset.
- Construct the features $x_{k+1} = (x - \xi_k)_+ - \theta_0 - \theta_1 x$.
- Fit the model $f(x) = \sum_{k=2}^{K} \theta_k \, x_k$ using linear methods.

Example: Piecewise Cubic Polynomials

We often prefer smoother functions, which can be achieved by increasing the order of the local polynomial.

For example, piecewise-cubic polynomials

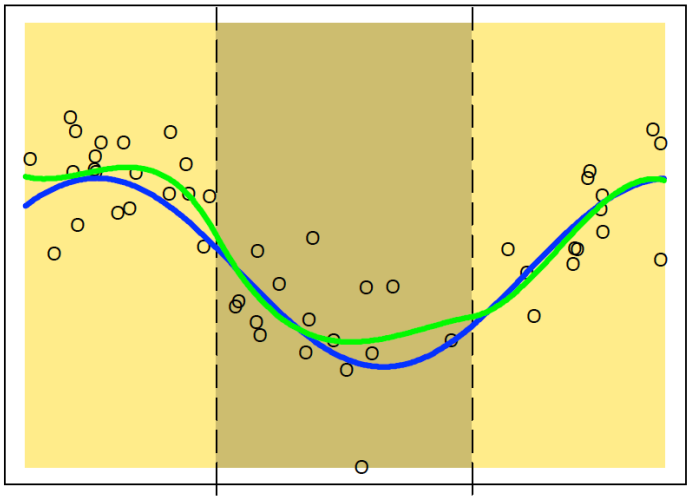


$f(\xi_i^-) = f(\xi_i^+)$ for i=1,2

➢ Cubic spline

Moving beyond linear functions, we can enforce continuity on the higher order derivatives. One the left we see a cubic fit with continuous first derivatives and on the right we also impose continuous second derivatives.

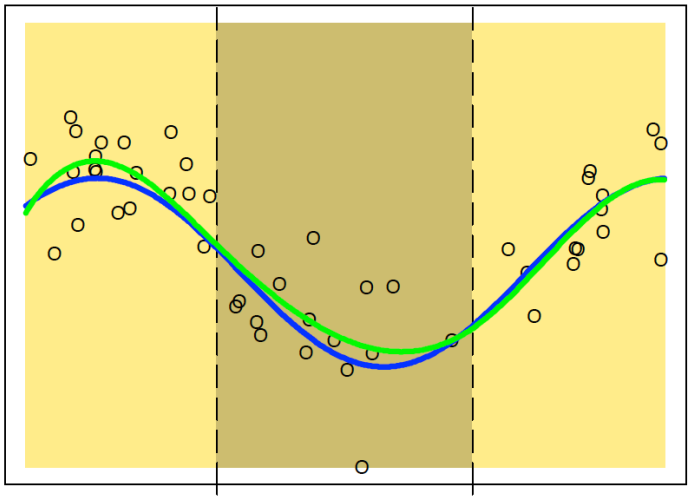
Continuous First Derivative



ξ_1 ξ_2

$f'(\xi_i^-) = f'(\xi_i^+)$ for i=1,2

Continuous Second Derivative



ξ_1 ξ_2

$f''(\xi_i^-) = f''(\xi_i^+)$ for i=1,2

cubic spline

A piecewise cubic with continuous second derivatives at the endpoints ξ is called a **cubic spline.**

Example. Cubic spline.

The function in the lower right panel is continuous $f(\xi_i^-) = f(\xi_i^+)$, and has continuous first derivatives $f'(\xi_i^-) = f'(\xi_i^+)$ and second derivatives $f''(\xi_i^-) = f''(\xi_i^+)$ at the knots $\xi_1 \ldots \xi_K$.

A basis for cubic spline with knots at ξ_1 and ξ_2 are

$$h_1(x) = 1$$

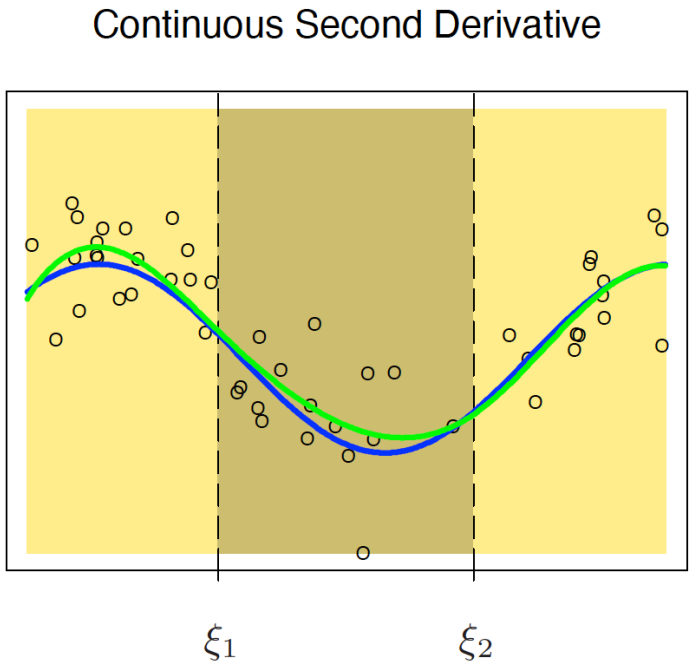
$$h_2(x) = x$$

$$h_3(x) = x^2$$

$$h_4(x) = x^3$$

$$h_5(x) = (x - \xi_1)^3{}_+$$

$$h_6(x) = (x - \xi_2)^3{}_+$$

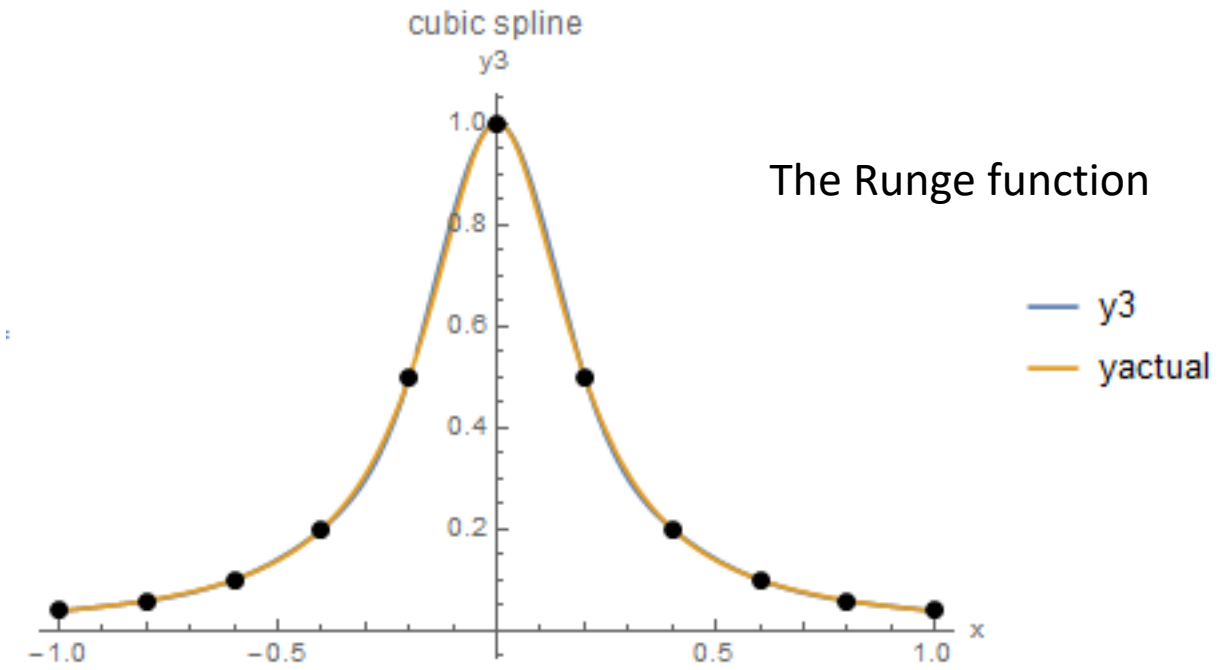


Number of free parameters for the model =

(3 regions)×(4 parameters per region) − (2 knots)×(3 constraints per knot) = 6.

Idea behind splines:

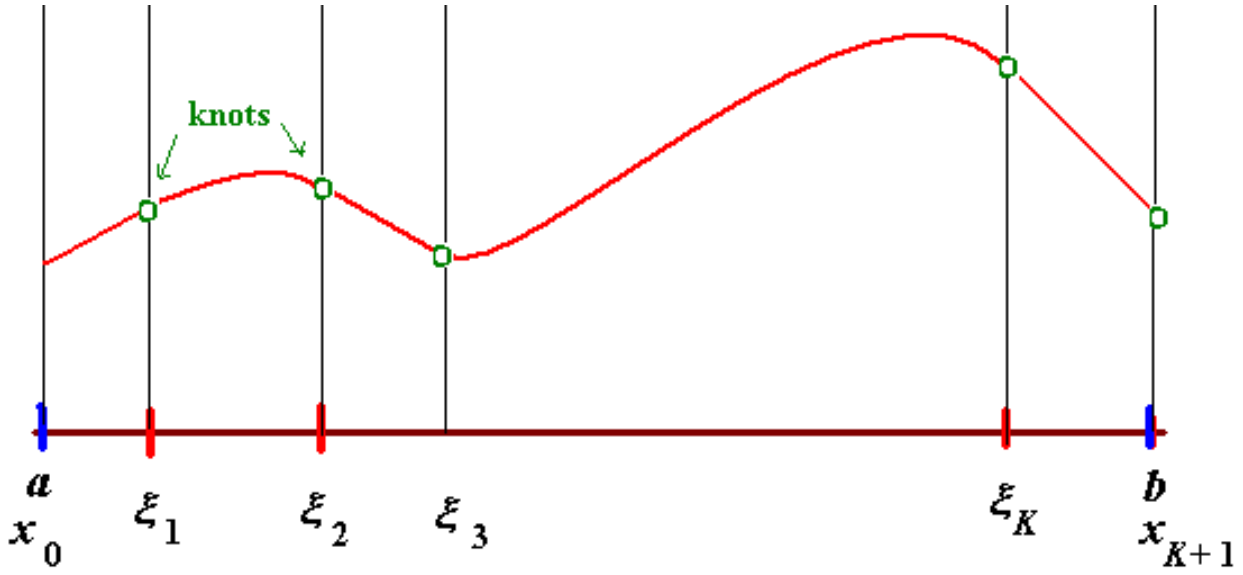
1. Use lower order polynomials to connect subsets of data points
2. Make connections between adjacent splines smooth
3. Lower order polynomials avoid oscillations and overfitting.



The Runge function

➢ Spline functions

More generally, Fix M∈ $\mathbb{N}$ (natural number) and allow only functions that are piecewise polynomials of some order $M-1$. We assume continuous derivatives to order $M-2$, even at knots (joining points of different polynomials).



Spline Space

Definition: The **spline space** $\mathcal{S}$ of order $M-1$ on an interval $[a,b] \in \mathbb{R}$ is the set of all functions $f(x)$ on $[a,b]$,

- with $M-2$ continuous derivatives everywhere, $f^{(M-2)}(x^-) = f^{(M-2)}(x^+)$
- and that are piecewise polynomial of order $M-1$ in all spline intervals defined by the mesh $\{\xi_1, \ldots, \xi_K\}$.

For example, A **cubic** spline is an order $M = 4$ spline.

Some terminologies:

- Such functions $f(x)$ are called splines of order $M-1$.
- One of the above intervals with endpoints in the set $\{a, \xi_1, \ldots, \xi_K, b\}$ is called a spline interval, (i.e., $R_k = [\xi_{k-1}, \xi_k]$).
- The joining points ξ_i are called knots.

Basis for spline space

Theorem: For any fixed M, the spline space of functions is a *vector space.*

Theorem: Then for fixed M, a **basis** for the **spline space** of functions is given by the following $M + K$ functions:

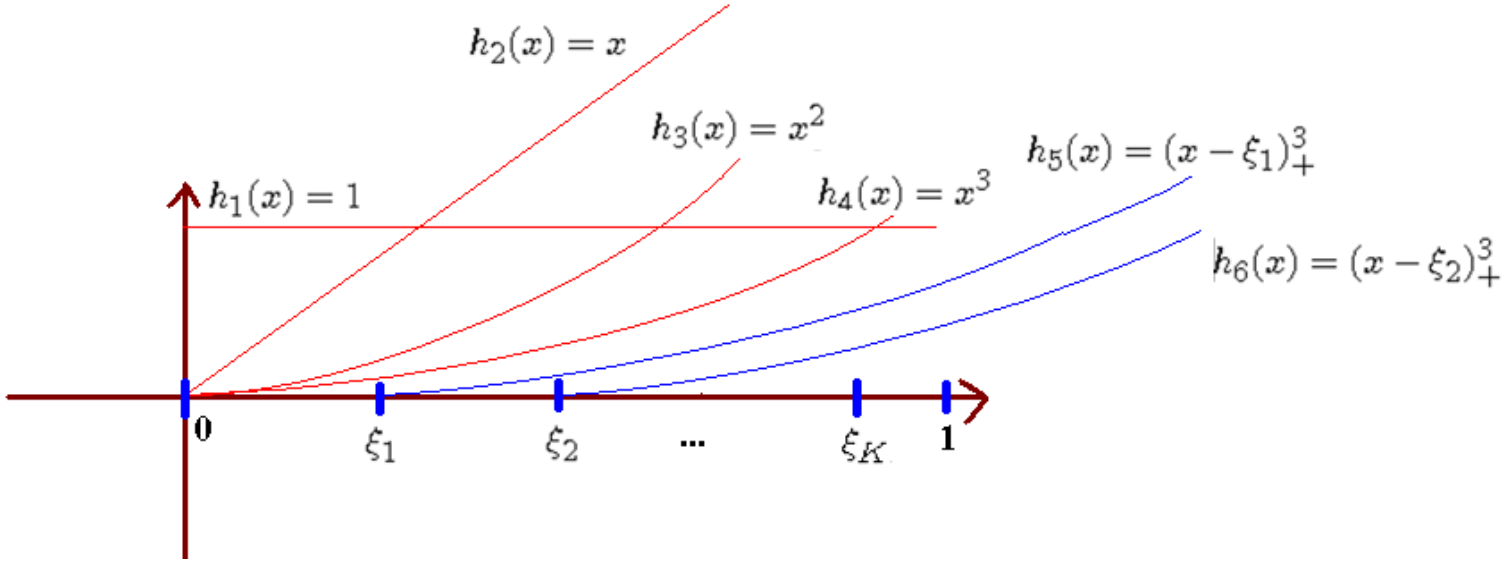
$$h_j(x) = x^{j-1}, \text{ for } j = 1, \dots, M$$

$$h_{M+l}(x) = (x - \xi_l)^{M-1}_{+}, \text{ for } l = 1, \dots, K.$$

Thus, we have $\boldsymbol{M + K}$ degrees of freedom (free coefficients of above basis functions) to specify our approximation functions. (i.e., the number of free coefficients θ_i of these functions)

$$f(x) = \sum_{i=1}^{M+K} \theta_i \cdot h_i(x) = \sum_{j=1}^{M} \theta_i \cdot h_i(x) + \sum_{l=1}^{K} \theta_{M+l} \cdot h_{M+l}(x)$$

For simplicity, we assume $[a, b] = [0,1]$



Case $M = 4$: basis functions for cubic splines

➢ Four variations of use splines:

1. (Standard) regression splines: Knot points $\xi_1, \ldots, \xi_K$ **fixed** beforehand, and use OLS (ordinary least squares) to fit a function of the form

$$f(x) = \sum_{i=1}^{M+K} \theta_i \cdot h_i(x)$$

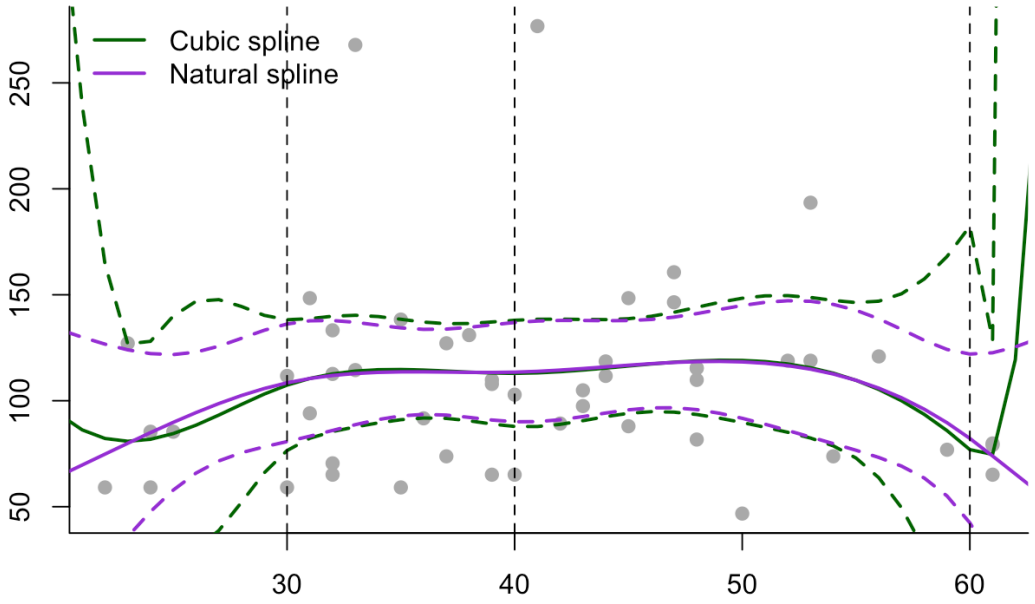
We can find θ_i with standard least squares again.

3. Natural cubic splines.

4. Smoothing splines: can also vary knot locations $\xi_1, \ldots, \xi_K$ use ridge regression to regularize and 'shrink' the θ_i.

➢ Natural Cubic Spline $(M = 4)$

The cubic spline basis we've described should only be trusted over the regions R_k , since outside these regions the boundary cubic polynomials quickly go to infinity.

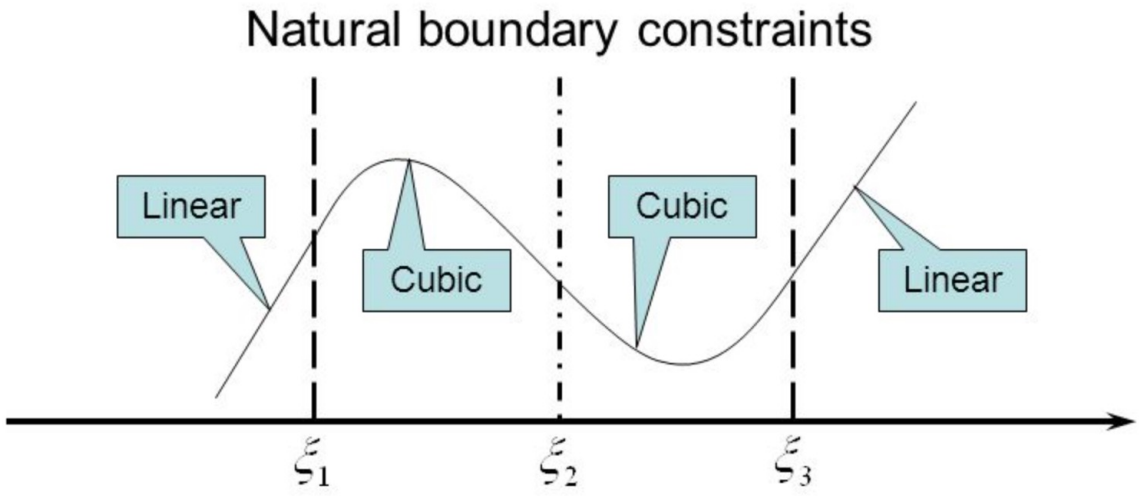


- Regression splines usually have *high variance* at the outer range of the predictor (the *tails*).
- The confidence intervals at the tails are wiggly (especially for small sample sizes)

Natural cubic splines are quite popular to solve the above problems.

They are like standard cubic splines with fixed (i.e., predetermined) knots at $\xi_1, \ldots, \xi_K$.

Adds a further **Natural** constraint that the fitted function is **linear** beyond the boundary knots $\xi_1 \; and \; \xi_K$, or on end regions $[a, \xi_1]$ and $[\xi_K, b]$.



Standard (regression) cubic splines have high variance at end regions $[a, \xi_1]$ and $[\xi_K, b]$, and this is reduced by the **linearity** constraints in natural cubic spline.

Natural cubic splines controls variance, at cost of more bias near the endpoints than cubic splines.

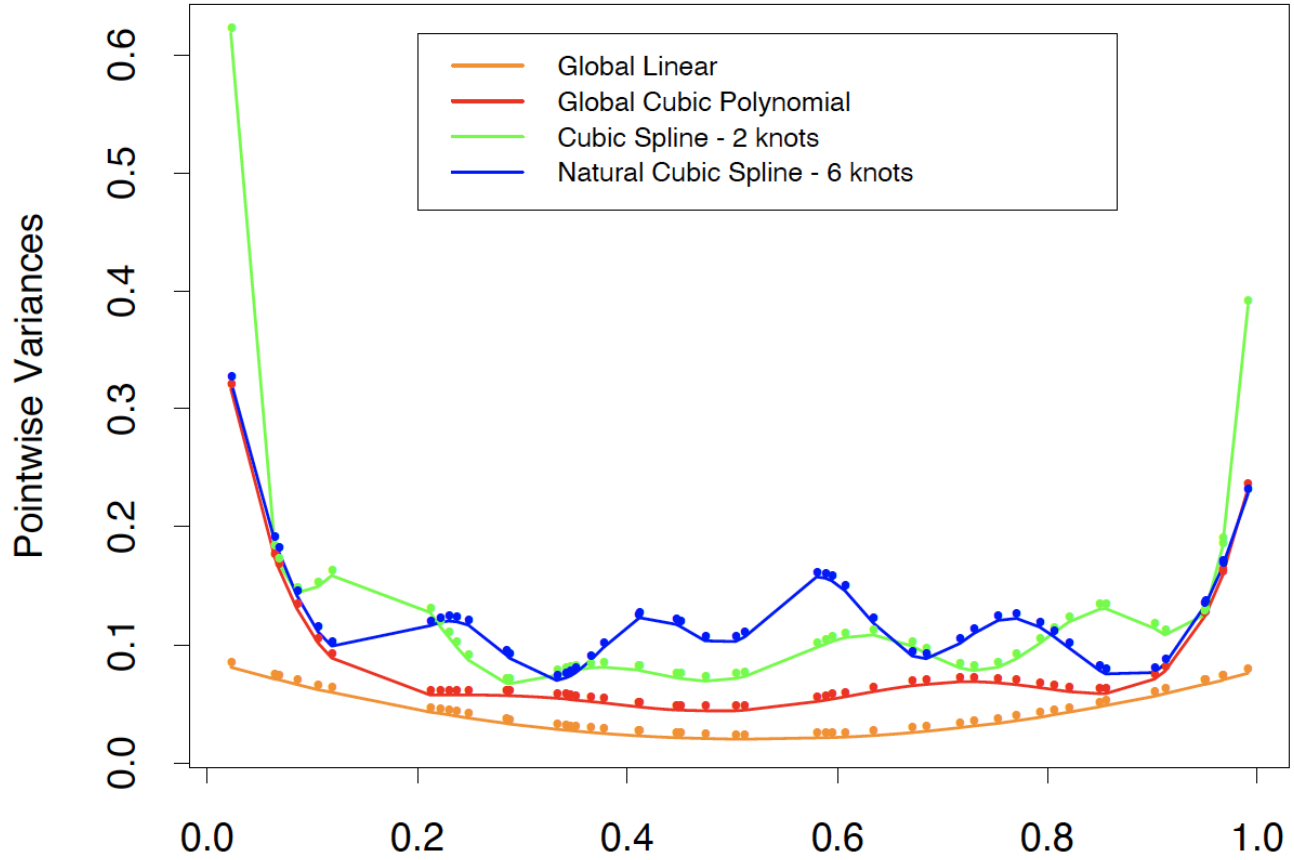


Figure from [Hastie], p. 145: for each of the 4 models pick an $f(x)$ that satisfies the model (e.g. green curve is a cubic spline $f(x)$ with given fixed knots at 0.33 and 0.66) adding random error ϵ_i of fixed variance σ. So $y^{(i)} = f\left(x^{(i)}\right) + \epsilon_i$. Then vary the training set $\mathcal{D} = \left\{\left(x^{(i)}, y^{(i)}\right)\right\}_{i=1}^{50}$ by fixing $x^{(i)}$ and varying the ϵ_i. The graph plots the variation at each training point of the estimate $\hat{y}^{(i)} = \hat{f}\left(x^{(i)}\right)$ as $\mathcal{D}$ varies.

Basis for space of Natural Cubic Spline

Theorem: A **natural cubic spline** model with *K* knots is represented by ***K*** **basis** functions:

$$N_1(x) = 1$$

$$N_2(x) = x$$

$$N_{k+2}(x) = d_k(x) - d_{K-1}(x) \qquad \text{for } k = 1, \dots, K-2$$

where,

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

Note that,

Each of these basis functions has zero 2nd and 3rd derivative outside the boundary knots.

The dimension for natural cubic spline space is $\boldsymbol{K}$. The dimension is $\boldsymbol{K+4}$ for cubic spline space.

Example

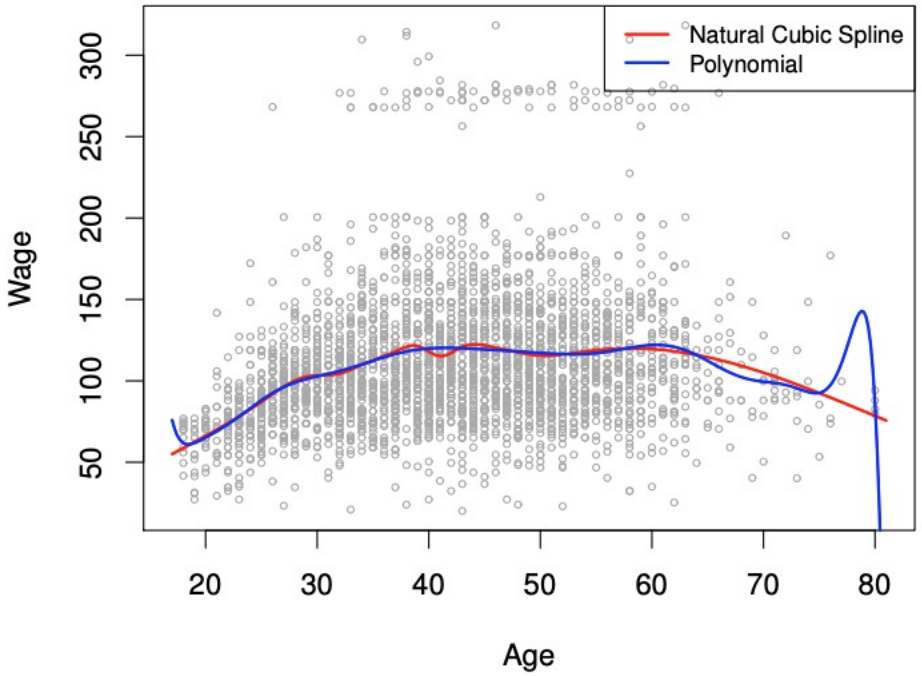


Figure: 7.7 from ISLR. Natural cubic splines are very nicely behaved at the tails of the data. Polynomial regression shows erratic behavior. (14 degrees of freedom used for both)

Some remarks:

The function $d_k(x)$ are not linear for $x > \xi_K$, but $d_k(x) - d_{k-1}(x)$ are linear.

These functions $N_k(x)$ are all linear for $x > \xi_K$ and $x < \xi_1$.

Standard (regression) cubic splines have high variance at end regions, i.e., $[a, \xi_1]$ and $[\xi_K, 1]$, and this is reduced by the linearity constraint.

Controls variance, at cost of more bias near the endpoints.

Cubic splines are a decent smoothing basis and there is seldom a good reason to go to higher order. However, for a large number of regions it can become computationally inefficient. In this case, there are other spline bases such as B-splines which use the Harr basis to define more computationally efficient spline fittings.

➢ Regression using cubic splines(dim=d)

In practice, assume input predictor variables $\vec{X} = (X_1, \ldots, X_d) \in \mathbb{R}^d$, and want to predict output Y.

Notice as usual $(\vec{X}, Y)$ initially random variables.

As usual, our model requires we use a predictor function (to predict y)

$$y \approx f(\vec{x}) = E\left(Y \middle| \vec{X} = \vec{x}\right)$$

In our cubic spline model, will assume

$$f(\vec{x}) = E\left(Y \middle| \vec{X} = \vec{x}\right) = \left[\sum_{j=1}^{d} \sum_{m=1}^{M} \theta_{jm} \cdot h_m(x_j)\right] + \theta_0$$

Note $f(\vec{x})$ is still modeled as sum of finite of x_j, but not linear ones.

So, we can use vector notation

$$f(\vec{x}) = \left[\sum_{j=1}^{d} \vec{\theta}_j^T \, \vec{h}(x_j)\right] + \theta_0$$

Here

$$\vec{\theta}_j = \begin{bmatrix} \theta_{j1} \\ \theta_{j2} \\ \vdots \\ \theta_{jM} \end{bmatrix} \qquad \vec{h}(x_j) = \begin{bmatrix} h_1(x_j) \\ h_2(x_j) \\ \vdots \\ h_M(x_j) \end{bmatrix}$$

Here, $\vec{h}(x_j)$is more than just x_j by itself as in linear regression.

But we will use the usual OLS (ordinary least squares) tools for computation

Use matrix notation

$$f(\vec{x}) = \left[\sum_{j=1}^{d}\sum_{m=1}^{M} \theta_{jm} \cdot h_m(x_j)\right] + \theta_0 = \left[\sum_{j=1}^{d} \vec{\theta}_j^T \, \vec{h}(x_j)\right] + \theta_0 = \boldsymbol{H\Theta}$$

Here,

$$\boldsymbol{H} = \begin{bmatrix} 1 & h_1(x_1) & \dots & h_M(x_1) \\ 1 & h_1(x_2) & \dots & h_M(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & h_1(x_d) & \dots & h_M(x_d) \end{bmatrix} \qquad \boldsymbol{\Theta} = \begin{bmatrix} \theta_0/d & \theta_{11} & \dots & \theta_{1M} \\ \theta_0/d & \theta_{21} & \dots & \theta_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_0/d & \theta_{d1} & \dots & \theta_{dM} \end{bmatrix}$$

The matrix product $\mathbf{H\Theta}$ is denotes a sum over all pairwise products of entries in $\mathbf{H}$ and $\mathbf{\Theta}$.

More questions about Splines:

Question: How many knots and where?

Up to now, we fix the knot points $\xi_1, \ldots, \xi_K$.

In principle, why not choose from among all f with any number of knots at any location (instead of fixed knots at pre-defined points $\xi_1, \ldots, \xi_K$)?

Options:

(1) Specify slightly over-rich set with variable knot locations ξ_k and use traditional variable selection tools to reduce too many degrees of freedom (df).

(2) Perform regression with standard fixed knot locations (less rich set, so less df and less chance to over-fit)

➢ Smoothing Splines (with shrinkage)

Based on the spline basis method to estimate $y = f(x) + \epsilon$:

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

Consider the one-dimension problem: among all functions $f(x)$ with continuous second derivatives, find the one that **minimizes** the penalized residual sum of squares.

$$RSS(f, \lambda) = \sum_{i=1}^{N} \left(y^{(i)} - f\left(x^{(i)}\right)\right)^2 + \lambda \int_a^b \left(f''(t)\right)^2 dt$$

Here λ is a fixed *smoothing parameter.*

That is, find

$$\hat{f}_\lambda(x) = \underset{f \in S[a,b]}{\text{argmin}} \; RSS(f, \lambda)$$

To make the $RSS(f, \lambda)$ well-defined, we need to consider only the (Sobolev) space $\mathcal{S}[a, b]$ of functions $f: [a, b] \to \mathbb{R}$ with **finite** $\int_a^b \left(f''(t)\right)^2 dt$.

$$\mathcal{S}[a, b] := \left\{ f: [a, b] \to \mathbb{R} \mid f'' \text{continuous and } \int_a^b \left(f''(t)\right)^2 dt \text{ is finite} \right\}$$

Two special cases are:

- $\lambda = 0$: f can be any function that interpolates the data.
- $\lambda = \infty$: f is the simple least squares line fit, since no second derivative can be tolerated.
- $\lambda \in (0, \infty)$: f is something in between.

Suppose the training data set is $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i-1}^{n}$

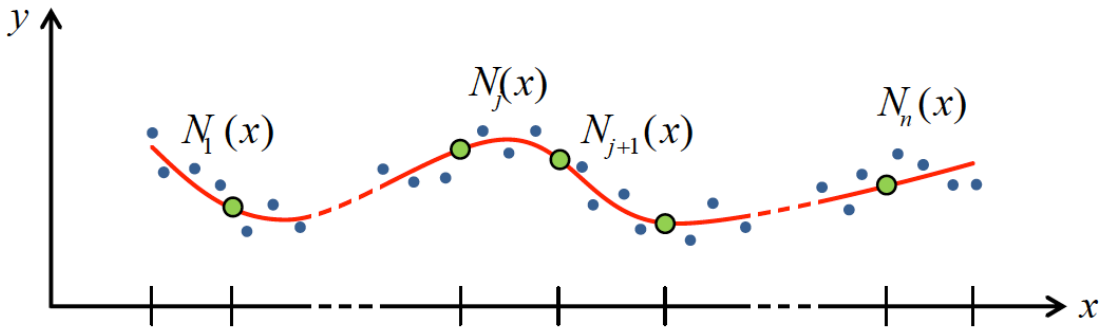
Theorem: The optimization question

$$\hat{f}_{\lambda}(x) = \underset{f \in \mathcal{S}[a,b]}{\operatorname{argmin}} \; RSS(f, \lambda)$$

has a **unique**, **finite-dimensional minimizer** given by a **natural cubic splines**

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

with knots at the datapoints $x^{(i)}$. The penalty λ becomes a dampening of the coefficients, which reduces the potential over fitting.



Writing

$$\{N\}_{ij} = N_j(x^{(i)}) \qquad \Omega_{jk} = \int_a^b N_j''(t)\, N_k''(t) dt$$

The penalized residual sum of squares

$$RSS(f, \lambda) = \sum_{i=1}^{n} \left(y^{(i)} - f\left(x^{(i)}\right)\right)^2 + \lambda \int_a^b \left(f''(t)\right)^2 dt$$

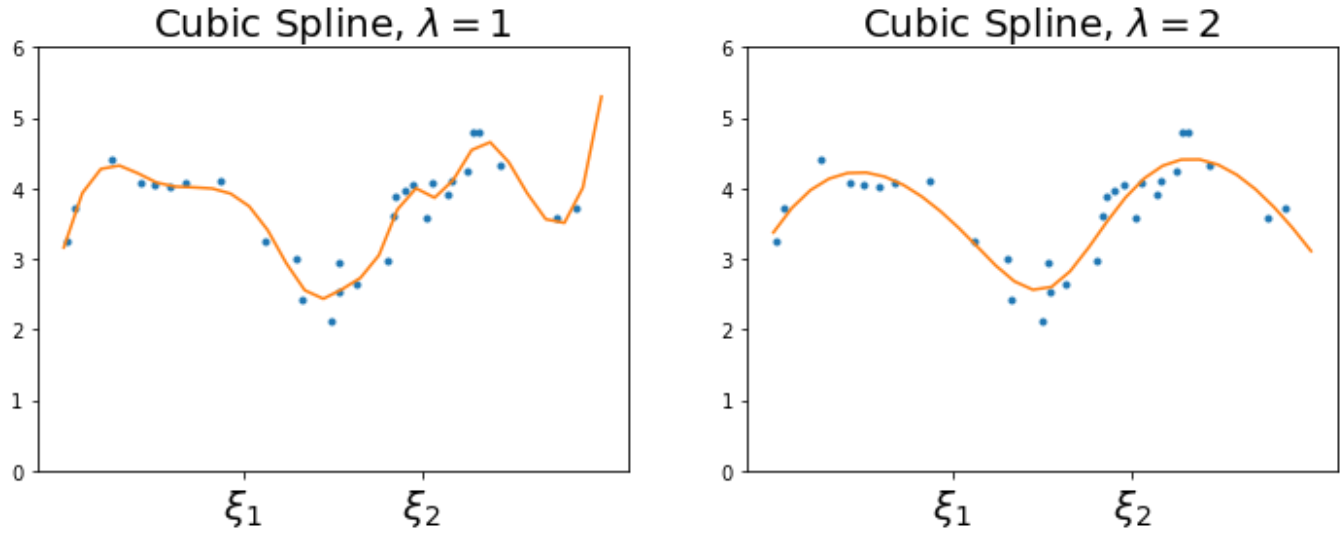
$$= \left(\vec{y} - N\vec{\theta}\,\right)^T \left(\vec{y} - N\vec{\theta}\,\right) + \lambda \vec{\theta}^T \Omega \vec{\theta}$$

It can then be solved for

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

This is just a generalized ridge regression problem.

Example: Non-Mixed Features



For example, we see the smoothing spline fit to the data from before for different values of λ. In this case, relatively close lambda values actually give a significantly different interpolation.

➢ Degrees of Freedom for Smoothing Splines

As a function of λ, a smoothing spline is a **linear smoother** since the parameters are linear functions of $y^{(i)}$. Indeed, after fitting

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

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

Denote $\hat{\mathbf{f}}$ the n-vector of fitted values $\hat{f}\left(x^{(i)}\right)$, then,

$$\hat{\mathbf{f}} = N(N^T N + \lambda\Omega)^{-1} N^T \vec{y} = S_\lambda \vec{y}$$

where S_λ is a linear operator known as the **smoother matrix**.

Here, N is an $n \times n$ matrix. $(N^T N + \lambda\Omega)$ is almost always invertible.

In particular, if we consider the case $\lambda = 0$ without penalty.

$$\hat{\mathbf{f}} = B(B^T B)^{-1} B^T \vec{y} = H \vec{y}$$

We need to assume that $\mathrm{n} > m$, i.e., no more free parameters m than data points n. Here m is the total number of spline basis functions.

Here B is an $n \times m$ matrix with rank $B = m$ (assumption to make sure $B^T B$ is invertible.)

Both S_λ and H are symmetric and positive definite.

H is is idempotent, i.e., $H^2 = H$. This implies $\mathrm{Tr}(H) = \mathrm{rank}\, H = M$

Degrees of Freedom

Degrees of Freedom of **cubic splines** is $df = \mathrm{Tr}(H) = M = \mathrm{rank}\, H$

Degrees of Freedom of **smoothing cubic splines** is

$$df = \mathrm{Tr}(S_\lambda) = \sum_{j=1}^{N} \mathrm{eig_j}(S_\lambda)$$

Denote $K = (N^T)^{-1}\Omega N$, then, the matrix S_λ can be written as,

$$S_\lambda = N(N^T N + \lambda\Omega)^{-1} N^T = (1 + \lambda K)^{-1}$$

Suppose K has eigenvalues d_k, then, S_λ has eigenvalues $\dfrac{1}{1 + \lambda d_k}$

$$df = \mathrm{Tr}(S_\lambda) = \sum_{\mathrm{k}=1}^{M} \frac{1}{1 + \lambda d_k}$$

➢ Classification using cubic splines

Nonparametric Logistic Regression. For categorical fitting, we can use smoothing methods to fit the probability discriminant for each category. Consider logistic regression with one-dimension x:

$$\log \frac{P(Y=1|X=x)}{P(Y=0|X=x)} = f(x) = \sum_{i=1}^{M+K} \theta_i \cdot h_i(x)$$

That means

$$P(Y=1|X=x) = \frac{e^{f(x)}}{1+e^{f(x)}} = \frac{1}{1+e^{-f(x)}}$$

Fitting $f(x)$ in a smooth fashion leads to a smooth estimate of the conditional probability $P(Y = 1|x)$, which can be used for classification or risk scoring.

We can also construct a penalized log-likelihood criterion:

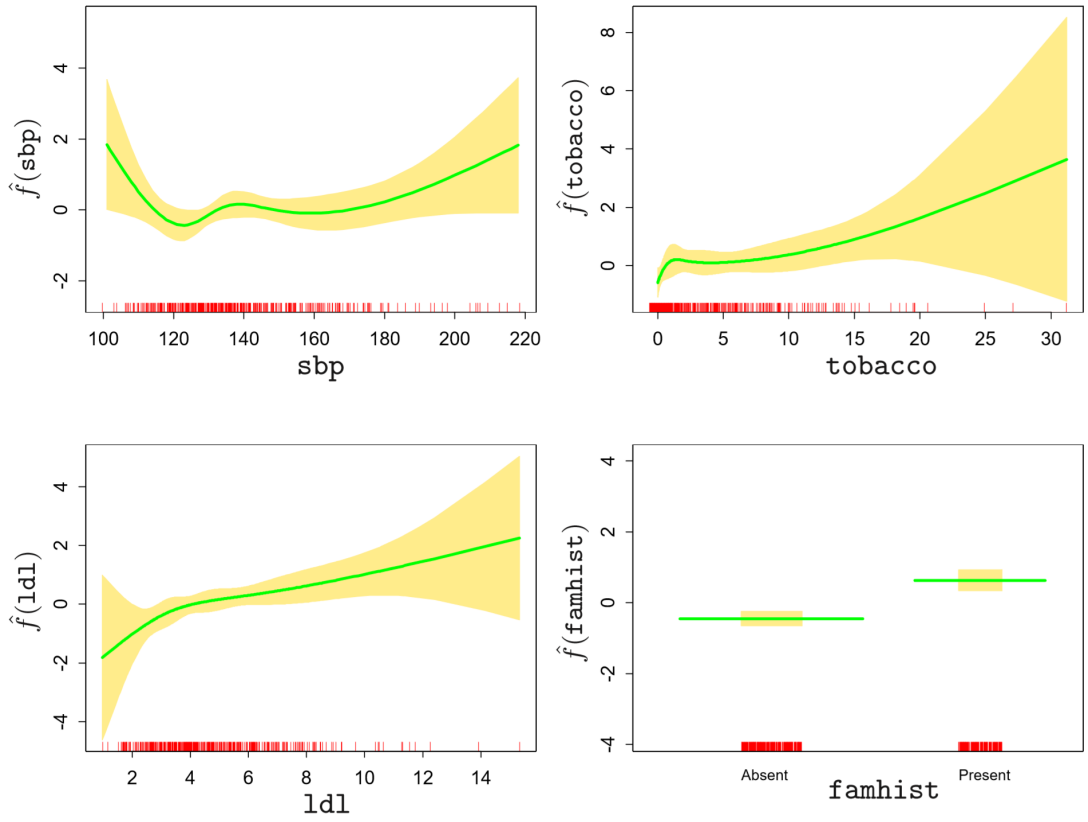
$$l(f,\lambda) = \sum_{i=1}^{N} \left(y^{(i)} \log p(x^{(i)}) + \left(1 - y^{(i)}\right) \log\left(1 - p\left(x^{(i)}\right)\right)\right) \; - \frac{1}{2}\lambda \int_a^b \left(f''(t)\right)^2 dt$$

$$= \sum_{i=1}^{N} \left(y^{(i)} f\left(x^{(i)}\right) + \log\left(1 + e^{f\left(x^{(i)}\right)}\right)\right) \; - \frac{1}{2}\lambda \int_a^b \left(f''(t)\right)^2 dt$$

Here, $p\left(x^{(i)}\right) = P(Y = 1|x)$

The optimal *f* is a natural spline with knots at the datapoint. we can proceed to fit it using Newton's method as we did for the linear logistic model.

Example: Non-Mixed Features



We can also use standard linear methods like backward subset selection, dropping basis elements according to some metric. In the above, features have been dropped to jointly minimize the training error and the number of degrees of freedom.

Other Bases

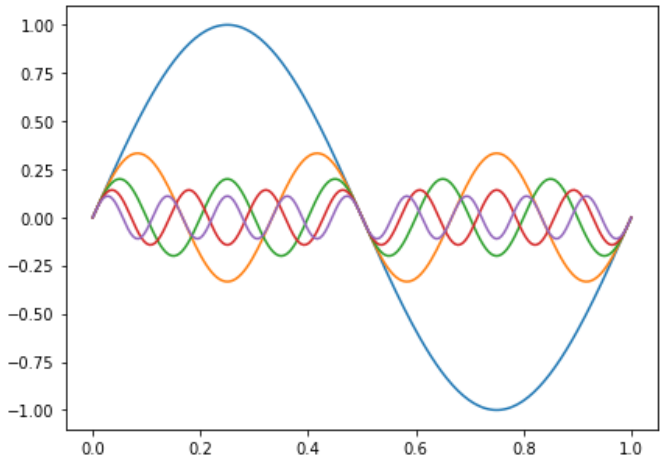
There are a few other bases we should mention.

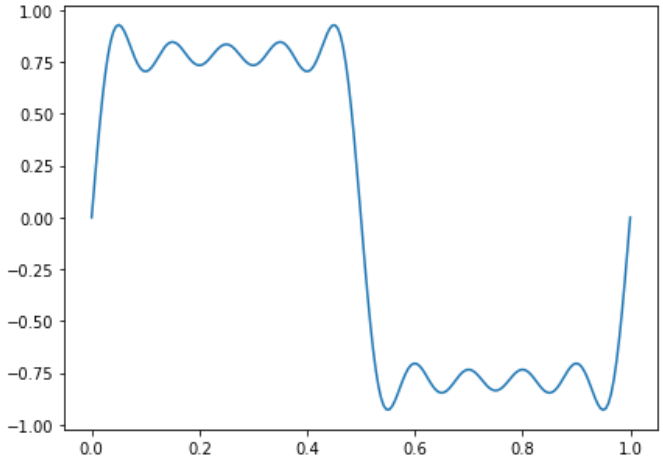
1. **Fourier Basis:** The canonical basis for wavelike data, separates waves into sums of frequencies.
2. **Wavelet Bases:** Any of a family of bases that consist of waves of a finite length. Unlike the Fourier basis this gives them both frequency and locality.
3. **Haar Basis:** A computationally efficient wavelet basis composed of piecewise step functions that cover a partition of the domain.
4. **B-Spline Basis**: Polynomial combinations of the Haar basis functions that provide a differenitable spline basis. The Bspline basis is what is typically used in actual spline computation.

1. Fourier Basis

The Fourier basis transforms repeating waves into sums of sine and cosine functions at different frequencies:

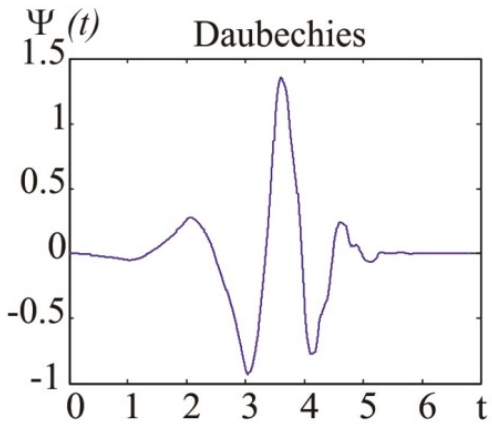
$$f(x) = \sum_{n=0}^{\infty} a_n \sin\left(\frac{2\pi n}{T}\right) + b_n \cos\left(\frac{2\pi n}{T}\right)$$

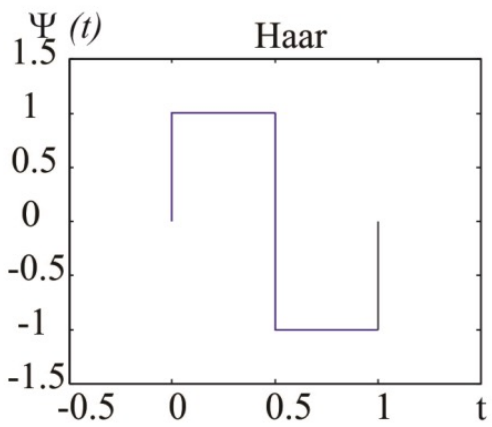


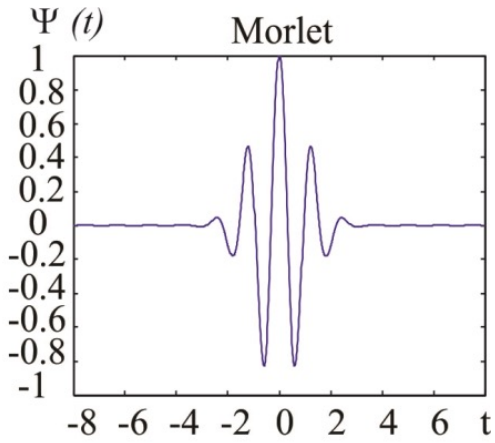


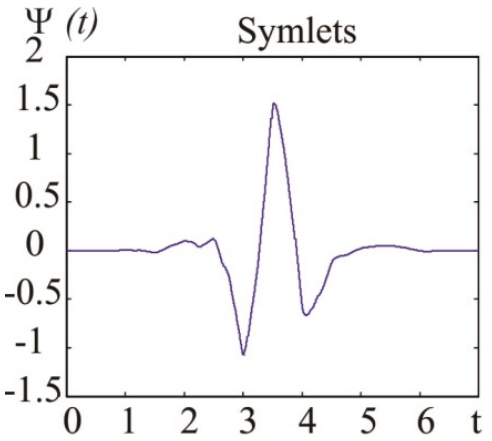
It is important to note that the Fourier basis truncated to a finite number of terms is inherently non-local.

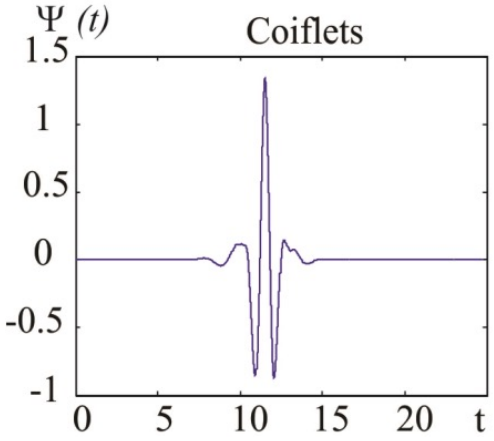
2. Wavelet Basis

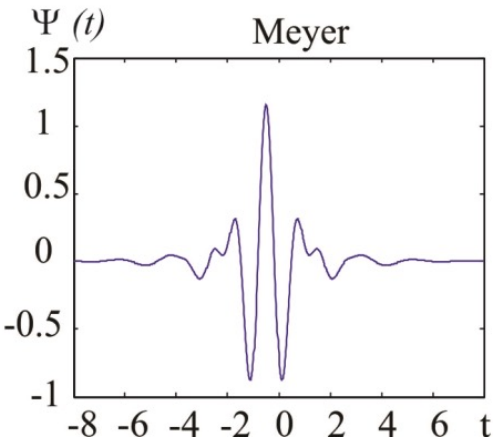






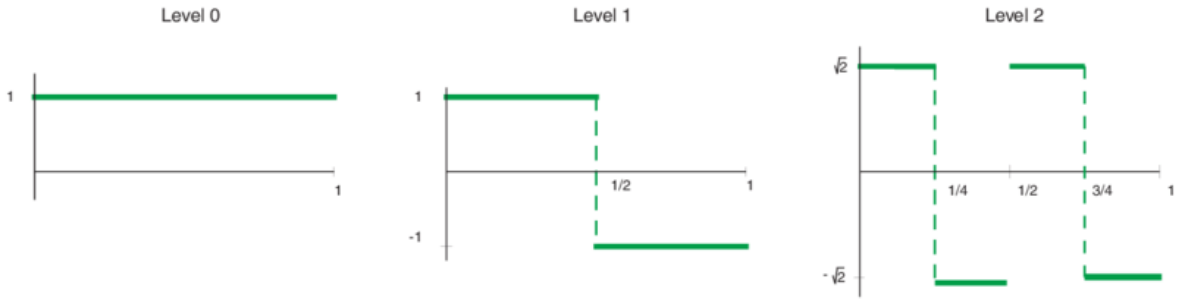






A **wavelet basis** is a basis of functions that try to capture both frequency and location. There are many adapted to various theoretical and practical uses, and are particularly used in image processing and storage.

3. Haar Basis

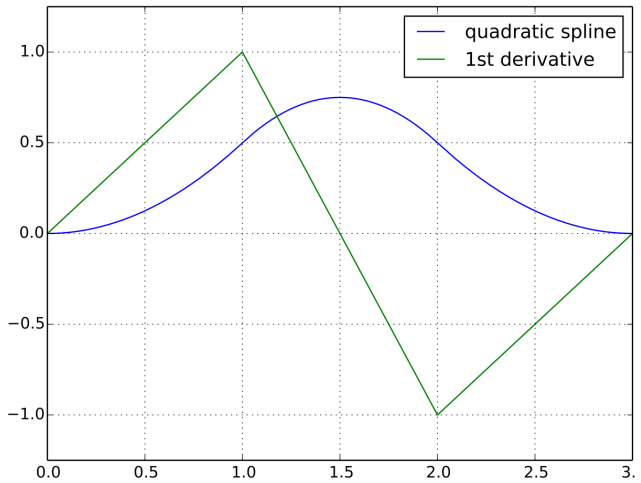


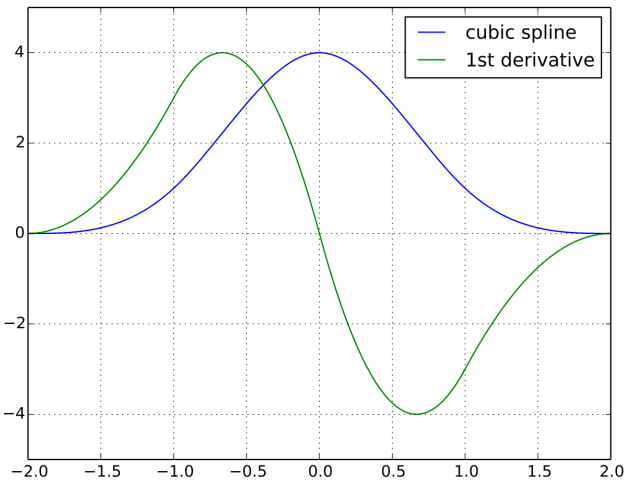
One of the building blocks of computationally efficient wavelet computations is the Haar basis. We construct the basis recursively, starting with a constant function $h_0(x)$ on a bounded domain, for example $R = [0, 1]$. Then,

$$h_1(x) = \begin{cases} 1 \text{ for } x < \frac{1}{2} \\ -1 \text{ for } x \geq \frac{1}{2} \end{cases}$$

h_0 and h_1 are orthogonal. We continue cutting each domain in half.

4. B-Spline Basis





B-splines are defined recursively from a Haar like basis. Let $B_{i,k}(x) = \mathbb{I}_{R_k}(x)$, the order m spline is

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

The recursive nature of the construction allows for much faster smoothing by B-splines of order 4 than by normal cubic splines.

For a large number N of regions, fitting by cubic splines can be shown to be $O(N^3)$ while under mild sparsity conditions fitting by B-splines is of order $O(N)$.

Definition:

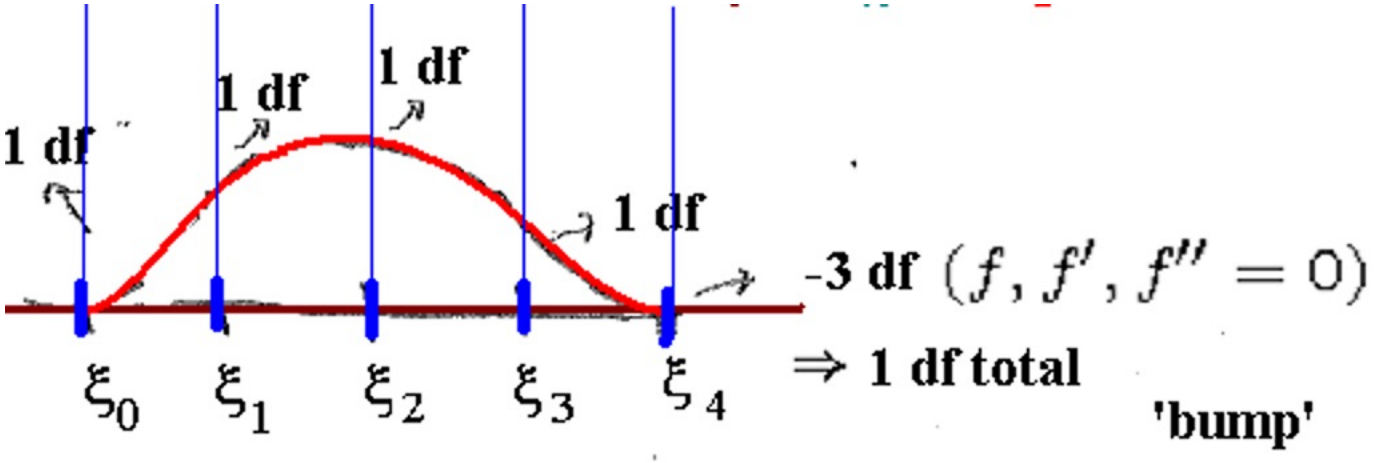
The **support** of a function $f(\vec{x})$ on $\vec{x} \in \mathbb{R}^d$ is the smallest *closed* set A containing the points where $f(\vec{x}) \neq 0$.

A function $f(\vec{x})$ on $\vec{x} \in \mathbb{R}^d$ has **compact support** if its support is a *bounded* set (i.e., contained in some ball in $\mathbb{R}^d$.)

B-spline: A basis function for the vector space $\mathcal{S}$ of splines with fixed knots at $\xi_1, \ldots, \xi_K$ supported on the smallest number of spline intervals.

- Thus B-spline is function with $M-2$ derivatives (including at the pre-assigned knots.
- It is a polynomial of order $M-1$ in each spline interval, and is 0 outside of M intervals.
- It cannot have support over less than M spline intervals.

An illustration for $M = 4$ (cubic B-spline) below:



Note we have 1 df (degree of freedom, i.e. free parameter) to choose in first interval above.

Reason:

Cubic polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3$ has 4 free parameters. However, we need 3 conditions:

$$f(\xi_0) = f'(\xi_0) = f''(\xi_0) = 0$$

in order to have $M - 2 = 2$ continuous derivatives just at ξ_0

Thus have $4 - 1 - 1 - 1 = 4 - 3 = 1$ free parameters in first interval.

Also have
1 more df in second piece,
1 more df in third piece,
1 more df in fourth piece,
and again $-1 - 1 - 1 = -3$ df at end of fourth piece.

Thus, a total of $1 + 1 + 1 + 1 - 3 = 1$ df.

Thus, there is only one free parameter in the above function (i.e. a free multiplicative constant determining the overall height).

The interpolation function One segment depending on the 4 control points A,B,C,D, is given by the polynomial:

$$Q = \frac{A(1 - 3^t + 3t^2 - t^3)}{6} + \frac{B(4 - 6t^2 + 3t^3)}{6} + \frac{C(1 + 3t + 3t^2 - 3t^3)}{6} + \frac{D(t^3)}{6}$$

thus the point at $t = 0$ is given by $A/6 + 4B/6 + C/6$,
and the point at $t = 1$ is given by $B/6 + 4C/6 + D/6$.

➢ Multidimensional Splines (2D)

The multivariate case follows directly from the one variable case. For example, in $\mathbb{R}^2$, if h_{1i} is a spline basis for x_1, and h_{2j} form a spline basis for x_2, we can form the **tensor product basis**

$$g_{ij}(\vec{x}) = h_{1i}(x_1)h_{2j}(x_2)$$

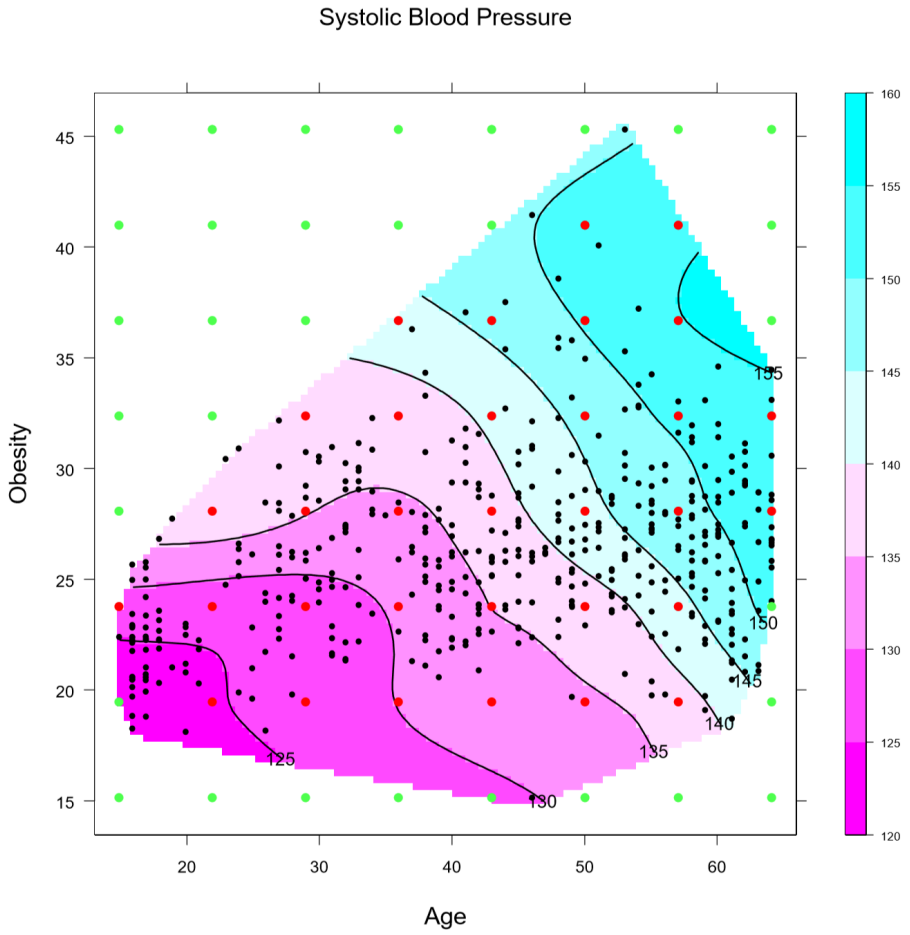
We can then fit

$$g(\vec{x}) = \sum_{i=1}^{M_1}\sum_{j=1}^{M_2} g_{ij}(\vec{x})\,\theta_{ij} = \mathrm{G}^{\mathrm{T}}\Theta$$

For example, consider $h_{1i}(x_1) = x_1^i$ *and* $h_{2j}(x_2) = x_2^j$ *for* $i, j = 1,2,3$. *Then, we have* $g_{ij}(\vec{x})$ *given by* $1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3$.

All the rest are the same as one dimensional.

In practice, the only difficulty is picking the lattice. For a maximum number of degrees of freedom M, one can separate the domain into a lattice of M points and then throw away all points outside the convex hull of the dataset.



Multidimensional Smoothing Splines ((2D))

For smoothing splines, we now need to minimize a function of the form

$$RSS(\vec{\theta}) = \sum_{i=1}^{N} \left(y^{(i)} - f(\vec{x}^{(i)})\right)^2 + \lambda \iint_{\mathbb{R}^2} \left[\left(\frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_2}\right)^2 f(\vec{x})\right]^2 dx_1 dx_2$$

The solutions are smooth 2D surfaces known as ***thin plate splines,***

$$f(\vec{x}) = \theta_0 + \vec{\theta}^T \vec{x} + \sum_{j=1}^{N} \alpha_j h_j(\vec{x})$$

And take the form of ***radial basis functions***

$$h_i(\vec{x}) = \left\|\vec{x} - \vec{x}^{(i)}\right\|^2 \log\left\|\vec{x} - \vec{x}^{(i)}\right\|$$

As before, for $\lambda = 0$ the solution approaches an interpolating function and for $\lambda \to \infty$, the solution approaches the least squares plane.

Textbooks:

Hastie: Chapter 5.

Clarke: Principles and Theory for data mining and machine learning. Chapter 3.

Kernel Smoothing: Principles, Methods and Applications
https://www.wiley.com/en-us/Kernel+Smoothing:+Principles,+Methods+and+Applications-p-9781118456057

Semiparametric Regression

https://bookdown.org/ssjackson300/Machine-Learning-Lecture-Notes/generalised-additive-models.html

Coding Resources:

Matlab:

https://www.mathworks.com/help/curvefit/construction.html

https://www.mathworks.com/help/curvefit/smoothing-splines.html

Python-Scipy:

https://docs.scipy.org/doc/scipy/tutorial/interpolate.html

B-spline:
https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.BSpline.html

https://github.com/madrury/basis-expansions

R:

Further reading:

SplineCNN: Fast Geometric Deep Learning with Continuous B-Spline Kernels

https://openaccess.thecvf.com/content_cvpr_2018/papers/Fey_SplineCNN_Fast_Geometric_CVPR_2018_paper.pdf