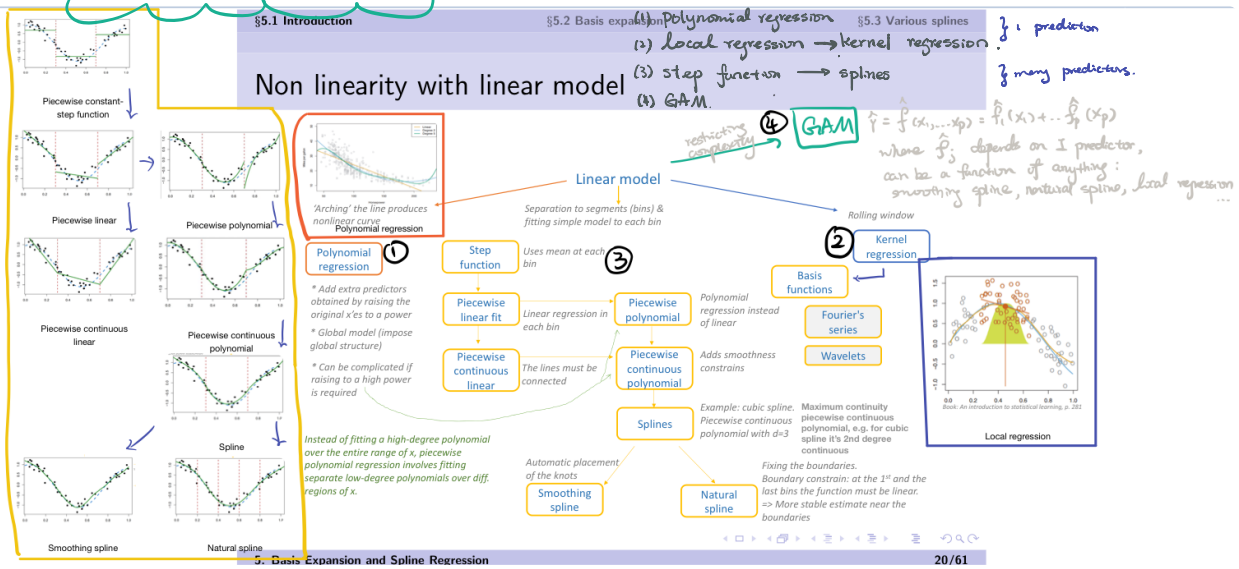
06 Nonlinear models



Suppose $x_{n\times p} = (X_1, \dots X_n)^T$ where $X_i = (x_{i1}, \dots x_{ip})$ and $y_{n\times 1} = (Y_1, \dots Y_n)^T$

A basic function is a function $h_m : \mathbb{R}^p \to \mathbb{R}$. Given a set of basis functions, we can construct a model:

$f(X_i) = \sum_{m=1}^{M} \beta_m h_m(X_i)$ to estimate $E(Y_i)$. =>

- $h_m(x_i) = x_{im}$ for $m = 1, \dots p$ => linear model
- $h_m(x_i) = x_{im}^k$, for $k = 0, 1, 2, \dots$ => polynomial.
- $h_m(x_i) = \max(x_{im}, c)$ for $m = 1, \dots d$ => spline
- $h_m(x_i) = x_{ij} x_{ik}$ => interactions
- $h_m(x_i) = \log(x_{im})$
- ⋮

(1) Polynomial Regression

$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \dots + \beta_d x_i^d + \varepsilon_i$

- linear in coefficient / non-linear in x.
- more interested in the fitted function than the coefficients : $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4$
- can use CV to choose d.
- logistic regression follows naturally, e.g. $P(\text{wage} > 250 | x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \dots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \dots + \beta_d x_i^d)}$ recall: $p = \frac{e^\theta}{1+e^\theta}$ for binomial family.

(2) local regression / kernel regression

local regression:

method: fit a simple linear regression for x_0 using points in the neighbourhood h (i.e. bandwidth); the closer the points to x_0, the higher its weight

note: • the size of neighbourhood (i.e bandwidth) h: $h\downarrow$ - local wiggly fit; $h\uparrow$ - global smooth fit.
• weight is determined by a weighting function.

Kernel regression:

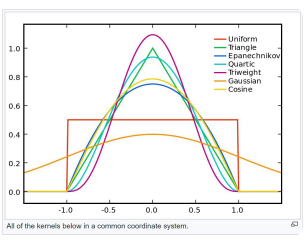
method:

local linear regression [most popular version] : at each x of X, α_x and β_x are chosen to minimise: $\sum_{i=1}^{N} (y_i - \alpha_x - \beta_x x_i)^2 K_\lambda(x - x_i)$ where $K_\lambda(x - x_i) = \frac{1}{\lambda} K\left(\frac{x - x_i}{\lambda}\right)$, $\lambda \in [-h, h]$.

- $f(x) = \alpha + \beta x$ → still behave like a linear function, in a small neighbourhood of each x.
- $K(x)$ is used to determin weight allocation, it needs to be
  - smooth
  - maximal at 0
  - symmetrical around 0.
  - decreasing with respect to $|x|$.
  - optional: $\int K dx = 1$

Common kernel functions are

Name	$K(x)$	Support
Epanechnikov	$\frac{3}{4}(1-x^2)\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1, 1]$
Gaussian	$(2\pi)^{-1/2}\exp\left\{\frac{-x^2}{2}\right\}$	$[-\infty, \infty]$
Biweight	$\frac{15}{16}(1-x^2)^2\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1, 1]$
Triweight	$\frac{35}{32}(1-x^2)^3\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1, 1]$
Uniform	$\frac{1}{2}\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1, 1]$
Tricube	$\frac{70}{81}(1-\lvert x\rvert^3)^3\mathbb{I}_{\{\lvert x\rvert<1\}}$	$[-1, 1]$



- Choosing a particular kernel function is often not important. (default in 'loxfit' package & tricube)
- The chose kernel is scaled by a bandwidth h, i.e. in kernel regression, we will use:
  $K_h(x) = \frac{1}{h} k(\frac{x}{h})$, $h \in [-h, h]$.

steps:

(1) at each x of X, α_x and β_x are chosen to minimise:
$$\sum_{i=1}^n (Y_i - \alpha_x - \beta_x X_i)^2 K_h(x - X_i)$$

(2) So it's effectively a **weighted least squares (WLS)** regression:
$$\Rightarrow (\hat{\alpha}_x, \hat{\beta}_x)^T = [X^T W(x) X]^{-1} X^T W(x) Y \quad \text{where}$$
$$X = \underset{n\times 2}{\begin{pmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \end{pmatrix}^T}, \quad Y = \underset{n\times 1}{(Y_1 \ \cdots \ Y_n)^T} \text{ and}$$
$$W(x) = \underset{n\times n}{\text{diag}[k_h(x - X_1), \ \cdots \ k_h(x - X_n)]}$$

(3) Then we will use $\hat{f}(x) = \hat{\alpha}_x + \hat{\beta}_x x$ to estimate $E(Y)$ at $X = x$.

(4) Reating (1) ~ (3) for a grid of x to give a curve for Y

note:

- $\hat{\alpha}_x$ and $\hat{\beta}_x$ are x-independent.
- the choice of h directly controls the bias-variance tradeoff.
  - h too small → overfitted models (high variance, low bias)
  - h too large → underfitted models (high bias, low variance).
  - can use CV or adaptive choice of h: i.e. using a fixed h_x that is to vary with x.

$t = \frac{\sum_i I_{\{|X_i - x| < h_x\}}}{n}$ (i.e. nearest neighbour bandwidth) ⇒ so h_x is chosen such that window always contains a fixed proportion of data t.
(t: remains constant; h_x: varies with x.)

- we can extend this to **local higher-order (i.e. pth order) polynomial regression smoothing curve**, i.e.
$$\sum_{i=1}^n (Y_i - \beta_{x_0} - \beta_{x_1}(x - X_i) - \beta_{x_2}(x - X_i)^2 - \cdots - \beta_{x_p}(x - X_i)^p]^2 K_h(x - X_i).$$

- **local constant regression smoother**: α_x is chose to minimise
$$\sum_{i=1}^n (Y_i - \alpha_x)^2 k_h(x - X_i) \Rightarrow \hat{\alpha}_x = [\sum_{i=1}^n k_h(x - X_i)]^{-1} \sum_{i=1}^n Y_i k_h(x - X_i)$$

- **kernel regression in higher dimension (higher dimension predictors)**,
  let $x = (x_1, \cdots x_p)^T$ and $k(x) = \prod_{j=1}^p K(x_j) \to$ multivariate kernel for p-vector x.
  Then the local multiple regression for $E(Y|x)$ is $f(x) = \alpha_x + x^T\beta_x$, where α_x and β_x are chosen to minimise
  $\sum_{i=1}^n (Y_i - \alpha_x - X_i^T\beta_x)^2 k_h(X_i - x)$

- The **'curse of dimension'** refers to the phenomenon that as $p \uparrow$, points in the sample tend to be further away with each other. Thus each local neighbourhood of X contains fewer and fewer observations. Hence assessing local behaviour of Y versus X becomes more difficult. ⇒ To cover the same amount of data as lower dimension would, we need to stretch in each direction to effectively select more data in each local regression.
  i.e. $h^p = C$, if C is constant, as $p \uparrow$, $h = C^{1/p} \uparrow$ for $0 < C < 1$.

Nadaraya-Watson estimator:

method:

(1) To find $E(Y|x) = \int_y f(y|x)dy = \int y \frac{f(x,y)}{f(x)} dy$:

(2) a kernel density estimator of $\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^n k(\frac{x - X_i}{h}) = \frac{1}{n} \sum_{i=1}^n K_h(x - X_i)$ ... (1)

(3) a kernel density estimator of $\hat{f}(x,y) = \frac{1}{nh^2} \sum_{i=1}^n k(\frac{x - X_i}{h}) k(\frac{y - Y_i}{h})$ ... (2)

(4) Then $\hat{E}(Y|x) = \hat{\alpha}_x = \underbrace{[\sum_{i=1}^n k_h(x - X_i)]^{-1} \sum_{i=1}^n Y_i k_h(x - X_i)}_{\text{local constant regression smoother}} = \int_y \frac{(2)}{(1)} dy$.

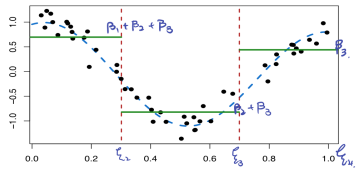
(3) Splines

method: $\beta_1, \beta_2, \ldots \beta_m$ are decided by least square. and $f(x) = \sum_m \beta_m b_m(x)$

(1) Piecewise constant fit:

$b_1(x) = I(x < \xi_2)$
$b_2(x) = I(x < \xi_3)$
$b_3(x) = I(x < \xi_4)$.

$\Rightarrow f(x) = \beta_1 b_1(x) + \beta_2 b_2(x) + \beta_3 b_3(x)$.



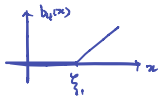
How do you like this as a fit?

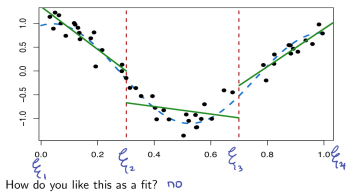
(2) Piecewise linear fit:

intercept terms.

$b_1(x) = I(x < \xi_2)$
$b_2(x) = I(x < \xi_3)$
$b_3(x) = I(x < \xi_4)$
$b_4(x) = \max(x - \xi_1, 0)$
$b_5(x) = \max(x - \xi_2, 0)$
$b_6(x) = \max(x - \xi_3, 0)$

$\Rightarrow f(x) = \sum_{m=1}^{6} \beta_m b_m(x)$.





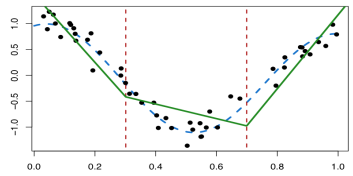
How do you like this as a fit? no

(3) Piecewise continuous linear fit: ($\beta_1 = \beta_2 = 0$)

intercept terms. $b_3(x) = I(x < \xi_4)$

$b_4(x) = \max(x - \xi_1, 0)$
$b_5(x) = \max(x - \xi_2, 0)$
$b_6(x) = \max(x - \xi_3, 0)$

$\Rightarrow f(x) = \sum_{m=3}^{6} \beta_m b_m(x)$.

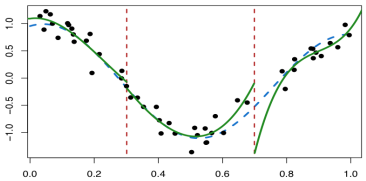


How do you like this as a fit?

(4) Piecewise cubic polynomial fit:

$b_1(x) = \mathbb{I}(x < \xi_2)$
$b_2(x) = \mathbb{I}(x < \xi_3)$
$b_3(x) = \mathbb{I}(x < \xi_4)$
$b_4(x) = \max(x - \xi_1, 0)$
$b_5(x) = \max(x - \xi_2, 0)$
$b_6(x) = \max(x - \xi_3, 0)$
$b_7(x) = \max(x - \xi_1, 0)^2$
$b_8(x) = \max(x - \xi_2, 0)^2$
$b_9(x) = \max(x - \xi_3, 0)^2$
$b_{10}(x) = \max(x - \xi_1, 0)^3$
$b_{11}(x) = \max(x - \xi_2, 0)^3$
$b_{12}(x) = \max(x - \xi_3, 0)^3$

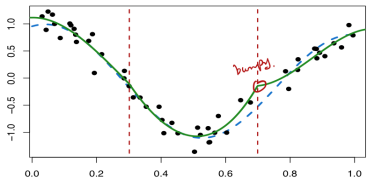
parameters = (# knots − 1)(degree + 1) = 3 × 4 = 12.



(5) Piecewise continuous cubic polynomial fit: ($\beta_1 = \beta_2 = 0$)

$b_3(x) = \mathbb{I}(x < \xi_4)$
$b_4(x) = \max(x - \xi_1, 0)$
$b_5(x) = \max(x - \xi_2, 0)$
$b_6(x) = \max(x - \xi_3, 0)$
$b_7(x) = \max(x - \xi_1, 0)^2$
$b_8(x) = \max(x - \xi_2, 0)^2$
$b_9(x) = \max(x - \xi_3, 0)^2$
$b_{10}(x) = \max(x - \xi_1, 0)^3$
$b_{11}(x) = \max(x - \xi_2, 0)^3$
$b_{12}(x) = \max(x - \xi_3, 0)^3$

parameters = (# knots − 1)(degree) + 1 = 3 × 3 + 1 = 10.

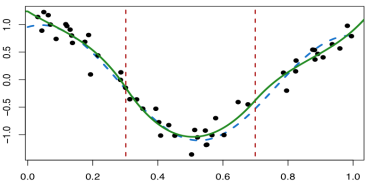


(6) Piecewise cubic fit with 2nd degree continuous ($\beta_1 = \beta_2 = \beta_5 = \beta_6 = 0$)

2nd degree continuous = 1st derivative continuous.

$b_3(x) = \mathbb{I}(x < \xi_4)$
$b_4(x) = \max(x - \xi_1, 0)$
$b_7(x) = \max(x - \xi_1, 0)^2$
$b_8(x) = \max(x - \xi_2, 0)^2$
$b_9(x) = \max(x - \xi_3, 0)^2$
$b_{10}(x) = \max(x - \xi_1, 0)^3$
$b_{11}(x) = \max(x - \xi_2, 0)^3$
$b_{12}(x) = \max(x - \xi_3, 0)^3$

parameters = (# knots − 1) (degree − 1) + 2 = 8

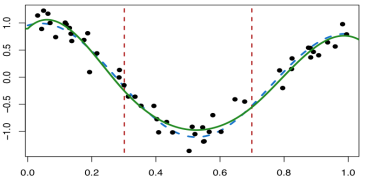


(7) Piecewise cubic fit with 3rd degree continuous. ($\beta_1 = \beta_2 = \beta_5 = \beta_6 = \beta_8 = \beta_9 = 0$)

3rd degree continuous = 2nd derivative continuous.

$b_3(x) = \mathbb{I}(x < \xi_4)$
$b_4(x) = \max(x - \xi_1, 0)$
$b_7(x) = \max(x - \xi_1, 0)^2$
$b_{10}(x) = \max(x - \xi_1, 0)^3$
$b_{11}(x) = \max(x - \xi_2, 0)^3$
$b_{12}(x) = \max(x - \xi_3, 0)^3$

parameters = (# knots − 1) (degree − 2) + 3 = 6



note:

- knot placement directly control bias-variance trade-off. { ↑ # knots ⇒ ↑ # parameters ⇒ overfitted model; ↓ # knots ⇒ ↓ # parameters ⇒ underfitted model
  (e.g. usually lower-quartile, median and upper-quartile).

Also known as cubic spline : { ξ_1 and ξ_k are boundary knots. up to 2nd derivative continuous.

$b_1(x) = 1$

$b_2(x) = x$

$b_3(x) = x^2$

$b_4(x) = \max(x - \xi_k, 0)^3$, where $k = 1, 2, \dots k-1$.

parameters = (# knot - 1)(degree - 2) + 3 = (k-1) + 3 = k + 2

$$\Rightarrow \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_1 \end{pmatrix} + \beta_2 \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \beta_3 \begin{pmatrix} x_1^2 \\ \vdots \\ x_n^2 \end{pmatrix} + \beta_4 \begin{pmatrix} (x_1 - \xi_1, 0)_+^3 \\ \vdots \\ (x_n - \xi_1, 0)_+^3 \end{pmatrix} + \dots + \beta_{k+2} \begin{pmatrix} [x_1 - \xi_{k-1}, 0)_+^3 \\ \vdots \\ (x_n - \xi_{k-1}, 0)_+^3 \end{pmatrix}$$

↳ Can be ignored because can be automatically generated. so there will be k+1 columns.

More generally, if we have p predictors, we need to use B-spline

$$\Rightarrow \begin{pmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{pmatrix} = \beta_2 \begin{pmatrix} x_{11} \\ \vdots \\ x_{1n} \end{pmatrix} + \dots + \beta_2 \begin{pmatrix} x_{p1} \\ \vdots \\ x_{pn} \end{pmatrix} + \dots + \beta_{k+2} \begin{pmatrix} [x_{11} - \xi_{k-1}, 0)_+^3 \\ \vdots \\ (x_{1n} - \xi_{k-1}, 0)_+^3 \end{pmatrix} + \dots + \beta_{k+2} \begin{pmatrix} [x_{p1} - \xi_{k-1}, 0)_+^3 \\ \vdots \\ (x_{pn} - \xi_{k-1}, 0)_+^3 \end{pmatrix}$$

i.e. repeat the basic functions for p times. ⇒ p(k+1) columns.

⇒ combine the columns together we have an $n \times (p(k+1))$ matrix ⇒ cubic spline basic matrix.

Since cubic splines near boundary knots are volatile, we can impose linearity beyond the boundary

⇒ natural cubic spline : { linear beyond boundary. up to 2nd derivative continuous. works well with p predictors as well

$b_1(x) = 1$

$b_2(x) = x$

$b_3(x) = d_k - d_{k-1}$, $k = 1, \dots k-2$ where $d_k = \dfrac{\max(x - \xi_k, 0)^3 - \max(x - \xi_{k_*}, 0)^3}{\xi_{k_*} - \xi_k}$, (k_* → # knots)

parameters = # basic functions = k. e.g. if k = 4, $b_1 = 1$, $b_2 = x$, $b_3 = d_1 - d_3$, $b_4 = d_2 - d_3$

If we don't want to place knot manually, we can use smoothing spline:

i.e. fit a model $f(x)$ that minimise :

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

(up to 2nd derivative continuous. → smoothing parameter.)

fitness: $\sum_{i=1}^{n} \{ y_i - f(x_i) \}^2$; smoothness: $\lambda \int \{ f''(t) \}^2 dt$

{ if $\lambda \to 0$, f is any twice differentiable function; if $\lambda \to \infty$, linear least square.

Theorem: RSS is uniquely minimized by natural cubic spline with knots at every observation.

$$\Rightarrow RSS = \sum_{i=1}^{n} (y_i - B_i^T \theta)^2 + \lambda \sum_{j,k=1}^{n} \theta_j \theta_k \int b_j''(t) b_k''(t) dt$$

$$= \sum_i (y_i - B_i^T \theta)^2 + \lambda \theta \Omega \theta^T \text{ where } \Omega_{j,k} = \int b_j''(t) b_k''(t) dt.$$

and $b_i''(t) = 6(x - \xi_i)$ because $b_i(t) = (x - \xi_i)^3$

$$\Rightarrow \hat{\theta} = (B^T B + \lambda \Omega)^{-1} B^T Y.$$

Let $S_\lambda = B(B^T B + \lambda \Omega)^{-1} B^T \Rightarrow \hat{f}(x) = \hat{Y} = B\hat{\theta} = S_\lambda Y$

$\Rightarrow S_\lambda$ = smoother matrix,

$\Rightarrow$ trace(S_λ) can be treated as degree of freedom

(4) GAM.

$f(x_{i1}, \dots x_{ip}) = \alpha + f_1(x_{i1}) + \cdots f_p(x_{ip})$ where $f_i(x)$ can take many forms but is a function of 1 variable.

for logistic regression: $\log\frac{P(y_i=1)}{\log(1-P(y_i=1))} = \alpha + f_1(x_{i1}) + \dots f_p(x_{ip})$

parameters are chosen to minimise

$$\sum_{i=1}^{n} \{ y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \}^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j .$$

fitness | smoothness.

but estimating all f_j's simultaneously is difficult. So backfitting algorithm is used:

(1) Initialize $\hat{\alpha} = \bar{y}$, all $\hat{f}_j = 0$.

(2) For $j = 1, \dots p$

$\hat{f}_j$ = smooth fit using $\{x_{ij}\}_{i=1}^n$ for $\{ y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik}) \}_{i=1}^n$ (without $\hat{f}_j(x_{ij})$)

(3) let

$\hat{f}_j = \hat{f}_j - \frac{1}{n}\sum_{i=1}^{n} \hat{f}_j(x_{ij})$

(4) repeat step 2 until convergence.