Moving Beyond Linearity

The truth is never linear!

Moving Beyond Linearity

The truth is never linear!
Or almost never!

Moving Beyond Linearity

The truth is never linear!
Or almost never!

But often the linearity assumption is good enough.

Moving Beyond Linearity

See board for distinction between linear regression & linear models

- C1(x), C2(x), ... Cd(x) are basis functions for linear model

The truth is never linear!
Or almost never!

But often the linearity assumption is good enough.

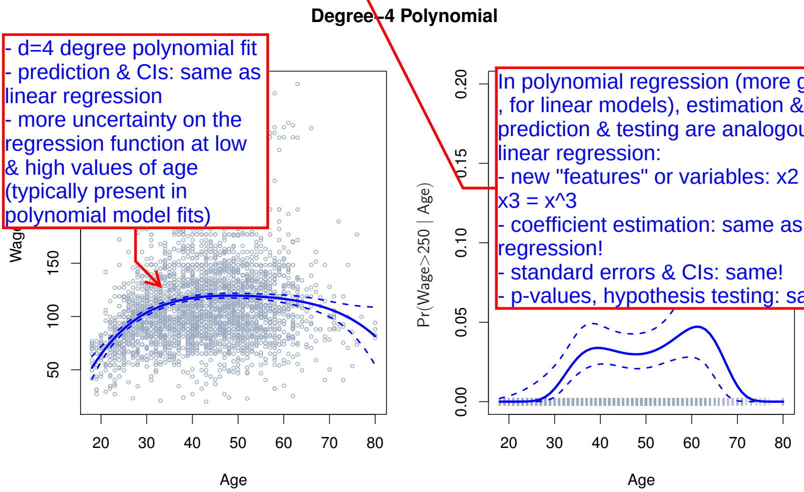
When its not ...

- polynomials,
- step functions,
- splines,
- local regression, and
- generalized additive models

offer a lot of flexibility, without losing the ease and interpretability of linear models.

Polynomial Regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \ldots + \beta_d x_i^d + \epsilon_i$$

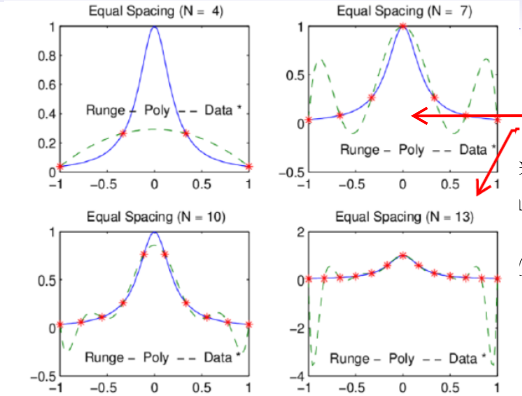


- d=4 degree polynomial fit
- prediction & CIs: same as linear regression
- more uncertainty on the regression function at low & high values of age (typically present in polynomial model fits)

In polynomial regression (more generally, for linear models), estimation & prediction & testing are analogous from linear regression:
- new "features" or variables: x2 = x^2, x3 = x^3
- coefficient estimation: same as linear regression!
- standard errors & CIs: same!
- p-values, hypothesis testing: same!

Details

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.



Runge's phenomenon:
- for certain non-polynomial regression functions, fitting a higher-order polynomial can in fact give poor (highly oscillatory) predictions near domain boundaries
- why? higher-order polynomials are highly volatile - they capture global trends, but cannot capture well local trends.
- this is the key danger of polynomial modeling

X_2

$\hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$

Details

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}_\ell$, can get a simple expression for *pointwise-variances* $\mathrm{Var}[\hat{f}(x_0)]$ at any value x_0. In the figure we have computed the fit and pointwise standard errors on a grid of values for x_0. We show $\hat{f}(x_0) \pm 2 \cdot \mathrm{se}[\hat{f}(x_0)]$.

Details

- Create new variables $X_1 = X$, $X_2 = X^2$, etc and then treat as multiple linear regression.
- Not really interested in the coefficients; more interested in the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4.$$

- Since $\hat{f}(x_0)$ is a linear function of the $\hat{\beta}_\ell$, can get a simple expression for *pointwise-variances* $\mathrm{Var}[\hat{f}(x_0)]$ at any value x_0. In the figure we have computed the fit and pointwise standard errors on a grid of values for x_0. We show $\hat{f}(x_0) \pm 2 \cdot \mathrm{se}[\hat{f}(x_0)]$.
- We either fix the degree d at some reasonably low value, else use cross-validation to choose d.

Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.

Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later).

Details continued

- Logistic regression follows naturally. For example, in figure we model

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

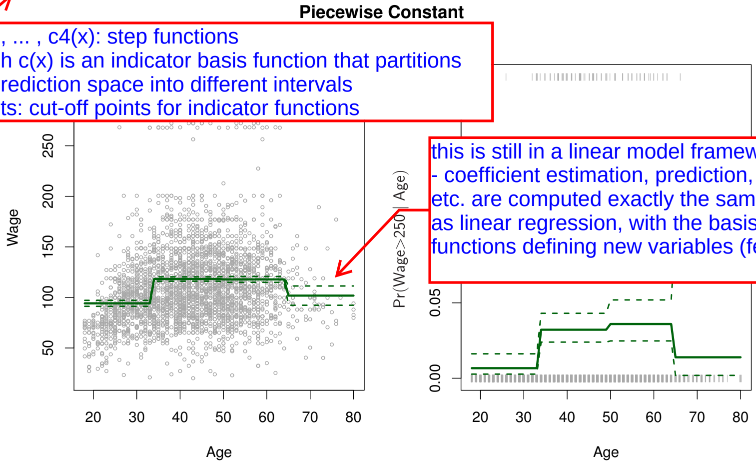
- To get confidence intervals, compute upper and lower bounds on *on the logit scale*, and then invert to get on probability scale.
- Can do separately on several variables—just stack the variables into one matrix, and separate out the pieces afterwards (see GAMs later).
- Caveat: polynomials have notorious tail behavior — very bad for extrapolation.
- Can fit using $\texttt{y} \sim \texttt{poly}(\texttt{x}, \texttt{degree} = \texttt{3})$ in formula.

Step Functions

Another way of creating transformations of a variable — cut the variable into distinct regions.

$$C_1(X) = I(X < 35), \quad C_2(X) = I(35 \leq X < 50), \ldots, C_3(X) = I(X \geq 65)$$

c1(x), ... , c4(x): step functions
- each c(x) is an indicator basis function that partitions the prediction space into different intervals
- knots: cut-off points for indicator functions



this is still in a linear model framework:
- coefficient estimation, prediction, CIs, etc. are computed exactly the same way as linear regression, with the basis functions defining new variables (features)

Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.

Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\texttt{Year} < 2005) \cdot \texttt{Age}, \quad I(\texttt{Year} \geq 2005) \cdot \texttt{Age}$$

  would allow for different linear functions in each age category.

Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\texttt{Year} < 2005) \cdot \texttt{Age}, \quad I(\texttt{Year} \geq 2005) \cdot \texttt{Age}$$

  would allow for different linear functions in each age category.
- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.

Step functions continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of `Year` and `Age`:

$$I(\texttt{Year} < 2005) \cdot \texttt{Age}, \quad I(\texttt{Year} \geq 2005) \cdot \texttt{Age}$$

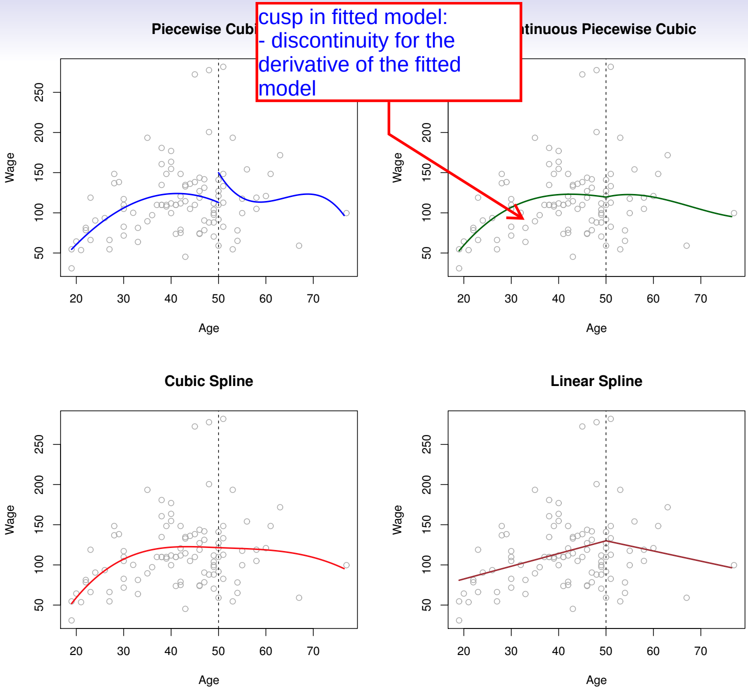
  would allow for different linear functions in each age category.
- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.
- Choice of cutpoints or *knots* can be problematic. For creating nonlinearities, smoother alternatives such as *splines* are available.

Piecewise Polynomials

- Instead of a single polynomial in X over its whole domain, we can rather use different polynomials in regions defined by knots. E.g. (see figure)

$$y_i = \begin{cases} \beta_{01} + \beta_{11}x_i + \beta_{21}x_i^2 + \beta_{31}x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12}x_i + \beta_{22}x_i^2 + \beta_{32}x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

- Better to add constraints to the polynomials, e.g. continuity.
- *Splines* have the "maximum" amount of continuity.



Piecewise Cubic
Continuous Piecewise Cubic
cusp in fitted model:
- discontinuity for the derivative of the fitted model
Cubic Spline
Linear Spline
Wage
Age
20
30
40
50
60
70
50
100
150
200
250

Linear Splines

A linear spline with knots at ξ_k, $k = 1, \dots, K$ is a piecewise linear polynomial continuous at each knot.

We can represent this model as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

where the b_k are *basis functions*.

Claim: linear splines can be represented as the following linear model:

Linear Splines

A linear spline with knots at ξ_k, $k = 1, \dots, K$ is a piecewise linear polynomial continuous at each knot.

typo: K+1 basis functions

We can represent this model as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

where the b_k are *basis functions*.

baseline linear function

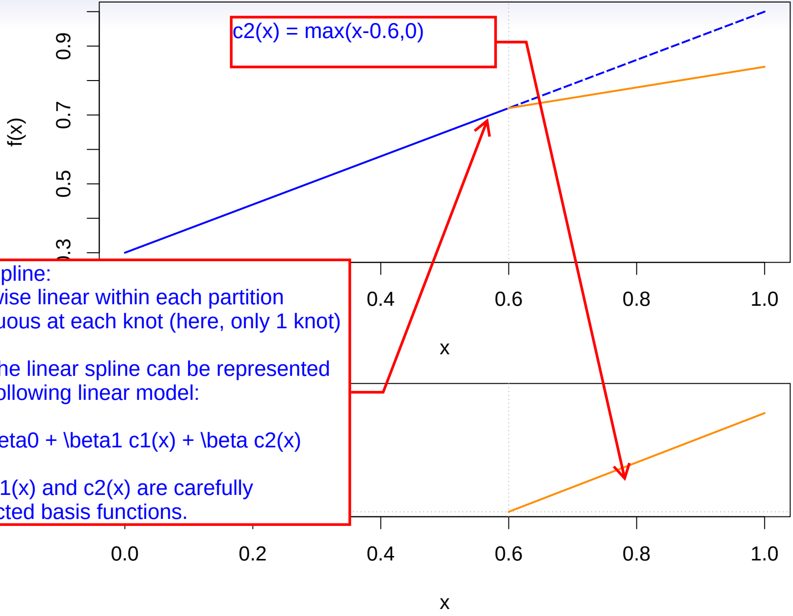
$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k \end{aligned}$$

allows for linear flexibility after knot \xi_k

Here the $()_+$ means *positive part*; i.e.

$$(x_i - \xi_k)_+ = \begin{cases} x_i - \xi_k & \text{if} \\ 0 & \text{ot} \end{cases}$$

this then becomes a linear model:
- coefficient estimation, standard errors, CIs, hypothesis testing: same as for linear regression



Linear spline:
- piecewise linear within each partition
- continuous at each knot (here, only 1 knot)

Claim: the linear spline can be represented by the following linear model:

f(x) = \beta0 + \beta1 c1(x) + \beta c2(x)

where c1(x) and c2(x) are carefully constructed basis functions.

degrees-of-freedom (df): "the number of model parameters to fit". we'll generalize this later for non-integer df.
- model df can be thought of as a proxy for model complexity

- cubic splines with K knots will have K+4 dfs
- linear splines with K knots will have K+2 dfs
- adding additional constraints reduces the number of dfs

...bic Splines

... $\xi_k,\ k = 1, \ldots, K$ *is a piecewise*

...*nuou*... *rder 2 at*

why can we not make this order 3? it would then be the same cubic function over different partitions - we lose our local modeling property!

mo... ower basis

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

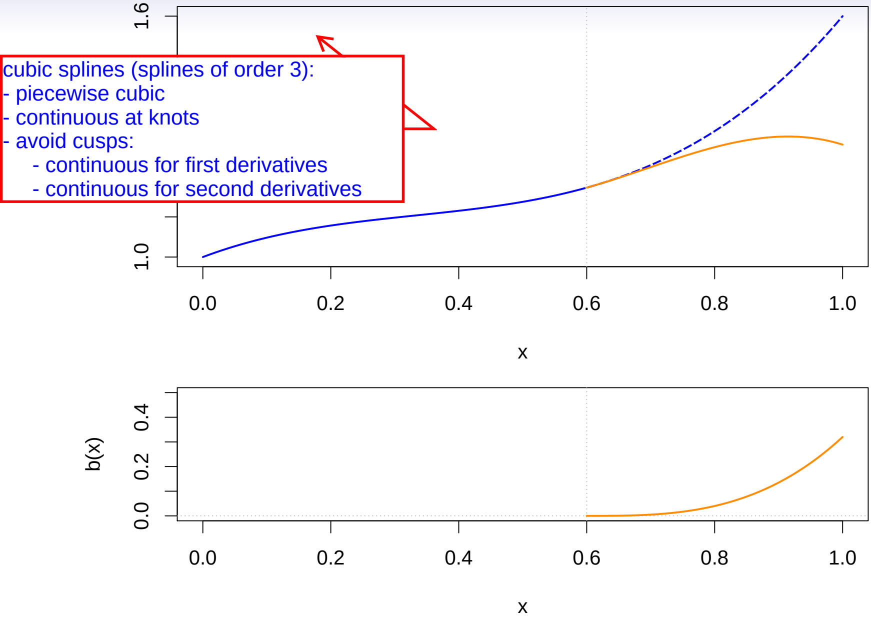
Claim: a cubic spline can be represented as the following linear model
- efficient coefficient estimation, standard errors, CIs, testing: same as what we did in linear regression

$$\begin{aligned} b_1(x_i) &= x_i \\ b_2(x_i) &= x_i^2 \\ b_3(x_i) &= x_i^3 \\ b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \ldots, K \end{aligned}$$

baseline cubic function

where

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$


cubic splines (splines of order 3):
- piecewise cubic
- continuous at knots
- avoid cusps:
- continuous for first derivatives
- continuous for second derivatives
1.6
1.0
0.0
0.2
0.4
0.6
0.8
1.0
x
b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

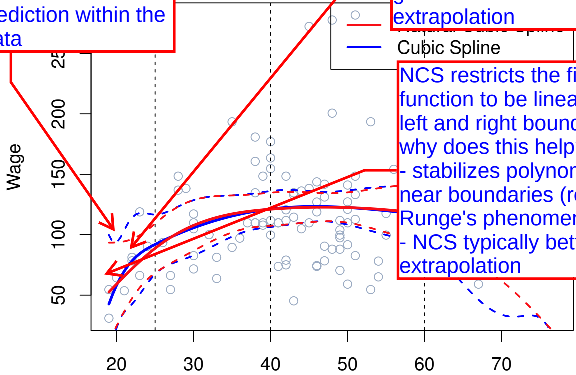
Natural Cubic

NCS with K knots => K dfs
(K+4) - 4 constraints (2 on each side)

A natural cubic spline extrapolates linearly beyond the boundary knots. This adds $4 = 2 \times 2$ extra constraints, and allows us to put more internal knots for the same degrees of freedom as a regular cubic spline.

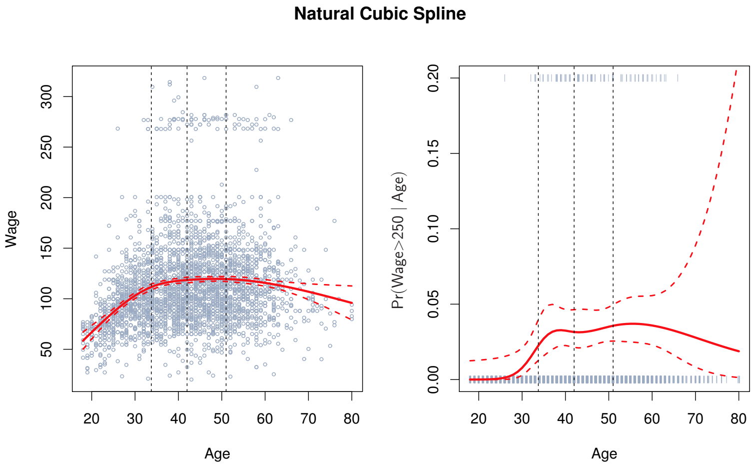
limitation of CS: it can be quite volatile near prediction edges, so it's not good / stable for extrapolation

extrapolate: prediction beyond the range of your data
interpolation: prediction within the range of your data



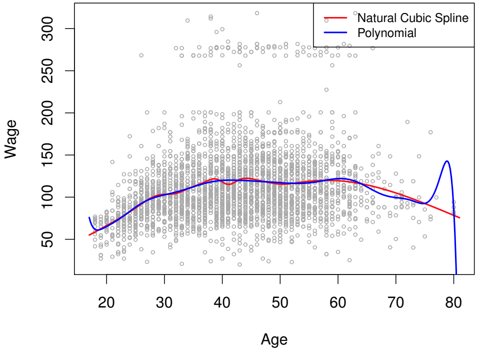
NCS restricts the fitted function to be linear at the left and right boundaries.
why does this help?
- stabilizes polynomial fits near boundaries (recall Runge's phenomenon)
- NCS typically better for extrapolation

Fitting splines in R is easy: `bs(x, ...)` for any degree splines, and `ns(x, ...)` for natural cubic splines, in package `splines`.



Knot placement

- One strategy is to decide K, the number of knots, and then place them at appropriate quantiles of the observed X.
- A cubic spline with K knots has $K + 4$ parameters or degrees of freedom.
- A natural spline with K knots has K degrees of freedom.



Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.

Knot placement

- One strategy is to decide K, the number of knots, and then

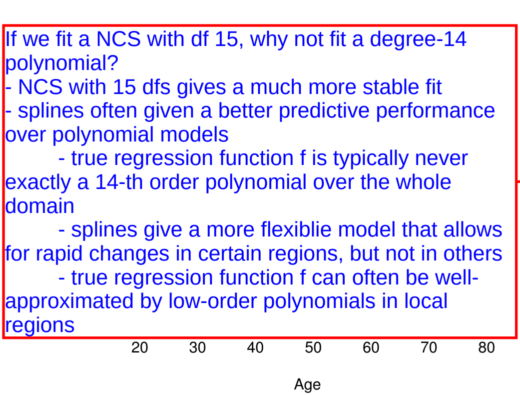
What are different ways to vary the bias-variance trade-off for splines?
- changing the number of knots K
- changing the degree of polynomials within each partition (linear splines, cubic splines, etc.)
- regularization (shrinkage) on the linear model representation of splines

…ntiles of the observed X.

…as $K + 4$ parameters or

… has K degrees of freedom.

If we fit a NCS with df 15, why not fit a degree-14 polynomial?
- NCS with 15 dfs gives a much more stable fit
- splines often given a better predictive performance over polynomial models
    - true regression function f is typically never exactly a 14-th order polynomial over the whole domain
    - splines give a more flexiblie model that allows for rapid changes in certain regions, but not in others
    - true regression function f can often be well-approximated by low-order polynomials in local regions

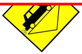


Comparison of a degree-14 polynomial and a natural cubic spline, each with 15df.

```
ns(age, df=14)
poly(age, deg=14)
```

Smoothing Sp

why smoothing splines?
- regularization (shrinkage) can help in estimating coefficients from linear models
- the choice of knot selection can also be subjective

This section is a little bit mathematical 

Consider this criterion for fitting a smooth function $g(x)$ to s

"RSS term": measures training error

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

penalty term:
- penalizes a measure of the smoothness of a function
- the penalty equals 0 when the function g is linear
- the penalty becomes large if the function is very wiggly

Smoothing Splines

This section is a little bit mathematical 

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.

Smoothing Splines

This section is a little bit mathematical 

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is. It is modulated by the *tuning parameter* $\lambda \geq 0$.

Smoothing Splines

This section is a little bit mathematical 

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is. It is modulated by the *tuning parameter* $\lambda \geq 0$.
  - The smaller λ, the more wiggly the function, eventually interpolating y_i when $\lambda = 0$.

Smoothing Splines

This section is a little bit mathematical 

Consider this criterion for fitting a smooth function $g(x)$ to some data:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- The first term is RSS, and tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is. It is modulated by the *tuning parameter* $\lambda \geq 0$.
  - The smaller λ, the more wiggly the function, eventually interpolating y_i when $\lambda = 0$.
  - As $\lambda \to \infty$, the function $g(x)$ becomes linear.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.
- The algorithmic details are too complex to describe here. In R, the function `smooth.spline()` will fit a smoothing spline.

Smoothing Splines continued

The solution is a natural cubic spline, with a knot at every unique value of x_i. The roughness penalty still controls the roughness via λ.

Some details

- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.
- The algorithmic details are too complex to describe here. In R, the function `smooth.spline()` will fit a smoothing spline.
- The vector of n fitted values can be written as $\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y}$, where $\mathbf{S}_\lambda$ is a $n \times n$ matrix (determined by the x_i and λ).
- The *effective degrees of freedom* are given by

$$df_\lambda = \sum_{i=1}^{n} \{\mathbf{S}_\lambda\}_{ii}.$$

Smoothing Splines continued — choosing λ

- We can specify *df* rather than λ!

  In R: `smooth.spline(age, wage, df = 10)`

Smoothing Splines continued — choosing λ

- We can specify *df* rather than λ!

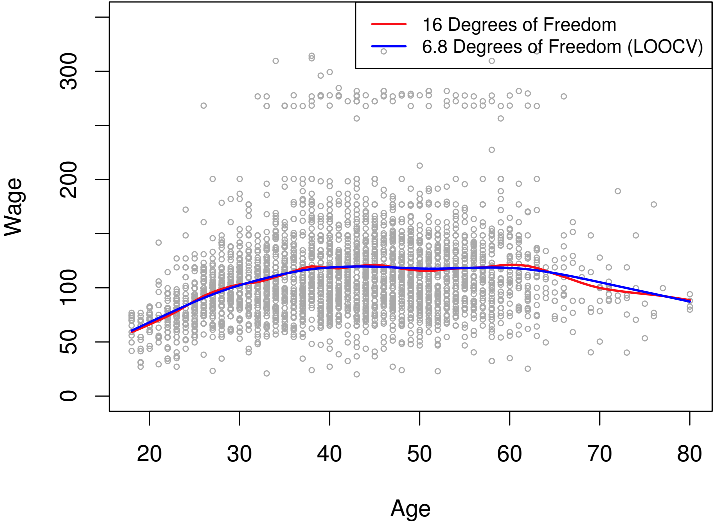
  In R: `smooth.spline(age, wage, df = 10)`

- The leave-one-out (LOO) cross-validated error is given by

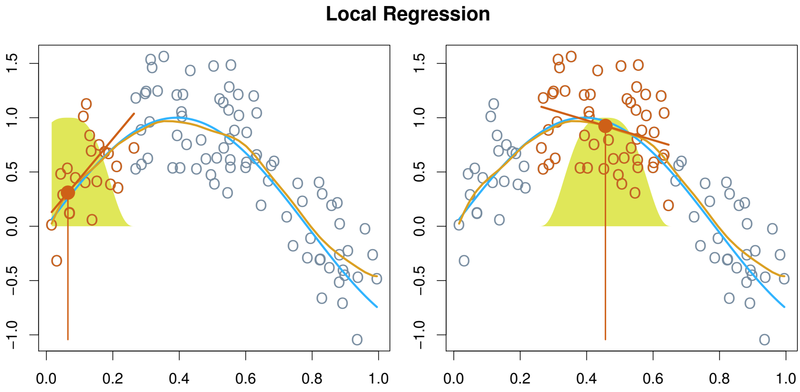
$$\mathrm{RSS}_{cv}(\lambda) = \sum_{i=1}^{n} (y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}} \right]^2.$$

  In R: `smooth.spline(age, wage)`

Smoothing Spline



Local Regression



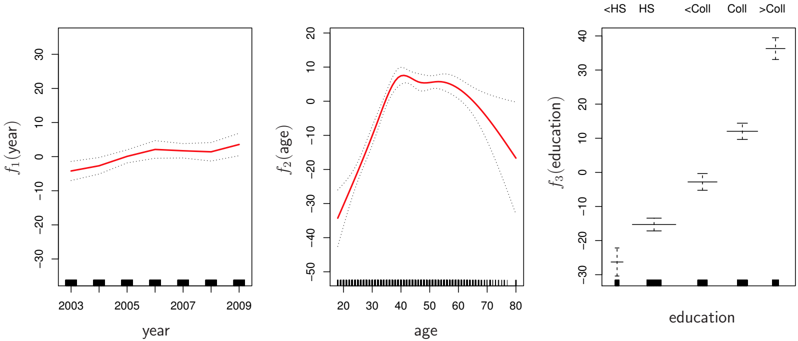
With a sliding weight function, we fit separate linear fits over the range of X by weighted least squares.

See text for more details, and `loess()` function in R.

Generalized Additive Models

Allows for flexible nonlinearities in several variables, but retains the additive structure of linear models.

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i.$$



GAM details

- Can fit a GAM simply using, e.g. natural splines:

  $\texttt{lm}(\texttt{wage} \sim \texttt{ns}(\texttt{year}, \texttt{df} = 5) + \texttt{ns}(\texttt{age}, \texttt{df} = 5) + \texttt{education})$

GAM details

- Can fit a GAM simply using, e.g. natural splines:

  $\texttt{lm}(\texttt{wage} \sim \texttt{ns}(\texttt{year}, \texttt{df} = 5) + \texttt{ns}(\texttt{age}, \texttt{df} = 5) + \texttt{education})$

- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.

GAM details

- Can fit a GAM simply using, e.g. natural splines:

  $\texttt{lm}(\texttt{wage} \sim \texttt{ns}(\texttt{year}, \texttt{df} = 5) + \texttt{ns}(\texttt{age}, \texttt{df} = 5) + \texttt{education})$

- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.

GAM details

- Can fit a GAM simply using, e.g. natural splines:

  $$\texttt{lm}(\texttt{wage} \sim \texttt{ns}(\texttt{year}, \texttt{df} = 5) + \texttt{ns}(\texttt{age}, \texttt{df} = 5) + \texttt{education})$$

- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.
- Can use smoothing splines or local regression as well:

  $$\texttt{gam}(\texttt{wage} \sim \texttt{s}(\texttt{year}, \texttt{df} = 5) + \texttt{lo}(\texttt{age}, \texttt{span} = .5) + \texttt{education})$$

GAM details

- Can fit a GAM simply using, e.g. natural splines:

  `lm(wage ~ ns(year, df = 5) + ns(age, df = 5) + education)`

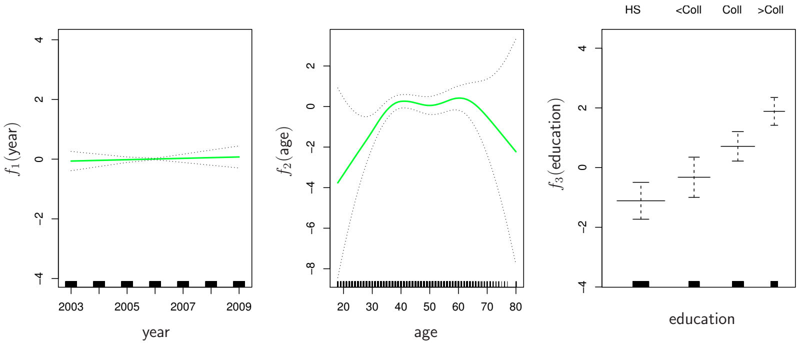
- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.
- Can use smoothing splines or local regression as well:

  `gam(wage ~ s(year, df = 5) + lo(age, span = .5) + education)`

- GAMs are additive, although low-order interactions can be included in a natural way using, e.g. bivariate smoothers or interactions of the form `ns(age,df=5):ns(year,df=5)`.

GAMs for classification

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p).$$



$\texttt{gam(I(wage} > \texttt{250)} \sim \texttt{year} + \texttt{s(age, df} = \texttt{5)} + \texttt{education, family} = \texttt{binomial)}$