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

The truth is never linear!
Or almost never!

But often the linearity assumption is good enough.

When its not . . .

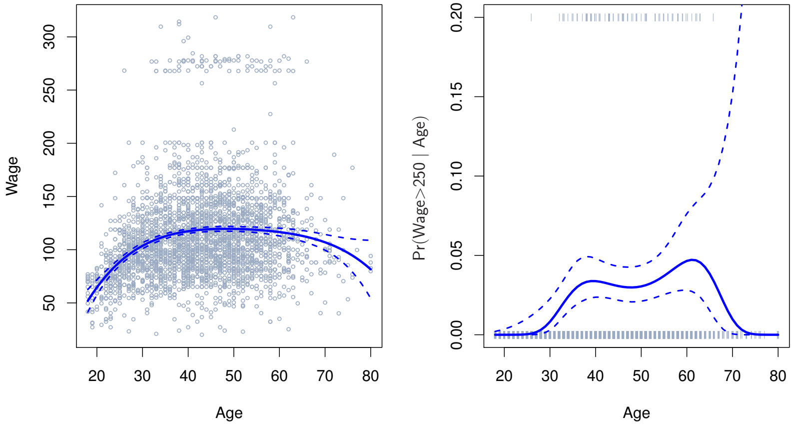
- polynomials,
- step functions,
- splines,
- local regression, and
- generalized additive models

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

Polynomial Regression

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

Degree−4 Polynomial



Details

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

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:

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

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:

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

```
x' = (1,x0,x0^2,x0^3,x0^4). fhat = x'*betahat
betahat ~ MVN(beta, Sigma); => Var(x'*betahat)=x'*Sigma*x;
```

- 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.

x' = (1,x0,x0^2,x0^3,x0^4). fhat = x'*betahat
betahat ~ MVN(beta, Sigma); => Var(x'*betahat)=x'*Sigma*x;

=> this gives the CI for logit(p(x0)); then apply "expit" (=logit^(-1)) to both sides

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

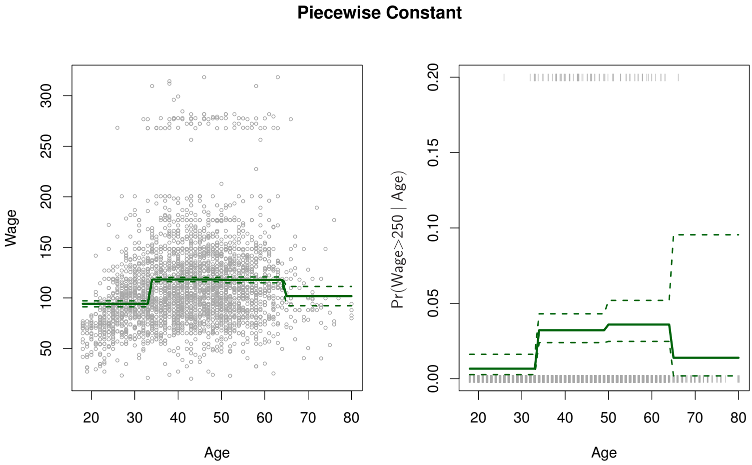
$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}{1 + \exp(\beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \ldots + \beta_d x_i^d)}.$$

- 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 $\mathtt{y} \sim \mathtt{poly}(\mathtt{x}, \mathtt{degree} = \mathtt{3})$ in formula.

Step Functions

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

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



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`:

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

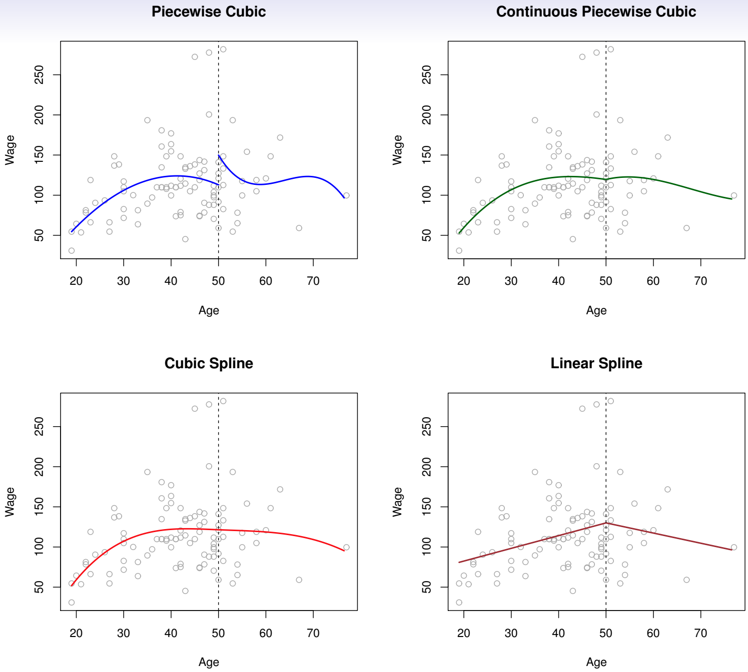
  would allow for different linear functions in each age category.
- In R: `I(year < 2005)` or `cut(age, c(18, 25, 40, 65, 90))`.
- Choice of cutpoints or *knots* can be problematic. For creating nonlinearities, smoother alternatives such as *splines* are available.

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
Cubic Spline
Linear Spline
Wage
Age
50
100
150
200
250
20
30
40
50
60
70

Linear Splines

A linear spline with knots at ξ_k, $k = 1, \ldots, K$ *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*.

Linear Splines

A linear spline with knots at ξ_k, $k = 1, \ldots, K$ 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*.

K+1 rather than K+3

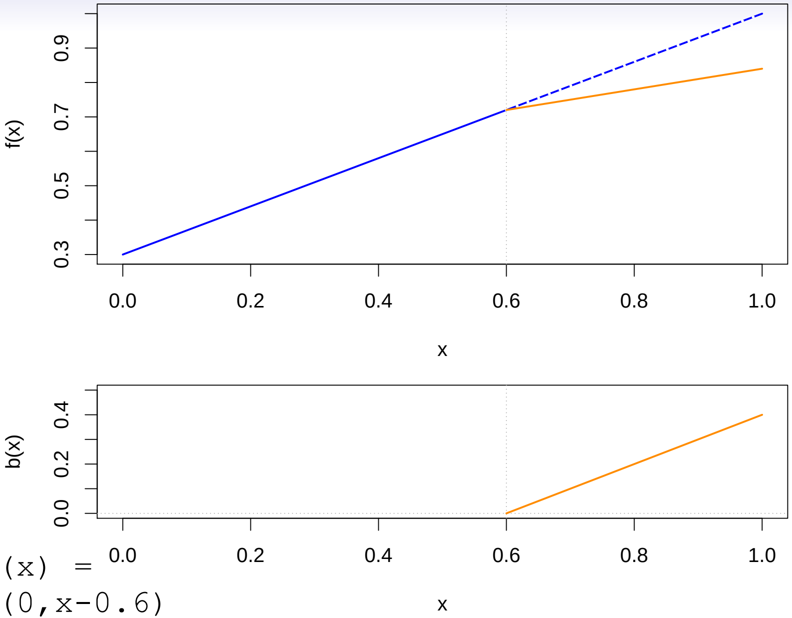
$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k = 1, \ldots, K \end{aligned}$$

= max(0, x_i - xi_k)

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

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

`1* b_1(x) - 0.5*b_2(x)`



`b_2(x) = max(0,x-0.6)`

Cubic Splines

A cubic spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot.

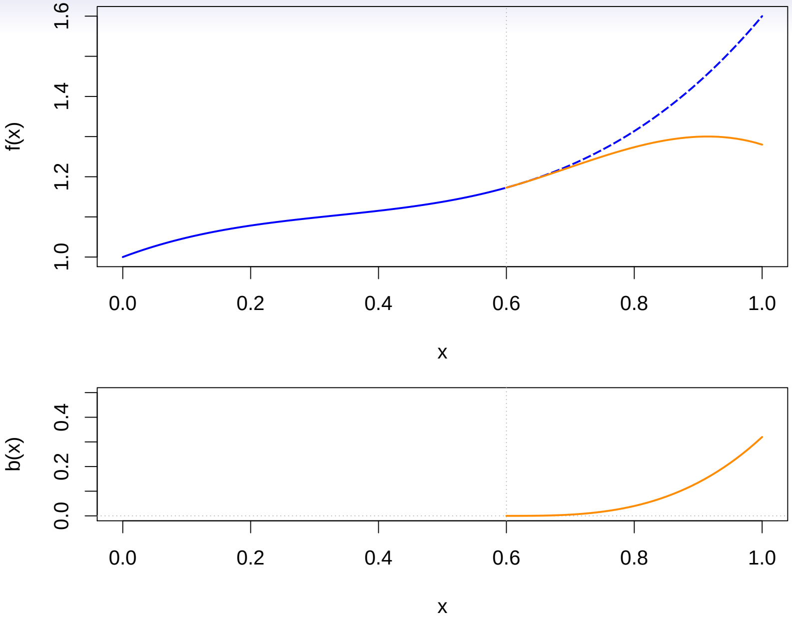
Again we can represent this model with truncated power basis functions

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

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

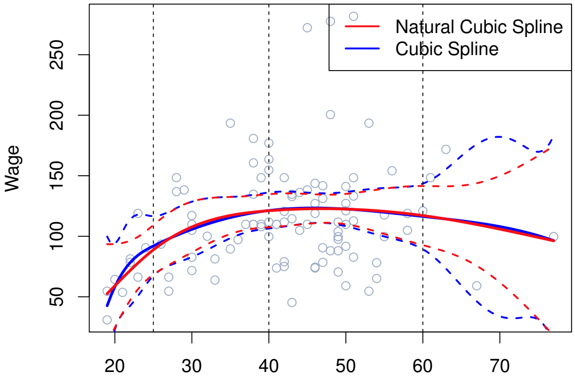
where

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

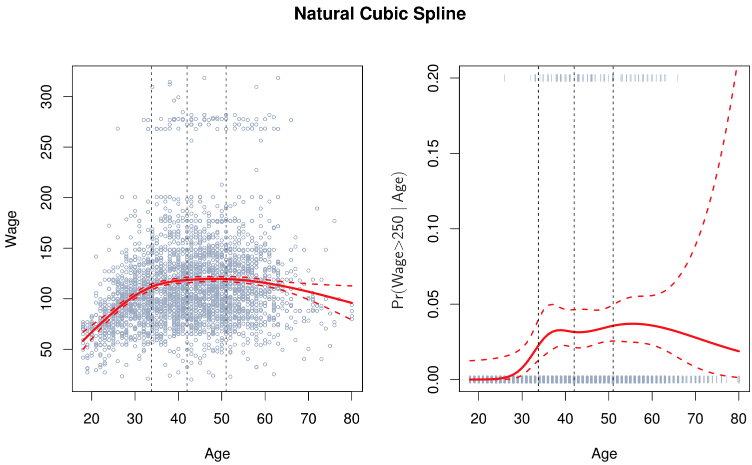

f(x)
1.0
1.2
1.4
1.6
0.0
0.2
0.4
0.6
0.8
1.0
x
b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

Natural Cubic Splines

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.

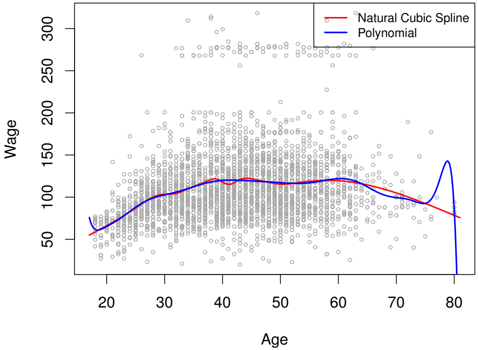


Fitting splines in R is easy: `bs(x, ...)` for any degree splines, and `ns(x, ...)` for natural cubic splines, in package `splines`.



Knot placement

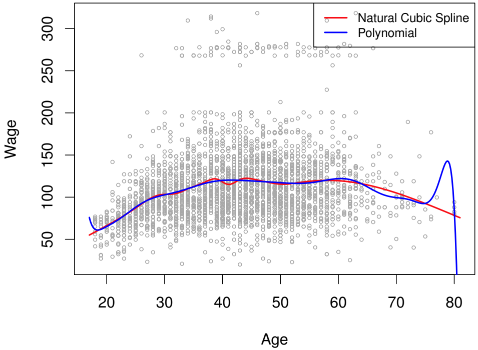
- One strategy is to decide K, the number of knots, and then place them at appropriate quantiles of the observed X.
- A cubic spline with K knots has $K + 4$ parameters or degrees of freedom.
- A natural spline with K knots has K degrees of freedom.



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 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.

`ns(age, df=14)`

`poly(age, deg=14)`

Smoothing Splines

This section is a little bit mathematical 

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

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

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

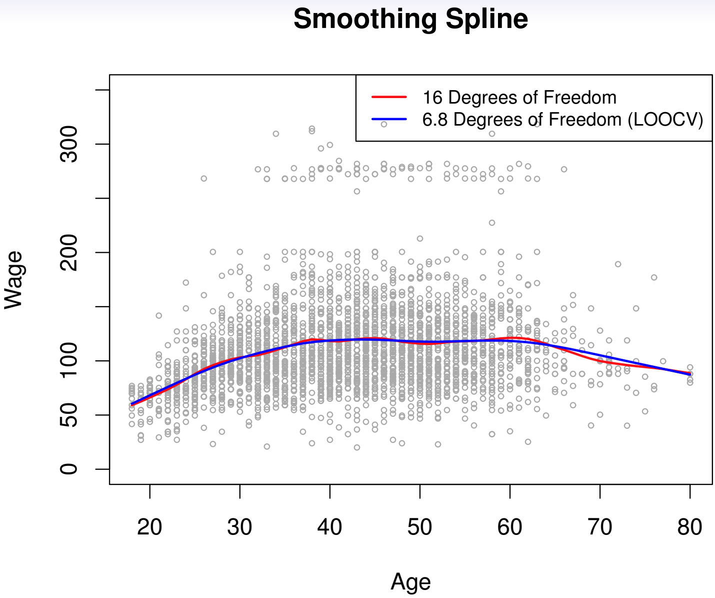
- We can specify *df* rather than λ!

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

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

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

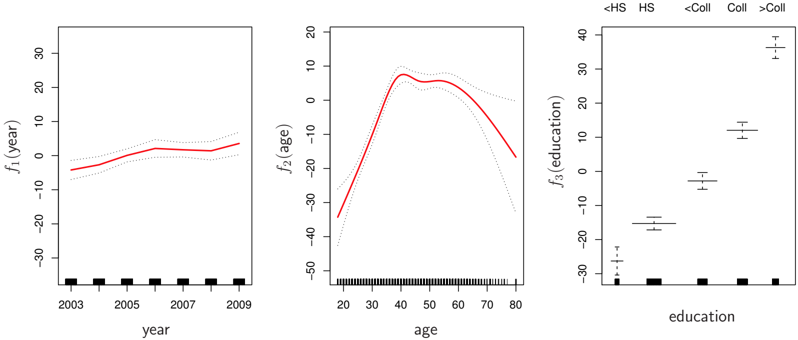
  In R: `smooth.spline(age, wage)`


Smoothing Spline
16 Degrees of Freedom
6.8 Degrees of Freedom (LOOCV)
Wage
0
50
100
200
300
20
30
40
50
60
70
80
Age

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:

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

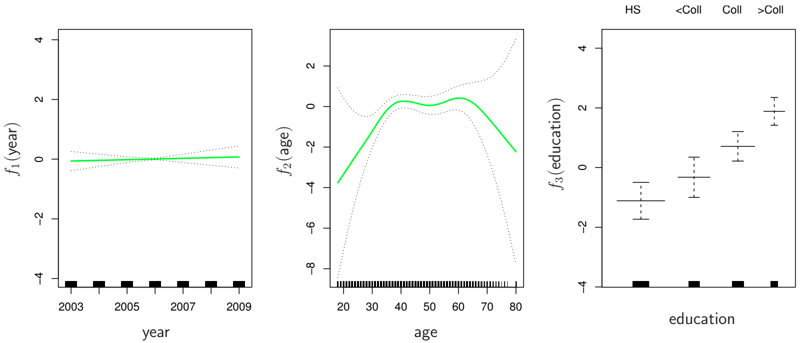
- Coefficients not that interesting; fitted functions are. The previous plot was produced using `plot.gam`.
- Can mix terms — some linear, some nonlinear — and use `anova()` to compare models.
- Can use smoothing splines or local regression as well:

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

- 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)`.

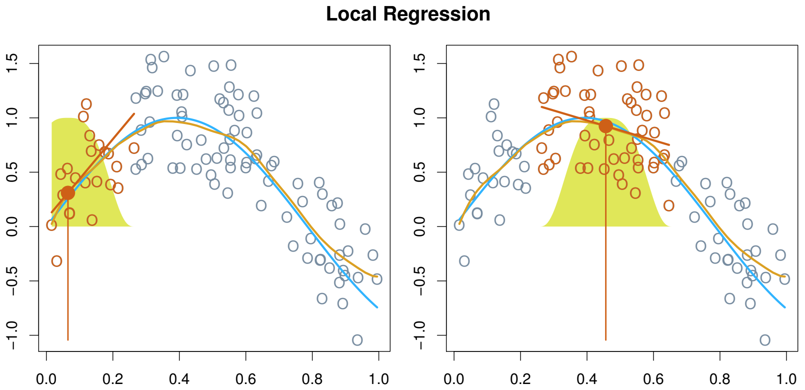
GAMs for classification

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



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

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.