Introduction to Statistical Data Science

Dr. Francois-Xavier Briol
Department of Statistical Science,
UCL

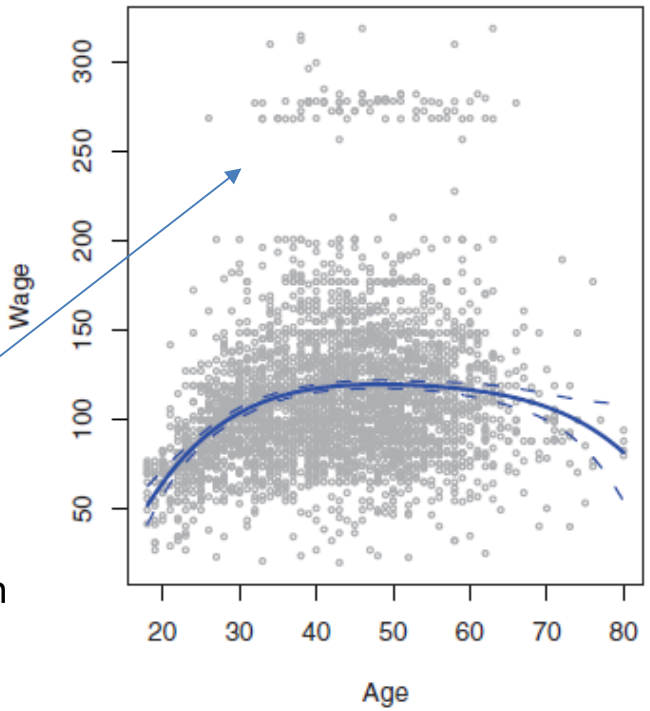
SMALL STEPS BEYOND LINEARITY

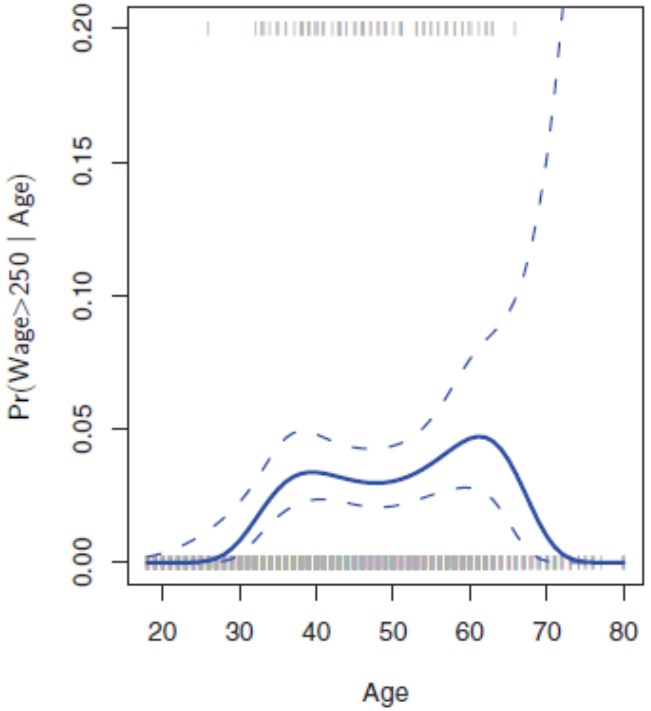
Polynomial Regression Revisited

- We can take a covariate x and expand it in terms of x, x^2, x^3 etc.
- What about confidence intervals in the original space? How do they behave?
  - Polynomials of degree > 3 are not particularly trustworthy. Avoid them for all practical purposes (Look up Runge's Phenomenon!).

Example: 4th Order Logistic Regression

$$\log \frac{P(Wage > 250 \mid Age = x)}{P(Wage \leq 250 \mid Age = x)} = \hat{\beta}_0 + \hat{\beta}_1 x + \hat{\beta}_2 x^2 + \hat{\beta}_3 x^3 + \hat{\beta}_4 x^4$$





Not great as a regression model

(ISLR, Chapter 7)

Flexible Models from Local Basis Functions: the Step Case

- Any parameter in a polynomial regression problem has a global implication (why?).
- Let's think in terms of more general **basis functions**, nonlinear transformations of our inputs.
  - Sometimes, a collection of basis functions is called a **dictionary**.
  - If you have a signal processing background, you may be familiar with concepts such as Fourier or wavelet decomposition.
- In particular, an alternative for flexible regression is the use of **local**, or **sparse**, basis functions: basis functions which are zero in most of the input space.

Binning

- Create cutpoints in the range of covariate *X*:

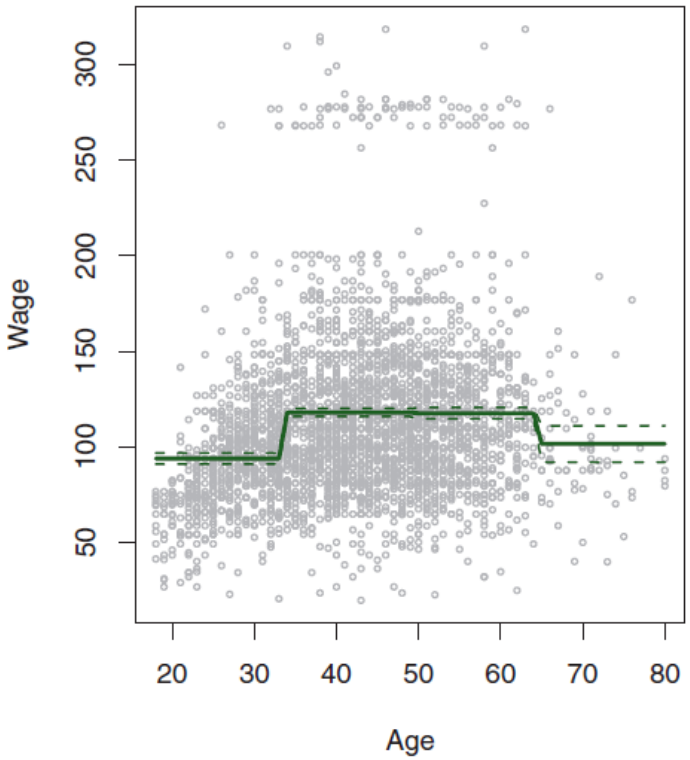
$$
\begin{aligned}
C_0(X) &= I(X < c_1) \\
C_1(X) &= I(c_1 \leq X < c_2) \\
C_2(X) &= I(c_2 \leq X < c_3) \\
&\cdots \\
C_{K-1}(X) &= I(c_{K-1} \leq X < c_K) \\
C_K(X) &= I(X \leq c_K)
\end{aligned}
$$

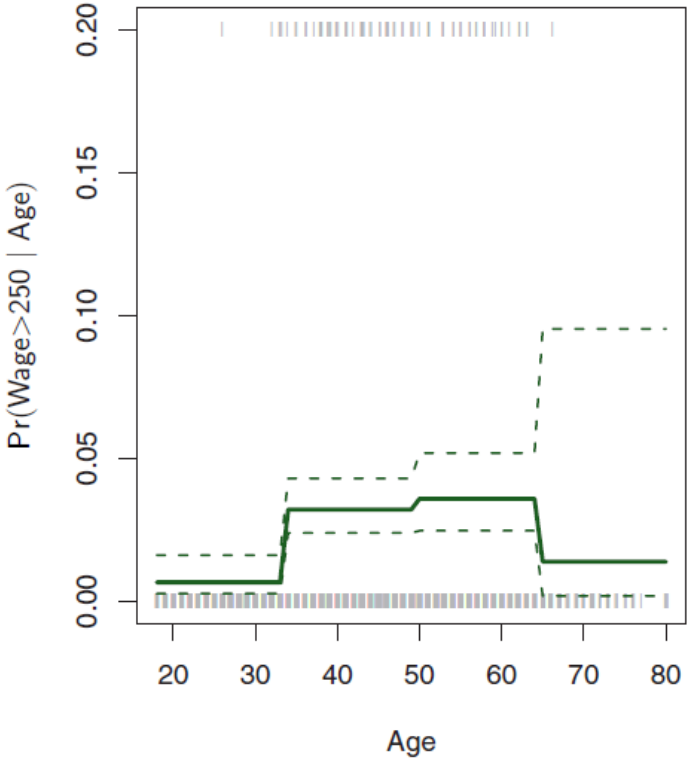
where once again remember that *I*() is an indicator function (outputs 0 or 1 depending whether argument is false or true).

Example

- What is the interpretation of β_j? *The average increase in the response when x is in the interval.*
- Which shortcomings do you see?

$$Y = \beta_0 + \beta_1 C_1(x) + \cdots + \beta_K C_K(x) + \epsilon$$





This week

- This week we will discuss a range of models which allow us to go beyond the linear framework.
- Most of these can be thought of as models using basis functions: more on that when we get to Generalised Additive Models.
- As we go through this material, try to think about the advantages and disadvantages of each approach.

REGRESSION SPLINES

Alternative: Regression Splines

- This combines ideas from the polynomial and step basis functions.
- Basic insight: fit **separate (low-degree) polynomials** over different regions of the input space. This is sometimes called a "piecewise" model.
- Most common building block: cubic polynomials.

$$Y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$$

Knots

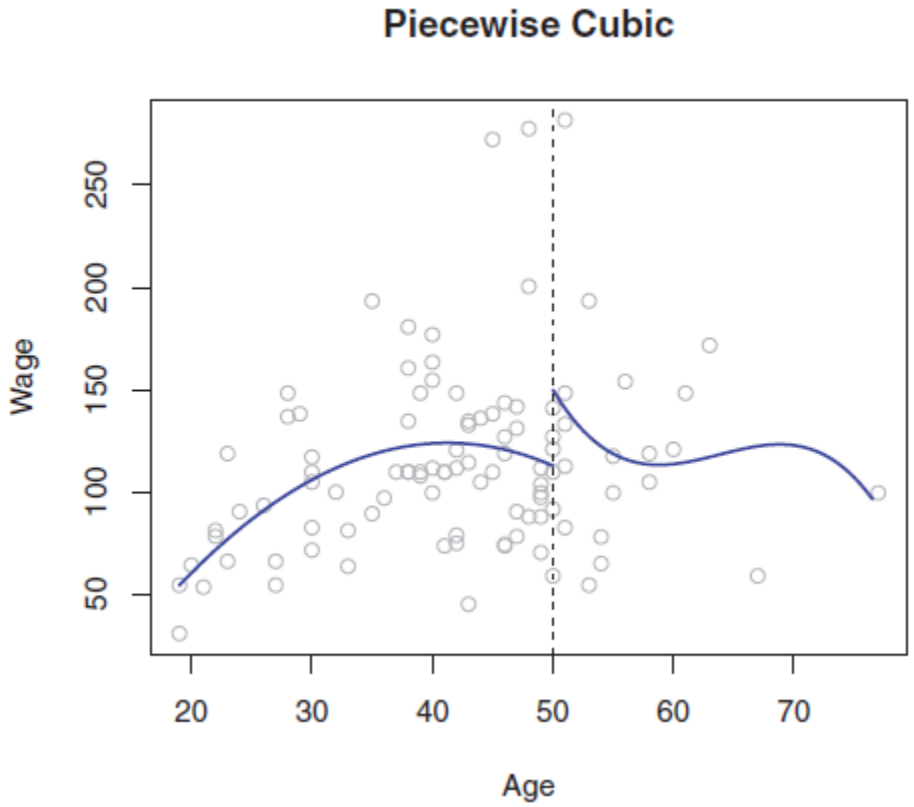
- The technical name given to points in the input space where coefficients change. For example, some point *c* such that:

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

- *K* knots will lead to *K* + 1 polynomials.
- Notice that the step function was a spline of degree 0 (“piecewise constant”).
- “Piecewise linear” is what we get by fitting linear models within each region.

First Attempt

- Let's get back to fitting a model for (a subset of) the wages data, splitting at age $c = 50$.



Needed

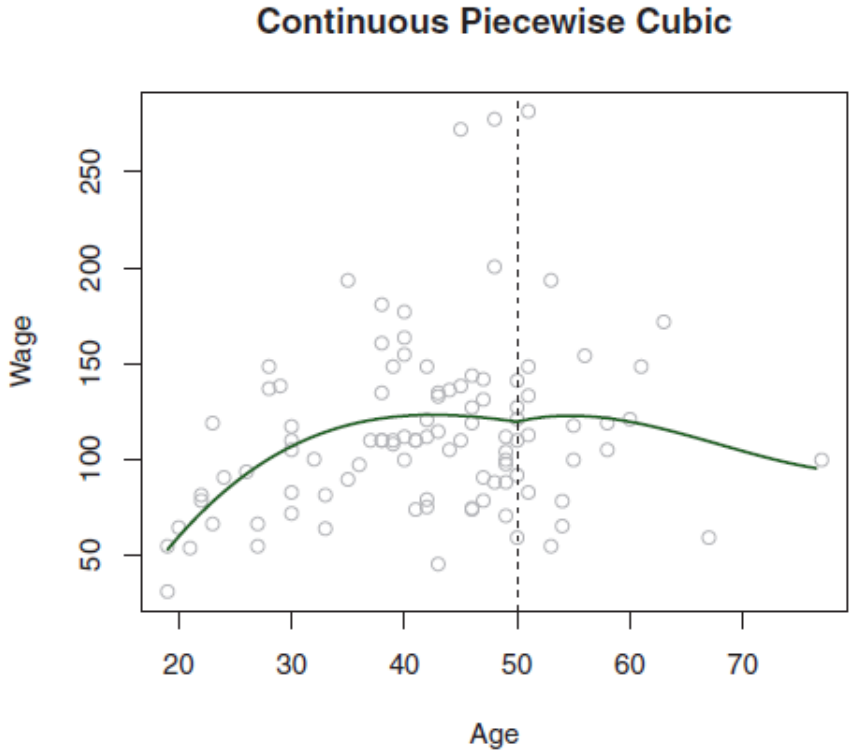
- A way of fitting parameters so that they agree at the knots.
- What does agreement mean? We can “tie the knot” by enforcing

$$\beta_{01} + \beta_{11}c + \beta_{21}c^2 + \beta_{31}c^3 = \beta_{02} + \beta_{12}c + \beta_{22}c^2 + \beta_{32}c^3$$

- How many free parameters do we get by fitting two cubic polynomials that agree at the know?

One Tie, One Fewer Degree of Freedom

- We can express one intercept as a function of other parameters, so we have 7 free parameters ("degrees of freedom").



- Still doesn't look right, does it?

More Smoothing

- Let's enforce not only agreement at the knot, but also agreement of the *first derivatives*, and agreement of the *second derivatives*.

$$\beta_{01} + \beta_{11}c + \beta_{21}c^2 + \beta_{31}c^3 = \beta_{02} + \beta_{12}c + \beta_{22}c^2 + \beta_{32}c^3$$

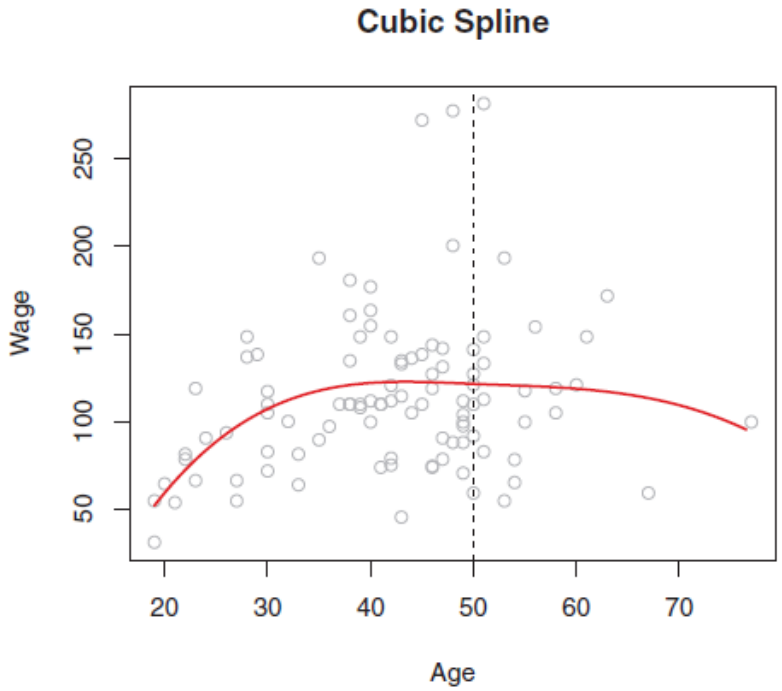
$$\beta_{11} + 2\beta_{21}c + 3\beta_{31}c^2 = \beta_{12} + 2\beta_{22}c + 3\beta_{32}c^2$$

$$2\beta_{21} + 6\beta_{31}c = 2\beta_{22} + 6\beta_{32}c$$

- We are left with 5 degrees of freedom.

Cubic Splines

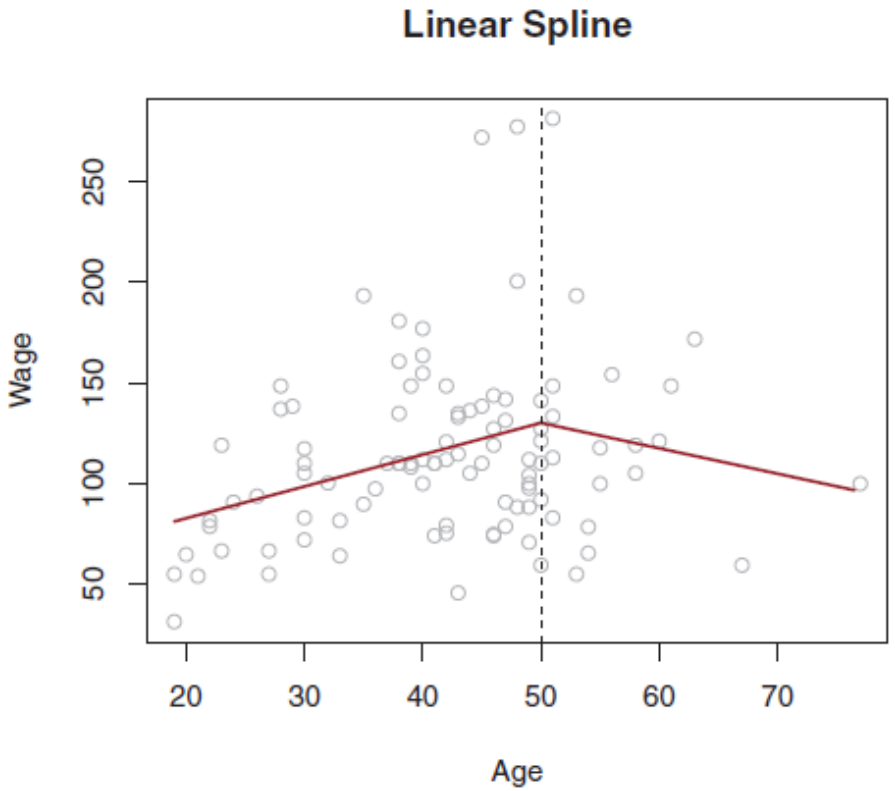
- The result is the unimaginatively named **cubic spline**.



- It is not hard to show that for *K* knots, we have 4 + *K* degrees of freedom.

Degree-*d* Spline

- We tie the derivatives up to degree *d* – 1. So we if had linear polynomials, “derivative 0” is just the original lines.



A More Suitable Formulation

- You may notice that optimising the parameters (say, by least squares) is a bit annoying because of the constraints.
- One idea is a **change of representation**, which in an abstract form relates to reparameterisation (recall GLMs!)
- What if we had $K + 3$ basis functions such that

$$Y = \beta_0 + \beta_1 b_1(x) + \beta_2 b_2(x) + \cdots + \beta_{K+3} b_{K+3}(x) + \epsilon$$

The Truncated Power Basis Representation

- For each knot ξ,

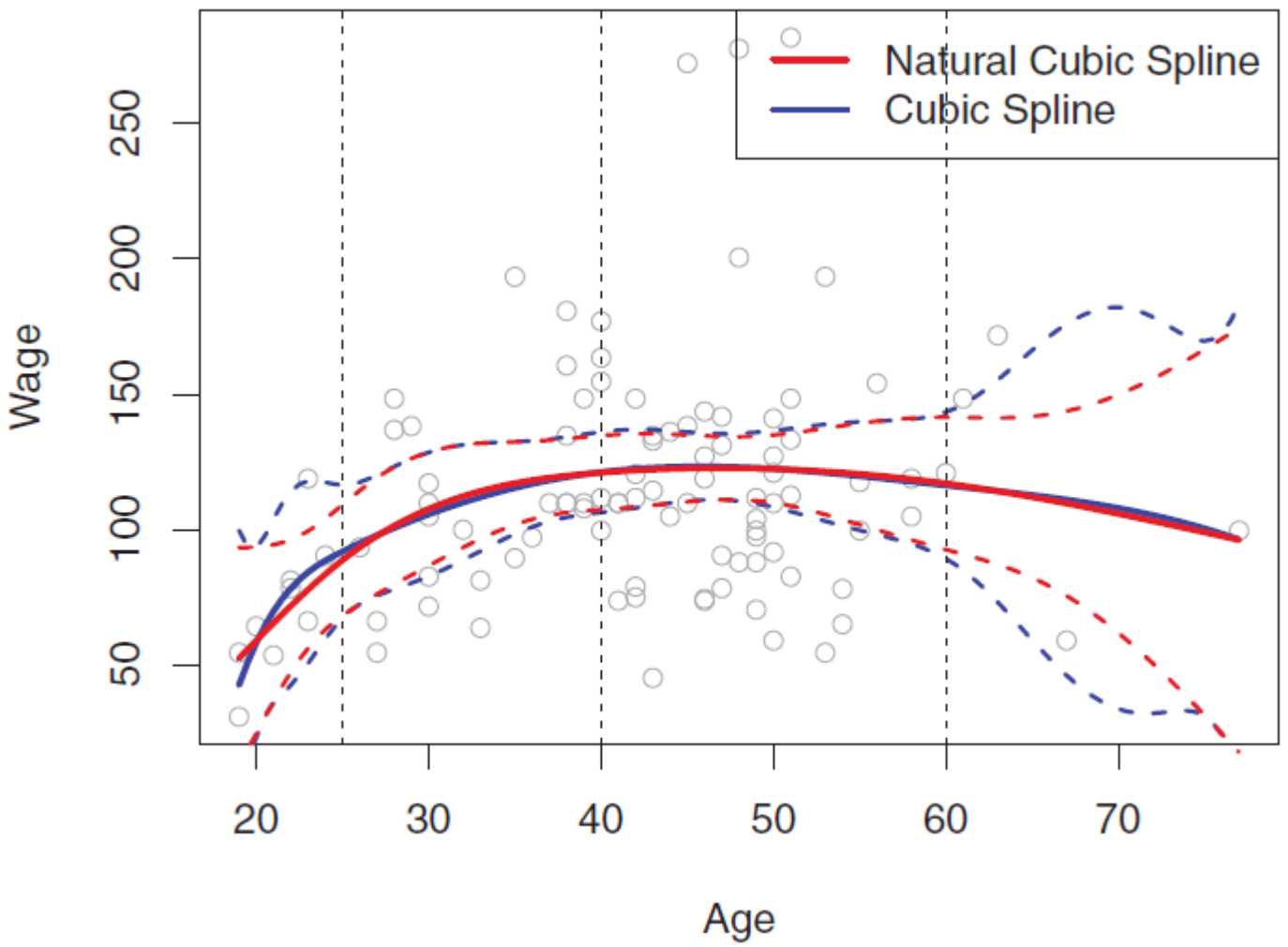
$$h(x,\xi) \equiv (x-\xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise} \end{cases}$$

- This function is discontinuous only at the third derivative
- We can use the variables $X, X^2, X^3, h(X,\xi_1), h(X,\xi_2), \ldots, h(X,\xi_K)$
- The coefficients are now unconstrained, and it is business as usual!

Further Corrections

- Points at the outer range of the training data (small/large values of *x*) are always the bane of regression. Splines are particularly sensitive.
- The smugly named **natural spline** "gives up" on nonlinearity at the "outer" knots: a linear function is used to fit points before the first knot/after the last knot.
  - If frees up 4 degrees of freedom, which can be cashed by spending them on 4 more knots.

Example (3 Knots)

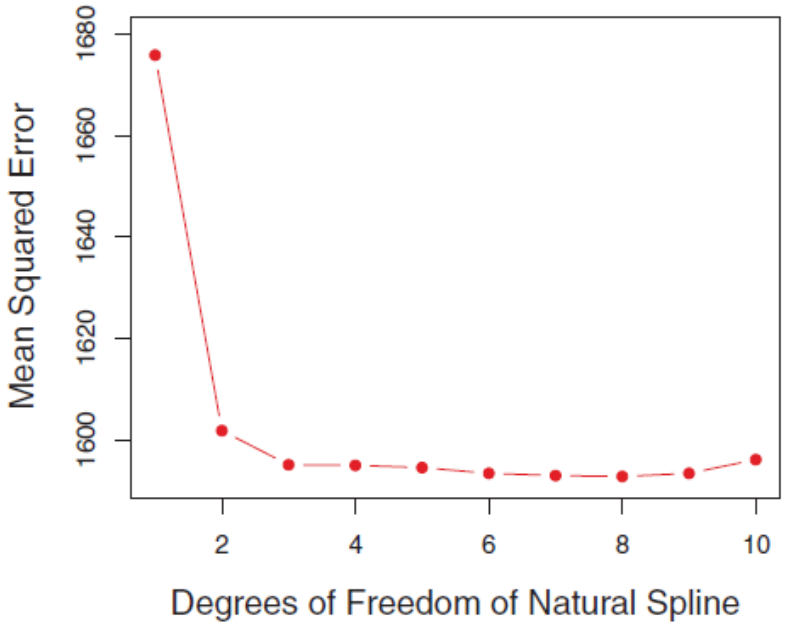


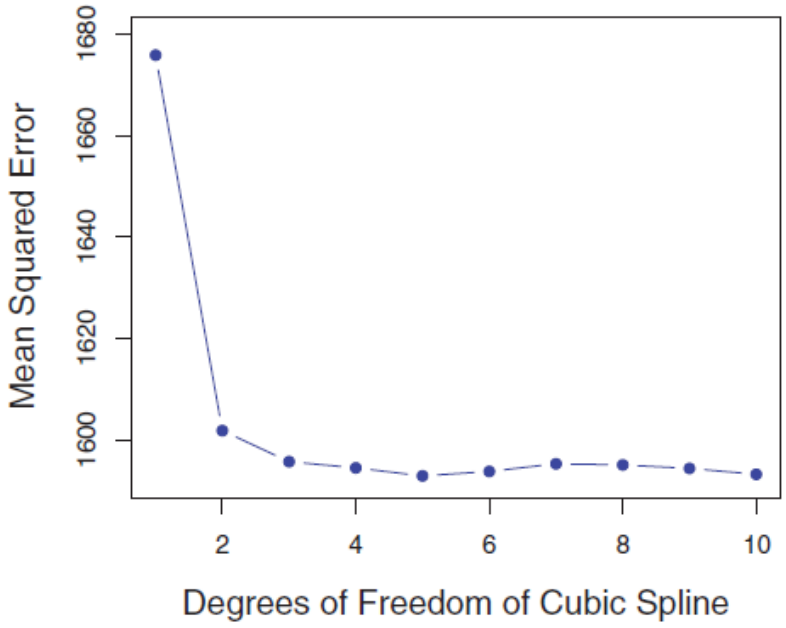
Choosing the Knots

- Intuition: putting more knots where the function is "wigglier", fewer where it is particularly smooth.
  - But of course, we *don't know* the function. That's what we want to estimate!
- In practice, placing them uniformly in the range of x is one of the most common strategies.
  - Empirical quantiles of X are typically used.
- But how many knots?

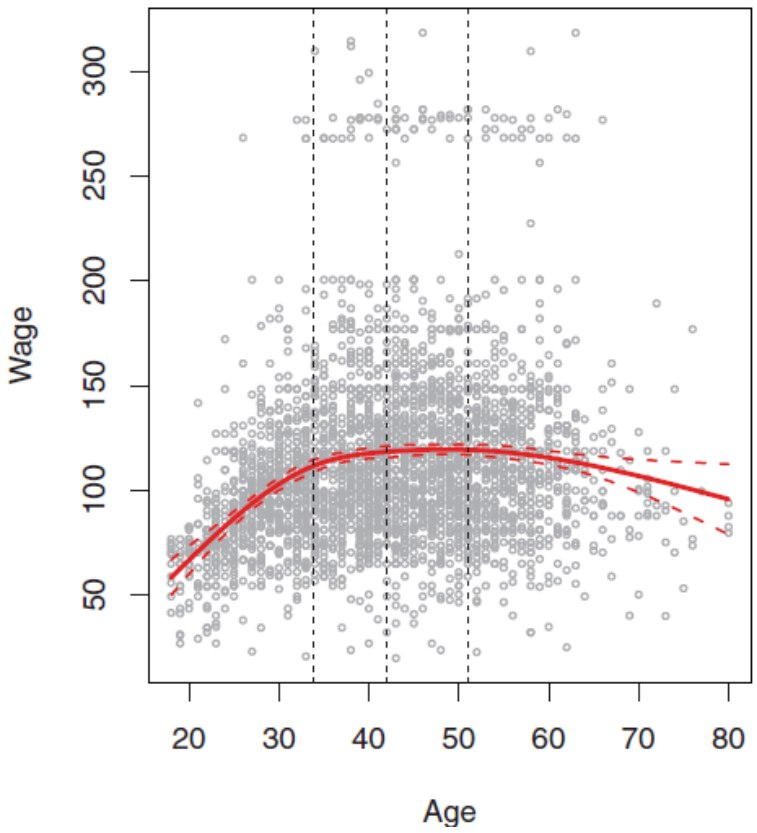
Example: Cross-Validation

- In practice, particularly when we have several inputs (details soon), it is not that uncommon to just fix the number of knots.

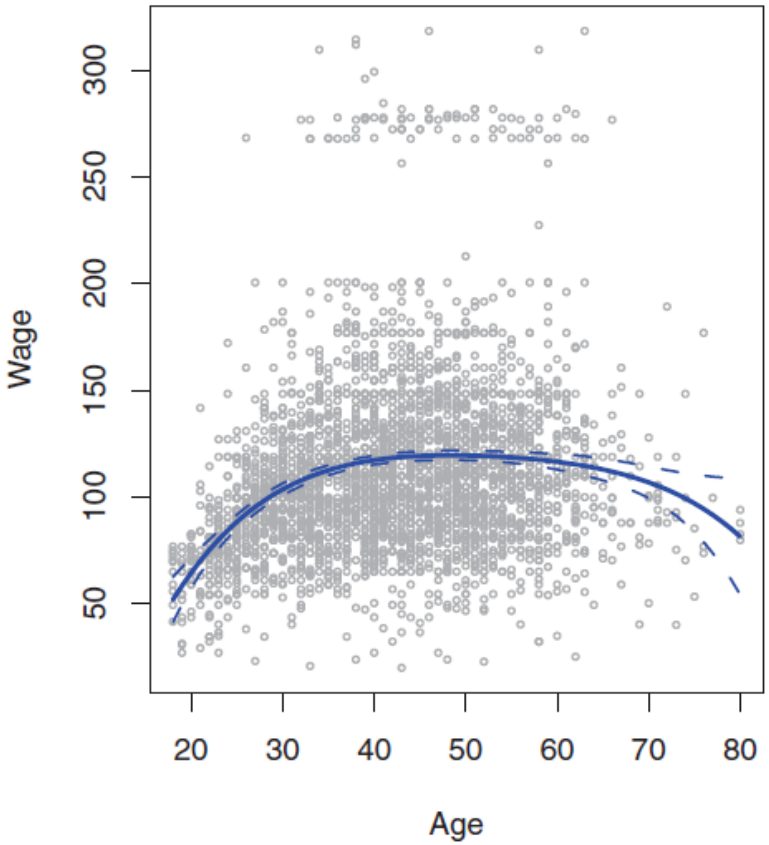




Example: 3 Knots @ the 0.25, 0.50 (median) and 0.75 Empirical Quantiles



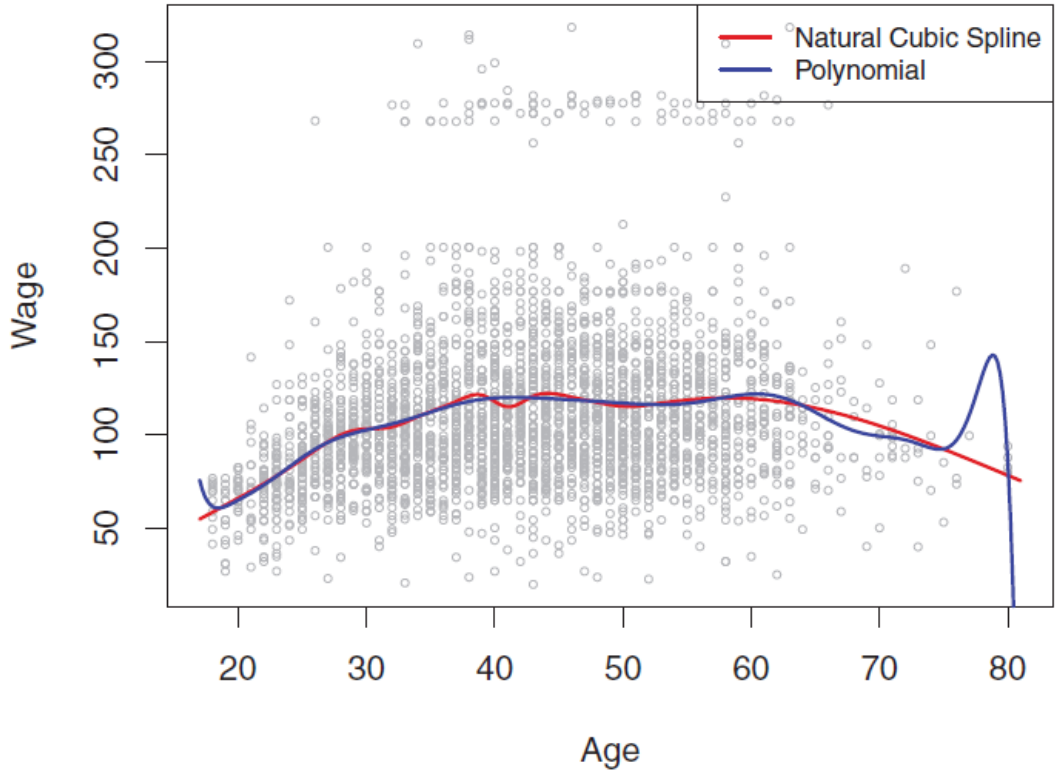
Natural Cubic Spline



Degree 4 Polynomial

Comparison

- Spending 15 degrees of freedom on a natural cubic spline vs. polynomial regression.



- Representation matters!

SMOOTHING SPLINES

Smoothing Splines

- Take no prisoners: *make every training point a knot*.
- How can we justify this? Cast least-squares in a very abstract sense

$$\text{minimise} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2 \text{ such that } f(\cdot) \text{ is a function}$$

- Wait, what? I will just set $f(x^{(i)}) = y^{(i)}$, thank you very much. Not sure what will happen outside the training data.

Regularization

$$\text{minimise} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2 \text{ such that } f(\cdot) \text{ is a function that is not "too wiggly"}$$

- Better phrased as $$\text{minimise} \sum_{i=1}^{n} (y^{(i)} - f(x^{(i)}))^2 + \lambda \int f''(t)^2 dt$$

- f''(t) can be thought of as a measure of how "wiggly" f is near t, and the integral can be thought of as taking an average over values of t.

- The result of this "function optimisation" problem is a natural cubic spline with a knot on every training point!

See Ex 5.7 of ESL

Nonparametrics

- As hinted before, this is a genuine example of nonparametric statistics. The parameter space is **infinite**, even if the range of *x* was bounded.

Start with

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_K \end{bmatrix} = \begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ \dots \\ f(K) \end{bmatrix}$$

But, then…

$$\begin{bmatrix} f(0) \\ f(1) \\ f(1.5) \\ f(2) \\ \dots \\ f(K) \end{bmatrix} \text{?...} \qquad \begin{bmatrix} f(0) \\ f(0.25) \\ f(0.5) \\ f(0.75) \\ f(1) \\ f(1.25) \\ f(1.5) \\ f(1.75) \\ f(2) \\ \dots \\ f(K) \end{bmatrix} \text{!...}$$

Nonparametrics

- But even if the parameter is a weird infinite-dimensional "vector", the **estimator** we get has a finite representation.
- It just happens to grow with the size of the training set…

$$Y = \beta_0 + \beta_1 b_1(x) + \beta_2 b_2(x) + \cdots + \beta_{K+3} b_{K+3}(x) + \epsilon$$

now $K = n$

Degrees of Freedom, Revisited

- When we introduce a regularizer, it is not a matter of parameter counting anymore.

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

- $\lambda = 0$ corresponds to *n* effective degrees of freedom.
- $\lambda = \infty$ corresponds to 2 effective degrees of freedom.
- Why do we care?

Linear Smoothers

- Recall an exercise in Sheet #4 where we showed that in regression, the fitted values are weighted combinations of the training *Y*.

- For natural cubic splines, the same applies:

$$\hat{\mathbf{f}}_\lambda = \mathbf{S}_\lambda \mathbf{y}$$

- Without doing the algebra, the diagonal entries of $\mathbf{S}_\lambda$ decrease with λ. The "effective degrees of freedom" is the sum of the diagonal entries of $\mathbf{S}_\lambda$.

(See ESL, Section 5.4, if you want the gory details.)

Linear Smoothers

- What is my point? You should be aware that *leave-one-out cross-validation* can be computed very efficiently for linear smoothers:

$$\sum_{i=1}^{n}(y^{(i)} - \hat{f}^{(-i)}(x^{(i)}))^2 = \sum_{i=1}^{n}\left[\frac{y^{(i)} - \hat{f}_\lambda(x^{(i)})}{1 - \{\mathbf{S}_\lambda\}_{ii}}\right]^2$$

Implications

- If for some reason you need to fit many nonlinear curves, a linear smoother like smoothing splines can be very handy!
- If a piece of software is choking on delivering the fit of many linear smoothers, maybe this software is junk.
- Also, many software packages report the "effective degrees of freedom" chosen by cross-validation instead of λ, because it is arguably more interpretable.

LOCAL REGRESSION

Local Regression

- Large-scale fitting is a problem when you have much data: smoothing splines cost* O(n^3) computational steps.
- An alternative is to build a model around the prediction point you are interested.
  - Old-fashioned machine learning had wonderful names for that, *lazy learning* or *memory-based learning*.
- The main idea is to fit least-squares around a point by nonlinear reweighting the data around it.
  - It does require storing your training data at test time.

*There are less naïve ways of doing it, see ESL, Appendix to Chapter 5.

Algorithm

Algorithm 7.1 *Local Regression At* $X = x_0$

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.

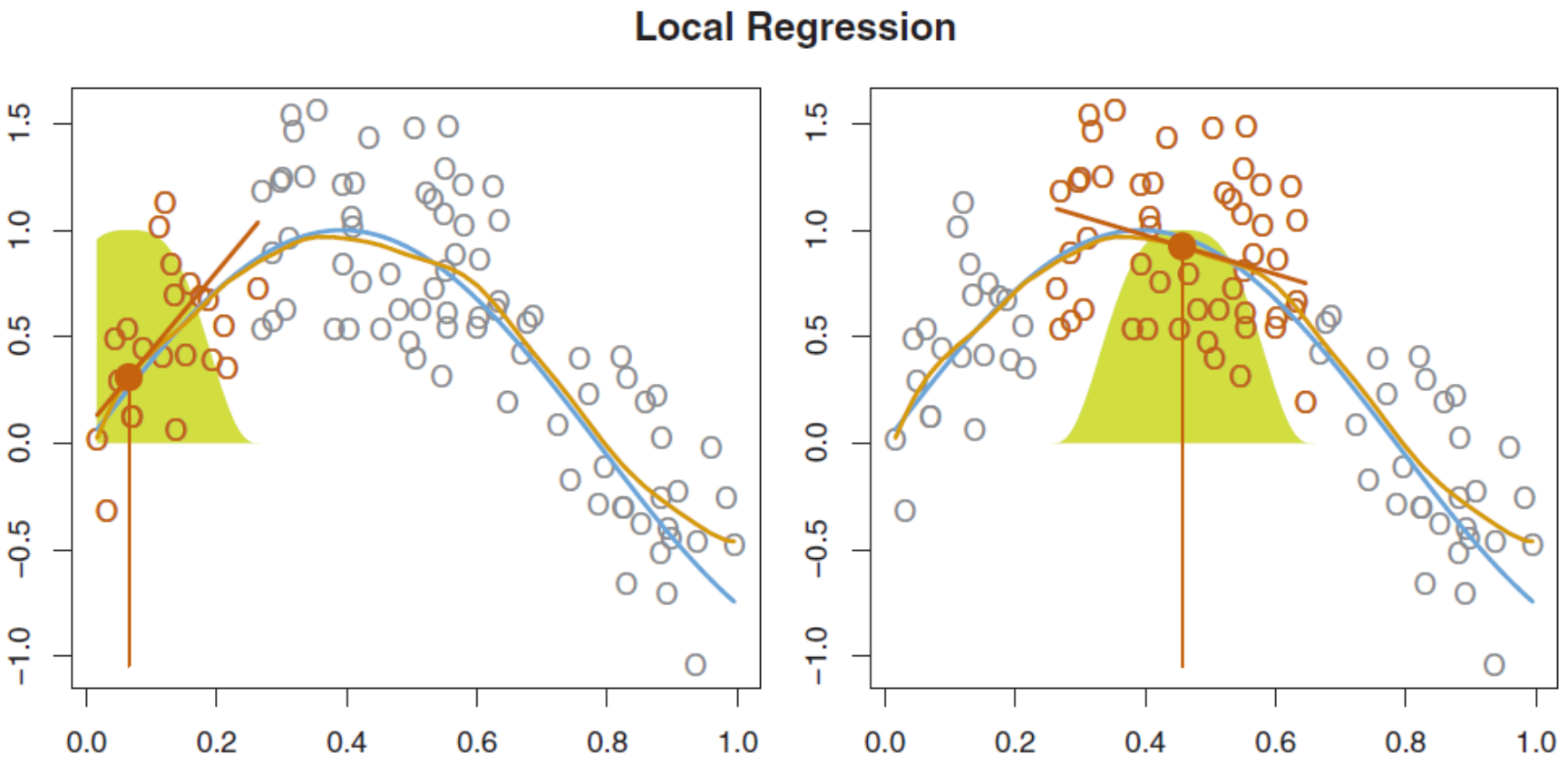
2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.

3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Illustration: Wage data again



Parameters in Local Regression

- How should we pick the number of training points to use for each prediction?
- What model should we build locally? Is a linear model sufficient, or should we do something else?
- How should we pick the weighting function? What is the impact on the regression curve?

*See ESL Chapter 6 for all the juicy details.

GENERALISED ADDITIVE MODELS

Summary

- We have seen a range of examples of basis functions which can be used for regression.

$$Y = \beta_0 + \beta_1 C_1(x) + \cdots + \beta_K C_K(x) + \epsilon$$

- More generally, we now have a nonlinear regression function f such that:

$$Y = f(x) + \epsilon$$

- However, all these methods were very much limited to cases with one covariate x.

Multidimensional Splines

- Imagine we have two covariates. We can define a **tensor product basis**:

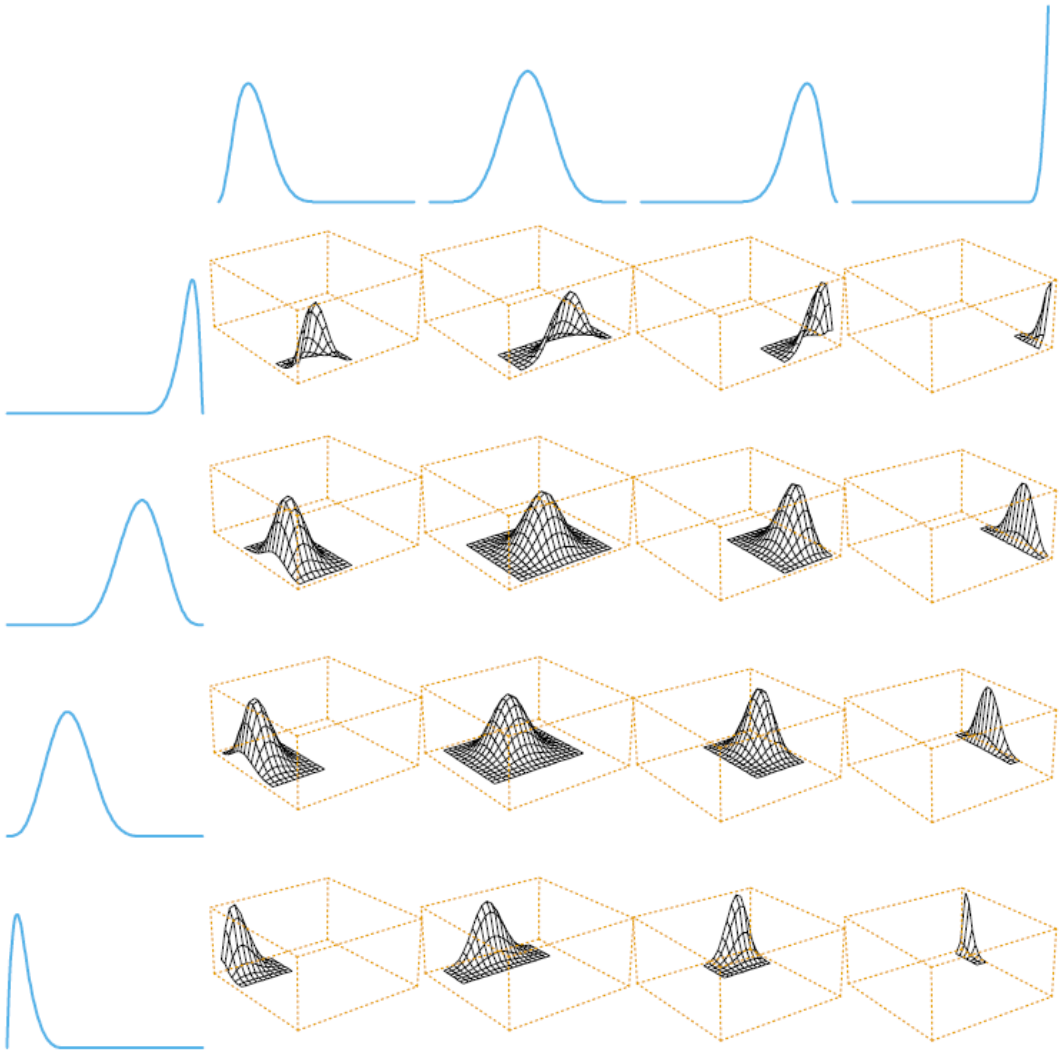
$$g_{jk}(x) = h_{1j}(x_1)h_{2k}(x_2)$$

for some $j = 1, \ldots, M_1$, $k = 1, \ldots, M_2$.

- This leads to the following parameterisation of the regression function

$$f(x) = \sum_{j=1}^{M_1} \sum_{k=1} \beta_{jk} g_{jk}(x)$$

Visualisation



(ESL, Chapter 5)

Problems

- The complexity of this tensor combination explodes as a function of dimensionality. Not really used even in few dimensions (e.g. 4+).
  - This is both computationally (takes too much time) and statistically (takes too much data) intractable.
  - There are alternatives: remember greedy search? It is used in the context of deciding which pieces of the tensor to be used.

- A useful compromise is **additive models**. Way fewer degrees of freedom, and also interpretable.

Additive Models

- An alternative idea is to use one basis function per covariate: this is called an additive model.

- Additivity here means on additivity on the covariates. On top of it, we can have additive errors (as in linear regression).

Typically, $\sum_{i=1}^{n} f(x_j^{(i)}) = 0$ is enforced.

$$Y = \beta_0 + \sum_{j=1}^{p} f_j(x_j) + \epsilon$$

Redundant?

- Algorithms for fitting these models can build upon existing methods for univariate regression.

Example: The Backfitting Algorithm

Algorithm 9.1 *The Backfitting Algorithm for Additive Models.*

1. Initialize: $\hat{\alpha} = \frac{1}{N}\sum_1^N y_i$, $\hat{f}_j \equiv 0, \forall i, j$.

2. Cycle: $j = 1, 2, \ldots, p, \ldots, 1, 2, \ldots, p, \ldots,$

$$\hat{f}_j \leftarrow \mathcal{S}_j\left[\{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\}_1^N\right],$$

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

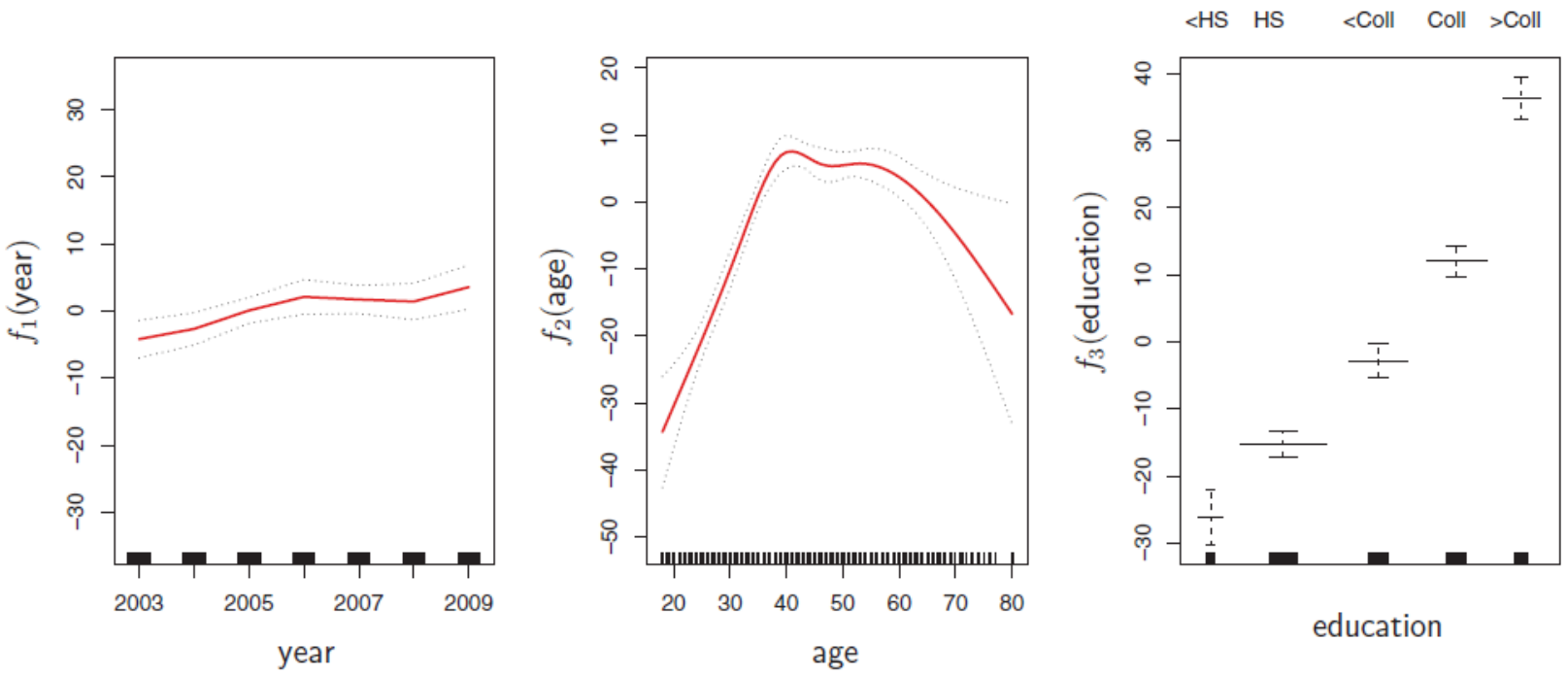
until the functions $\hat{f}_j$ change less than a prespecified threshold.

Some smoother e.g. natural smoothing splines

(ESL, Chapter 9)

Example: Wage Data

(ISLR, Chapter 7)



- We took f_1 and f_2 as natural splines and f_3 to be a constant per level.
- What is the interpretation?
- How does it differ with respect to linear models?

Generalised Additive Models

- We can combine this idea of additive models with any GLM: these are called generalised additive models.

- The logistic regression case:

$$\log \frac{P(Y=1 \mid x)}{P(Y=0 \mid x)} = \beta_0 + \sum_{j=1}^{p} f_j(x_j) + \epsilon$$

- Beware of interpretation and algorithm. The idea now is similar to Newton-Raphson, iterated per covariate.

Example: Backfitting for Additive Logistic Regression

Algorithm 9.2 *Local Scoring Algorithm for the Additive Logistic Regression Model.*

1. Compute starting values: $\hat{\alpha} = \log[\bar{y}/(1-\bar{y})]$, where $\bar{y} = \text{ave}(y_i)$, the sample proportion of ones, and set $\hat{f}_j \equiv 0 \; \forall j$.

2. Define $\hat{\eta}_i = \hat{\alpha} + \sum_j \hat{f}_j(x_{ij})$ and $\hat{p}_i = 1/[1+\exp(-\hat{\eta}_i)]$.

   Iterate:

   (a) Construct the working target variable

   $$z_i = \hat{\eta}_i + \frac{(y_i - \hat{p}_i)}{\hat{p}_i(1-\hat{p}_i)}.$$

   (b) Construct weights $w_i = \hat{p}_i(1-\hat{p}_i)$

   (c) Fit an additive model to the targets z_i with weights w_i, using a weighted backfitting algorithm. This gives new estimates $\hat{\alpha}, \hat{f}_j, \; \forall j$

3. Continue step 2. until the change in the functions falls below a prespecified threshold.

(ESL, Chapter 9)

Take-Home Messages

- Nonlinear regression/classification/etc. assumes many facets.
- The spline/additive approach is very popular in the Statistics community because of:
  - Computational simplicity
  - In many cases, confidence intervals are easy to calculate
    - Many approaches boil down to least-squares with data-independent regularization!
- Additivity is certainly not the greatest assumption, but it can be as good as it gets in problems with many variables and not "that much" data.

Take-Home Messages

- Much of this chapter also illustrates the development of statistical thinking: a mix-and-match combination of many fundamental building blocks.
- R packages such as **gam** and **glmnet** can go a long way in your practical applications, even if these are not particularly optimised pieces of code.
- Sparse/(Generalised) Additive Models complement nicely the Machine Learning toolbox of SVMs, neural nets etc. particularly for interpretation purposes and the selection of variables. But don't rely on additivity for additivity's sake.