New York University: Machine Learning in Economics

Lecture 4: Basis Expansions and Regularization

Dr. George Lentzas

Introduction

Moving Beyond Linearity

- Linear models are easy to use and understand but in practice they can have significant limitations in terms of predictive power.
- This is because the linearity assumption is almost always an approximation. Now we will try to relax this assumption!
- We will consider simple extensions of the linear models such as:
  1. **Polynomial Regression**: adds extra predictors by raising the original predictors to a power.
  2. **Step Functions**: cut the range of the X variable into K distinct regions and fit a piecewise constant function.
  3. **Regression Splines**: cut the range of the X variable into K distinct regions and fit a different polynomial to each region (joined smoothly).
  4. **Smoothing Splines**: minimize a residual sum of squares criterion subject to a smoothness penalty.
  5. **Local Regression**: similar to splines but with overlapping regions.
  6. **Generalized Additive Models**: extend the above to deal with multiple predictors.
- In the below we assume a single predictor X.

Polynomial Regression

- The standard way to extend the model

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

for non-linearity is with a polynomial regression

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 {x_i}^2 + \cdots + \beta_d {x_i}^d + \epsilon_i$$

- This can be estimated using least squares and in practice $d = 3$ or $d = 4$ is used.
- Using polynomial functions of the features as predictors imposes global structure on the non-linearity of X. This can be a problem!

Polynomial Regression

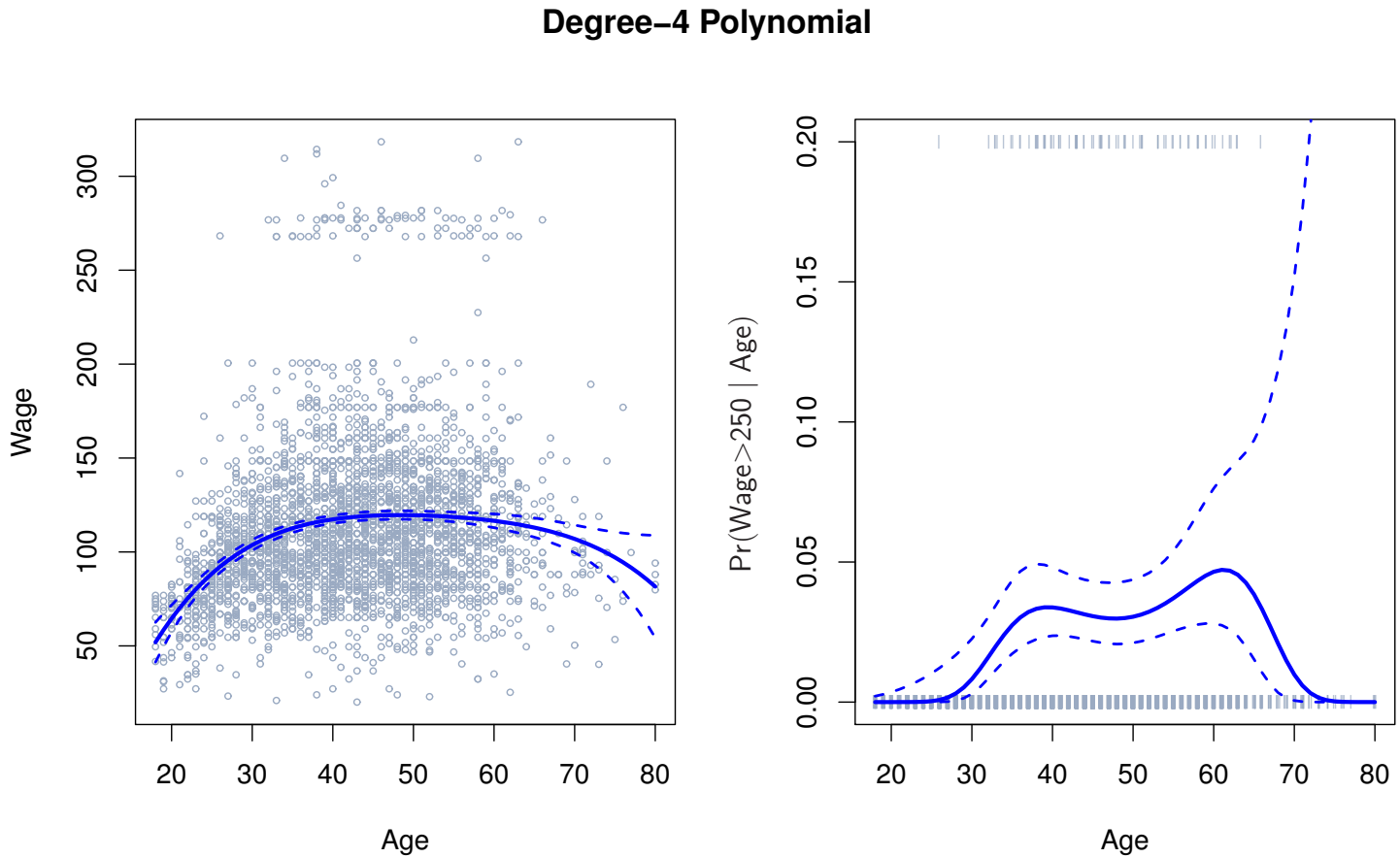


Figure: 4th Degree Polynomial

Step Functions

- Step functions work by breaking the range of X into bins and fit a different constant to each bin. The model is estimated by

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_d C_d(x_i) + \epsilon_i$$

- Here

$$C_i(x_i) = I(c_i \leq X < c_{i+1})$$

  and $c_1, \ldots, c_K$ are the breakpoints in the range of X.
- Unfortunately, unless there are some natural breakpoints in the data step functions will likely not work well.

Step Functions

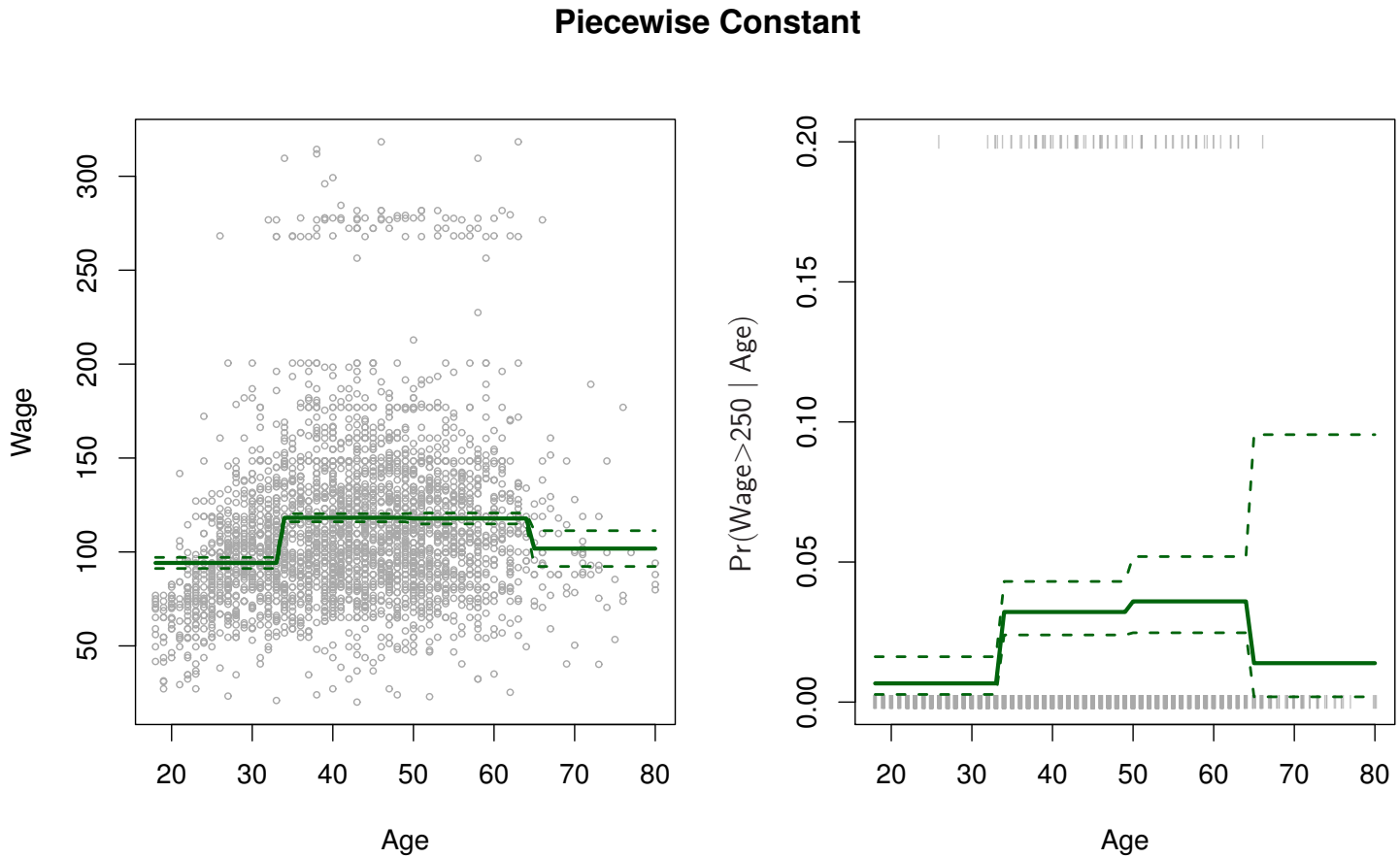


Figure: Step Functions

Basis Functions

- Polynomial and piecewise-constant regression models are actually special cases of the basis function approach.
- The idea here is to have at hand a family of functions that can be applied to a variable X

$$b_1(X), b_2(X), \ldots, b_d(X)$$

- The use them to fit the model

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_d b_d(x_i) + \epsilon_i$$

- Many alternative other than polynomial or step functions exist, for example wavelets or Fourier transformations.
- Another common choice for a basis function is the regression spline

Regression Splines

Piecewise Polynomials

- Piecewise polynomial regression fits separate low degree polynomials over different regions of X.
- For example a third order polynomial with a single 'knot' looks like:

$$y_i = \beta_{01} + \beta_{11} x_i + \beta_{21} {x_i}^2 + \beta_{31} {x_i}^3 + \epsilon_i \text{ if } x_i < c$$

$$y_i = \beta_{02} + \beta_{12} x_i + \beta_{22} {x_i}^2 + \beta_{32} {x_i}^3 + \epsilon_i \text{ if } x_i \geq c$$

- In general if we place K different knots we will have to fit K+1 polynomials.
- This sounds like a good idea but suffers from one immediate drawback: the estimated function is discontinuous!

Regression Splines

Constraints and Splines

- The obvious solution is to fit a piecewise polynomial under the constraint that the fitted curve is continuous. However this still looks unnatural!
- Actually what we probably prefer is to ensure that the fitted function is continuous as well as smooth. In practice we impose two constraints: both the first and the second derivatives of the piecewise polynomials are continuous. The resulting curve is called a *cubic spline*.
- In general a degree d spline is a piecewise degree d polynomial with continuity in derivatives up to a degree d-1 at each knot.

Regression Splines

The Spline Basis Representation

- In practice how do we fit a cubic spline?
- We can use the basis model to represent a cubic spline as follows:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 {x_i}^2 + \beta_3 {x_i}^3 + \beta_4 b_4 (x_i) + \cdots + \beta_{K+3} b_{K+3} (x_i) + \epsilon_i$$

where

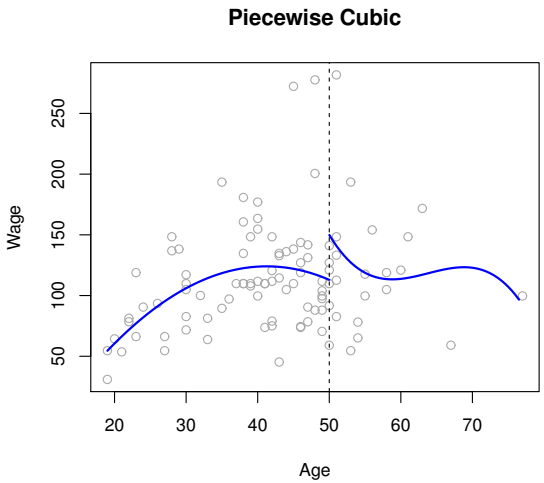
$$b_i (x) = h (x, \xi) = (x - \xi)_+^3$$

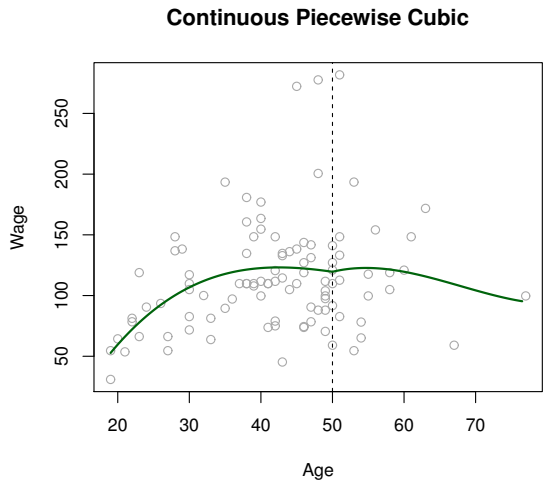
with $\xi_1, \ldots, \xi_K$ the knots.

- One issue with splines is that they tend to have high variance at the outer range of the predictors. To solve this we use *natual splines*. These are splines with the additional constraint that the function is required to be linear at the boundaries.

Regression Splines

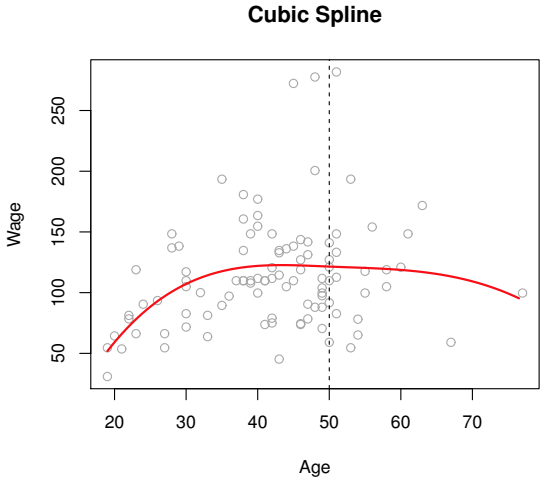
cubic polyn with 1 knot

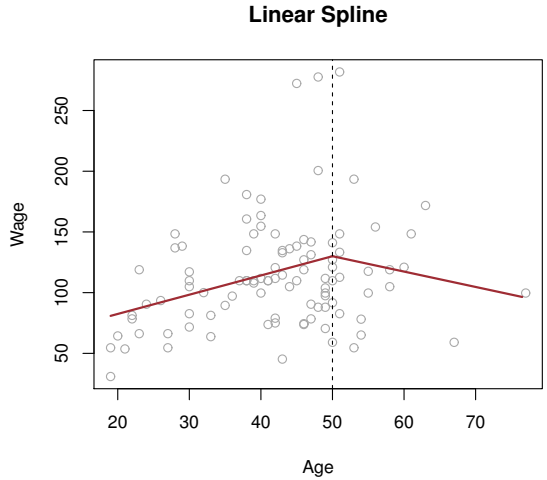




cubic polyn with continuous first derivative: two b_i = (x-xi)^2

cubic polyn with continuous first & second derivative





linear fit with knot

Figure: Various Piecewise Polyn0mials

Regression Splines

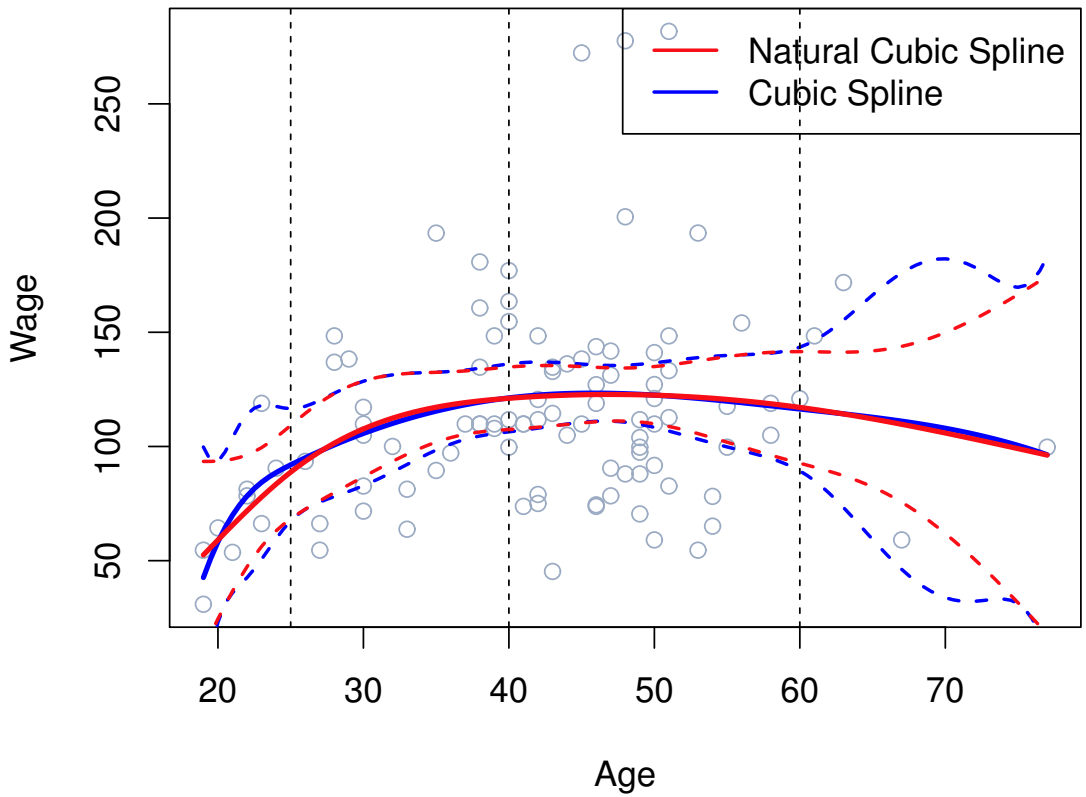


Figure: Natural Cubic Splines

Regression Splines

What about the Knots?

- How do we choose the number and location of the knots?
- In practice it is common to place knots uniformly across the data. Which leaves the question, how many knots?
- One good option is to use cross validation to estimate the optimal number of knots.

Splines vs Polynomial Regression

- Splines often give superior results than polynomial regression. This is because (unlike polynomials which use a high degree to increase flexibility) splines introduce flexibility by increasing the number of knots, which produces more stable estimates.

Regression Splines

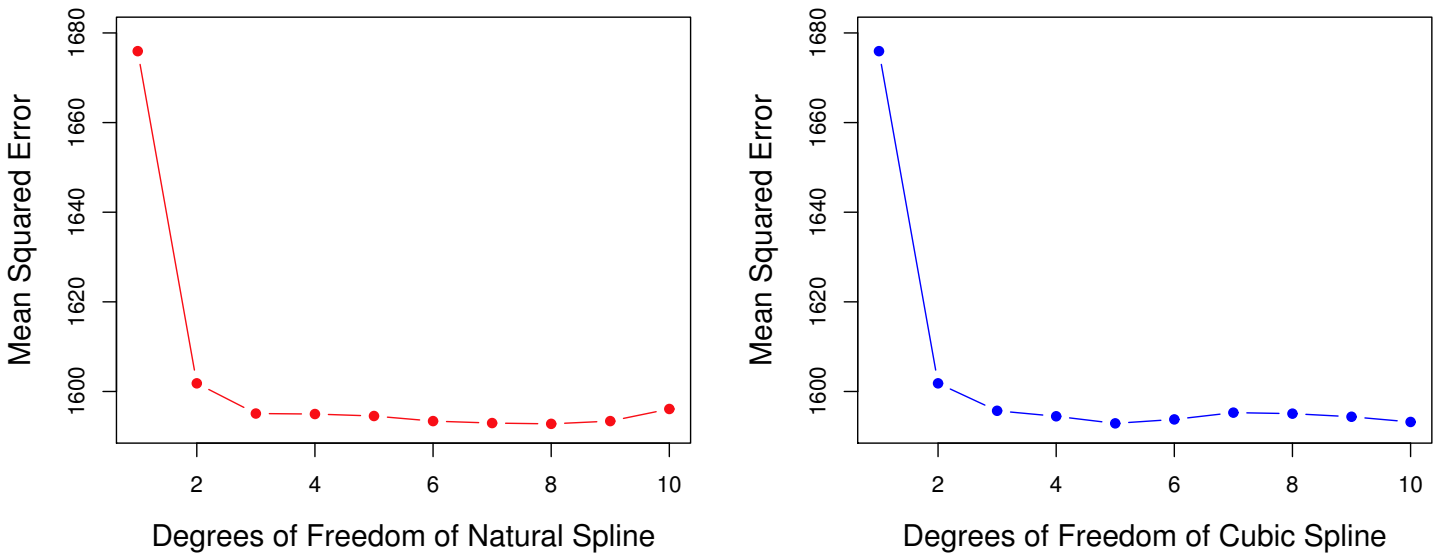


Figure: Cross Validation for Knots

Regression Splines

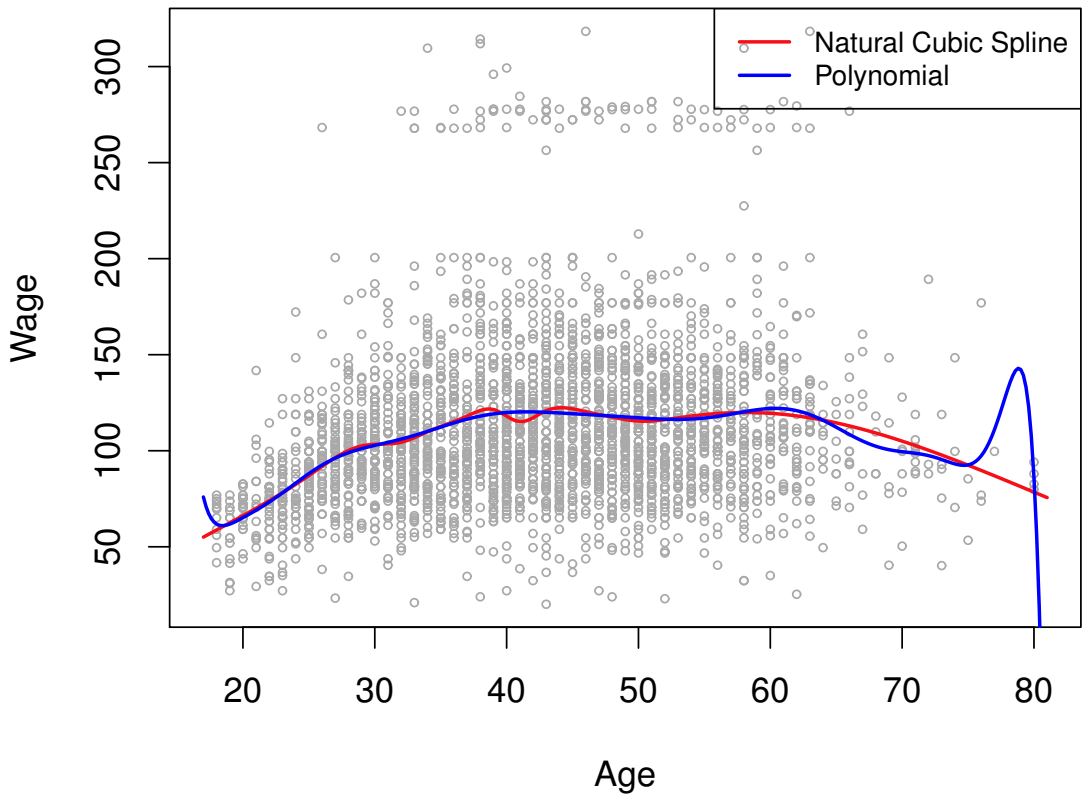


Figure: Splines vs Polynomial Regression

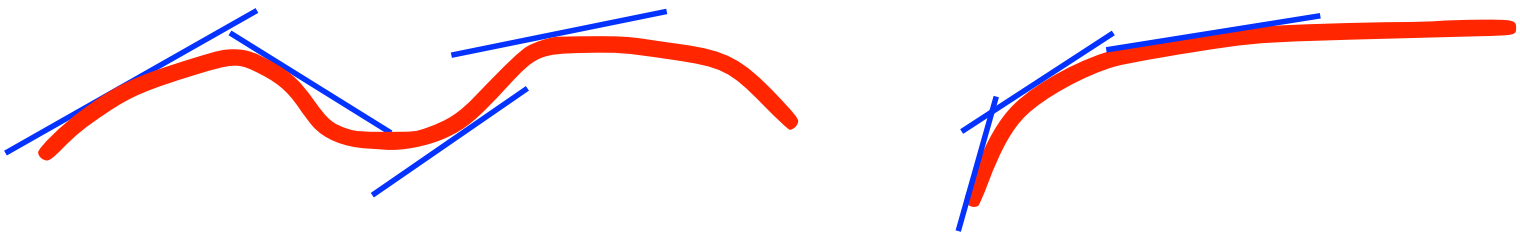
Smoothing Splines

- An alternative way to approach this problem is from first principles. We want if to find a function $g(x)$ that fits the observed data well and that is also smooth.
- A natural way to do this is to minimize

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

- This is similar to the "loss" + "penalty" formulation that we have use in Ridge regression or LASSO.
- The function g that minimizes the above is known as a *smoothing spline*. The penalty is a measure of the total change in the first derivative over its entire range; it is in other words a measure of roughness.

g'' measures how much the 1st deriv changes, so measuring how stable it is



Smoothing Splines

- The function that minimizes the above can be shown to be a natural cubic spline with knots at the unique values of $x_1, \ldots, x_n$.
- It is however a shrunken version of the natural cubic spline that we would get by simply fitting the above to the data, where the value of the tuning parameter λ controls the level of shrinkage and the bias-variance trade-off as usual.
- The tuning parameter is typical calculated by cross validation; in fact it is can be shown that the LOOCV error can be calculated analytically whch makes this choice very fast computationally.

Leave-one-out CV

Smoothing Splines

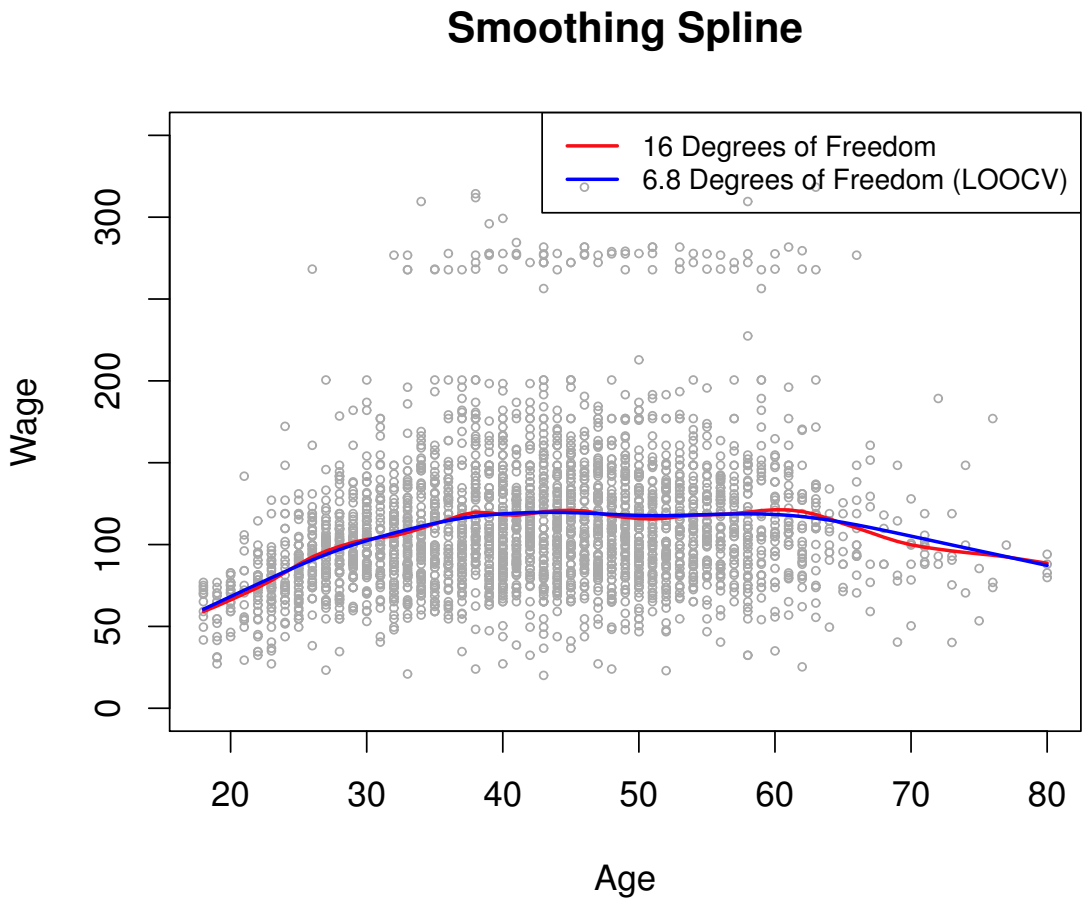


Figure: Smoothing Splines

Local Regression

- Local regression involves computing multiple fits each using only the nearby training observations.
- At each point x_0 a new weighted least squares regression model is estimated.
- For local regression one must choose how to define the weighting function L, and whether to fit a linear or higher order regression. More importantly, the *span* $s = \frac{k}{n}$ of training points closest to x_0 must be chosen which controls the flexibility of the linear fit, typically done by cross-validation. The regression is then

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

  where K is the weighting function.
- Local regression generalizes well to models that are local in a pair of variables, but fails for dimensions 3 or more due to the curse of dimensionality.

Local Linear Regression

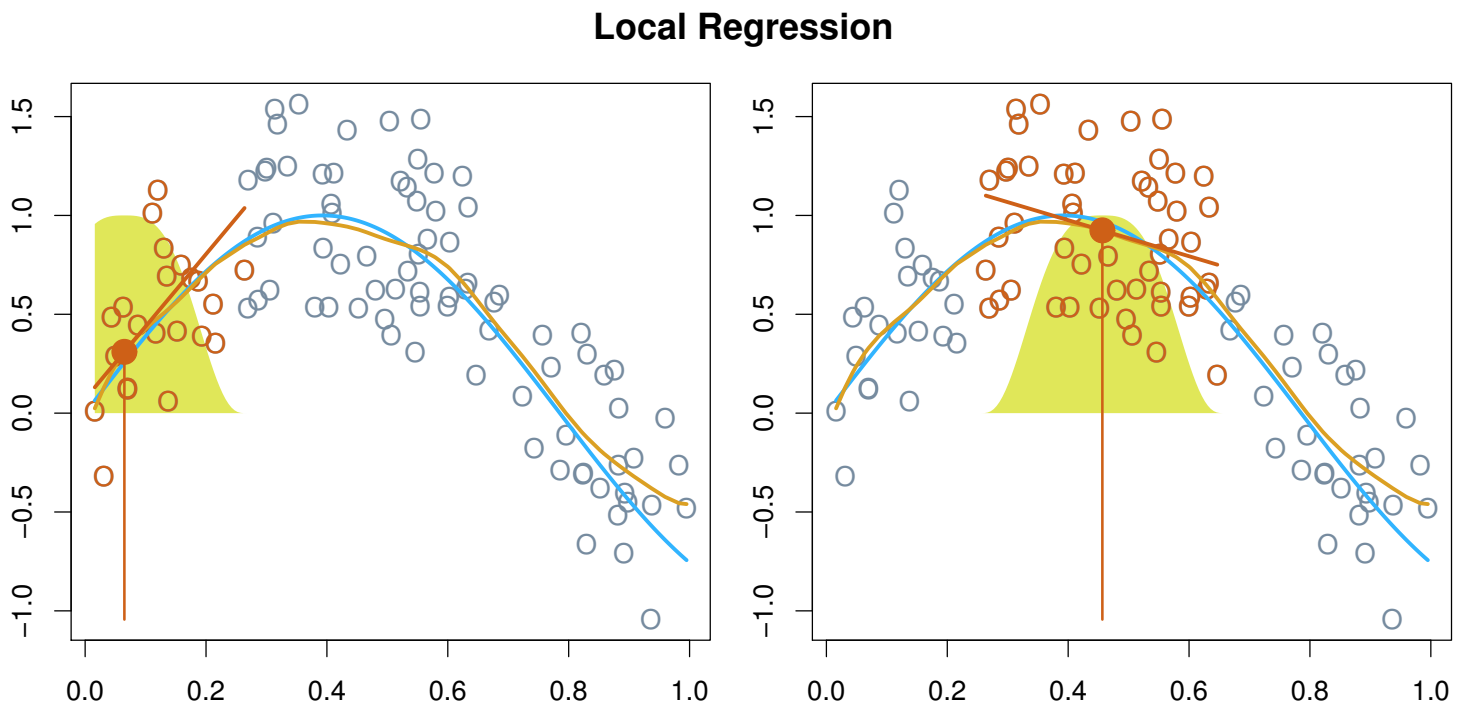


Figure: GAM Components (Natural Splines)

Generalized Additive Models

- So far we have seen a number of approaches for predicting a response Y using one feature X. Now we turn to the problem of predicting Y using several features, $X_1, \ldots, X_p$.
- Generalized Additive Models (GAMs) are a framework for extending standard linear regression by allowing non-linear function of each of the X variables while still maintaining additivity.
- A (regression) GAM can be written as

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

- You can see the additive nature of the model as a separate f_j is used for each X_j and then all the contributions are added together.
- All the methods we discussed previously (splines, basis functions, etc) can be used as building blocks for a GAM.

Estimation of GAMs

- In the case of natural splines (or basis functions in general) estimation can be done via least squares. The GAM model is simply one big regression where all the f_js are replaced by their linear forms.
- In the case of smoothing splines this is not possible so instead an approach called *backfitting* is used. Backfitting involves iteratively holding all but one coefficient constant and update only that coefficient, continuing the process until convergence. Another way to think about this

Generalized Additive Models

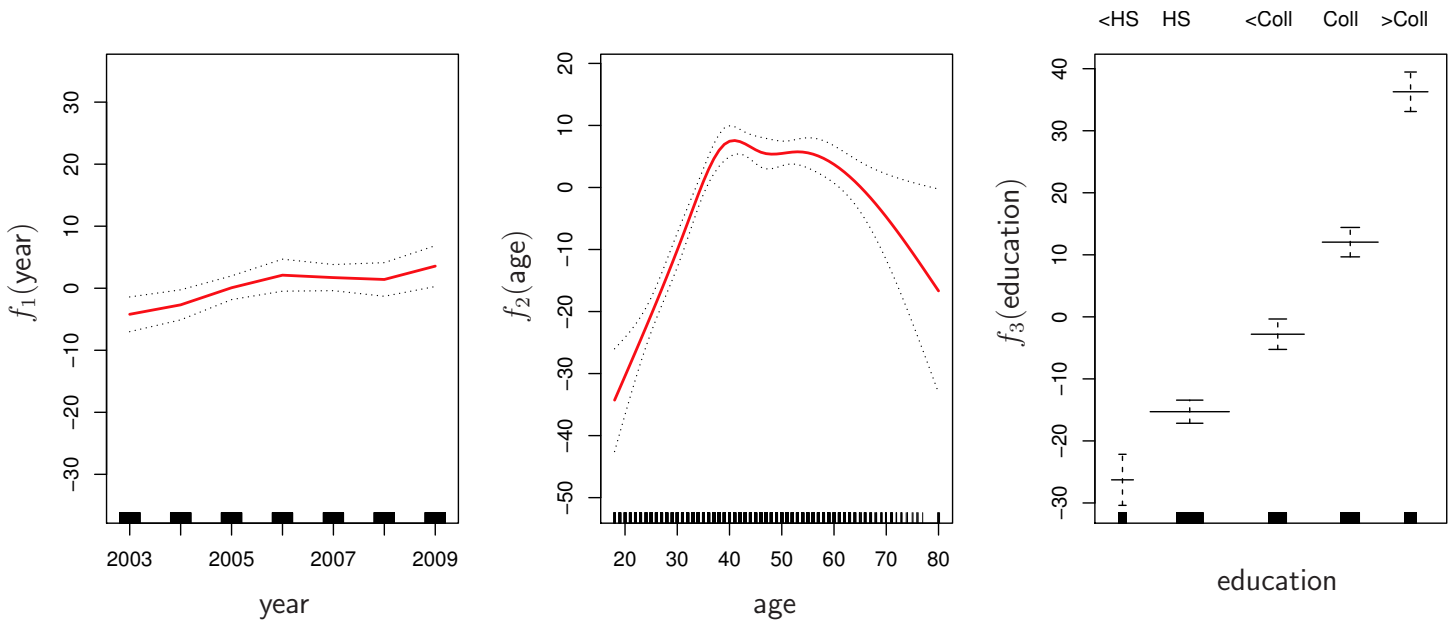


Figure: Generalized Additive Model Using Splines

Pros and Cons of GAMs

- GAMs are usually more accurate than multivariate linear models (+).
- GAMs are relatively straightforward to fit and allows us to automatically model complex non-linear relationships without trying multiple transformations of the data (+).
- GAMs retain the interpretability of linear models and allow us to understand partial effects (+).
- GAMs assume additivity so important interaction terms might be missed (-).

Acknowledgements

- These notes follow closely "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- Also, some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.