Lecture 7: Beyond linearity

STATS 202: Statistical Learning and Data Science

Linh Tran

tranlm@stanford.edu

Department of Statistics
Stanford University

July 14, 2025

Announcements

- HW1 is graded.
  - Regrade requests are allowed up to 1 week from publication
- HW2 is due today (solutions posted tmrw night)
- Midterm is this Wednesday
  - In-person
  - Accommodations should be confirmed
  - No calculators necessary
- Annonymous course survey will be posted on Wednesday
  - Closes Tuesday afternoon
  - Up to 10 additional bonus points on exam (conditional on % participation)
- No section this Friday

Outline

- Extending linear models
- Basis functions
- Piecewise models
- Smoothing splines
- Local models
- Geneneralized additive models

Recall

We can use linear models to for regression / classification, e.g.

$$\mathbb{E}[Y|\mathbf{X}] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... \tag{1}$$

- When relationships are non-linear, could e.g. include polynomial terms (e.g. $X^2, X^3, ...$) or interactions (e.g. $X_1 * X_2, ...$)
- Could then use e.g. stepwise regression to do model fitting
  - n.b. Recall that using too many parameters can result in overfitting

Recall

We can use linear models to for regression / classification, e.g.

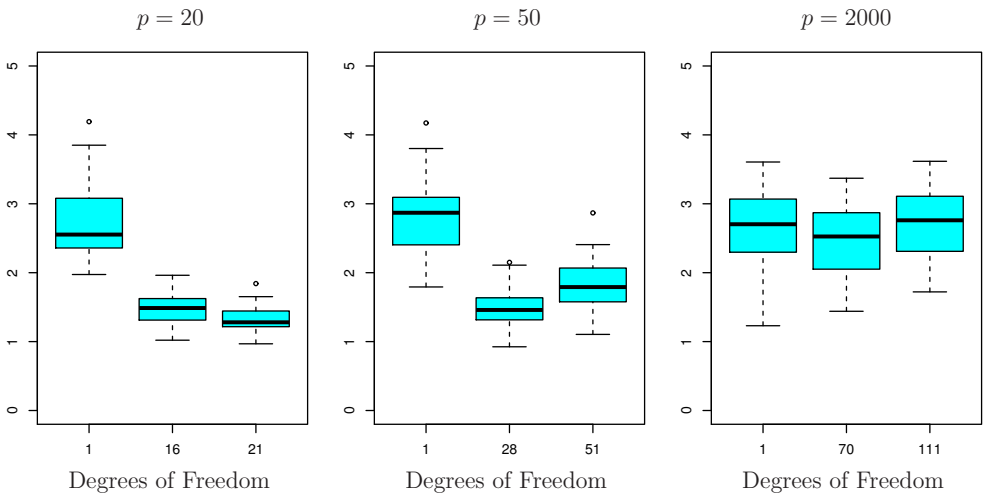
$$\mathbb{E}[Y|\mathbf{X}] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... \tag{1}$$

- When relationships are non-linear, could e.g. include polynomial terms (e.g. $X^2, X^3, ...$) or interactions (e.g. $X_1 * X_2, ...$)
- Could then use e.g. stepwise regression to do model fitting
  - n.b. Recall that using too many parameters can result in overfitting

Question: Assuming $n >> p$, why not just do every transformation we can think of and throw all of them into e.g. Lasso?

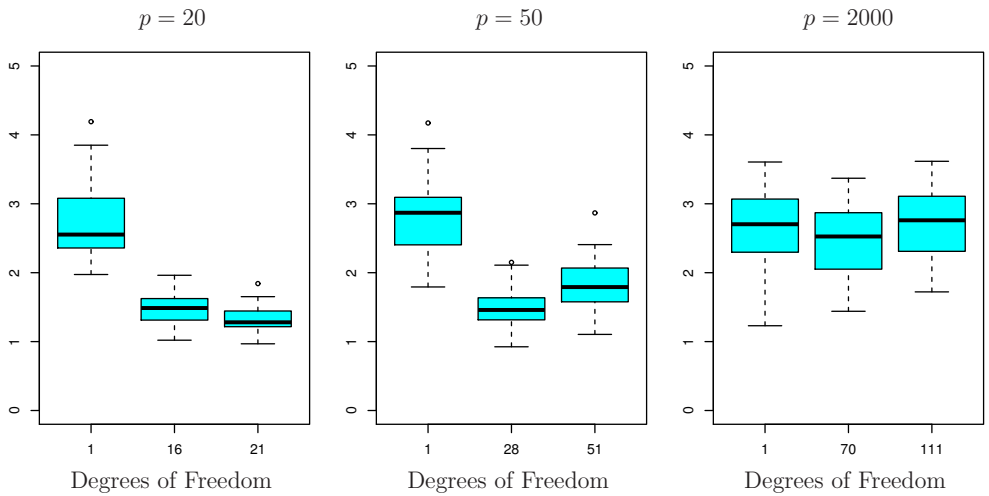
Example

Consider a distribution where only 20 predictors are truly associated to the outcome (out of 20, 50, or 2000 total predictors)



Example

Consider a distribution where only 20 predictors are truly associated to the outcome (out of 20, 50, or 2000 total predictors)



- Test error increases with more predictors and parameters
- *Lesson*: Using too many predictors/parameters can hurt performance
  - Selecting predictors carefully can/will lead to better performance

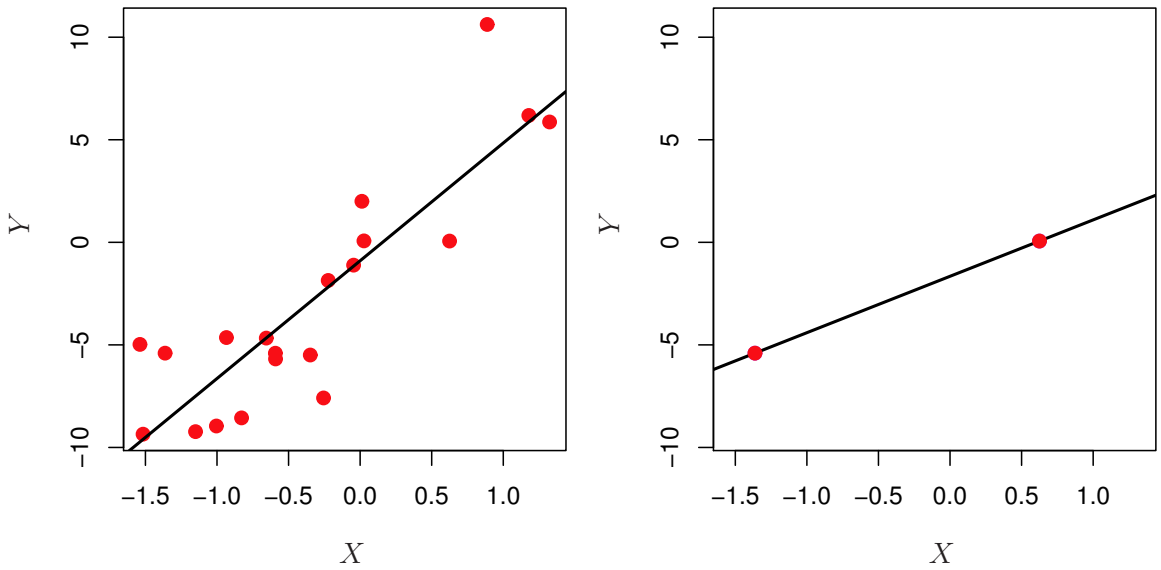
Follow-up

Question: What about if we have $p \geq n$? (pretty common now, since collecting data has become cheap and easy.)
Examples:

- **Medicine:** Instead of regressing heart disease onto just a few clinical observations (blood pressure, salt consumption, age), we use in addition 500,000 single nucleotide polymorphisms.
- **Marketing:** Using search terms to understand online shopping patterns. A *bag of words* model defines one feature for every possible search term, which counts the number of times the term appears in a person's search. There can be as many features as words in the dictionary.

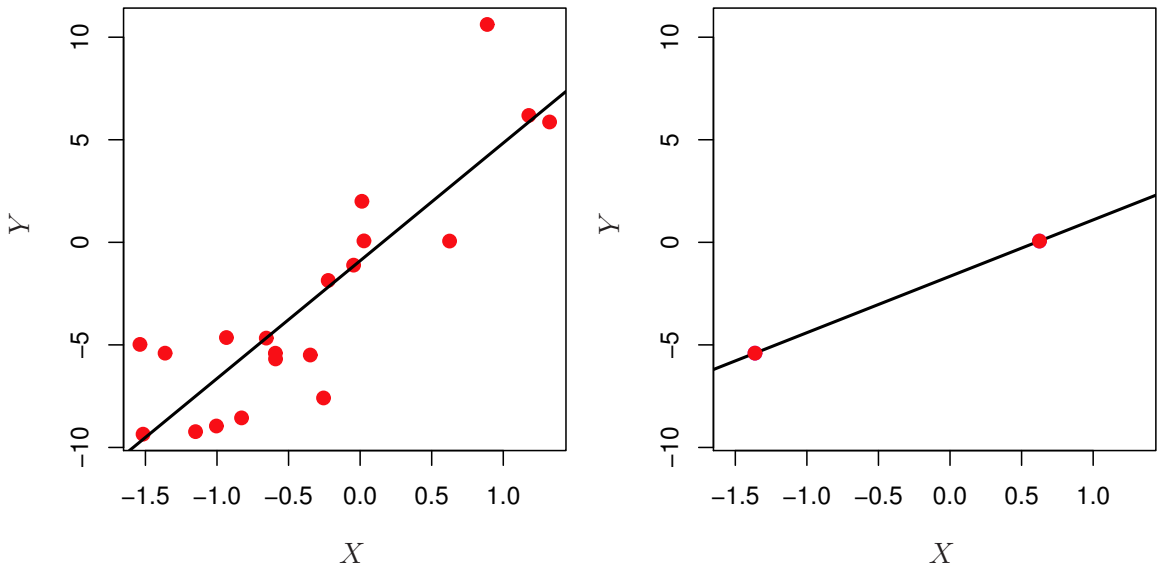
Follow-up

When $n = p$, we can find a perfect fit for the training data, e.g.



Follow-up

When $n = p$, we can find a perfect fit for the training data, e.g.

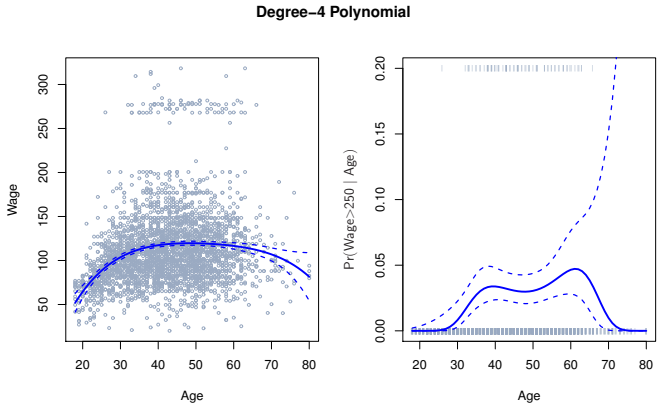


When $p > n$, OLS doesn't have a unique solution.

- Can use e.g. stepwise variable selection, ridge, lasso, etc.
- Still have to deal with estimating $\hat{\sigma}^2$.

Basis functions

Back to using polynomial terms



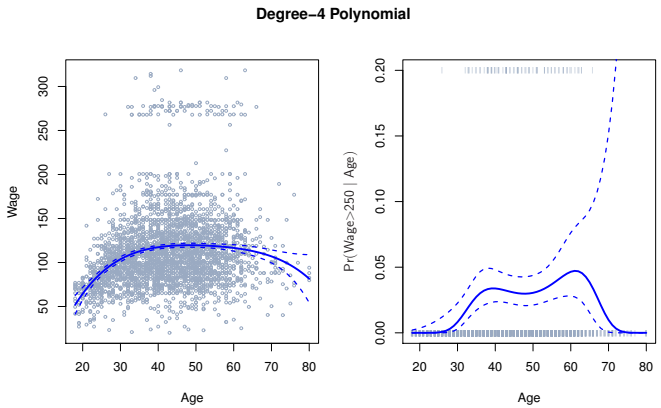
- Generalizing the notation, we have

$$\mathbb{E}[Y|\mathbf{X}] = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + ... + \beta_d b_d(X) \qquad (2)$$

  - where $b_i(x) = x^i$

Basis functions

Back to using polynomial terms



- Generalizing the notation, we have

$$\mathbb{E}[Y|\mathbf{X}] = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + ... + \beta_d b_d(X) \qquad (2)$$

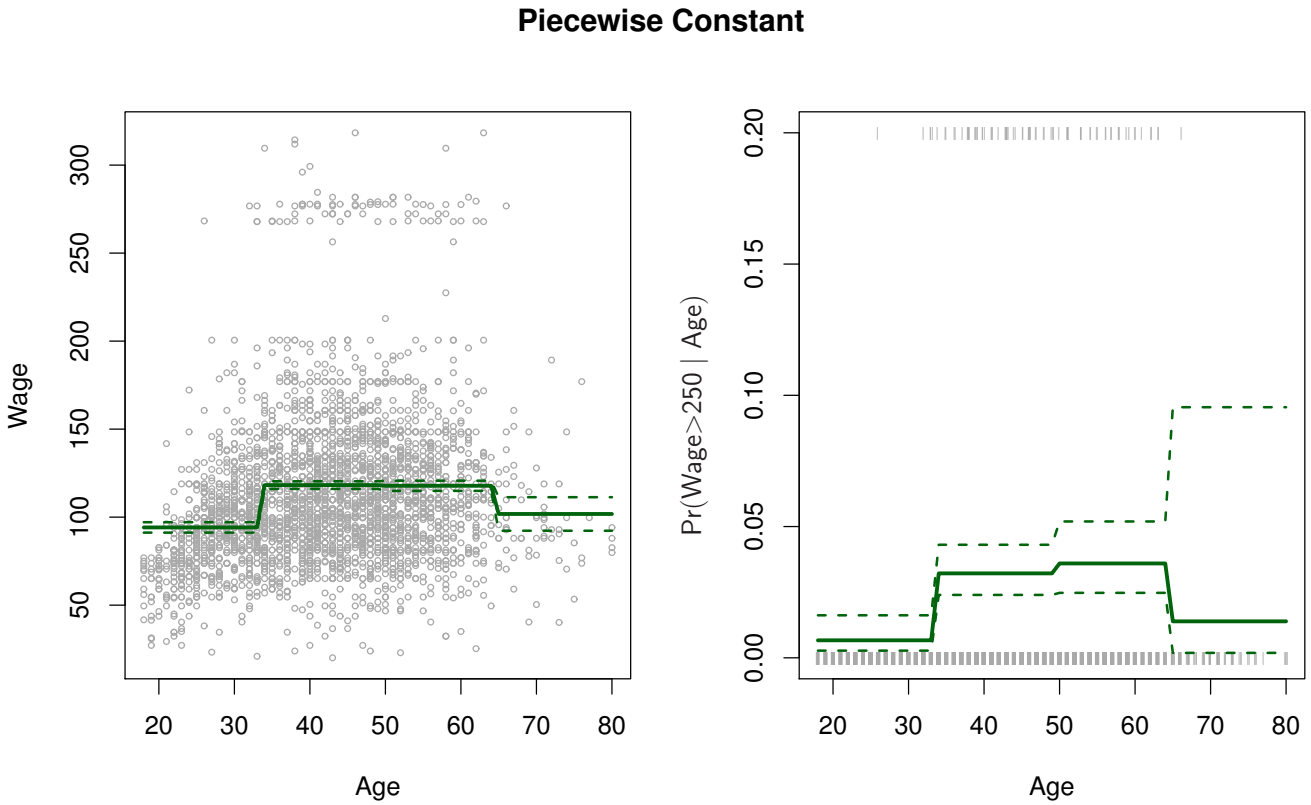
  - where $b_i(x) = x^i$

- We don't have to just use polynomials (e.g. could use cutpoints instead)

  - i.e. $b_i(x) = \mathbb{I}(c_i \leqslant x < c_{i+1})$

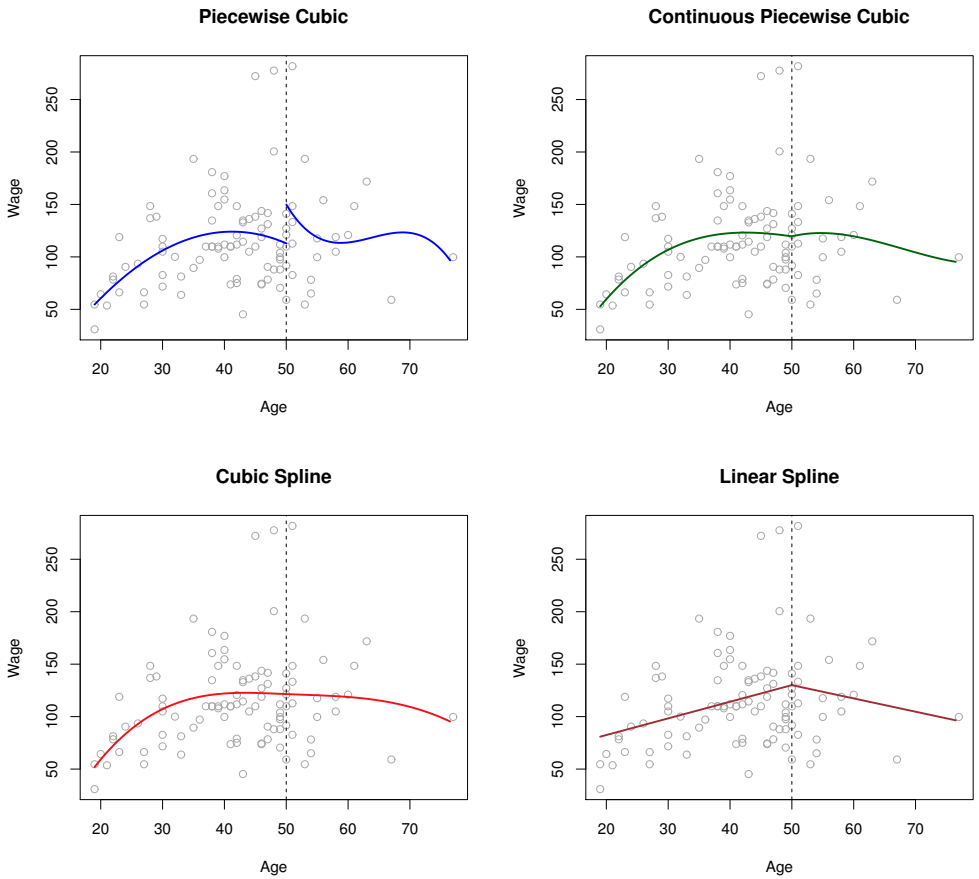
Basis functions: Example

Example using *Wage* data.



Basis functions options

We can also use other *basis functions*,
e.g. piecewise polynomials



Note: points where coefficients change are called a *knots*.

Piecewise polynomials

Example of a piecewise cubic model with 1 knot & $d = 3$:

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

i.e. We're fitting two regression models (for $x_i < c$ and $x_i \geq c$)

- This can be generalized to any number of knots $\leq n$
- Problem: resulting function is discontinuous

Cubic splines

Very popular, since they give very smooth predictions over X.

- Define a set of knots $\xi_1 < \xi_2 < \cdots < \xi_K$.

Cubic splines

Very popular, since they give very smooth predictions over X.

- Define a set of knots $\xi_1 < \xi_2 < \cdots < \xi_K$.
- We want the function $Y = f(X)$ to:
  1. Be a cubic polynomial between every pair of knots ξ_i, ξ_{i+1}.
  2. Be continuous at each knot.
  3. Have continuous first and second derivatives at each knot.

Cubic splines

Very popular, since they give very smooth predictions over X.

- Define a set of knots $\xi_1 < \xi_2 < \cdots < \xi_K$.
- We want the function $Y = f(X)$ to:
  1. Be a cubic polynomial between every pair of knots ξ_i, ξ_{i+1}.
  2. Be continuous at each knot.
  3. Have continuous first and second derivatives at each knot.

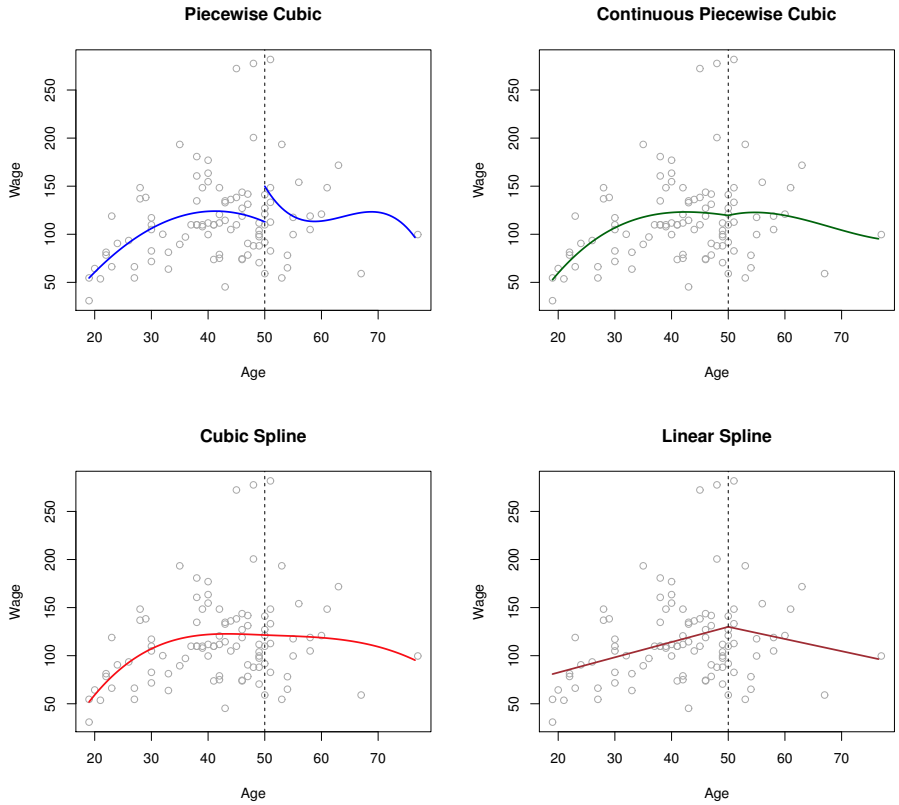
Fact: Given constraints, we need $K + 3$ basis functions:

$$f(X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 + \beta_4 h(X, \xi_1) + \cdots + \beta_{K+3} h(X, \xi_K) \tag{4}$$

where,

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

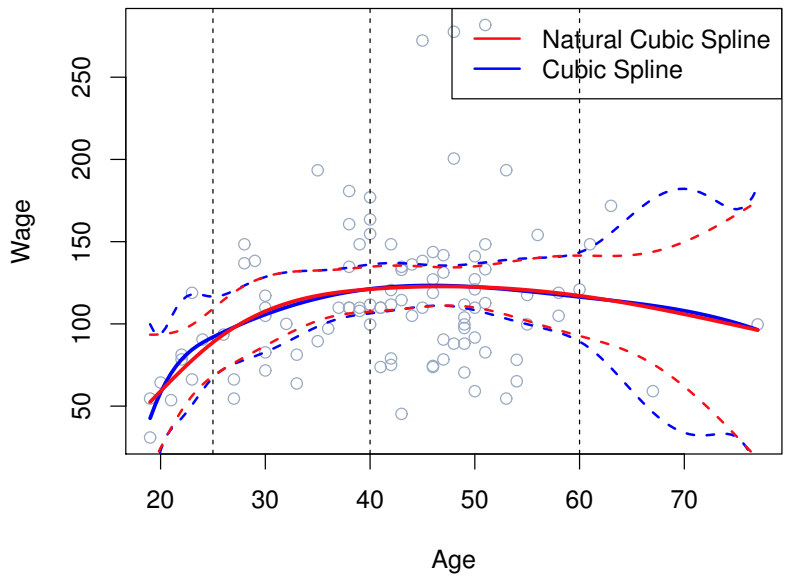
Piecewise polynomials vs cubic splines



- Piecewise model has $df = 8$.
- Cubic spline has $df = 5$.
  - In general, k knots result in $df = 4 + K$

Natural cubic splines

Spline which is linear instead of cubic for $X < \xi_1$ & $X > \xi_K$

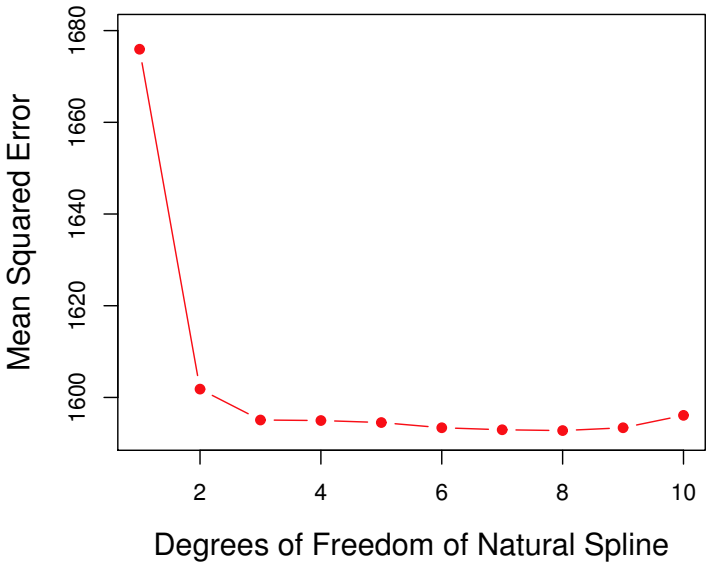


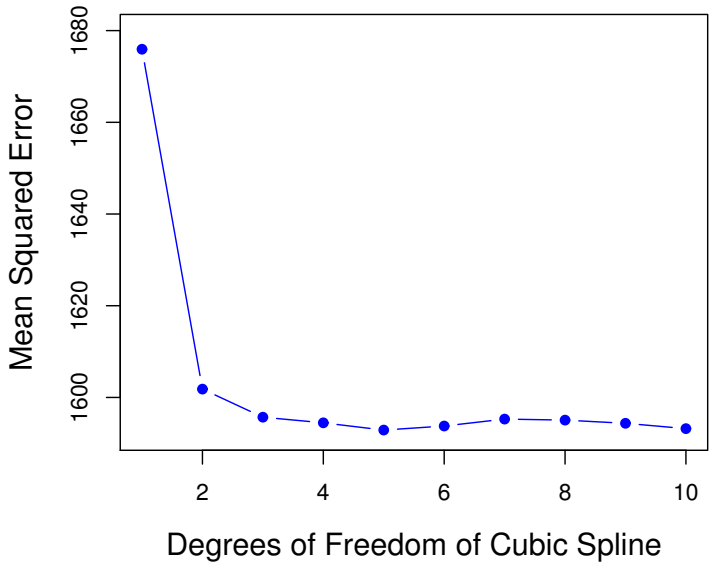
Makes predictions more stable for extreme values of X

- Will typically have less degrees of freedom than cubic splines

Number and location of knots

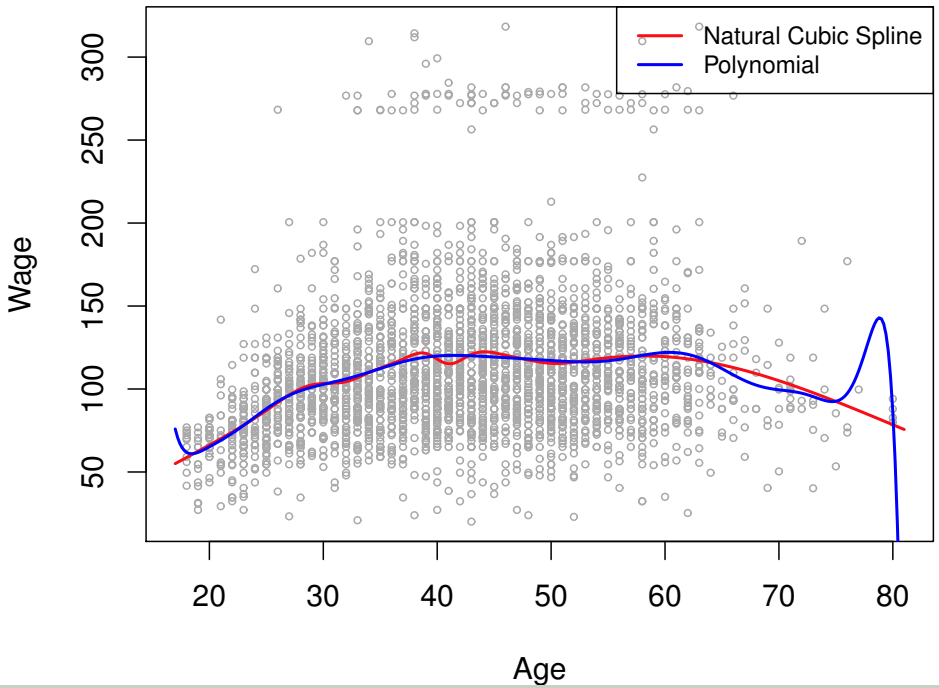
- The locations of the knots are typically quantiles of X.
- The number of knots, K, is chosen by cross validation





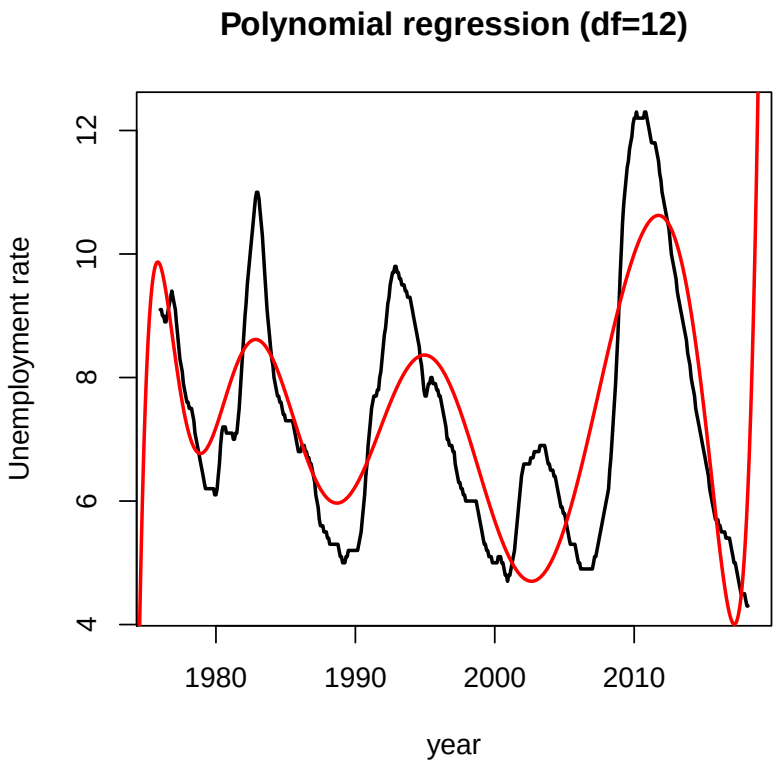
Splines vs polynomial regression

- Splines can fit complex functions with few parameters
- Polynomials require high degree terms to be flexible
  - May also result in non-convergence
- High-degree polynomials (like vanilla cubic splines) can be unstable at the edges



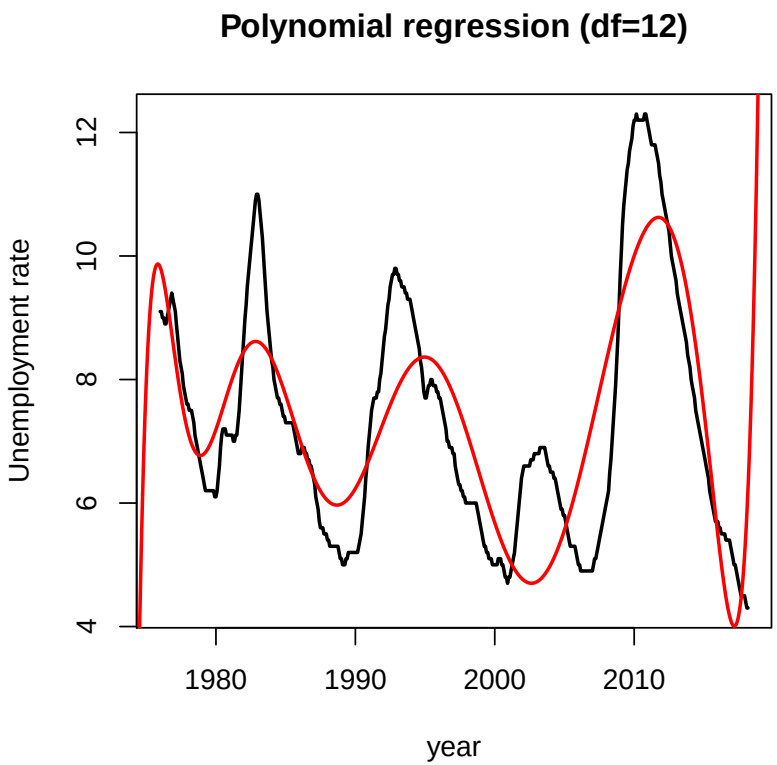
Splines vs polynomial regression

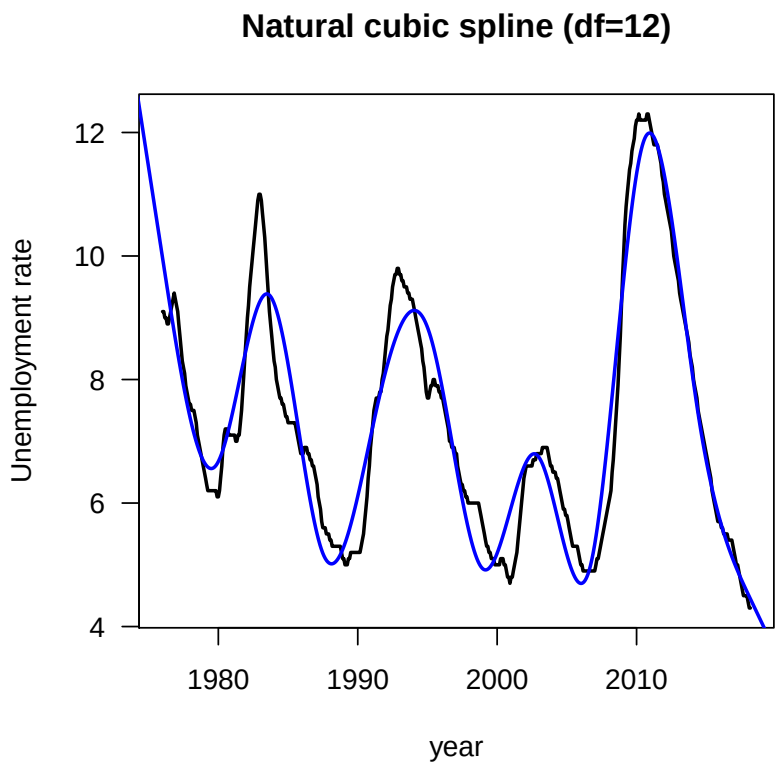
Applied example: California unemployment rates



Splines vs polynomial regression

Applied example: California unemployment rates





Smoothing splines

Our goal is to find the function f which minimizes

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$$

- The RSS of the model.
- A penalty for the roughness of the function.

Smoothing splines

Our goal is to find the function f which minimizes

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$$

- The RSS of the model.
- A penalty for the roughness of the function.

For regularization, we have that $\lambda \in (0, \infty)$

- When $\lambda = 0$, f can be any function that interpolates the data.
- When $\lambda = \infty$, f will be the simple least squares fit

Smoothing splines

Notes:

- $\hat{f}$ becomes more flexible as $\lambda \to 0$.
- The minimizer $\hat{f}$ is a natural cubic spline, with knots at each sample point $x_1, \ldots, x_n$.
  - Though usually not the same as directly applying natural cubic splines due to λ
  - Our function (Representer Theorem) can therefore be represented as

$$\hat{f}(x) = \sum_{j=1}^{n} N_j(x)\hat{\theta}_j \tag{5}$$

- Obtaining $\hat{f}$ is similar to a Ridge regression, i.e.

$$\hat{\theta} = (\mathbf{N}^\top \mathbf{N} + \lambda \mathbf{\Sigma}_N)^{-1} \mathbf{N}^\top \mathbf{y} \tag{6}$$

Natural cubic splines vs. Smoothing splines

Natural cubic splines

- Fix $K \leq n$ and choose the locations of K knots at quantiles of X.
- Find the natural cubic spline $\hat{f}$ which minimizes the RSS:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2$$

- Choose K by cross validation.

Smoothing splines

- Put n knots at $x_1, \ldots, x_n$.
- Obtain the fitted values which minimizes:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int f''(x)^2 dx$$

- Choose λ by cross validation.

LOOCV

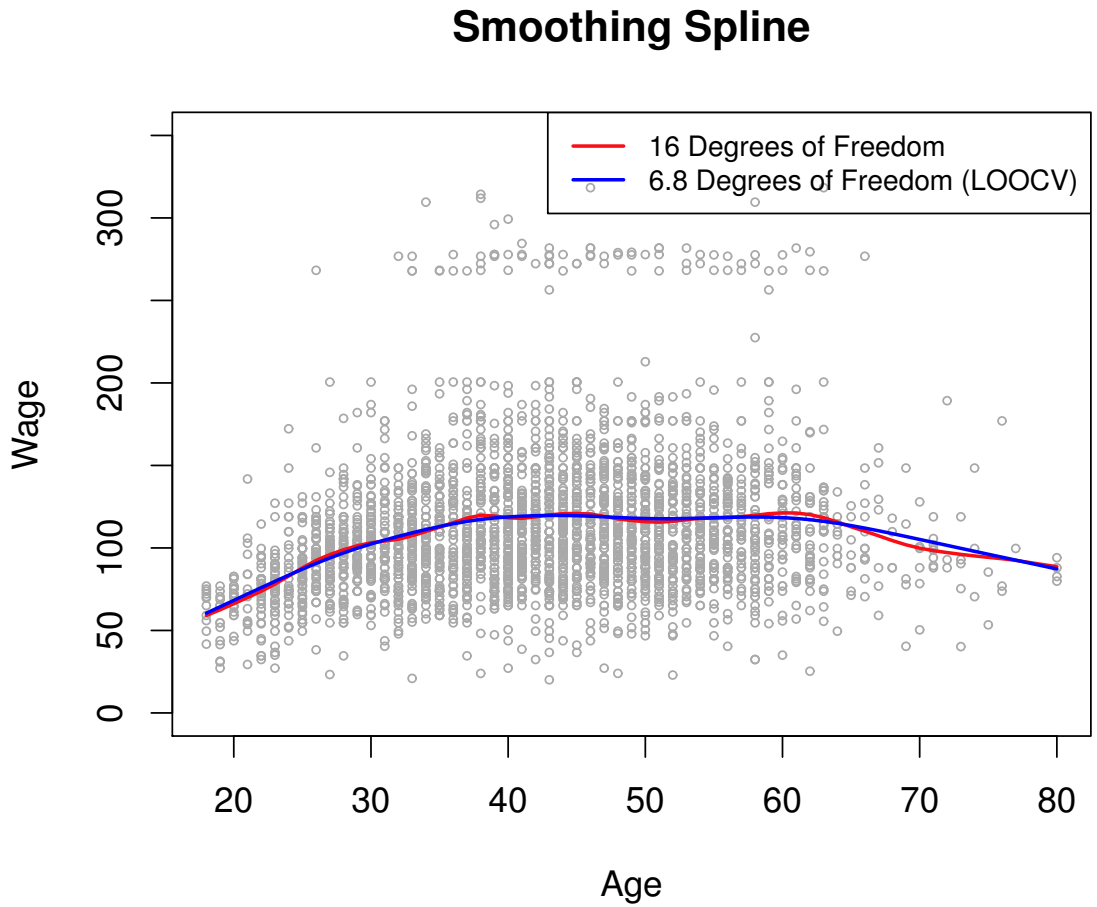
Similar to linear regression, there's a shortcut for LOOCV:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_\lambda^{(-i)}(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{y_i - \hat{f}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}}\right]^2 \quad (7)$$

for the $n \times n$ matrix

$$\mathbf{S}_\lambda = \mathbf{N}(\mathbf{N}^\top\mathbf{N} + \lambda\boldsymbol{\Sigma}_N)^{-1}\mathbf{N}^\top \quad (8)$$

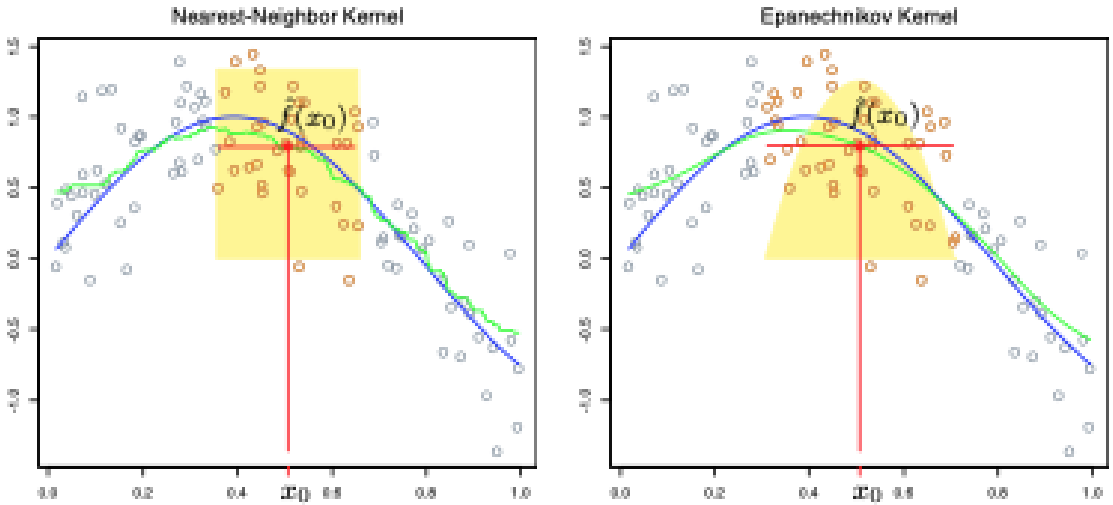
Example



n.b. We're measuring **effective** degrees of freedom

Kernel smoothing

Idea: Why not just use the subset of observations *closest* to the point we're predicting at?

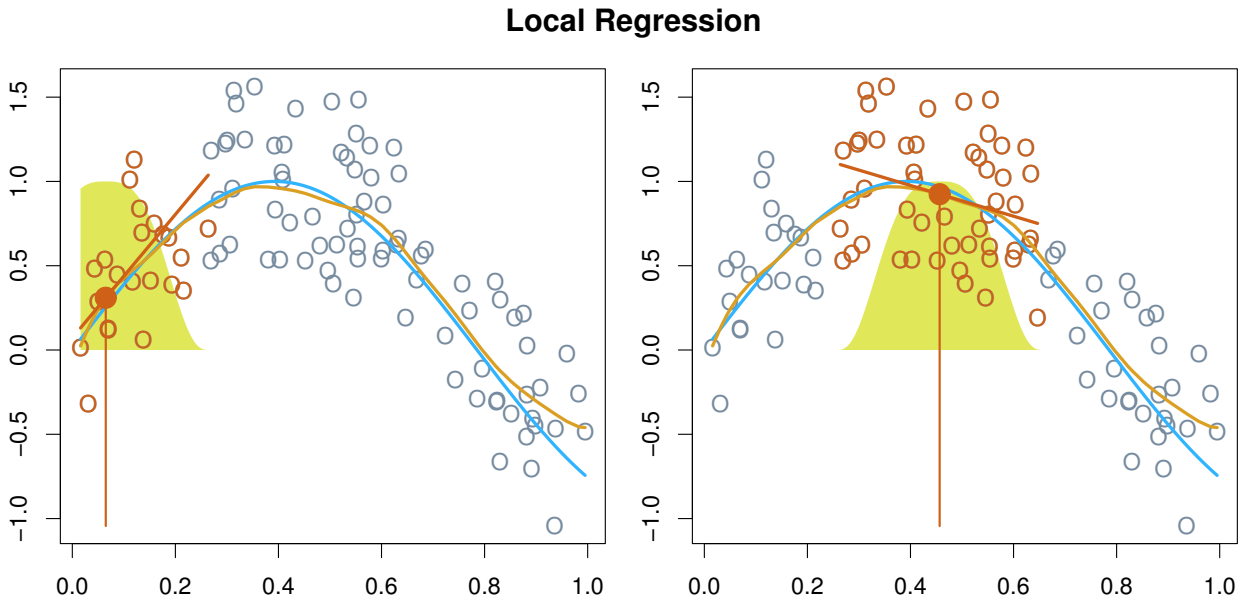


- Observations averaged *locally* for predictions.
- Can use different weighting kernels, e.g.

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{h_\lambda(x_0)}\right) \tag{9}$$

Local linear regression

Idea: Kernel smoothing with regression.



e.g. Perform regression *locally*.

Local linear regression

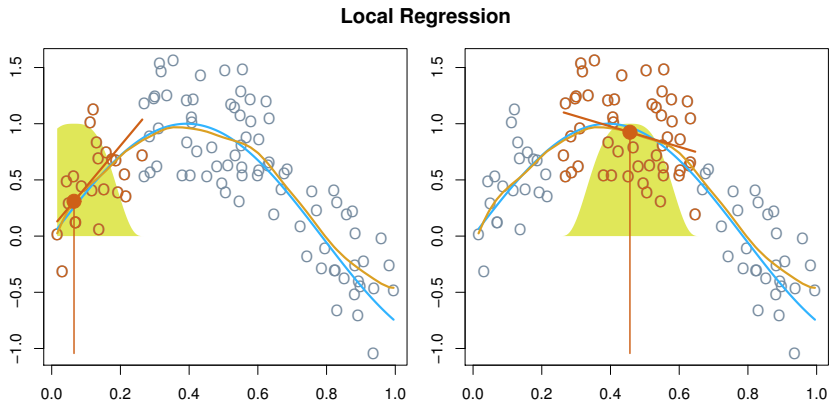
To predict the regression function f at an input x:

1. Assign a weight K_i to the training point x_i, such that:
   - $K_i = 0$ unless x_i is one of the k nearest neighbors of x.
   - K_i decreases when the distance $d(x, x_i)$ increases.
2. Perform a weighted least squares regression; i.e. find (β_0, β_1) which minimize

$$\sum_{i=i}^{n} K_i(y_i - \beta_0 - \beta_1 x_i)^2.$$

3. Predict $\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$.

Local linear regression

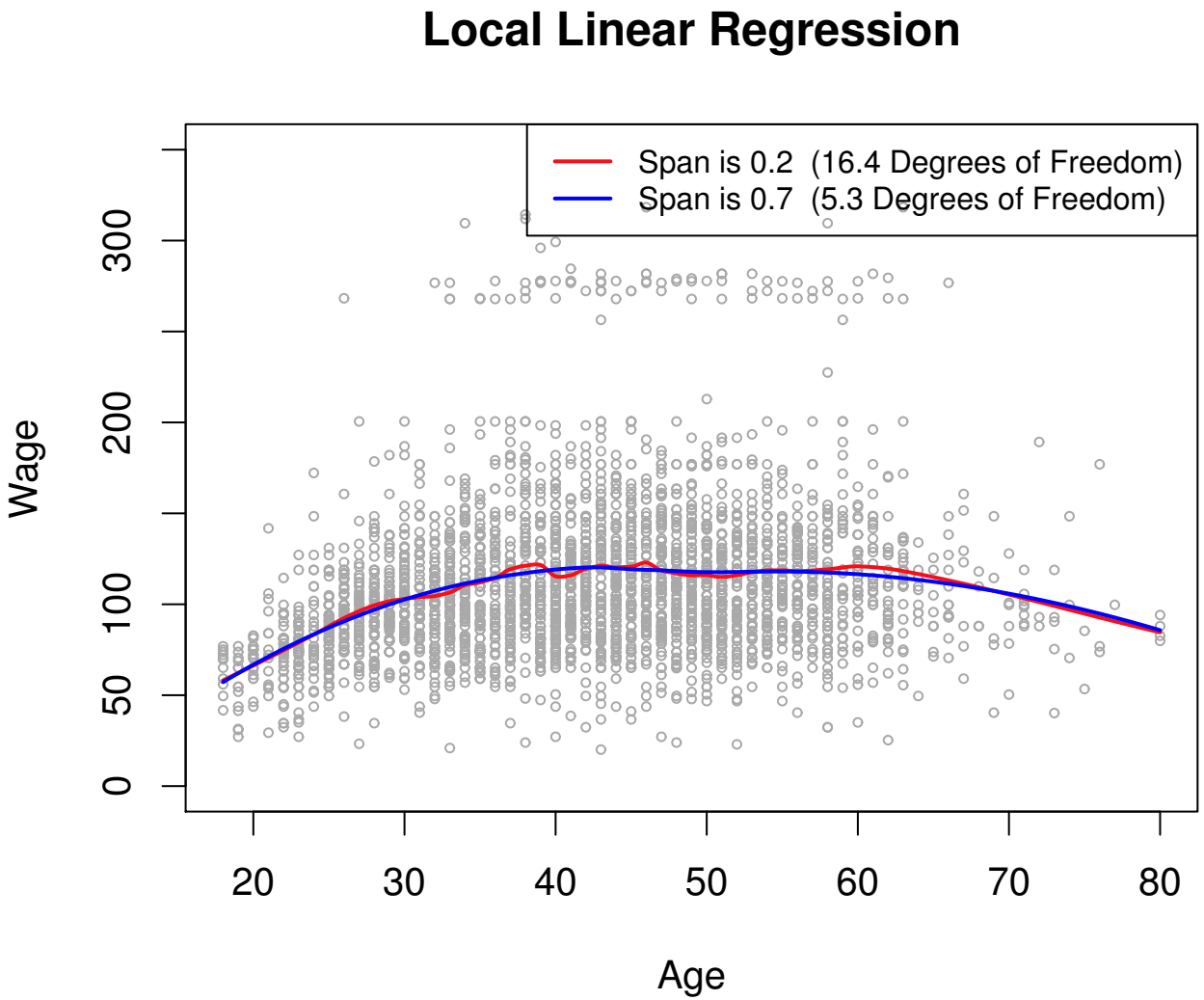


Some notes:

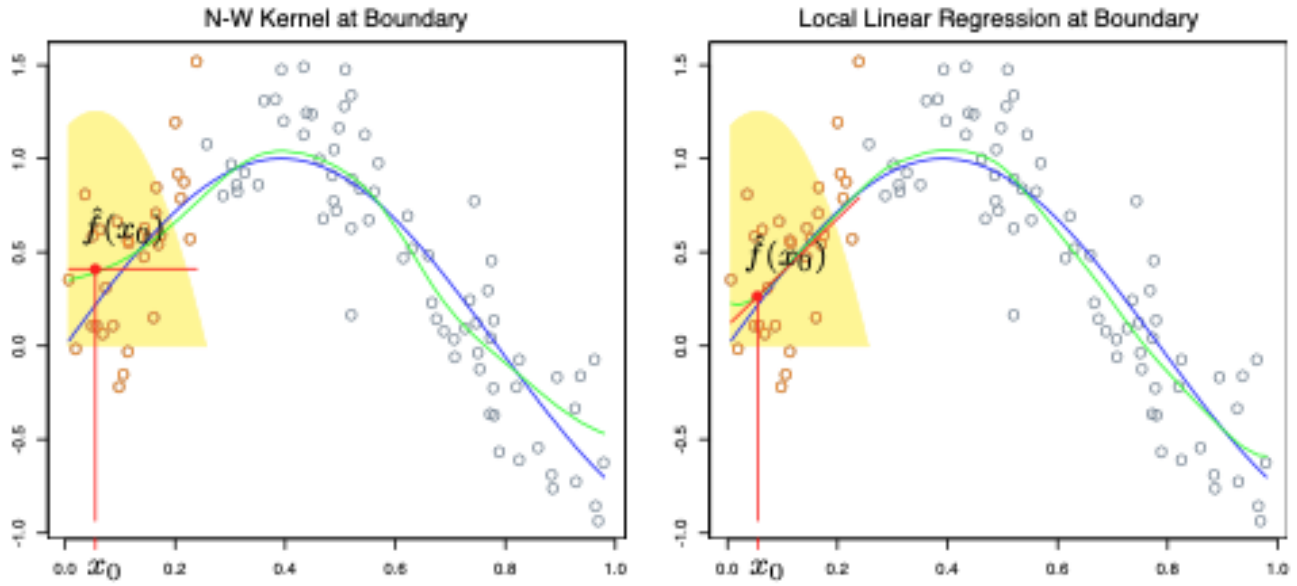
- The **span** k/n is the fraction of training samples used in each regression (the most important hyperparameter for the model)
- Sometimes referred to as a *memory-based* procedure
- Can mix this with linear models (e.g. global in some variables, but local in others)
- Performs poorly in large dimensional spaces

Local linear regression

Example



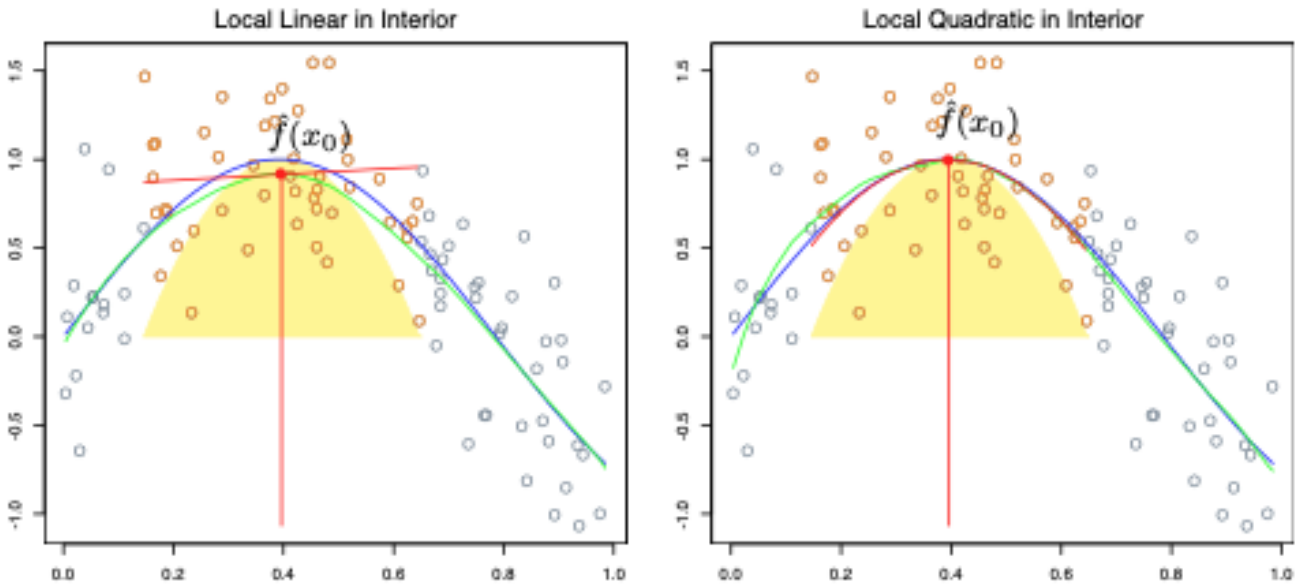
Kernel smoothing vs Local linear regression



The main differences occur in the edges.

Local polynomial regression

We don't have to stick with local linear models, e.g.



Can deal with the issue of "*trimming the hills*" and "*filling the valleys*".

Generalized additive models

The extension of basis functions to multiple predictors (while maintaining additivity) , e.g.

Linear model

$$\texttt{wage} = \beta_0 + \beta_1 \texttt{year} + \beta_2 \texttt{age} + \beta_3 \texttt{education} + \epsilon \tag{10}$$

Additive model

$$\texttt{wage} = \beta_0 + f_1(\texttt{year}) + f_2(\texttt{age}) + f_3(\texttt{education}) + \epsilon \tag{11}$$

The functions $f_1, \ldots, f_p$ can be polynomials, natural splines, smoothing splines, local regressions, etc.

Fitting GAMs

If the functions f_j have basis representations, we can use least squares

- Natural cubic splines
- Polynomials
- Step functions

Otherwise, we can use *backfitting* to fit our functions

- Smoothing splines
- Local regression

Backfitting a GAM

- Otherwise, we can use **backfitting**:
  1. Keep $f_2, \ldots, f_p$ fixed, and fit f_1 using the partial residuals:

     $$y_i - \beta_0 - f_2(x_{i2}) - \cdots - f_p(x_{ip}),$$

     as the response.
  2. Keep $f_1, f_3, \ldots, f_p$ fixed, and fit f_2 using the partial residuals:

     $$y_i - \beta_0 - f_1(x_{i1}) - f_3(x_{i3}) - \cdots - f_p(x_{ip}),$$

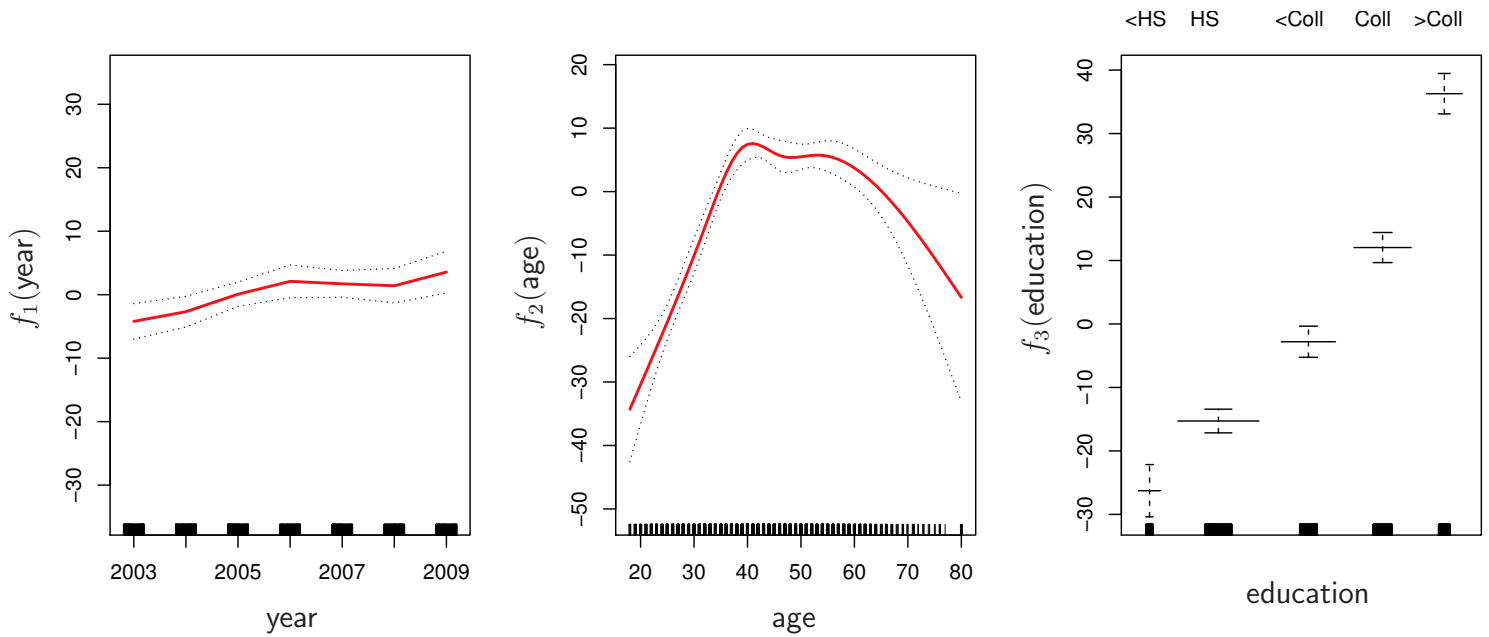
     as the response.
  3. ...
  4. Iterate
- This works for smoothing splines and local regression.

GAM properties

- GAMs are a step from linear regression toward a nonparametric method
  - Mitigates need to manually try out many different transformations on each variable individually
  - We can report degrees of freedom for most non-linear functions (as a way of representing model complexity)
  - The only constraint is additivity, which can be partially addressed through adding interaction variables, e.g. $X_i X_j$
- Similar to linear regression, we can examine the significance of each of the variables

Example: Regression for Wage

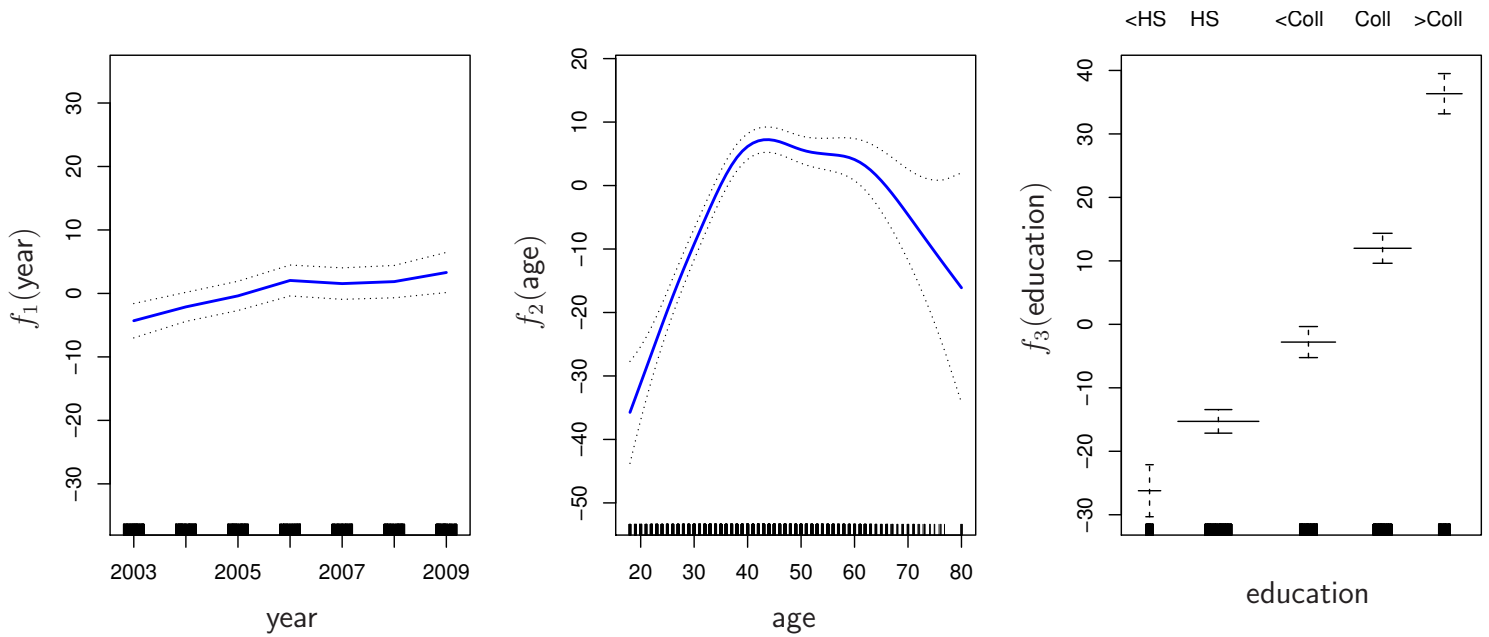
Using natural cubic splines:



- `year`: natural spline with df=4.
- `age`: natural spline with df=5.
- `education`: step function.

Example: Regression for Wage

Using smoothing splines:



- `year`: smoothing spline with df=4.
- `age`: smoothing spline with df=5.
- `education`: step function.

GAMs for Classification

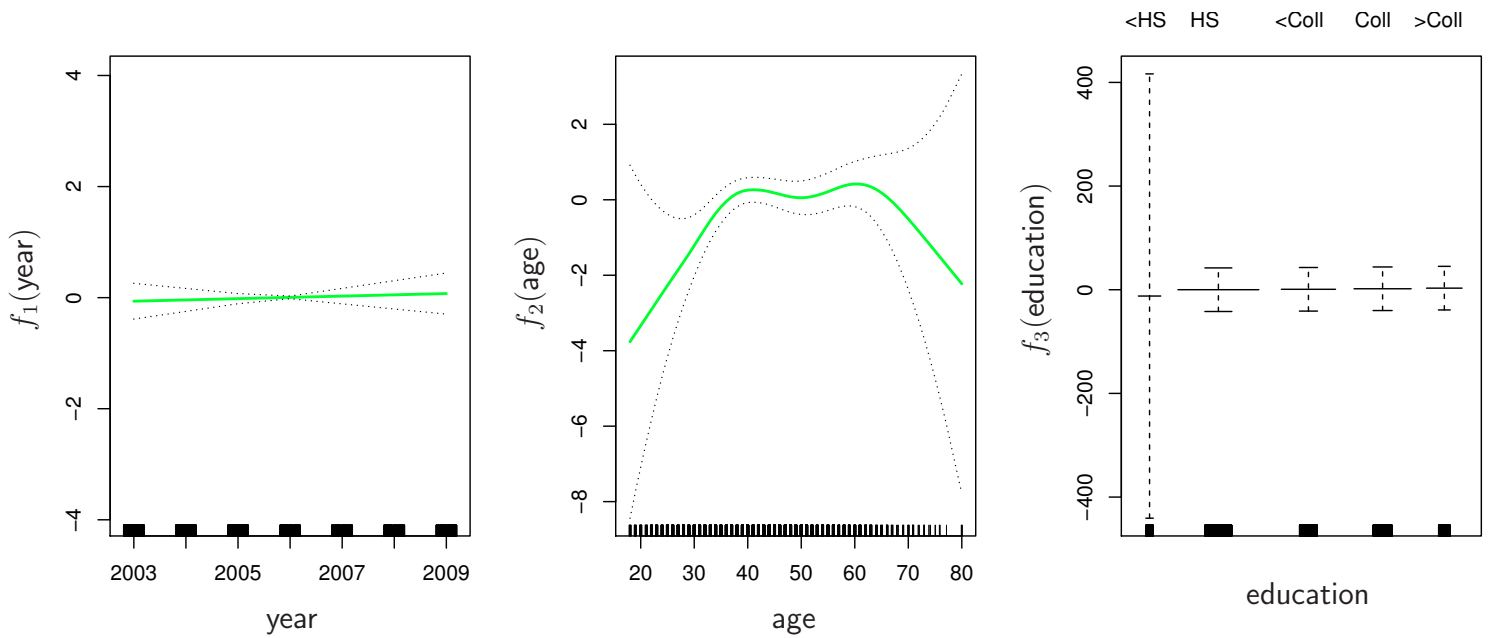
Recall: We can use logistic regression for estimating our conditional probabilities.

GAMS very naturally apply over to this:

$$\log \frac{P(Y = 1 \mid X)}{P(Y = 0 \mid X)} = \beta_0 + f_1(X_1) + \cdots + f_p(X_p).$$

The fitting algorithm is a version of backfitting, but we won't discuss the details.

Example: Classification for `Wage>250`



- `year`: linear
- `age`: smoothing spline with df=5
- `education`: step function

n.b. The confidence interval for `<HS` is very large

References

[1] ISL. Chapter 6.4-7

[2] ESL. Chapter 5