STATISTICAL LEARNING

CHAPTER 7: MOVING BEYOND LINEARITY

INSTRUCTOR: SEOKHO LEE

HANKUK UNIVERSITY OF FOREIGN STUDIES

2015 SPRING

- Linear models are relatively simple to describe and implement, and have advantages in terms of interpretation and inference
- It can have significant limitations in predictive power if the linear assumption is not appropriate
- We relax the linearity assumption while still attempting to maintain as much interpretability as possible
  - Polynomial regression
  - Step functions
  - Regression splines
  - Smoothing splines
  - Local regression
  - Generalized additive models

Polynomial Regression

- **Polynomial regression** is the standard way to extend linear regression
  - For the standard linear model $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$, we can extend it to

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_d x_i^d + \epsilon_i \tag{7.1}$$

  - For large enough degree d, a polynomial regression allows us to produce an extremely non-linear curve
  - The coefficients in (7.1) can be easily estimated using least squares linear regression
  - Generally, it is unusual to use d greater than 3 or 4 because for large values of d, the polynomial curve can become overly flexible and can take on some very strange shapes, especially near the boundary of the X variable

- Wage data
  - Polynomial regression with 4-degree (See left panel of Figure 7.1)

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4 \tag{7.2}$$

  - Logistic regression with 4-degree (See right panel of Figure 7.1)

$$\Pr(y_i > 250 | x_i) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4)} \tag{7.3}$$

Polynomial Regression

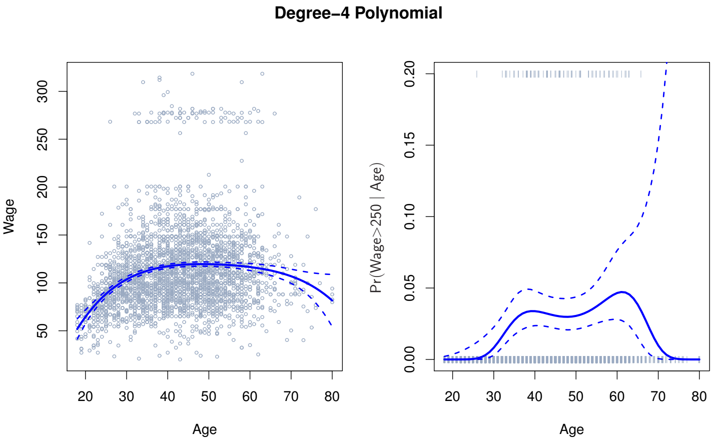

Figure 7.1: The Wage data. Left: The solid blue curve is a degree-4 polynomial of wage (in thousands of dollars) as a function of age, fit by least squares. The dotted curves indicate an estimated 95% confidence interval. Right: We model the binary event wage>250 using logistic regression, again with a degree-4 polynomial. The fitted posterior probability of wage exceeding $250,000 is shown in blue, along with an estimated 95% confidence interval.

Step Functions

- We can use **step functions** in order to avoid imposing a global structure as in polynomial regression
- Piecewise-constant regression
  - We break the range of X into **bins**, and fit a different constant in each bin
  - **Ordered categorical variable** with cutpoints $c_1, c_2, \ldots, c_K$

$$\begin{aligned} C_0(X) &= I(X < c_1) \\ C_1(X) &= I(c_1 \leq X < c_2) \\ C_2(X) &= I(c_2 \leq X < c_2) \\ &\vdots \\ C_{K-1}(X) &= I(c_{K-1} \leq X < c_K) \\ C_K(X) &= I(c_K \leq X) \end{aligned} \tag{7.4}$$

  - Then, use least squares to fit a linear model using $C_1(X), C_2(X), \ldots, C_K(X)$ as predictors:

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_K C_K(x_i) + \epsilon_i \tag{7.5}$$

item Or fit the logistic regression

$$\Pr(y_i > 250|x_i) = \frac{\exp(\beta_0 + \beta_1 C_1(x_i) + \cdots + \beta_K C_K(x_i))}{1 + \exp(\beta_0 + \beta_1 C_1(x_i) + \cdots + \beta_K C_K(x_i))} \tag{7.6}$$

Step Functions

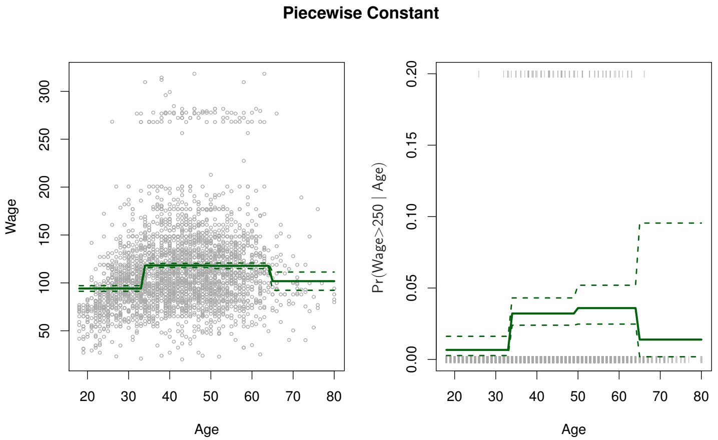

Figure 7.2: The Wage data. Left: The solid blue curve displays the fitted value from a least squares regression of wage (in thousands of dollars) using step functions of age. The dotted curves indicate an estimated 95% confidence interval. Right: We model the binary event wage>250 using logistic regression, again using step functions of age. The fitted posterior probability of wage exceeding $250,000 is shown, along with an estimated 95% confidence interval.

Basis Functions

- **Basis function** approach
  - The idea is to fit a linear model with predictors $b_1(X), b_2(X), \ldots, b_K(X)$ that is a family of functions or transformation of X:
  $$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \ldots + \beta_K b_K(x_i) + \epsilon_i \qquad (7.7)$$
  - The basis functions $b_1(\cdot), b_2(\cdot), \ldots, b_K(\cdot)$ are fixed and known
  - Polynomial and piecewise-constant regression models are special cases of a basis function approach
    - Polynomial regression: $b_j(x_i) = x_i^j$
    - Piecewise-constant regression: $b_j(x_i) = I(c_j \leq x_i < c_{j+1})$
  - Other basis functions: wavelets, Fourier series, etc.
  - **Regession splines** : polynomial regression + piecewise-constant regression

Regression Splines

- Regression splines is a flexible class of basis functions that extends upon the polynomial regression and piecewise-constant regression approaches

Piecewise Polynomials

- **Piecewise polynomial regression** involves fitting separate low-degree polynomials over different regions of X
  - Cubic regression model
$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \epsilon_i \tag{7.8}$$
  - Instead of fitting (7.8) over all region of X, the coefficients $\beta_0, \beta_1, \beta_2$, and β_3 differ in different parts of the range of X. The points where the coefficients change are called **knots**
  - A piecewise cubic polynomial with a single knot at a point c takes the form of
$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_3 + \epsilon_i & \text{if } x_i < c \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_3 + \epsilon_i & \text{if } x_i \geq c \end{cases}$$
  - In general, if we place K different knots throughout the range of X, then we will end up fitting $K + 1$ different cubic polynomials
  - See Figure 7.3
    - The immediate drawback is that the function is discontinuous and looks ridiculous!
    - Since each polynomial has four parameters, we are using a total of eight **degrees of freedom** in fitting this piecewise polynomial model

Constraints and Splines

- Figure 7.3 for Wage data
  - Top left panel looks wrong because the fitted curve is just too flexible
  - Top right panel imposes the continuity: V-shaped join looks unnatural (not continuous)

- **Cubic spline**[1]
  - Add two additional constraints: both the first and second *derivatives* of the piecewise polynomials are continuous at knots
  - In other words, the piecewise polynomial be not only continuous at knots, but also very **smooth**
  - See the bottom left panel of Figure 7.3
    - We are using 8 df, but impose 3 constraints (continuity, continuity of the first derivative, and continuity of the second derivative) $\Rightarrow$ 8-3=5 df
    - In general, a cubic spline with K knots uses a total of $4 + K$ df

[1] Cubic splines are popular because most human eyes cannot detect the discontinuity at the knots

Constraints and Splines

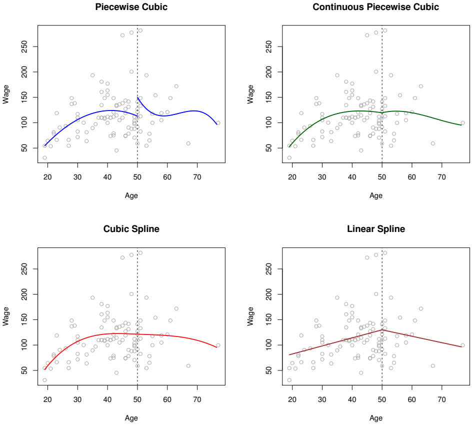

Figure 7.3: Various piecewise polynomials are fit to a subset of the Wage data, with a knot at age=50. Top Left: The cubic polynomials are unconstrained. Top Right: The cubic polynomials are constrained to be continuous at age=50. Bottom Left: The cubic polynomials are constrained to be continuous, and to have continuous first and second derivatives. Bottom Right: A linear spline is shown, which is constrained to be continuous.

The Spline Basis Representation

- How can we fit a piecewise degree-d polynomial under the constraints?
    - It turns out that we can use the basis model (7.7) to represent a regression spline
    - A cubic spline with K knots can be modeled as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i \qquad (7.9)$$

    - Basis functions $b_1, b_2, \ldots, b_{K+3}$ for cubic spline at K knots $\xi_1, \xi_2, \ldots, \xi_K$

$$b_1(x) = x, \quad b_2(x) = x^2, \quad b_3(x) = x^3$$
$$b_k(x) = h(x, \xi_k) \quad \text{for } k = 1, 2, \ldots, K$$

with

$$h(x, \xi) = (x - \xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise} \end{cases} \qquad (7.10)$$

    - This amounts to estimating a total of $K + 4$ regression coefficients; for this reason, fitting a cubic spline with K knots uses $K + 4$ df

The Spline Basis Representation

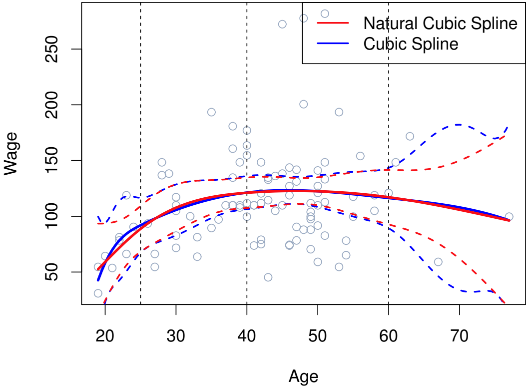

Figure 7.4: A cubic spline and a natural cubic spline, with three knots, fit to a subset of the Wage data.

The Spline Basis Representation

- Splines can have high variance at the outer range of the predictors
- A **natural spline** is a regression spline with additional **boundary constraints**:
  - The function is required to be linear at the boundary
  - This additional constraint means that natural splines generally produce more stable estimates at the boundaries
  - See Figure 7.4

Choosing the Number and Locations of the Knots

- The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly
  - One option is to place more knots in places where we feel the function might vary most rapidly, and place fewer knots in places where it seems more stable
- In practice, it is common to place knots in a uniform fashion
  - Specify the desired degrees of freedom, and then have the software automatically place the corresponding number of knots at uniform quantiles of the data
  - See Figure 7.5 for natural cubic spline applied to Wage data
- How many knots should we use, or how many df should our spline contain?
  - One option is to try out different numbers of knots and see which produces the best looking curve
  - A more objective approach is to use cross-validation
  - See Figure 7.6 for CV results from Wage data

Choosing the Number and Locations of the Knots

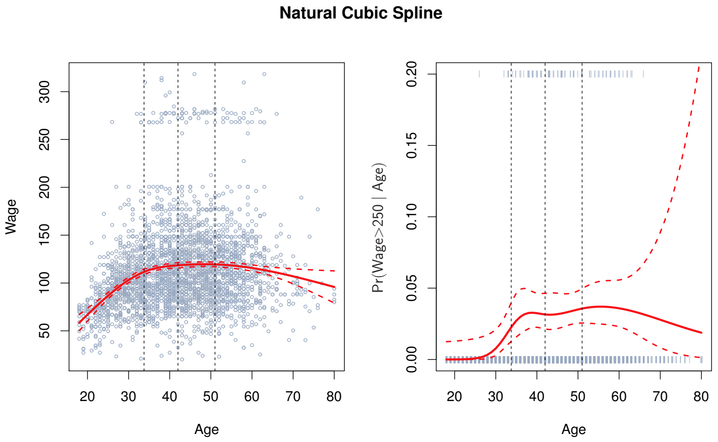

Figure 7.5: A natural cubic spline function with four degrees of freedom is fit to the Wage data. Left: A spline is fit to wage (in thousands of dollars) as a function of age. Right: Logistic regression is used to model the binary event wage>250 as a function of age. The fitted posterior probability of wage exceeding $250,000 is shown.

Choosing the Number and Locations of the Knots

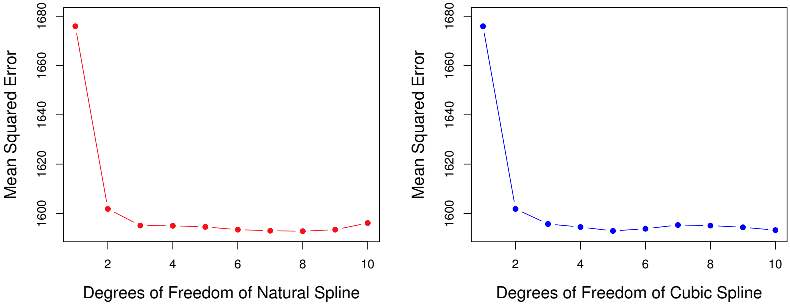

Figure 7.6: Ten-fold cross-validation mean squared errors for selecting the degrees of freedom when fitting splines to the Wage data. The response is wage and the predictor age. Left: A natural cubic spline. Right: A cubic spline.

Comparison to Polynomial Regression

- Regression splines often give superior results to polynomial regression
  - Usually polynomials must use a high degree (exponent in the highest monomial term, e.g. X^{15}) to produce flexible fits
  - Splines introduce flexibility by increasing the number of knots but keeping the degree fixed. Generally, this approach produces more stable estimates
  - Splines also allow us to place more knots, and hence flexibility, over regions where the function f seems to be changing rapidly, and fewer knots where f appears more stable
  - See Figure 7.7

Comparison to Polynomial Regression

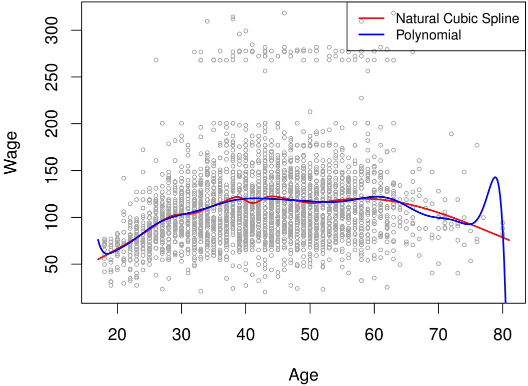

Figure 7.7: On the Wage data set, a natural cubic spline with 15 degrees of freedom is compared to a degree-15 polynomial. Polynomials can show wild behavior, especially near the tails.

An Overview of Smoothing Splines

- **Smoothing spline** is a different approach that also produces a spline
- Motivation of smoothing spline
  - What we really want to do is to find some function, say $g(x)$, that fits the observed data well: that is, we want RSS $= \sum_{i=1}^{n}(y_i - g(x_i))^2$ to be small
  - If we don't put any constraints on $g(x_i)$, then we can always make RSS zero simply by choosing g such that it **interpolates** all of the y_i
  - Such a function would woefully overfit the data–it would be far too flexible
  - What we really want is a function g that makes RSS small, but that is also *smooth*
- Smoothing spline approach is to find the function g that minimizes

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt \tag{7.11}$$

  - λ is a nonnegative **tuning parameter**
  - The function g that minimizes (7.11) is known as a **smoothing spline**

An Overview of Smoothing Splines

- Meaning of smoothing spline
  - Equation 7.11 takes the "Loss+Penalty" formulation as in the context of ridge regression and the lasso in Chapter 6
  - **Loss function**: the term $\sum_{i=1}^{n}(y_i - g(x_i))^2$ encourages g to fit the data well
  - **Penalty term**: the term $\lambda \int g''(t)^2 dt$ penalizes the variability in g
  - The second derivative $g''(t)$ corresponds to the amount by which the slope is changing.
  - The second derivative of a function is a measure of its **roughness**
    - It is large in absolute value if $g(t)$ is very wiggly near t, and it is close to zero otherwise
    - The second derivative of a straight line is zero; note that a line is perfectly smooth
    - $\int g''(t)^2 dt$ is simply a measure of the total change in the function $g'(t)$, over its entire range

An Overview of Smoothing Splines

- $\lambda \int g''(t)^2 dt$ encourages g to be smooth
    - When $\lambda = 0$, the the penalty term in (7.11) has no effect, and so the function g will be very jumpy and will exactly interpolate the training observations
    - When $\lambda \to \infty$, g will be perfectly smooth–it will just be a straight line that passes as closely as possible to the training points. In such case g will be the linear least squares line
    - For the intermediate value of λ, g will approximate the training observations but will be somewhat smooth
    - λ controls the bias-variance trade-off of the smoothing spline
- Smoothing spline is a **shrunken** version of natural cubic spline
    - The function $g(x)$ that minimizes (7.11) is a natural cubic spline with knots at $x_1, \dots, x_n$
    - The tuning parameter λ controls the level of shrinkage

Choosing the Smoothing Parameter λ

- The tuning parameter λ controls the roughness of the smoothing spline, or the **effective degrees of freedom**
  - In fitting a smoothing spline, we do not need to select the number or location of the knots
  - It might seem that a smoothing spline will have far too many degrees of freedom, since a knot at each data point allows a great deal of flexibility
  - As λ increases from 0 to ∞, the effective degrees of freedom (df_λ) decrease from n to 2
- Effective degrees of freedom
  - Usually degrees of freedom refer to the number of free parameters, such as the number of coefficients fit in a polynomial or cubic spline
  - df_λ is a measure of the flexibility of the smoothing spline–the higher it is, the more flexible (and the low-bias but high-variance) the smoothing spline

Choosing the Smoothing Parameter λ

- Effective degrees of freedom
  - We can write the smoothing spline as a linear combination of response
    $$\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y} \tag{7.12}$$
  - $\mathbf{S}_\lambda$ is an $n \times n$ hat matrix
  - The effective degrees of freedom is defined as the trace of the hat matrix
    $$df_\lambda = \mathrm{tr}(\mathbf{S}_\lambda) = \sum_{i=1}^{n} \{\mathbf{S}\}_{ii} \tag{7.13}$$
  - Note that, in the linear regression, the hat matrix is $\mathbf{H} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ and its trace is $p + 1$, which is the number of coefficients including the intercept
- Choosing λ
  - Cross-validation can be used to select λ
  - It turns out that the **leave-one-out** cross-validation error (LOOCV) can be computed very efficiently for smoothing splines

Choosing the Smoothing Parameter λ

- LOOCV
  - LOOCV computation essentially cost the same as computing a single fit, using the following formula:

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

  - $\hat{g}_\lambda^{(-i)}(x_i)$ indicates the fitted value evaluated at x_i, where the fit uses all of the training observations except the the ith observation (x_i, y_i)
  - $\hat{g}_\lambda(x_i)$ indicates the smoothing spline function fit to all of the training observations and evaluated at x_i
  - This remarkable formula says that we can compute each of these *leave-one-out* fits using only $\hat{g}_\lambda$, the original fit to *all* of the data!
  - The exact formulas for computing $\hat{g}(x_i)$ and $\mathbf{S}_\lambda$ are very technical; however, efficient algorithms are available for computing these quantities
  - See (5.2) which is very similar for least square linear regression

Choosing the Smoothing Parameter λ

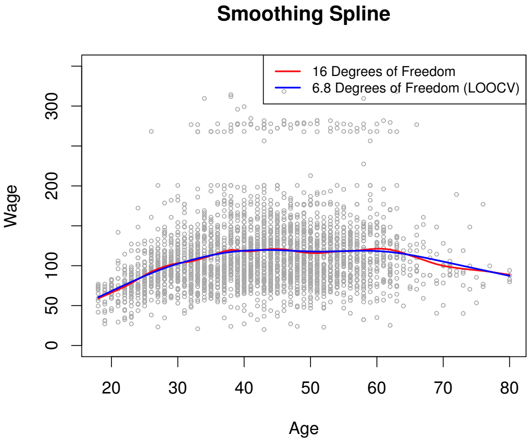

Figure 7.8: Smoothing spline fits to the Wage data. The red curve results from specifying 16 effective degrees of freedom. For the blue curve, λ was found automatically by leave-one-out cross-validation, which resulted in 6.8 effective degrees of freedom.

Local Regression

- **Local regression** is a different approach for fitting flexible non-linear functions, which involves computing the fit at a target point x_0 using only the nearby training observations
- Figure 7.9 illustrates the idea
- See Algorithm 7.1 for local regression
- Some components to be chosen for local regression
  - Weighting function K
  - linear, constant, or quadratic regression in Step 3 ((7.14) corresponds to a linear regression)
  - **span** s: the most important choice (See Figure 7.10)
    - The span plays a role like that of the tuning parameter λ in smoothing spline
    - It controls the flexibility of the non-linear fit. The smaller value of s, the more **local** and wiggly will be our fit; alternatively, a very large value of s will lead to a global fit to the data using all of the training observations

Local Regression

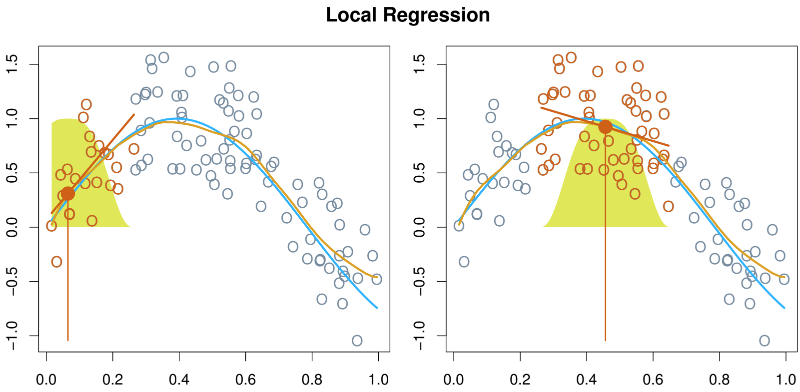

Figure 7.9: Local regression illustrated on some simulated data, where the blue curve represents $f(x)$ from which the data were generated, and the light orange curve corresponds to the local regression estimate $\hat{f}(x)$. The orange colored points are local to the target point x_0, represented by the orange vertical line. The yellow bell-shape superimposed on the plot indicates weights assigned to each point, decreasing to zero with distance from the target point. The fit $\hat{f}(x_0)$ at x_0 is obtained by fitting a weighted linear regression (orange line segment), and using the fitted value at x_0 (orange solid dot) as the estimate $\hat{f}(x_0)$.

Local Regression

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 furthers 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 minimizes

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

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

Local Regression

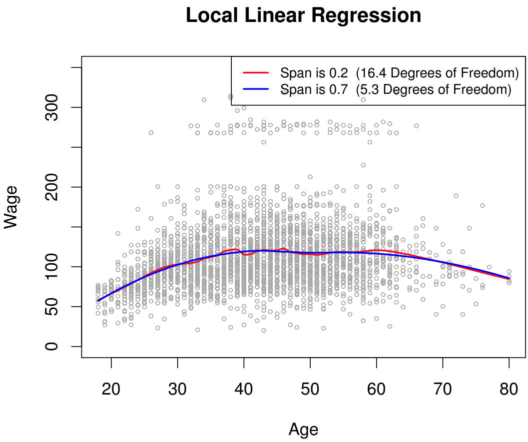

Figure 7.10: Local linear fits to the Wage data. The span specifies the fraction of the data used to compute the fit at each target point.

Local Regression

- Local regression also generalizes very naturally for multi-dimensional predictors
- Local regression can perform poorly if p is much learner than about 3 or 4
  - There will generally be very few training observations close to x_0, as nearest-neighbors regression suffers from a similar problem in high dimensions

Generalized Additive Models

- In the previous slides, we present a number of approaches for flexibly predicting a response Y on the basis of a single predictor X
- **Generalized additive models** (GAMs) are extension of multiple linear regression
  - GAMs allow non-linear functions of each of the variables, while maintaining **additivity**
  - GAMs can be applied with both quantitative and qualitative responses (**generalized** linear models)

GAMs for Regression Problems

- Multiple linear regression model with p predictors

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

- GAMs

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i \qquad (7.15) \end{aligned}$$

  - Each linear component $\beta_j x_{ij}$ in linear regression is replaced with a (smooth) non-linear function $f_j(x_{ij})$
  - We can use the methods for a single predictor as building blocks for fitting an additive model

GAMs for Regression Problems

- Wage data example (Figures 7.11 and 7.12)
  - Natural splines for two quantitative predictors year and age, dummy variable for a qualitative predictor education

$$\text{wage} = \beta_0 + f_1(\text{year}) + f_2(\text{age}) + f_3(\text{education}) + \epsilon \qquad (7.16)$$

  - Figure 7.11
    - Natural spline can be constructed using an appropriately chosen set of basis functions. Hence the entire model is just a big regression onto spline basis variables and dummy variables, all packed into one big regression matrix
  - Figure 7.12
    - In this time, f_1 and f_2 are smoothing splines with 4 and 5 degrees of freedom, respectively
    - Fitting a GAM with a smoothing splines cannot be performed using least squares
    - Standard software such as the gam() function in R can be used to fit GAMs using smoothing splines, via an approach known as **backfitting**
    - The backfitting method fits a model involving multiple predictors by repeatedly updating the fit for each predictor in turn, holding the others fixed

GAMs for Regression Problems

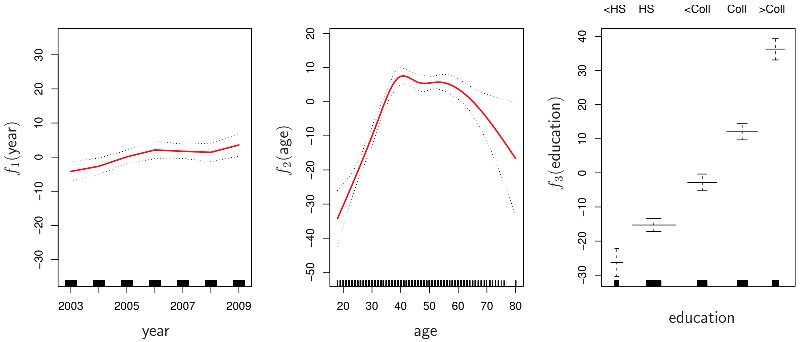

Figure 7.11: For the Wage data, plots of the relationship between each feature and the response, wage, in the fitted model (7.16). Each plot displays the fitted function and point wise standard errors. The first two functions are natural splines in year and age, with four and five degrees of freedom, respectively. The third function is a step function, fit to the qualitative variable education

GAMs for Regression Problems

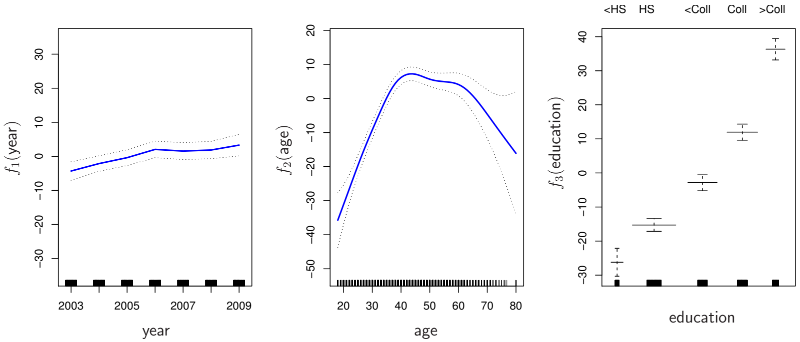

Figure 7.12: Details are as in Figure 7.11, but now f_1 and f_2 are smoothing splines with four and five degrees of freedom, respectively.

Pros and Cons of GAMs

P. GAMs allow us to fit a non-linear f_j to each X_j, so that we can automatically model non-linear relationships that standard linear regression will miss. This means that we do not need to manually try out many different transformations on each variable individually

P. The non-linear fits can potentially make more accurate predictions for the response Y

P. Because the model is additive, we can still examine the effects of each X_j on Y individually while holding all of the other variables fixed. Hence if we are interested in inference, GAMs provides a useful representation

P. The smoothness of the function f_j for the variable X_j can be summarized via degrees of freedom

C. The main limitation of GAMs is that the model is restricted to be additive. With many variables, important interactions can be missed

- However, as with linear regression, we can manually add interaction terms to the GAM model by including additional predictors of the form $X_j \times X_k$
- In addition we can add low-dimensional interaction functions of the form $f_{jk}(X_j, X_k)$ into the model; such terms can be fit using two-dimensional smoothers such as local regression, or two-dimensional splines (thin-plate spline)

GAMs for Classification Problems

- Logistic regression with p predictors

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p \qquad (7.17)$$

- GAMs as an extension of (7.17)

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) \qquad (7.18)$$

- Wage data example (Figure 7.13)

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 \times \text{year} + f_2(\text{age}) + f_3(\text{education}) \qquad (7.19)$$

with $p(X) = \Pr(\text{wage} > 250 | \text{year}, \text{age}, \text{education})$

GAMs for Classification Problems

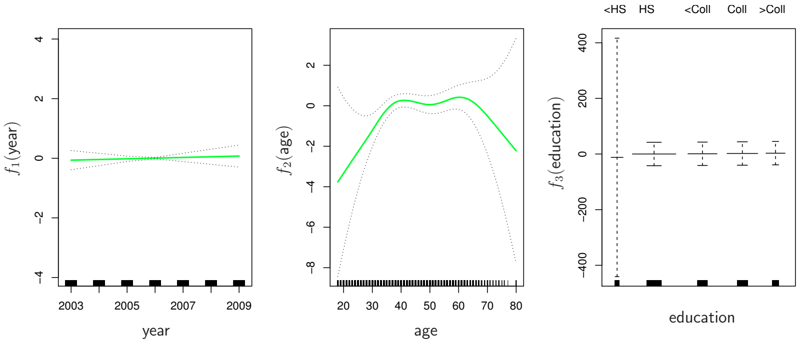

Figure 7.13: For the Wage data, the logistic regression GAM given in (7.19) is fit to the binary response I(wage$>$ 250). Each plot displays the fitted function and point wise standard errors. The first function is linear in year, the second function a smoothing spline with five degrees of freedom in age, and the third a step function for education. There are very wide standard errors for the first level $<$HS of education.

Lab: Non-linear Modeling