Statistical Learning and Prediction

Splines and GAMs

Marjolein Fokkema

Leiden University

Modeling non-linearity

- Can use polynomial regression, e.g., cubic:

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

- More generally, of degree d:

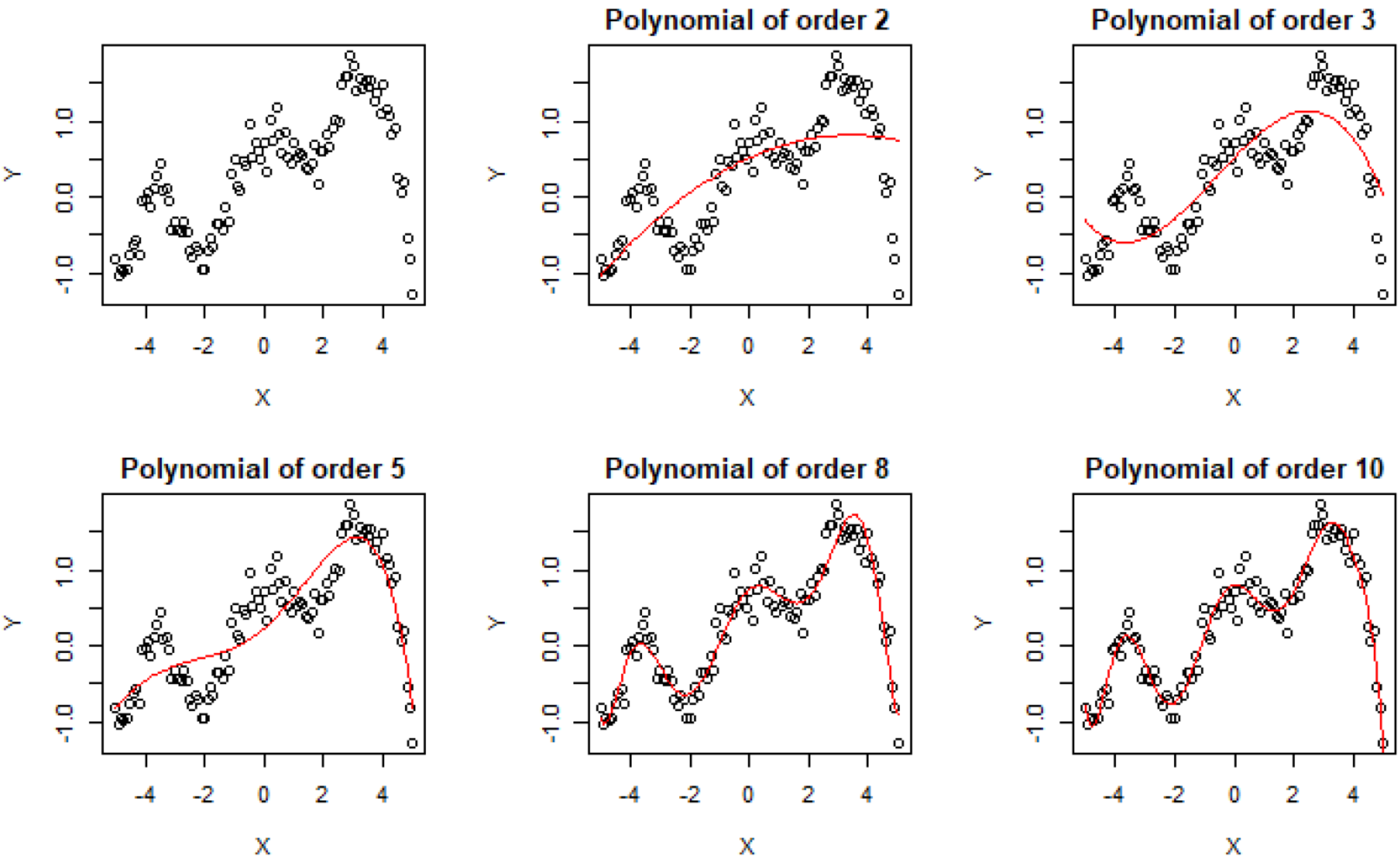
$$y_i = \beta_0 + \sum_{j=1}^{d} \beta_j x_i^j + \epsilon_i$$

$$\mathbb{E}(y_i | x_i) = \beta_0 + \sum_{j=1}^{d} \beta_j x_i^j$$

- Can generalize as usual within the GLM, e.g., for binary outcome:

$$\log\left(\frac{\pi_i}{1 - \pi_i}\right) = \beta_0 + \sum_{j=1}^{d} \beta_j x_i^j$$

Flexibility may been needed, but ...



Why splines?

- Splines *localize* the fitted model $\hat{f}(x)$ while enforcing *smoothness*.

- The *smoother* $\hat{f}(x)$, the less change there is in the *derivatives* w.r.t. x of $\hat{f}(x)$.

- Q: What is the derivative of $\hat{f}(x)$ w.r.t. x in a linear model? Does it change as a function of x?

Polynomial regression, generalized additive models

- Interest in inference and in flexible approximation of shape of association; no interest in individual coefficients (difficult to interpret).

- Model selection and hypothesis testing through:

  1. Statistical tests (t and/or F tests, ANOVA, likelihood ratio tests)
  2. Information criteria (AIC, BIC)
  3. Cross validation

Step functions

- Define cut points $\xi_1, \xi_2, \ldots, \xi_K$ and with these, step functions:

$$\begin{aligned}
h_0(X) &= I(X < \xi_1) \\
h_1(X) &= I(\xi_1 \le X < \xi_2) \\
\ldots & \\
h_{K-1}(X) &= I(\xi_{K-1} \le X < \xi_K) \\
h_K(X) &= I(\xi_K \le X)
\end{aligned}$$

- Can use $h_k(X)$ as predictors in a regression:

$$y_i = \beta_0 + \sum_{k=1}^{K} \beta_k h_k(x_i) + \epsilon_i$$

Basis functions

- Polynomials and step functions are *basis functions* of the predictors.
- We can define basis functions of each predictor X_j: $h_{j,1}(X_j), h_{j,2}(X_j), \ldots, h_{j,K_j}(X_j)$
- Generalized Additive Model: Use basis functions instead of original X_j as predictors in a (G)LM:

$$f(x_i) = \beta_0 + \sum_{j=1}^{p} \sum_{k_j=1}^{K_j} \beta_{k_j} h_{k_j}(x_{ij}) = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij})$$

- If functions h_{k_j} are known in advance, parameters can be estimated 'as usual' (with OLS, ML, ...).
- Better: Precise shape is unknown in advance, so use penalized estimation (i.e., smoothing splines).

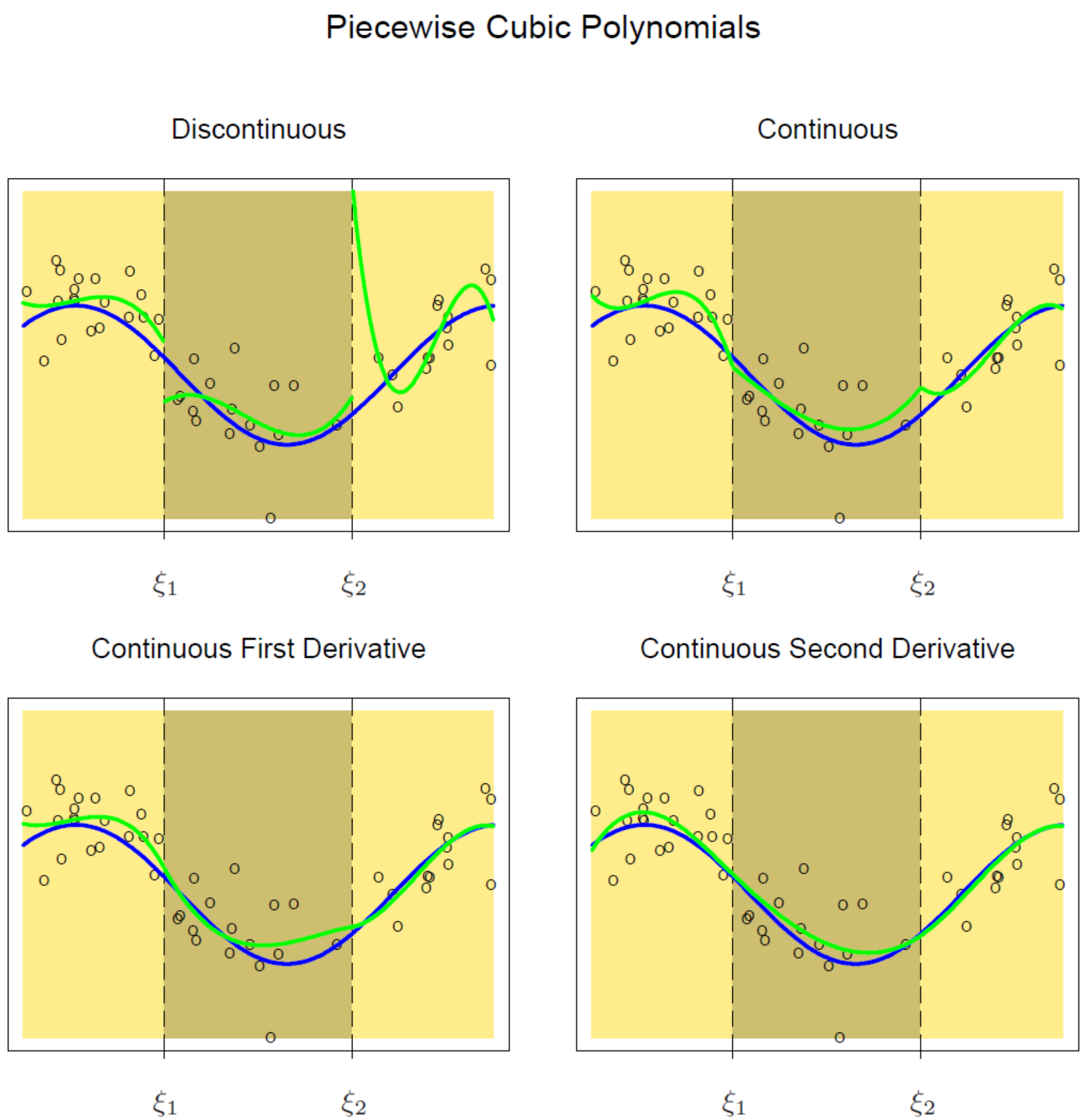
Piecewise polynomial

- Combine step idea with polynomial functions
- Create a cutpoint ξ:

$$\hat{y}_i = \begin{cases} \hat{\beta}_{01} + \hat{\beta}_{11} x_i + \hat{\beta}_{21} x_i^2 + \hat{\beta}_{31} x_i^3 \text{ if } x_i < \xi \\ \\ \hat{\beta}_{02} + \hat{\beta}_{12} x_i + \hat{\beta}_{22} x_i^2 + \hat{\beta}_{32} x_i^3 \text{ if } x_i \geq \xi \end{cases}$$

- This is a *piecewise polynomial* with 1 knot (cutpoint).
- Can still give erratic behavior near boundaries:

Piecewise polynomial



Regression Splines

- We rather have a *smooth* curve, so we add constraints:
  - f should be continuous.
  - f' should be continuous.
  - f'' should be continuous.

- General definition:
  - A degree-d spline is a piecewise polynomial of degree d,
  - with continuity in the derivatives up to degree $d-1$.

- It uses $d+1+K$ degrees of freedom. A cubic spline with K knots uses $K+4$ degrees of freedom (see next slide!).

Cubic spline

- The basis functions are given by

$$
\begin{aligned}
h_1(X) &= X \\
h_2(X) &= X^2 \\
h_3(X) &= X^3 \\
h_{3+k}(X, \xi_k) &= (X - \xi_k)_+^3 = \begin{cases} (X - \xi_k)^3 & \text{if} \quad X > \xi_k \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

  for $k = 1, \ldots, K$

- The intercept uses up an additional degree of freedom

Natural spline

- The *natural spline* additionally constrains the function to be linear at the boundaries.
- Uses lower degrees of freedom: only linear effect at both boundaries.
- Which basis functions are omitted or change? What is the difference in degrees of freedom?

$$
\begin{aligned}
h_1(X) &= X \\
h_2(X) &= X^2 \\
h_3(X) &= X^3 \\
h_{3+k}(X, \xi_k) &= (X - \xi_k)_+^3 = \begin{cases} (X - \xi_k)^3 & \text{if} \quad X > \xi_k \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

Smoothing Spline

- To avoid (mis)specification of number and location of knots, we simply place way too many and use penalization to obtain a smooth model.

- As in lasso and ridge regression, use a *fit + penalty* approach:

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

- The function f that minimizes this PRSS is known as a *smoothing spline*: A natural spline with a knot at every unique observed value x_i.

- The design matrix for the smoothing spline has n columns! With so many basis functions of x_j, it is heavily overparameterized!

- Penalty $\lambda \int f''(t)^2 dt$ shrinks coefficients of many columns towards zero (cf. ridge regression).

From univariate splines to GAMs

- Often we have multiple predictor variables: $X_1, \ldots, X_p$
- We can generalize the ideas:

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

- The f_j are estimated through minimizing 9.7 (p. 297):

$$PRSS = \sum_{i=1}^{N} \{y_i - \beta_0 - \sum_{j=1}^{p} f_j(x_{ij})\}^2 + \sum_{j=1}^{p} \left[\lambda_j \int f_j''(t_j)^2 dt_j\right]$$

- The f_j need not be (smoothing) splines. What value would the right-hand (penalty) take for a linear function $f(x_{ij}) = x_{ij}\beta_j$?

Penalized likelihood estimation

- Current state-of-the-art: R package `mgcv` (Wood, 2004, 2011), which takes a *penalized likelihood* estimation approach to fitting smoothing splines.

- Allows for using (RE)ML estimation: specify `method = "REML"` in the call to function `gam`).

- The *linear* basis function is estimated as a *fixed effect*, i.e., its parametric coefficient β_j is estimated in an unpenalized manner (cf. *PRSS* formula).

- The *non-linear* basis functions are treated as random effects, for which only the variance σ_b^2 of the coefficients is estimated; the coefficients of the non-linear functions are thereby estimated in a *penalized manner* (cf. *PRSS* formula).

- There is a one-to-one correspondence between λ (or λ_j) in the *PRSS* formula and the variance of the random effect σ_b: $\lambda = \frac{\sigma_\epsilon^2}{\sigma_b^2}$

- Major advantage of using REML estimation: Random effects can also be included to account for *dependency* of observations! See `?mgcv::s` and `mgcv::?smooth.terms` to see how random effects can be specified using the `bs` argument of function `s()`.