Week 5: Local and Spline Regression

- Generalized Additive Models, T.J. Hastie and R.J. Tibshirani (1990)
  - Chapter 2
- An Introduction to Statistical Learning: with Applications in R, J. Gareth, D. Witten, T. Hastie and R. Tibshirani (2021)
  - Chapter 7 Sections 7.1 to 7.6
- The elements of statistical learning: data mining, inference and predictions, T. Hasties, R. Tibshirani and J. Friedman (2001)
  - Chapter 5 Sections 5.1 to 5.7 ⟶ more on theory

5.1 Local smoothing

Local smoothing

We are going to discuss a class of regression techniques that

- achieve flexibility in estimating the regression function $f(X)$ over the domain $\boldsymbol{R}^p$ by fitting a different but simple model separately at each point x_0.
  - use only the observations close to the target point x_0 to fit the simple model,
  - the resulting estimated function $f(X)$ is smooth in $\boldsymbol{R}^p$,
  - This localisation is achieved via a weighting function or kernel, which assigns a weight to x_i based on its distance from x_0.

So far, we have discussed **linear models**, for which **the mean of response variable** is a **linear function** of **the predictors**.

$g(\mu_i) = x_i^T \beta$

- In many practical applications, a standard model assumption is that the regression model is **nonlinear**:

$$y_i = f(x_i) + \epsilon_i \longrightarrow$$

$E(\varepsilon_i) = 0$
$Var(\varepsilon_i) = \sigma^2$

ε_i and ε_j are uncorrelated

  where y_i is the response, x_i is a vector of predictors and the ϵ_i are zero mean uncorrelated errors with a common variance σ^2.

- The interest is in the ”underlying trend” in a scatter plot, where scatter plot points are simply treated as a collection of points on a plane, **without much regard to an underlying probabilistic model**.

→ no assumption on the probabilistic behaviour of response variable

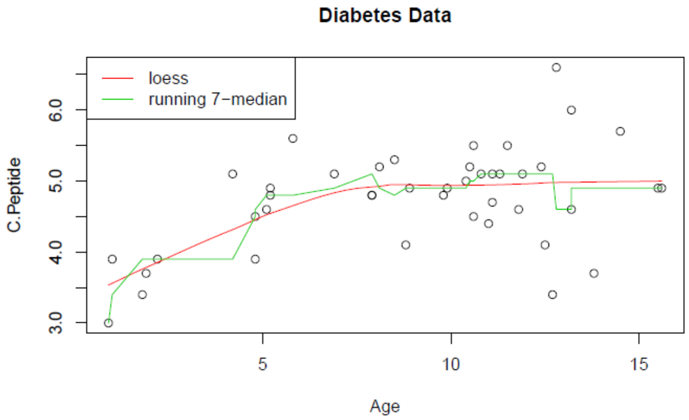
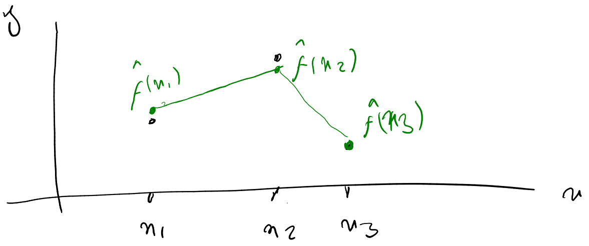


Figure 1: Data from a study (Sockett et al. 1987) of the factors affecting patterns of insulin-dependent diabetes mellitus in children.

- A scatter plot smoother is a function of the data which defines an estimator $\hat{f}(x)$ of $f(x)$ over the range of the predictor values.
  - In general, smoothers are methods or algorithms used to create a smooth curve that fits or approximates the underlying trend in a set of data points.
  - The goal of a smoother is to capture the essential pattern of the data, ignoring the noise, and making the structure of the data more apparent.
- Smoothers are often defined by specifying an estimate of f for the observed x_i and then using interpolation.
  - Once you have an estimate of the function f that describes the trend of the data, interpolation is used to fill in the gaps to predict what f would be at values of x that were not directly observed.

In the following, assume the x_i's are distinct, and $x_1 < ... < x_n$.



Nearest neighbour smoothers (Moving average)

The **symmetric** k nearest neighbourhood of x_i consists of x_i and the k nearest predictor values on either side (take as many values as you can if there aren't k on either side):

3-mean

$$\{x_j : j \in N_k^S(x_i)\} \text{ where } N_k^S(x_i) = \{\max(1, i-k), \cdots, \min(n, i+k)\}.$$

3obser

and **moving average** or **running mean** smoother is defined as

$$\hat{f}(x_i) = ave_{j \in N_k^S(x_i)} y_j.$$

x_i n

$$\frac{1}{7} \sum_{j=i-3}^{i+3} y_j = y_i$$

Note that here we do not provide the general idea of the nearest neghborhood and just focus on the symmetric from.

- This method is a popular method in time series analysis.
- In practice it does not work well.
  - too wiggly that makes it hard to called it smoother.
  - flatten out trends near the endpoints and results in higher bias. (Fig 2)

if we have n observation

3-mean

x_{n-1}

$\rightarrow x_{n-4}, \dots, x_{n-1}, x_n$

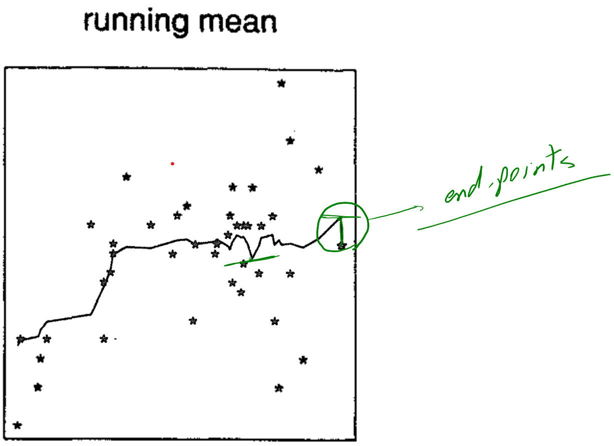


Figure 2: Running mean Smoother- Hastie and Tibshirani (1990)

The running median smoother

3-median

$\hat{y}_i = \text{median}(y_{i-3}, \ldots, y_{i+3})$

The running median smoother at x_i computes the median $\hat{f}(x_i)$ of the responses of the k nearest predictors $x \in N_k^S(x_k)$, and interpolates these values.
Question: What is the advantage of this method compare to running mean?

- the smoother is resistant to outliers.

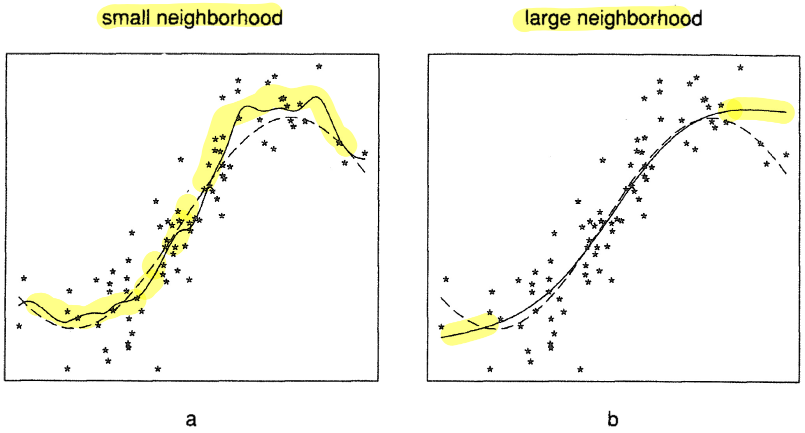
The running smoother is easily computed, but has several disadvantages:

- Boundary bias: there may be substantial bias in $\hat{f}$ near the endpoints x_1 or x_n since we may average over an asymmetric neighbourhood, see Fig. 2.

- The estimated f is not differentiable for running medians/means.

Smoothing Parameter

k is a smoothing parameter. If *k* is too large, we won't capture any sharp changes in f ($\hat{f}$ will have an appreciable bias). If *k* is too small, $\hat{f}$ will be highly variable.



k small
high variance
low bias

k large
low variance
high bias

Figure 3: Bias-variance trade-off for small and large neighbors, Hastie and Tibshirani (1990)

general rule : k is odd.

3-nearest neighborhood

The running line smoother

predictor $\rightarrow \begin{pmatrix} x_{i-3} \\ \vdots \\ x_0 \\ \vdots \\ x_{i+3} \end{pmatrix} \quad \begin{pmatrix} y_{i-3} \\ \vdots \\ y_0 \\ \vdots \\ y_{i+3} \end{pmatrix} \rightarrow$ responce

find $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ based on these vectors

- It achieves more smoothness and better boundary behaviour.
- It fits a linear regression model locally to the data in the symmetric nearest neighbourhood about x_0, and interpolates the fitted values $\hat{f}(x_0)$):

$$\hat{f}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0, \qquad i = 1, \cdots, n.$$

- The slope in the local regression captures the trend near the boundary (asymmetric neighborhood), and thus bias is reduced.
- For the interior points, the fitted value is close to the neighborhood mean.
- The fitted $\hat{f}(x)$ is not differentiable, since it is a linear interpolation of the fitted values $\hat{f}(x_i)$.

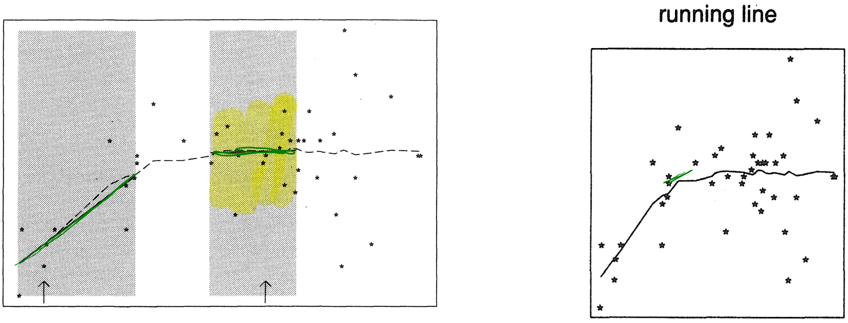


Figure 4: Running line smoother- Hastie and Tibshirani (1990)

Loess smoothers

Suppose that the fitted value $\hat{f}(x_0)$ is to be computed at x_0, where data are given at (x_i, y_i), $i = 1, \ldots, n$. Loess (locally weighted scatterplot smoothing) considers the weighted least squares criterion

$$\text{minimize} \sum_i \underbrace{w_i(x_0)}_{\text{weight function (based on the distance to } x_0 \text{)}}(y_i - \beta_0(x_0) - \beta_1(x_0)x_i)^2$$

at the point x_0. The weights w_i are smaller for x_i which are further away from x_0.

- It fits a line where points with x_i close to x_0 get more weight in the fit.
- The line is discarded, however, and only $\hat{f}(x_0) = \hat{\beta}_0(x_0) + \hat{\beta}_1(x_0)x_0$ is kept.

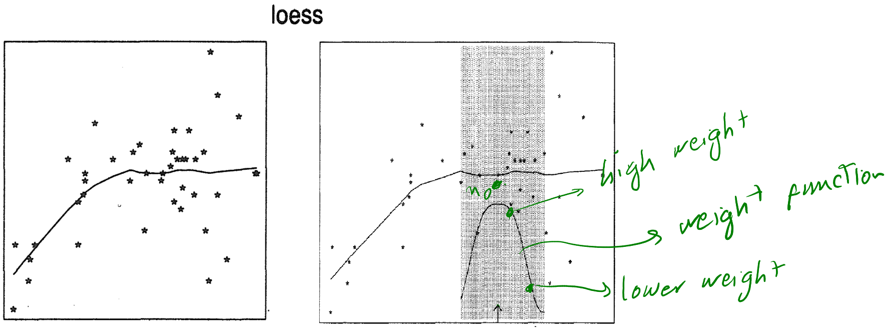


Figure 5: Loess smoother- Hastie and Tibshirani (1990)

Below the Loess algorithm of Cleveland (1979) is outlined:

Step 1. Pick a point x_0 (not necessarily one of the x_i). Find the k nearest x_i values to x_0, set of indices $N_k(x_0)$.

- Calculate $\Delta(x_0) = \max_{i \in N_k(x_0)} |x_0 - x_i|$.

Step 2. Assign weights to each point as $K\left(\frac{|x_0 - x_i|}{\Delta(x_0)}\right)$ where

$$K(u) = \begin{cases} (1-u^3)^3 & \text{for } 0 \le u \le 1 \\ 0 & \text{otherwise.} \end{cases} .$$

tri-cube kernel

Step 3. Calculate the weighted least squares line within the neighbourhood defined by $N_k(x_0)$ and take the fitted value at x_0 as $\hat{f}(x_0)$.

Step 4. Repeat for every desired value of x_0.

Remarks:

- As $K(u) = 0$ for $|u| \geq 1$,

  $$K\left(\frac{|x_0 - x_i|}{\Delta(x_0)}\right) = 0$$

  $> \; . < u < 1$

  if $|x_0 - x_i| \geq \Delta(x_0)$. So only x_i closer than $\Delta(x_0)$ contribute to the fit.

- $K(u)$ has its maximum at $u = 0$ and decreases as $|u|$ increases. So the weight is given to x_i decreases as x_i gets further from x_0.

- There are variations on the basic loess algorithm which attempts to down weight the influence of outliers.

R Code: Different kinds of weighting function

R Code: Scatterplot smoothers

Bias/variance trade-off

There is a natural bias/variance tradeoff in loess smooth as we change the width of the averaging window, which is most explicit for local averages:

- If the **window is narrow**, $\hat{f}(x_0)$ is an average of a small number of y_i close to x_0, and its **variance will be relatively large** – close to that of an individual y_i. The **bias will tend to be small**, again because each of the $\boldsymbol{E}[y_i] = f(x_i)$ should be close to $f(x_0)$.

- If the **window is wide**, the **variance of $\hat{f}(x_0)$ will be small** relative to the variance of any y_i, because of the effects of averaging. The **bias will be higher**, because we are now using observations x_i further from x_0 , and there is no guarantee that $f(x_i)$ will be close to $f(x_0)$.

Higher Order Loess smoothers

Question:Why only consider loess with local linear fits?
We can fit local polynomials of any degree d, by minimizing

$$\sum_{i=1}^{n} K_h(x_0, x_i) \left[y_i - \beta_0(x_0) - \sum_{j=1}^{d} \beta_j(x_0) x_i^j \right]^2 .$$

The solution

$$\overset{\hat{f}(x_0)}{\hat{f}(x0)} = \hat{\beta}_0(x_0) + \sum_{j=1}^{d} \hat{\beta}_j(x_0) x_0^j$$

is less prone to ”trimming the hills and filling the valleys,” (local linear fits tend to be biased in regions of curvature of the true function) and boundary bias will also be reduced. The price to be paid for this bias reduction is, of course, an increased variance.

Kernel smoothers

Kernel smoothing calculates $\hat{f}(x_0)$ by a weighted average of the responses y_i:

$$\hat{f}(x_0) = \sum_i w_i(x_0) y_i$$

The weight $w_i(x_0)$ given to y_i is

$$w_i(x_0) = \frac{K((x_i - x_0)/h)}{\sum_j K((x_j - x_0)/h)}, \qquad \text{Note: } \sum_i w_i(x_0) = 1,$$

where

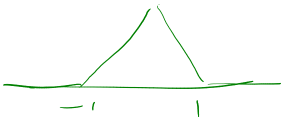
- $K(u)$ is a symmetric function decreasing in $|u|$ called the **kernel function**
- $h > 0$ is a **smoothing parameter** (called the **bandwidth**).

The estimate $\hat{f}(x_0)$ is known as the **Nadaraya-Watson kernel estimator**.

The kernel functions can be selected from the following choices:

- **Epanechikov kernel**: $K(t) = \left\{ \begin{array}{ll} \frac{3}{4}(1-t^2) & \text{for } |t| \leq 1 \\ 0 & \text{otherwise} \end{array} \right\}.$
- **Box kernel**: $K(t) = \left\{ \begin{array}{ll} 1 & |t| \leq 1 \\ 0 & \text{otherwise} \end{array} \right\}.$

- **Triangular kernel**: $K(t) = \left\{ \begin{array}{ll} 1-|t| & |t| \leq 1 \\ 0 & \text{otherwise} \end{array} \right\}.$



- **Tri-cube kernel**: $K(t) = \left\{ \begin{array}{ll} (1-|t|^3)^3 & |t| \leq 1 \\ 0 & \text{otherwise} \end{array} \right\}.$
- **Gaussian kernel**: $K(t)$ is the standard normal density function.

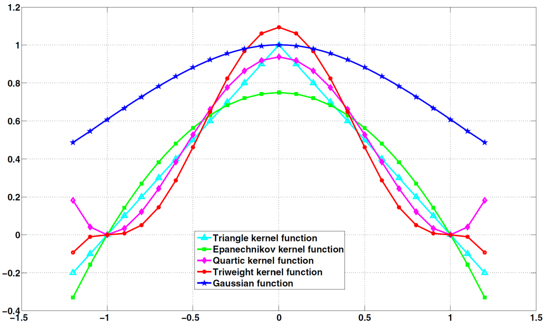
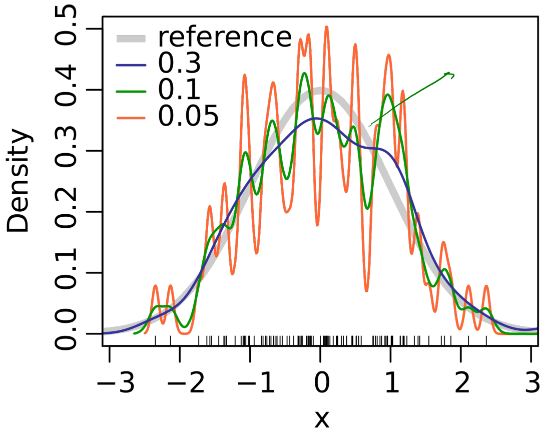


Figure 6: Kernels- Wang and Huang (2010)

- The fitted functions are continuous/smooth whenever $K(u)$ is continuous/smooth.
- As the window is moving through the interval, data points enter the neighbourhood initially with weight zero, and then their contributions slowly increases.
- The choice of kernel is less crucial to the predictive performance than the choice of smoothing parameter h.
- Large values of h imply lower variance but higher bias. In other words, a small bandwidth will do less smoothing than a large one.



large h

variance is low

bias is high

small h

variance high

bias low

Figure 7: The effect of bandwidth h- Wikipedia

R Code: Kernel smoothing

$\hat{f}(x_0) = \sum w_i(x_0) y_i$ kernel smoother

loess $y_i = \beta_0 + \varepsilon_i$

The equivalent kernel

Consider local regression where instead of fitting a line, we do a weighted fit with just a constant term:

$$\text{minimize} \sum_i w_i(x_0)(y_i - \beta_0(x_0))^2 \text{ with respect to } \beta_0(x_0).$$

Differentiating with respect to $\beta_0(x_0)$ and writing $\hat{\beta}_0(x_0)$ for the minimizer, we have

$$-2\sum_i w_i(x_0)(y_i - \hat{\beta}_0(x_0)) = 0 \quad \Rightarrow \quad \hat{\beta}_0(x_0) = \sum_i w_i(x_0)y_i,$$

since $\sum_i w_i(x_0) = 1$.

$$\hookrightarrow \sum_i w_i(x_0) y_i = \underline{\sum_i w_i(x_0)} \hat{\beta}_0(x_0) = \hat{\beta}_0(x_0)$$

$$\Rightarrow \hat{y}_i = \hat{\beta}_0(x_0) = \sum w_i(x_0) y_i'$$

- Kernel smoothing can be thought of as local weighted fitting of a model containing just a constant term.
- The locally weighted averages suffer from boundary bias, due to the asymmetry of the kernel in that region.
- Fitting straight lines rather than constants can remove this bias exactly to first order, which means that it corrects for the linear component of the bias but might not fully correct for higher-order effects .

Local vs. nearest neighbour

- For nearest neighbour methods (such as loess) the smoothing window size (or bandwidth) is dynamic and adjusted so that it always contains the same number of data points, regardless of their density in the input space.

  - Because the number of points within each local smoothing window is constant, Loess tends to produce smooths with constant variance across the input space. The method adapts to the data density: in denser regions, the window narrows, and in sparser regions, it widens, but always includes the same number of points. This adaptiveness helps maintain uniform smoothness and variance.

  - Where data are sparse, nearest neighbour methods are biased, since far away points with a different mean enter the regression.

- For kernel smoothers, on the other hand, the smoothing window size (bandwidth) is constant.

  - The kernel function assigns weights to points within this window, typically decreasing with distance from the target point. x_0

  - The fixed bandwidth means that in areas of the input space where data points are densely packed, many points contribute to the smoothing, whereas in sparse areas, fewer points are available.

  - Kernel smoothers, with their fixed bandwidth, face a trade-off between bias and variance, especially in sparse regions. A wide bandwidth might reduce variance by including more points but can increase bias by smoothing over too broad an area, potentially obscuring local trends.

  - Keeping the bandwidth fixed, the variance of kernel smoothers increases with data sparsity.

Local likelihood

Suppose that at x_0, the response follows a distribution from an exponential family with mean $\mu(x_0) = \beta_0 + \beta x_0$ and log-likelihood $\ell(y_0; \mu(x_0))$.
The local likelihood at x_0 can then be defined as

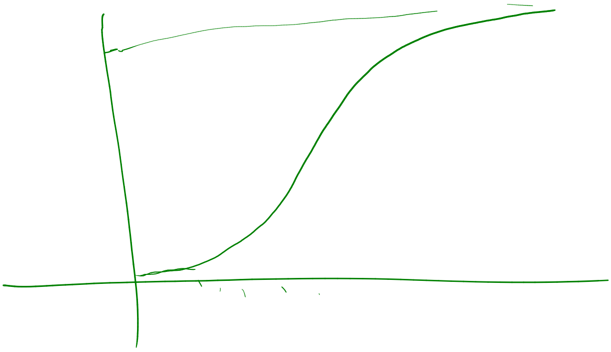
$$\text{maximize} \qquad \ell(\beta_0, \beta; x_0) = \sum_{i=1}^{n} K_\lambda(x_0, x_i)\ell(y_i, \mu(x_0)).$$

Its optimisation then yields the estimate of the mean at x_0: $\hat{\mu}(x_0) = \hat{\beta}_0 + \hat{\beta} x_0$, which may be plotted in all x_0 of interest.

R Code: Local likelihood

Activity in R: Loess Smoother

Activity in R: Kernel Smoothe



5.2 Polynomial regression and step functions

Polynomial regression and step functions

- In regression problems the underlying function $f(X)$ is typically nonlinear and non-additive in X.
- However, representing $f(X)$ by a linear model is a convenient and sometimes necessary approximation:
  - **Convenient**: linear model is easy to interpret, it is a first-order Taylor approximation to $f(X)$;
  - **Necessary**: with N small and/or p large, linear model might be all we can fit to the data without overfitting.
- Polynomial regression extends the linear model by adding extra predictors, obtained by raising each of the original predictors to a power.
  - The core idea in polynomial regression is to augment the vector of input X with additional variables, which are transformations of X
  - use linear models in this new space of derived input features.

Cont. Polynomial regression and step functions

- For simple linear regression, $y_i = \beta_0 + \beta_1 x_1 + \varepsilon_i$, the design matrix is

$$X = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}_{n\times 2}.$$

$$\underset{\sim}{y} = X \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \underset{\sim}{\varepsilon}$$

  - we say that 1, x is a basis.

- For a quadratic model, $y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \varepsilon_i$ the design matrix is

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix}$$

$$\underset{\sim}{y} = X \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} + \underset{\sim}{\varepsilon}$$

  - The basis 1, x, x^2.

R Code: Polynomial regression

R Code: polynomial logistic regression

Linear basis expansion

Denote by $h_m(X) : \boldsymbol{R}^p \to \boldsymbol{R}$ the mth transformation of X, $m = 1, \ldots, M$. We then model

$$f(X) = \sum_{m=1}^{M} \underbrace{\beta_m}_{\text{parameters of the model}} \underbrace{h_m(X)}_{\text{basis function}},$$

which is a linear basis expansion in X.

Note: once basis functions h_m are established, the models are linear in these new variables and the fitting proceeds as before.

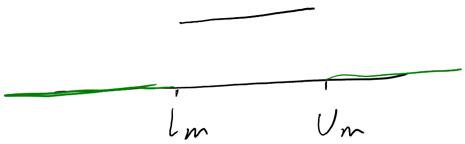
$f(X) = \sum_{j=0}^{p} \beta_j X_j$, $X_0 = 1$

Examples of basis functions:

- $h_m(X) = \overset{\text{basis}}{X_m}$, $m = 1, \ldots, p$ - recovers the original linear model;
- $h_m(X) = X_j^2$ or $X_j X_k$ - polynomial terms. Note that the number of variables grows exponentially in the degree of the polynomial. A full quadratic model with p variables requires $O(p^2)$ square and cross-product terms, or more generally $O(p^d)$ for a degree-d polynomial. $\frac{p(p+1)}{2}$ the number of basis
- $h_m(X) = \log(X_j)$, $\sqrt{X_j}, \ldots$ - other nonlinear transformations of single input;
- $h_m(X) = I(L_m < X_k \le U_m)$, and indicator for a region of X_k - By breaking the range of X_k up into M_k (bins) such nonoverlapping regions results in a model with a piecewise constant contribution for X_k.

R Code: step functions

Activity in R: Polynomial regression



5.3 Regression splines

Regression Splines

- Regression splines are more flexible than polynomials and step functions, and in fact, are an extension of the two.
- We divide the range of X into K distinct regions and in each region fit a polynomial function to the data.
- The polynomials are constructed in such a way that they join smoothly at the region boundaries or knots in contrast to step functions.

Lets assume that X is one-dimensional. A piecewise polynomial $f(X)$ is obtained by

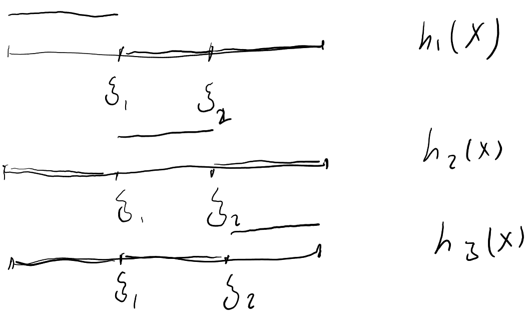
- dividing the domain of X into continuous intervals
- representing f by a separate polynomial in each interval.
  - The piecewise **constant** polynomial has the following basis functions for the intervals defined by ξ_1 and ξ_2 (called knots):

    $$h_1(X) = I(X \leq \xi_1), \;\; h_2(X) = I(\xi_1 < X \leq \xi_2), \;\; h_3(X) = I(\xi_2 < X).$$

    These results in a fit that has discontinuities.
  - For piecewise **linear** polynomial fit without discontinuities, we define the basis functions as follows:

    $$h_1(X) = 1, \;\; h_2(X) = X, \;\; h_3(X) = (X - \xi_1)_+, \;\; h_4(X) = (X - \xi_2)_+,$$

    where t_+ denotes the positive part.



$$f(X) = \sum_{m=1}^{3} \beta_m h_m(X)$$

$$\Rightarrow \hat{f}(X) = \hat{\beta}_m = \bar{y}_m$$

mean of the response variables if X_i belongs to mth interval

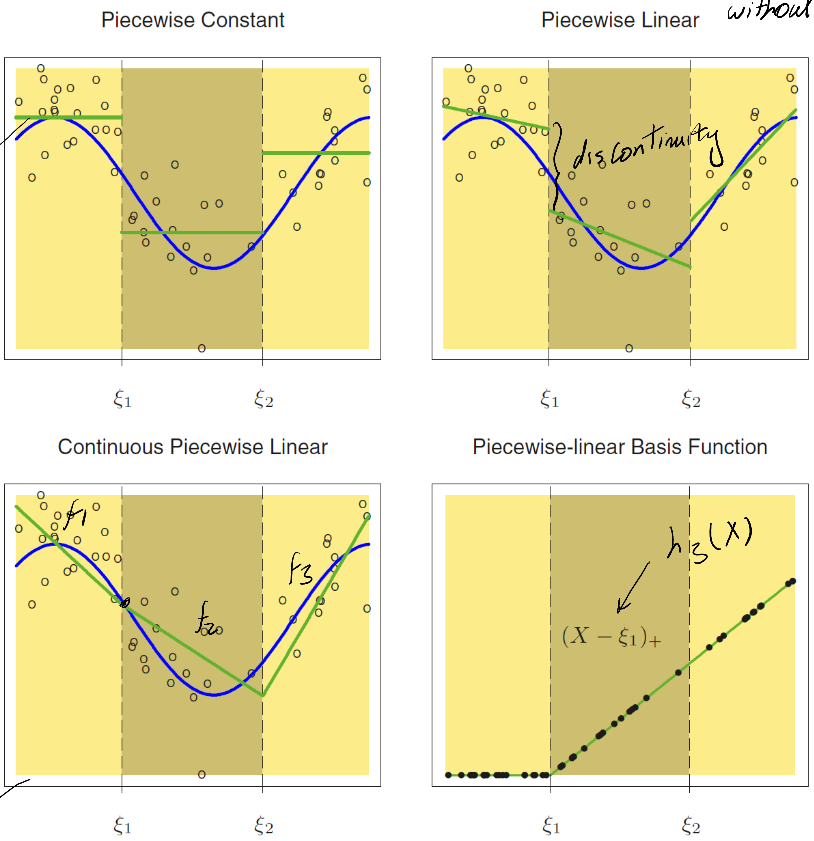
$$h_3(X) = (X - \xi_1)_+ = \begin{cases} X - \xi_1 & X \geq \xi_1 \\ 0 & X < \xi_1 \end{cases}$$

we fit the model without any restriction

$h_1(X) = 1$

$h_2(X) = X$

mean of the response variable



Piecewise Constant

Piecewise Linear

Continuous Piecewise Linear

Piecewise-linear Basis Function

Figure 8: Piecewise constant and piecewise linear- Hasties et al. (2001).

constrain → $f_1(\xi_1^-) = f_2(\xi_1^+)$

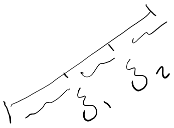
We often prefer smoother functions, which is achieved by increasing the order of the local polynomial.

- A cubic spline with knots at ξ_1 and ξ_2 has the following basis:

$$h_1(X) = 1,\ h_2(X) = X,\ h_3(X) = X^2,\ \ h_4(X) = X^3,$$
$$h_5(X) = (X - \xi_1)^3_+,\ h_6(X) = (X - \xi_2)^3_+.$$

An order M spline with K knots ξ_j, $j = 1, \ldots, K$ is a piecewise-polynomial of order M.

$K = 2$



- It is continuous and has continuous derivatives up to order $M - 1$.
- It has $K + M - 1$ degrees of freedom since

  $\beta_0, \beta_1, \ldots, \beta_M$: $M+1$ parameter

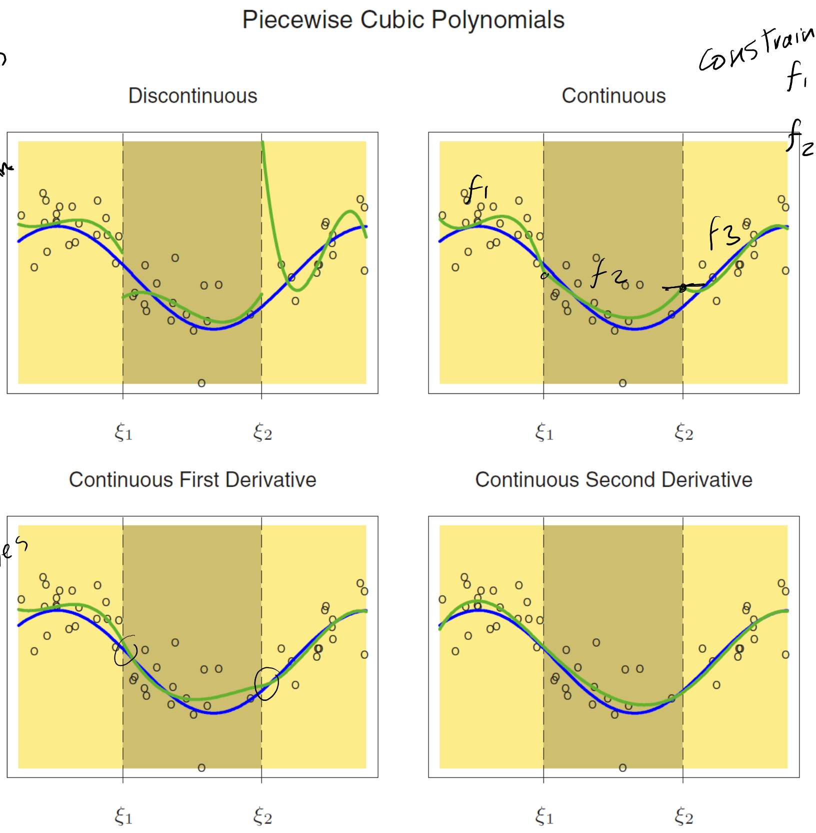
  - there are $(K + 1)$ polynomials with $M + 1$ parameters
  - at each of the K knots, we impose continuity (1 degree of freedom) and continuity of the derivatives up to order $M - 1$.

  Therefore the number of degrees of freedom is $(K + 1) \times (M + 1) - K \times \overbrace{M}^{M-1+1} = K + M + 1$.

  total number of parameter

  - For a cubic spline we have $M = 3$ and $K + 4$ degrees of freedom.
- The fixed-knot splines are known as regression splines.
- To fit a spline, we need to select the order of the spline, the number of knots and their placement.

$$M = 3 \xrightarrow{df} K + M + 1 = K + 4$$



polynomials of order 3, no constrain

constrain $f_1(\xi_1^-) = f_2(\xi_1^+)$

$f_2(\xi_2^-) = f_3(\xi_2^+)$

constrain first derivatives continous

constrain second derivatives continuous

Figure 9: Piecewise Cubic Polynomials- Hasties et al. (2001).

- A quadratic spline basis is given by

$$1, x, x^2, (x - \kappa_1)^2_+, \ldots, (x - \kappa_k)^2_+,$$

  and any linear combination of this basis defines a differentiable function with continuous first derivative.

- More generally, a p-th order basis is

$$1, x, \ldots, x^p, (x - \kappa_1)^p_+, \ldots, (x - \kappa_k)^p_+,$$

  from which functions with continuous $\overset{p-1}{\cancel{p}}$-th order derivatives can be constructed.

  - This is also known as the **truncated power basis of degree** p.

- Model selection criterion such as CV and Mallow's C_p can be used to select the number of knots. However, this is not feasible when the number of knots K is large.

Example: `bs` in R

- `bs(x, df = 7)` in R generates a basis matrix of cubic spline functions evaluated at the N observations in `x`, with ($df - degree = knote$) $7 - 3 = 4$ interior knots at the respective percentiles of `x`. default value of degree in bs = 3

- `bs(x, degree = 1, knots = c(0.2, 0.4, 0.6))` generates a basis for linear splines, with the three interior knots. $df = 3+1 = 4$

- **Warning**: by default the bs() function does not take the intercept into account. Therefore $df = length(knots) + degree$. The intercept can be included by specifying $bs(..., intercept = TRUE)$.

R Code: A linear spline with two knots (red) and 50 knots (green).

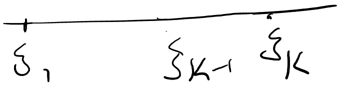
Natural cubic splines

- The behaviour of polynomials fit to data tends to be erratic near the boundaries and extrapolation can be dangerous.
- These problems are worsen with splines.
- The polynomials fit beyond the boundary knots behave even more wildly than the corresponding global polynomials in that region.
- Using splines instead of polynomials worsens this issue.
- A **natural cubic spline** adds additional constraints.
  - It makes the function linear beyond the boundary knots.
  - A natural cubic spline with K knots is represented by K basis functions:

$$N_1(X) = 1, \quad N_2(X) = X, \quad N_{k+2} = d_k(X) - d_{K-1}(X),$$

where

$$d_k(X) = \frac{(X - \xi_k)^3_+ - (X - \boxed{\xi_K})^3_+}{\xi_K - \xi_k}$$



last knot

  - Each of these basis functions has zero second and third derivatives for $X \geq \xi_K$.

R Code: natural spline (red) and ordinary spline (blue)

Activity in R: Spline Regression

exercise

$K = 2 \longrightarrow \xi_1, \xi_2$

natural cubic splines for 2 knots

5.4 Smoothing splines

Smoothing splines are similar to regression splines, but arise in a slightly different setting.

- If there are too many knots, the fitted curve would be too rough.
- One way to overcome this problem is via shrinkage.
- This avoids the knot selection problem entirely by using a maximal set of knots.
- The complexity of the fit is controlled by regularisation.

In this problem we find a function f (with two continuous derivatives) that minimizes the penalized residual sum of squares

$$RSS(f,\lambda) = \underbrace{\sum_{i=1}^{N}(y_i - f(x_i))^2}_{\text{controls the closeness of the fit to the data}} + \lambda \overbrace{\int (f''(t))^2 dt}^{\text{penalty part} \rightarrow \text{control the curvature of the function}},$$

where λ is a fixed smoothing parameter.

- The first term measures closeness to the data
- the second term penalizes curvature in the function.
- λ establishes a tradeoff between the two.

$\lambda = 0$ ——→ f is any function that interpolates the points

$\lambda \rightarrow \infty$ ——→ simple least square line fit

This problem has an explicit, finite-dimensional, unique minimizer which is a natural cubic spline with knots at the unique values of the x_i, $i = 1, \ldots, N$ and the solution can be expressed as

$$f(x) = \sum_{j=1}^{N} N_j(x)\theta_j,$$

natural cubic spline basis

where $N_j(x)$ are an N-dimensional set of basis functions for representing this family of natural splines.

The above criterion can be rewritten as

$$RSS(\theta, \lambda) = \|\boldsymbol{y} - \boldsymbol{N}\theta\|^2 + \lambda\theta^\top \boldsymbol{\Omega}_N \theta,$$

inner product

where $\boldsymbol{N}_{ij} = N_j(x_i)$ and $\boldsymbol{\Omega}_{N\,jk} = \int N_j''(t) N_k''(t)dt$. The solution is here

$$\hat{\theta} = (\boldsymbol{N}^\top \boldsymbol{N} + \lambda \boldsymbol{\Omega}_N)^{-1} \boldsymbol{N}^\top \boldsymbol{y},$$

$$\Rightarrow \hat{f} = \underbrace{N(N^\top N + \lambda \Omega)^{-1} N^\top}_{\text{hat matrix}} y$$

a generalized ridge regression. The fitted smoothing spline is given by

$$\hat{f}(x) = \sum_{j=1}^{N} N_j(x)\hat{\theta}_j.$$

Degrees of freedom

Denote by $\hat{\boldsymbol{f}}$ the N-vector of fitted values $\hat{f}(x_i)$ at the training predictors x_i. Then

$$\hat{\boldsymbol{f}} = \boldsymbol{N}(\boldsymbol{N}^\top \boldsymbol{N} + \lambda \boldsymbol{\Omega}_N)^{-1} \boldsymbol{N}^\top \boldsymbol{y} = \boldsymbol{S}_\lambda \boldsymbol{y}.$$

hat matrix

- Note that the fit is linear in $\boldsymbol{y}$, and the finite linear operator $\boldsymbol{S}_\lambda$ depends only on x_i and λ.

- The effective degrees of freedom of a smoothing spline are

$$df_\lambda = \text{trace}(\boldsymbol{S}_\lambda).$$

- Since df_λ is monotone in λ for smoothing splines, we can invert the relationship and specify λ by fixing df_λ.

- Fixing the degrees of freedom is an intuitive way of finding λ, which is an alternative for cross-validation.

Example: R Code
For example in `R` one can use `smooth.spline(x, y, df = 6)` to specify the amount of smoothing. This encourages a more traditional mode of model selection, where we might try a couple of different values of `df`, and select one based on approximate F-tests, residual plots and other more subjective criteria. This is, in particular, useful when comparing many different smoothing methods. (Used in generalised additive models).

Cross-validation curve

The N-fold cross-validation curve is:

drop the ith observation LOO-CV

original fit without dropping any observation

$$CV(\hat{f}_{\lambda}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}^{-i}\lambda(x_i))^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{f}_\lambda(x_i)}{1 - S_\lambda(i,i)}\right)^2,$$

which can be computed for each value of λ, from the original fitted values and the diagonal elements $S_\lambda(i,i)$ of $\boldsymbol{S}_\lambda$. Recall that

- the notation $\hat{f}_\lambda^{-i}(x_i)$ indicates the fitted value for the smoothing spline evaluated at x_i, where the fit uses all of the training observations except for the ith observation (x_i, y_i).
- This is an example of leave-one-out cross-validation.

Genearalized Cross-Validation

Generalised Cross-validation (GCV) replaces S_{ii} above by $tr(\boldsymbol{S}_\lambda)/N$, which achieves greater numerical stability. We then have

$$\text{GCV} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{y_i - \hat{f}_\lambda^{-i}(x_i)}{1 - \boldsymbol{S}_\lambda(i,i)}\right)^2,$$

which we can rewrite as

$$\text{GCV} = \frac{1}{N}\frac{\text{RSS}}{(1 - \text{tr}(\boldsymbol{S}_\lambda)/N)^2}.$$

R Code:Generalized Cross-Validation

5.5 Multidimensional splines

Multidimensional splines

Each of the approaches discussed in this weeks course material has multidimensional analogues.

- Assume that $X \in \boldsymbol{R}^2$ and we have a basis functions $h_{1k}(X_1)$, $k = 1, \ldots, M_1$ for representing functions of coordinate X_1, and likewise a set of M_2 functions $h_{2k}(X_2)$ for X_2.
- The $M_1 \times M_2$ dimensional tensor product basis is defined by

$$g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2), \quad j = 1, \ldots, M_1, k = 1, \ldots, M_2$$

and it can be rewritten as a two-dimensional function:

$$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \underbrace{\theta_{jk}}_{\text{coefficients}} \underbrace{g_{jk}(X)}_{} \longrightarrow \text{new basis function}.$$

$x \in \mathbb{R} \longrightarrow$ basis function 1

$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \longrightarrow$ basis function

$$x_1 \leftarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \rightarrow x_2$$

$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \in \mathbb{R}^3 \longrightarrow$ basis function

$$x_1 \leftarrow \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \underset{\downarrow \\ x_2}{\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \rightarrow x_3$$

Tensor product

Cont. Multidimensional splines

One dimensional smoothing splines generalize to higher dimensions, too.

- For x_i, y_i with $x_i \in \boldsymbol{R}^d$ we seek a d-dimensional regression function $f(x)$. The idea is to set up the problem

$$\min_f \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 + \lambda J[f],$$

  where J is an appropriate penalty functional for stabilizing a function f in $\boldsymbol{R}^d$.

- For example, in $\boldsymbol{R}^2$ we have

$$J[f] = \int\int_{R^2} \left[\left(\frac{\partial^2 f(x)}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(x)}{\partial x_2^2} \right)^2 \right] dx_1 dx_2.$$

  Optimizing the above criterion leads to a smooth two-dimensional surface, known as **thin-plate spline**.

- The solution here has the form

$$f(x) = \beta_0 + \beta^\top x + \sum_{j=1}^{N} \alpha_j h_j(x),$$

  where $h_j(x) = ||x - x_j||^2 \log ||x - x_j||$.

- h_j is an example of radial basis functions.

- The coefficients α_j are found by plugging this solution back into the optimization criterion.

R Code: Fitting a thin plate spline in R

5.6 Activities