Machine Learning II
Week 2 Lecture – 16th October 2019
Non-linear regression methods I

Contents

- Polynomial regression and step functions
- Basis Functions
- Spline fitting
- Spline smoothing

Introduction

The regression models in ML1 including ridge regression and the lasso are linear supervised learning models. Often the assumptions used in a linear model are inappropriate and we should fit a more flexible function using non-linear modelling methods.

We start off with an example already covered in ML1, polynomial regression, and use this as a starting point to develop spline regression, smoothing splines and local regression.

James et. al covers polynomial regression, step functions and basis function in detail. We will look only briefly at these as a means to developing the methods used in practice.

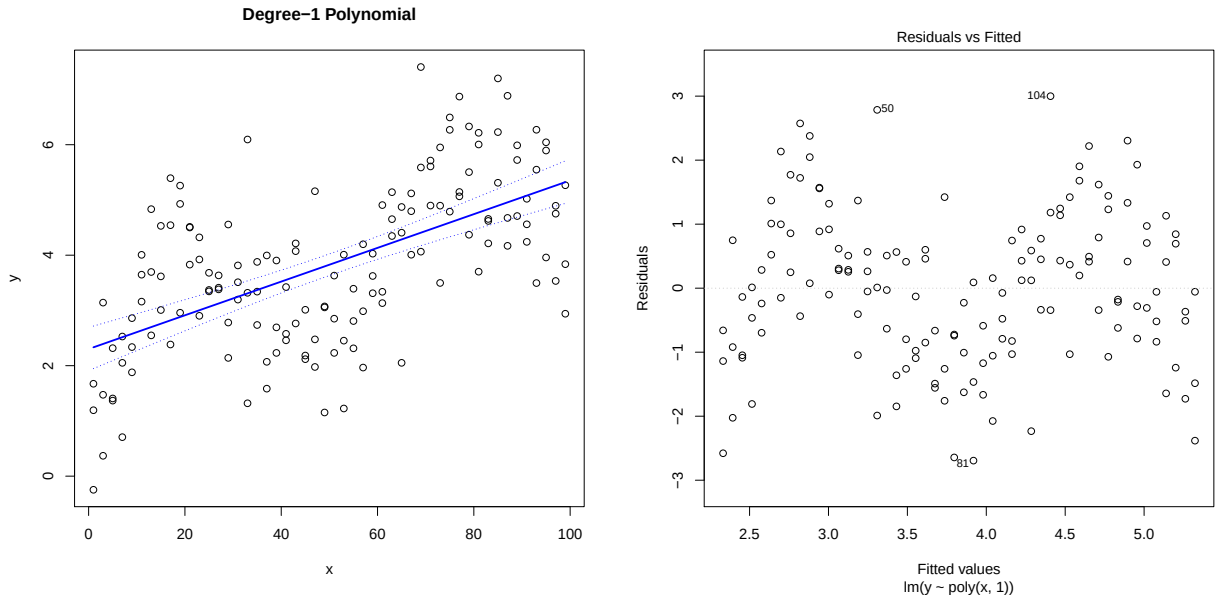
The methods this week are all using just one predictor variable x.

Generalised additive models (GAMs) adapt these ideas to multiple predictor variables (covered next week).

The aim is to find a function $f(x)$ which fits the data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ well.

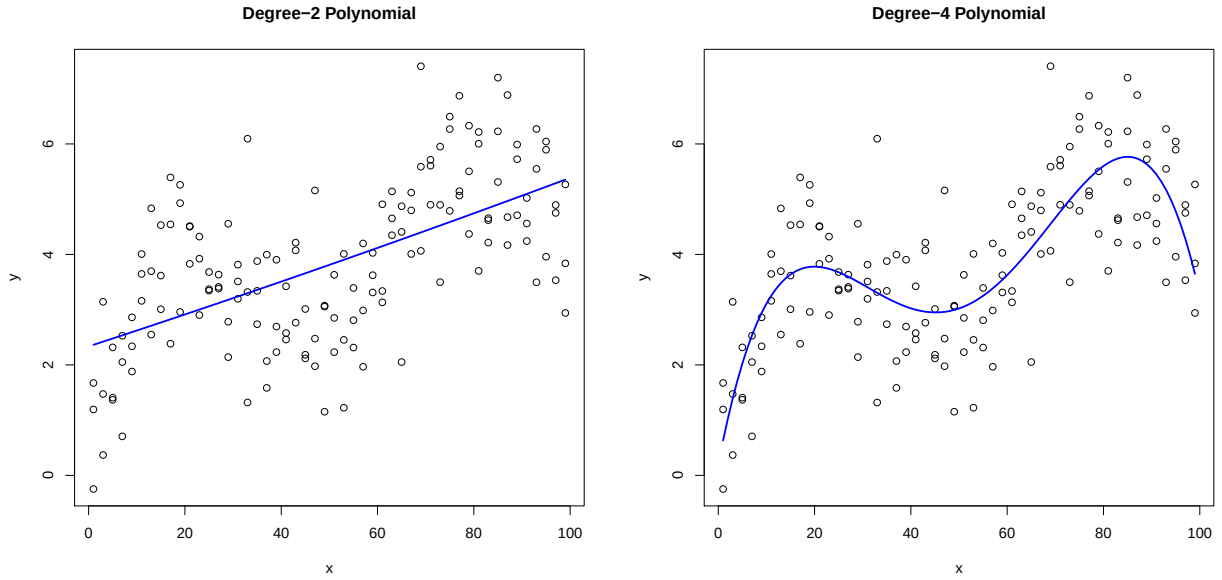
In linear regression the predictor function is $f(x) = \beta_0 + \beta_1 x$ and we choose the coefficients β_0 and β_1 to fit the data the best that a straight line can fit the data.

We will consider the following example data set for the lecture.



A linear regression clearly doesn't work for these data, and there is noticeable structure in the residual plot (right).

The “obvious” idea is to increase the degree of polynomial until you get a decent fit.



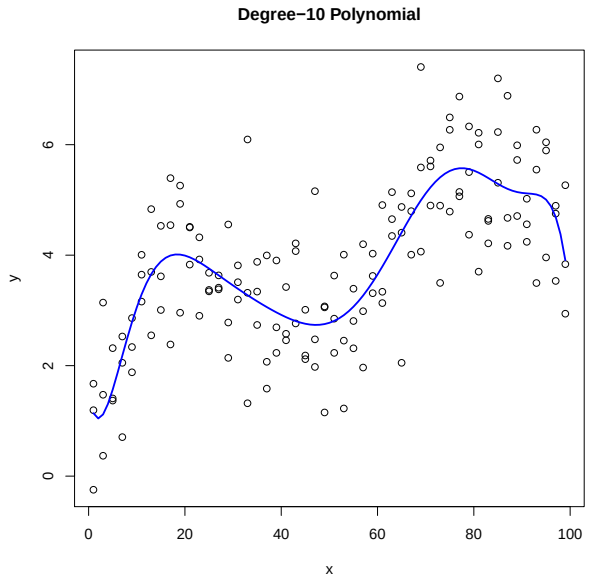
A quadratic regression (left) is almost exactly the same as the linear regression. Quartic regression (right) fits these data fairly well.

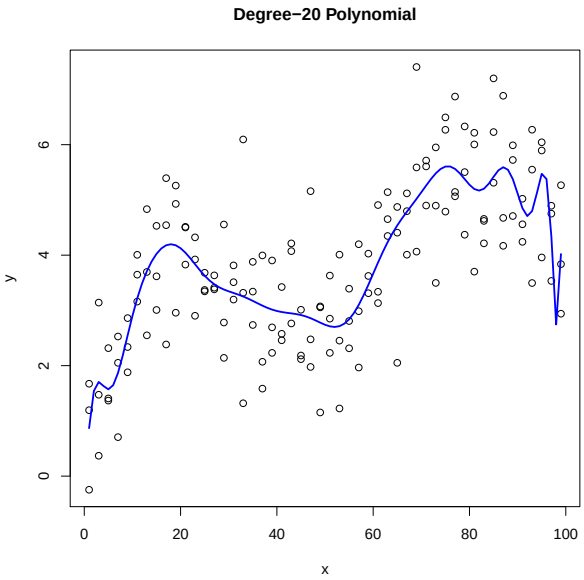
$$y_i = f(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + ... + \beta_d x_i^d + \epsilon_i$$

This approach is known as polynomial regression.

Polynomial regression is a linear model and can be fitted with `lm`.

In practice fitting a high order polynomial $d > 3$ is rarely a good idea. In general a polynomial of order d has $d - 1$ turning points so the the polynomial curve can become overly flexible and can take on some very strange shapes. This is especially true near the boundary of the X variable.





Step functions

Using a polynomial function to fit the features in the data imposes a global structure on the non-linear function of X.

We will now consider fitting several local functions. Functions that are zero outside a certain window. We start by using step functions, which are constant within the window. We break the range of X into bins, and fit a different constant in each bin. This amounts to converting a continuous variable into an ordered categorical variable.

We create cut points $c_1, c_2, \ldots, c_K$ in the range of X, and fit $K + 1$ step functions.

Define the window functions as

$$C_0(x) = \mathrm{I}(x < c_1),$$
$$C_1(X) = \mathrm{I}(c_1 \leqslant x < c_2), \quad \text{etc.}$$

$\mathrm{I}(\cdot)$ is an indicator function, that returns 1 if the condition is true, and returns 0 otherwise.

The fitted predictor function is

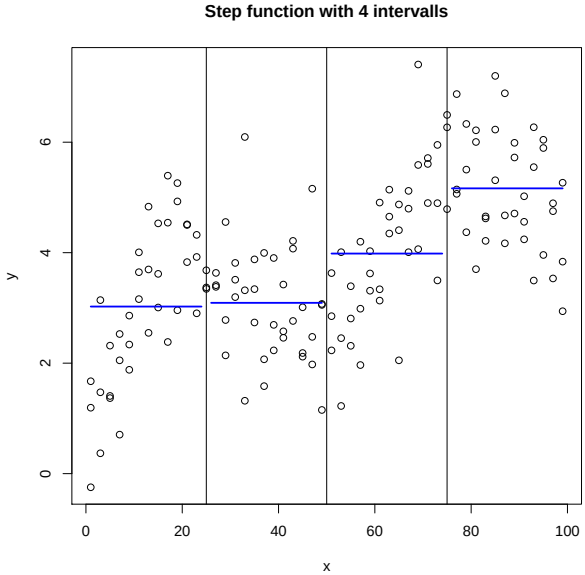
$$f(x) = \beta_0 C_0(x) + \beta_1 C_1(x) + \beta_2 C_2(x) + \cdots + \beta_K C_K(x)$$

The practical approach is fitting the function is to use least squares to fit a linear model.

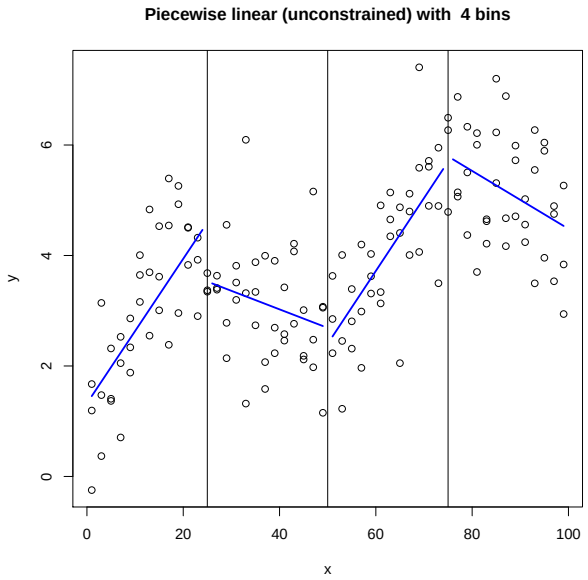
$$y_i = f(x_i) = \beta_0 C_0(x_i) + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \cdots + \beta_K C_K(x_i) + \epsilon_i$$

ϵ_i are the residuals and $\beta_0, \ldots, \beta_K$ are the parameters to be estimated.
For a given x_i, exactly one $C_0, C_1, \ldots, C_K$ is non-zero.

This has the effect of fitting a piecewise constant function. Here K=3



Alternatively we could fit a piecewise linear regression in each interval:



The predictor function is now

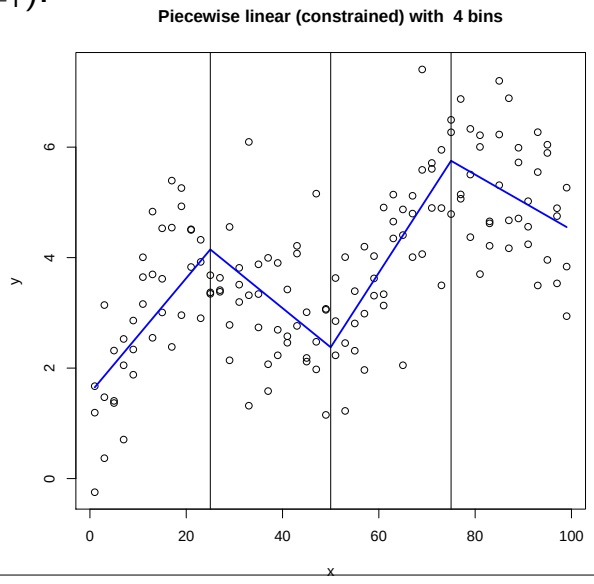
$$f(x) = f_0(x)C_0(x) + f_0(x)C_0(x) + \cdots f_K(x)C_K(x),$$

where each $f_k(x) = \beta_{0k} + \beta_{1k}x$ is a linear function.

The previous diagram suffers from the problem that there is a jump (discontinuity) at the interval, which is usually inappropriate.
We can constrain the piecewise linear (pwl) function so that the function is continuous.

$$y_i = f(x_i) = b_0(x_i)C_0(x_i) + b_1(x_i)C_1(x_i) + \cdots b_K(x_i)C_K(x_i) + \epsilon_i$$

where each $b_k(x_i) = \beta_{0k} + \beta_{1k}x$ **and** the boundary constraints are $b_k(c_{k+1}) = b_{k+1}(c_{k+1})$.



Basis Functions

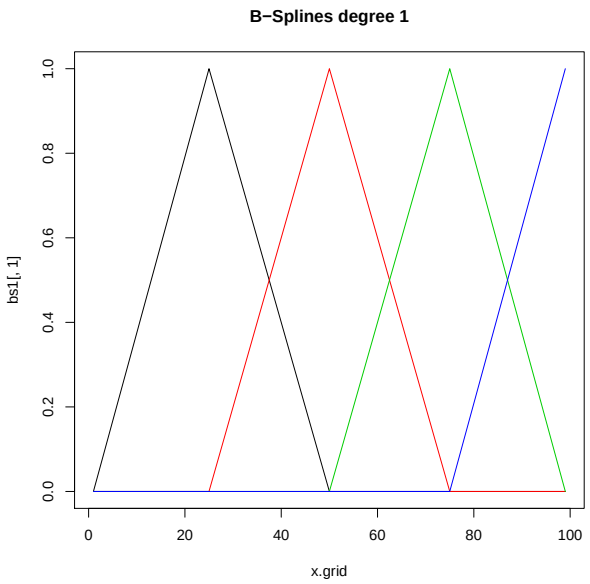
For all of methods so far today, f can be expressed as a linear combination of basis functions.

$$y_i = f(x_i) = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_k b_K(x_i) + \epsilon_i.$$

b_k are fixed and known basis functions.

In polynomial regression $b_k(x) = x^k$
and for piecewise constant functions $b_k(x) = C_k(x_i)$

In the constrained piecewise linear function we use basis functions that are in the shape of a hat over two intervals.



The fitted coefficients using the basis functions on the left are:
$\beta_0 = 1.65, \beta_1 = 2.50, \beta_2 = 0.73, \beta_3 = 4.10, \beta_4 = 2.90.$

The predictor function is the sum of the components.
$f(33) = 1.65 + 2.50 \cdot 0.68 + 0.73 \cdot 0.32 + 4.10 \cdot 0 + 2.90 \cdot 0 = 3.5836$
β_0 is required because the 1st function is equal to 0 on the far left.

Piecewise continuous cubic polynomials

The constrained piecewise linear function is continuous but has sharp points at the joins, called **knots**. In mathematical terms the sharp points are caused because the first derivative of f is discontinuous at the knots.

The continuity constraint was that the function value agreed on both sides of each knot[1] $f(c_k-) = f(c_k+)$.

A smoothness constraint requires that the derivative value agrees on both sides of each knot $f'(c_k+) = f'(c_k-)$.

This is not possible when each b_k is linear, so we specify that each piece is a quadratic function. Although piecewise continuous quadratic polynomials are much smoother, you still get a visible join in the resulting function at the knots.

[1] $c+$ is lim $x \to c_k$ from above, $c-$ is lim $x \to c_k$ from below

In practice piecewise cubic functions instead of piecewise quadratic functions are fitted. These are called **cubic splines**.

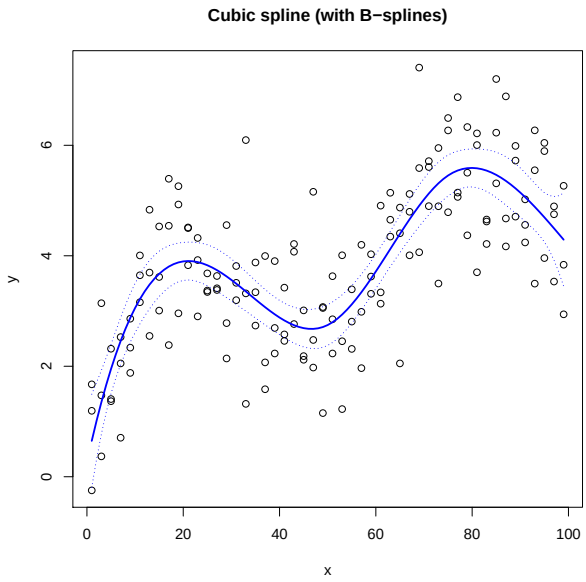
A cubic spline is a piecewise cubic function:

$$f(x) = b_k(x) \quad \text{for } c_{k-1} \leqslant x \leqslant c_k$$
$$\text{with} \quad b_k(x) = \beta_{0k} + \beta_{1k}x + \beta_{2k}x^2 + \beta_{3k}x^3$$

The function is continuous, with continuous first *and* second derivatives.

$$f(c_k-) = f(c_k+)$$
$$f'(c_k-) = f'(c_k+)$$
$$f''(c_k-) = f''(c_k+)$$

We only need to specify the constraints at the knots, because between the knots these conditions are a result of f being a cubic polynomial.



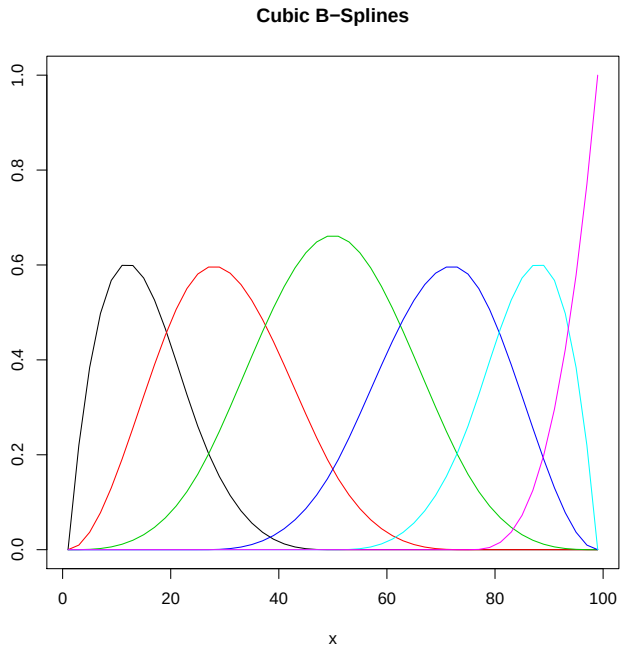
The cubic spline for the example data and knots at 25, 50 and 100.

With piecewise linear (pwl) continuous regression we expressed the function f as a linear combination of basis functions, which were hat functions. With $K = 3$ knots we had a constant parameter β_0 and four basis functions.

In general for polynomials of degree d and K knots, then we require $K + d$ basis functions plus a constant parameter, so for cubic splines the is $K + 3$ basis functions, $K + 4$ parameters.

The shape of the basis functions for pwl are unique.
For cubic splines the basis functions require another condition so that they are uniquely defined.
The two usual methods are *B-splines* and *natural splines*.

B-Splines are chosen to be efficient to compute via an iterative algorithm, details are in Hastie, Tibshirani and Friedman. In R use the function `bs(x,knots=c(),degree=3)` or `bs(x,df=,degree=3)`.
If `df` (degrees of freedom) is specified then the function will choose df-3 knots based on equally spaced quantiles of `x`.



The other type of spline basis commonly used are the natural splines.

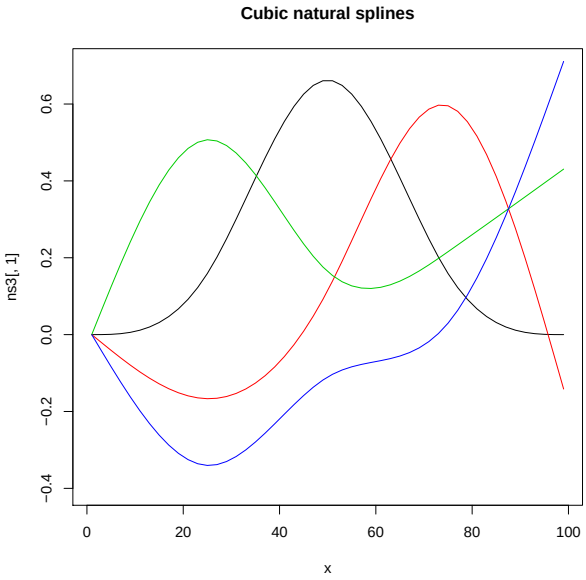
Natural cubic splines have the additional constraints that the second and third derivatives are zero at the extremes of the x-data.

If x_1 is the smallest and x_n the largest x-value then

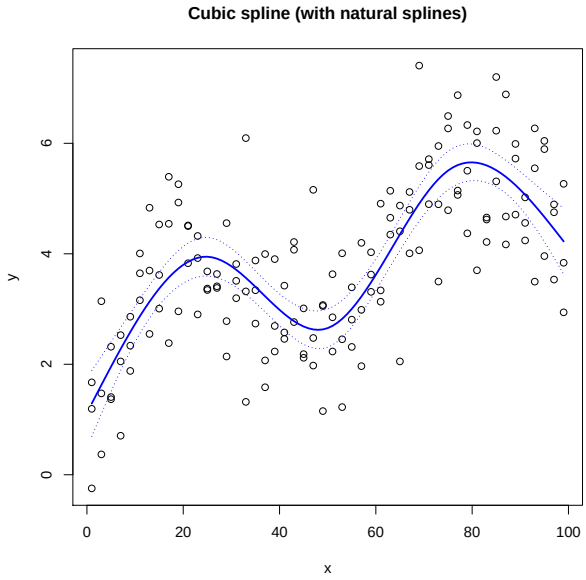
$$f''(x_1) = f'''(x_1) = f''(x_n) = f'''(x_n) = 0.$$

The basis functions are less visually appealing, but there is a data analytical advantage to using natural splines.

B-splines can give a predictor function with a large variance of the at the edges, this problem is reduced with natural splines



The additional constraints at the edges means that we require only $K + 1$ basis functions.



Smoothing Splines

We return to the penalised least squares concept you learnt in ridge regression last semester:

minimise: squared error of fitted model + $\boldsymbol{\lambda}$ penalty term

We can choose a function f which fits the data well. I.e. minimise squared residuals

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

But we do not want f to be too "wiggly", as a very wiggly function will over-fit the data. Wiggliness of a function is measured by the second derivative of f. f'' measures of how quickly the *slope* of f changes.

An overall measure of the wiggliness of our data is then

$$\int_{x_1}^{x_n} f''(x)^2 dx$$

With the integral over the range of the x-values.

The penalised sum of squares becomes

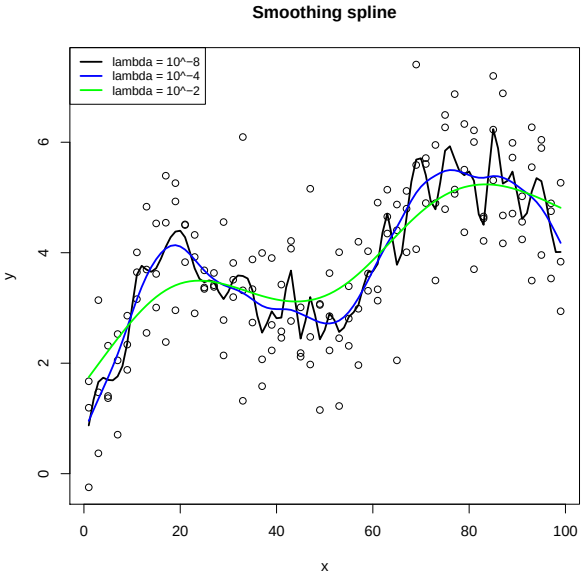
$$\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \int_{x_1}^{x_n} f''(x)^2 dx \tag{1}$$

If $\lambda = 0$ then there is no penalty for wiggliness and the f will interpolate the data if there are no duplicated x-values.If there are duplicated x-values, then imagine taking the mean of the y-values at each unique x-value. f will interpolate these mean values. For ease, we will call this the interpolating function.

If $\lambda = \infty$ then $f''(x) = 0$ meaning that f is linear and the result is a linear regression.

There is a theorem which gives the surprising result:
for any given λ the function $f(x)$ which minimises Equation 1, the penalised least squares formula, will be a natural cubic spline.

The knot points for this cubic spline are the unique values of $x_1, \ldots, x_n$. The predictor function is a smoothed version of the interpolating function.



Effective degrees of freedom

Let N be the number of unique values of $x_1, \ldots, x_n$.

The interpolating function ($\lambda = 0$) depends on N pairs of x and y values.

A linear regression line ($\lambda = \infty$) can be defined using two pairs of x and y values.

The effective degrees of freedom is a value between 2 and N which corresponds to the level of smoothing, the value of λ.

In R you can specify either λ or the effective degrees of freedom. The latter is usually on a more intuitive scale.

Algorithm and cross validation

You do not need to learn the algorithm which fits the spline smoothing, but details are given in Hastie, Tibshirani & Friedman.

The algorithm involves fixing λ and solving a set of linear equations which is an algorithm order n (i.e. fast).

For ridge regression you learnt that cross validation is a method of choosing a good value of λ. A problem with calculating the leave one out cross validation (LOOCV) score in general is that is computationally expensive.

With the spline smoothing algorithm, the LOOCV score can be calculated directly from the numerical solution.

Effectively we get the LOOCV score for free!

Today's Workshop:

- Work through the relevant Lab in James et al.
- Apply what you have learnt to a data set which is a difficult problem in regression/smoothing methods.

Next week: we will look at a similar method to spline smoothing called localised regression (loess) and Generalised Additive Models (GAMS).