Machine Learning II

Prof. Dr. Tim Downie

Week 1 Lecture – 10th October 2022

Non-linear regression methods I

Version: 7. Oktober 2022

Berliner Hochschule für Technik

Machine Learning II – Course Introduction

Topics in Machine Learning I

Unsupervised Learning

- K-Means
- K-Medoids
- Hierarchical Clustering
- Soft Clustering: EM Algorithm

Supervised Learning

- Regression Problems
  - Linear and multiple regression, ridge regression, the lasso.
- Classification Problems
  - Bayes classifier (theoretical), logistic regression, LDA
- Tree Methods Regression & Classification
- Ensemble Methods (applied to tree models)
- Dealing with Missing Data

Machine Learning II

This course builds on the ideas learnt in ML I and deals exclusively with *supervised* learning.

The main topics are

- Non-linear regression methods incl. spline smoothing and GAMs
- Support Vector Machines
- Projection Pursuit Regression
- Artificial Neural Networks
- Simple Bayesian methods: naive Bayes classifier, conditional independence, hidden Markov Models.

A provisional week by week semester plan is given in Moodle.
As in ML I, the methods will be applied using the software R.

Assessment

To pass the course you need to gain at least 45 out of 100 course marks.

The course is examined by a 90 minute written exam worth 65%. Everything covered in the course including computational aspects are examinable unless explicitly stated by the lecturer. Aspects covered in ML1, that are directly relevant to ML 2, such as MSE, are also examinable.

In addition there will be a major project worth 35% of the Marks. The project is not compulsory, but in practice passing the the course will be difficult without it.

The exams will be held on:

- In the 1st exam period: Monday 30th January 2023.
- In the 2nd exam period: ?Monday 27th March 2023.

You should expect that the exams will be held in-person on the BHT campus.

If the pandemic situation worsens, they might be switched to an online exam.
A final decision about the format of the exam will be made in early January.

Information on all exam timetabling can be found at:
https://pruefungen.beuth-hochschule.de/M-DS

Project

- You will work in small groups analysing a medium sized data set and comparing two supervised learning regression methods.
- A regression method means that the outcome variable is continuous or at least numeric.
- Your choice of 2 methods can include:
  - Any method learnt in *Machine Learning 2*
  - Some of the methods learnt in *Machine Learning 1*.

  Full details about which methods are allowed will be specified December.

- Finding a suitable data set is a part of the project. You should check with the lecturer that the data are appropriate before analysing the data.
- The provisional time plan for the project is:
  - 5th December – Project details given out.
  - 21st December or before – details of your proposed data set and the students in your group to be submitted to the lecturer.
  - Tuesday 17th January 2023 – Deadline for submitting your work via Moodle.

Text Books

As in Machine Learning I, you will be using James et al. as the main text book, which is available to download for free. For the more advanced subjects, you will use Hastie et al., which also covers more technical details of the subjects covered in James.

James, Witten, Hastie and Tibshirani *An Introduction to Statistical Learning with Applications in R*. Second edition, Springer-Verlag (2021). `https://www.statlearning.com/` and click on *download the second edition*.

Hastie, Tibshirani and Friedman *The Elements of Statistical Learning (2nd edition).* Springer-Verlag (2009). `https://web.stanford.edu/~hastie/ElemStatLearn/`

Baumer, Kaplan and Horton *Modern Data Science with R.* Chapman and Hall (2017).

W.N. Venables and B.D. Ripley *Modern Applied Statistics with* S (fourth edition, Springer (2002)).

Chantal D. Larose, Daniel T. Larose (2019) *Data Science Using Python and R*, Wiley.

Non-linear regression methods, week 1

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

The subject will be continued next week with

- Spline smoothing
- Local Regression Smoothing
- General Additive Models

Introduction

Many 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, which are used in practice.

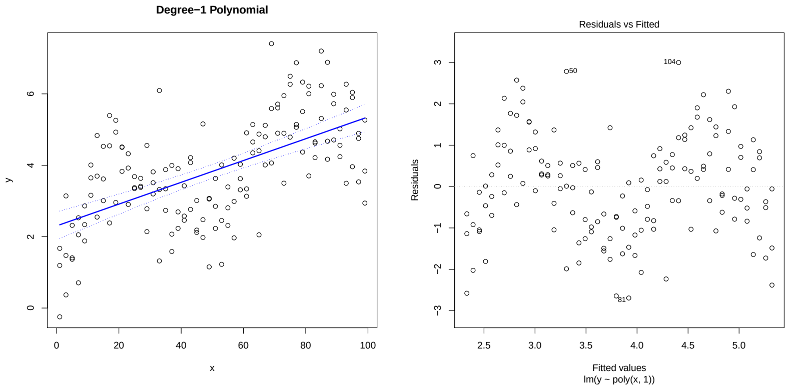
All the methods this week just use one predictor variable x.

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.

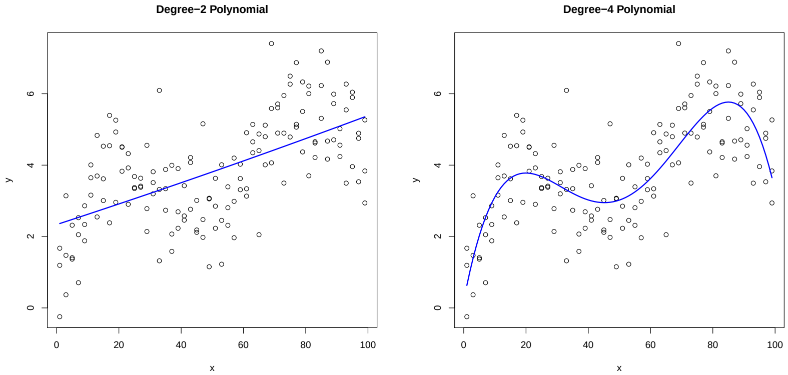
In linear regression the predictor function is $f(x) = \beta_0 + \beta_1 x$ and we choose the coefficients β_0 and β_1 to fit the data the best that a straight line can fit the data.

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.



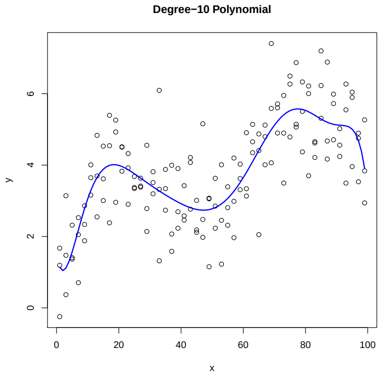
A quadratic regression (left) is almost exactly the same as the linear regression. Quartic regression (right) fits these data fairly well.

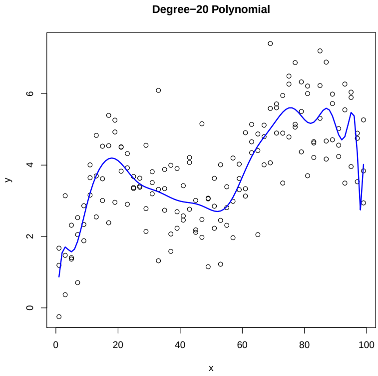
$$y_i = f(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + ... + \beta_d x_i^d + \epsilon_i$$

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.


Degree−10 Polynomial
y
0
2
4
6
x
0
20
40
60
80
100


Degree−20 Polynomial
y
0
2
4
6
x
0
20
40
60
80
100

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.
Local functions are zero outside a certain window.

We will 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.

We choose $K-1$ cut points $c_1, c_2, \ldots, c_{K-1}$ within in the range of X, and fit K step functions.

Define the window functions as

$$C_1(x) = \mathrm{I}(x < c_1),$$
$$C_2(x) = \mathrm{I}(c_1 \leqslant x < c_2), \quad \text{etc.}$$

$\mathrm{I}(\cdot)$ is an indicator function, that returns 1 if the condition is true, and returns 0 otherwise. The formal definition of e.g. $\mathrm{I}(c_1 \leqslant x < c_2)$ is

$$\mathrm{I}(c_1 \leqslant x < c_2) = \begin{cases} 1 & c_1 \leqslant x < c_2 \\ 0 & otherwise. \end{cases}$$

A piecewise constant predictor function has the form

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

We can now find the least squares estimates of the model

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

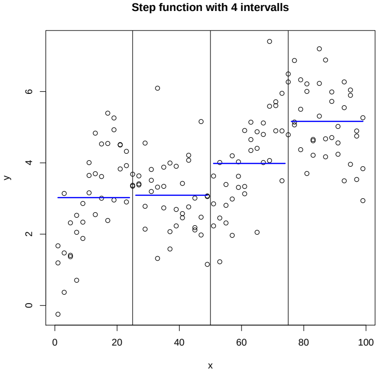
where ϵ_i are the residuals and $\beta_1, \ldots, \beta_K$ are the parameters to be estimated. Reminder: The least squares estimates $\widehat{\beta_1}$, $\widehat{\beta_1}$, ..., $\widehat{\beta_K}$, are the values which minimise

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

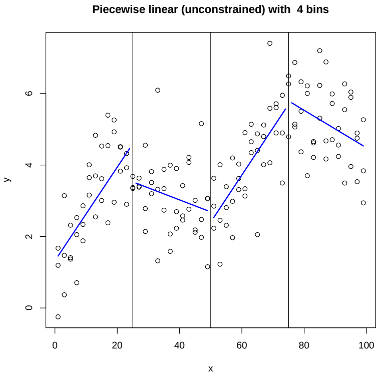
For each x_i, exactly one $C_1, \ldots, C_K$ is non-zero.

The result is that $\widehat{f}(x_i)$ is a piecewise constant function taking the mean values of y in each bin.

This is the result if $K=4$ where the bins are equally sized.



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



The predictor function is now

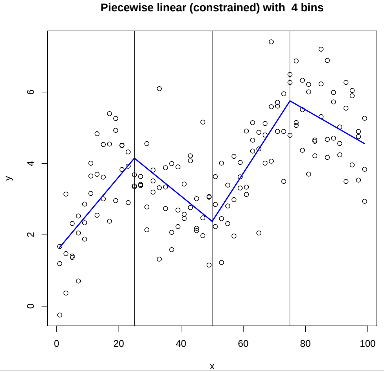
$$f(x) = f_1(x)C_1(x) + \cdots f_K(x)C_K(x),$$

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.

$$y_i = f(x_i) = f_1(x_i)C_1(x_i) + \cdots f_K(x_i)C_K(x_i) + \epsilon_i$$

where each $f_k(x_i) = \beta_{0k} + \beta_{1k}x$ **and** the boundary constraints are $f_k(c_k) = f_{k+1}(c_k)$.



Basis Functions

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

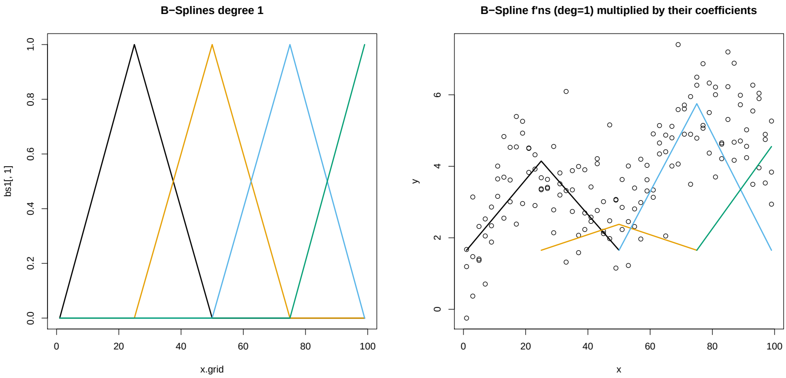
$$y_i = f(x_i) = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_k b_K(x_i) + \epsilon_i,$$

where $b_k(x)$ are fixed functions called basis functions.

Note that there is now a constant term β_0, which makes the fitting of the fixed basis functions easier.

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 spanning *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.
$\widehat{f}(33) = 1.65 + 2.50 \cdot 0.68 + 0.73 \cdot 0.32 + 4.10 \cdot 0 + 2.90 \cdot 0 = 3.5836$
33 is in the second bin, so only β_0, β_1 and β_2 contribute to $\widehat{f}(33)$.

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**.

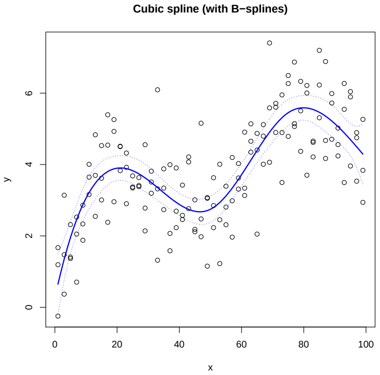
A cubic spline is a piecewise cubic function:

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

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

$$\begin{aligned} f(c_k-) &= f(c_k+) \\ f'(c_k-) &= f'(c_k+) \\ f''(c_k-) &= f''(c_k+) \end{aligned}$$

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.[2]

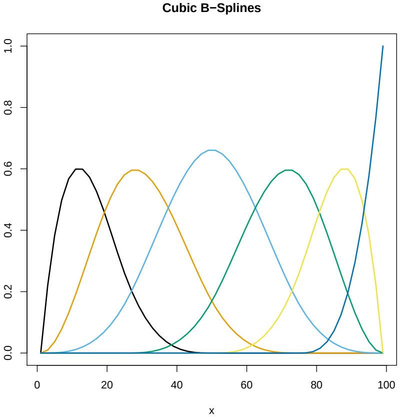
In general for polynomials of degree d and K knots, we require $K+1+d$ parameters, so for cubic splines $K+4$ parameters are needed.

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*.

[2] Note that the definition of K changes slightly here to be consistent with other texts! For spline fitting it is usual to specify K equal to the number of knots, rather than the number of bins.

B-splines are chosen to be efficient to compute via an iterative algorithm, details are in Hastie, Tibshirani and Friedman.

One parameter is used for the global constant parameter so $K+3$ B-splines are required. In our example $K=3$ so we get 6 B-splines shown below.



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.

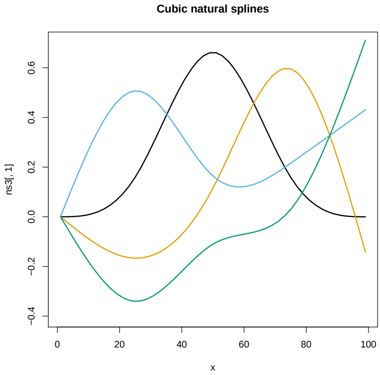
Natural cubic splines have the additional constraints that the second and third derivatives are zero at the extremes of the x-data.

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

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

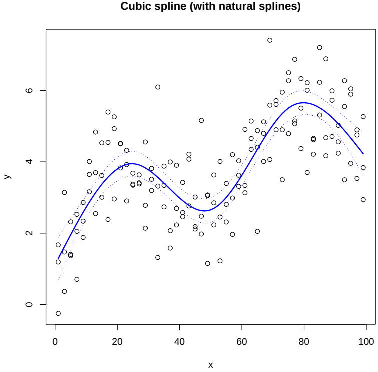
The basis functions are less visually appealing, but there is a data analytical advantage to using natural splines.

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 now require only $K+1$ basis functions.

Cubic spline (with natural splines)



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.