ASL Sheet 7: Spline Regression & Regression stumps

Exercise 1: Spline regression

Consider again the data set pw.csv from sheet 6. Now we want to model the data using splines.

a) **Fit a cubic spline model using the truncated power basis. Use the three inner knots (2.5, 5, 7.5).**
Construct the basis matrix yourself without using pre-defined spline functions and estimate the coefficient vector by simple matrix multiplication.
Visualize the data and the estimated model a little beyond the range of x in the given data set. Moreover, visualize the model function together with all basis functions scaled by their respective coefficient values.

We created a python code that does the following:

- Built the truncated power basis manually using $B_j(x)$ and creates the matrix Figure 1.
  - $1, x, x^2, x^3 \rightarrow the\ global\ polynomial\ terms.$
  - $(x-\zeta_i)_+^3$ for each three knots, for adding local flexibility.

b1(x) = 1	b2(x) = x	b3(x) = x^2	b4(x) = x^3	b5(x) = (x-2.5)^3_+
1.0	0.066	0.004	0.000	0.0
1.0	0.089	0.008	0.001	0.0
1.0	0.361	0.131	0.047	0.0
1.0	0.457	0.209	0.095	0.0
1.0	0.586	0.344	0.201	0.0
1.0	0.691	0.478	0.331	0.0
1.0	0.712	0.507	0.361	0.0
1.0	0.731	0.535	0.391	0.0
1.0	0.841	0.707	0.595	0.0
1.0	0.843	0.711	0.599	0.0

b6(x) = (x-5)^3_+	b7(x) = (x-7.5)^3_+
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0
0.0	0.0

Figure 1: Matrix with Basis Functions

- Uses the knots at 2.5, 5 and 7.5.
- Estimates the coefficients using the least square's function: $\hat{B} = (X^T X)^{-1} X^T y$
- Predicts and plots the spline function from Figure 2.

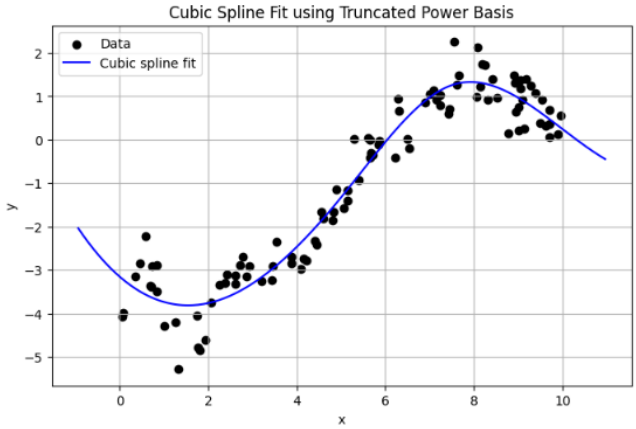

Figure 2: Cubic Spline Fit using Truncated Power Bias

We successfully fitted a cubic spline model using the truncated power basis, and the plot Figure 2 confirms that the spline has captured the non-linear pattern in the data.

There is **local flexibility** near the knots:

- Around $x = 2.5$, the curve bends upward.
- Around $x = 5$, it transitions smoothly.
- Around $x = 7.5$, it starts bending downward.

The function goes slightly wild at the boundaries. This is an expected behaviour because **truncated power bases lack natural constraints outside the knot range**.

If we know observe the scaled basis functions $\mathrm{b_j} * \hat{\beta}_j$ in Figure 3 where each line represents:

- $\mathrm{b}_1(x) * \beta_1 =$ constant function (intercept)
- $\mathrm{b}_2(x) * \beta_2 =$ linear term
- $\mathrm{b}_3(x) * \beta_3 =$ quadratic term
- $\mathrm{b}_4(x) * \beta_4 =$ cubic term
- $\mathrm{b}_5(x) * \beta_5, \mathrm{b}_6(x) * \beta_6, \mathrm{b}_7(x) * \beta_7 =$ truncated cubic terms for knots at 2.5, 5 and 7.5.

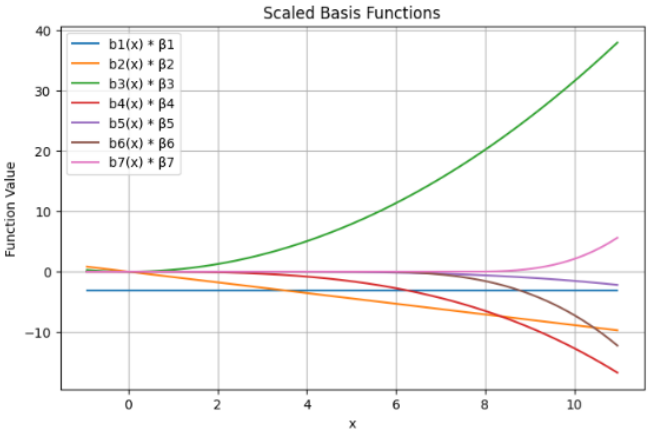

Figure 3: Scaled Basis Functions

In the polynomial terms, from $\mathrm{b}_1(x)$ to $\mathrm{b}_4(x)$ are contributing to the global structure of the spline:

- The $b_3(x) * \beta_3$ term (green line) grows sharply, meaning there's a strong quadratic curvature.
- The oranges and red ($b_2(x) * \beta_2 \; and \; b_4(x) * \beta_4$) go down, shaping the slope and flexibility.

In the other hand, the truncated terms, from $b_5(x) * \beta_5$ to $b_7(x) * \beta_7$ are zero to the left of the knot and then it changes, representing the local flexibility on these knots:

- Around $x = 2.5$, the purple line curve pulls gently down.
- Around $x = 5$, the brown line influences the curvature.
- And near $x = 7.5$, the pink curve becomes active by rising significantly.

We can conclude that the **sum of all these scaled basis functions** gives the spline in Figure 2. Also, the **local variations** in the spline correspond to the **activation of these truncated terms**. And that the **boundary behaviour** beyond the last knot is driven by the unbounded cubic term x^3 and any active truncated terms, which explains the slight downturn or upturn at the edges.

b) Fit a natural cubic spline model using the K = 5 knots (0, 2.5, 5, 7.5, 10) (now including the boundary knots 0 and 10). Visualize the data and the estimated model a little beyond the range of x. How do the models of a) and b) behave in the range of x and beyond the range of x?

We see the following Figure 4 showing the natural cubic spline model using the $k = \{0,2.5,5,7.5,10\}$ knots.

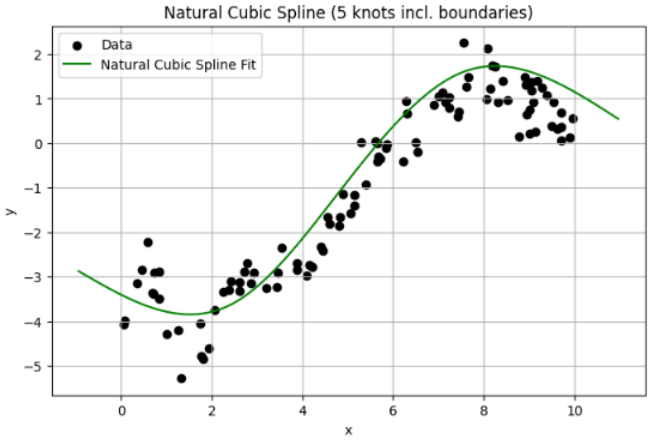

Figure 4: Natural Cubic Spline model (5 knots)

The spline in Figure 4 smoothly interpolates the trend in the data while it avoids sharp jumps. The function behaves linearly outside the range $[0,10]$, so it is consistent with the natural spline constraint. Inside the range, the spline fits a nonlinear pattern of the data by joining the cubic pieces into the 5 different knots.

If we compare Figure 4 with Figure 2, this second one has no unnatural bends beyond the data range; this model avoids erratic behaviour at the boundaries, since the second derivative is forced to be 0 beyond the endpoints.

c) Now use cubic P-splines (penalized B-splines). You can now make use of existing spline functions in your programming language. Fit cubic P-

spline models with the inner knots (0.5, 1, . . ., 9, 9.5) and the boundary knots 0 and 10. Use penalty values λ ∈ {0.01, 0.1, 1, 10, 100} and penalize second order differences. Visualize the data and the five estimated models and comment on your results.

Now we are using the P-spline with a B-spline basis of degree 3 (cubic spline) with evenly spaced inner knots $k = \{0{,}0.5{,}1{,}\ldots{,}9.5{,}10\}$ with 5 different penalty values $\lambda \in \{0.01{,}0.1{,}1{,}10{,}100\}$. In Figure 5 we see the plot of the splines with the various penalties.

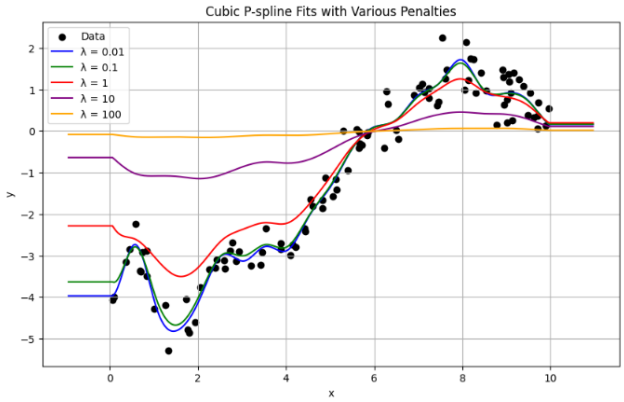

Figure 5: Cubic P-spline Fits with Various Penalties

As we can see, each curve corresponds to a different penalty λ:

- 0.01 (blue line) is very wiggly, it follows the data points closely, it's an overfitted model.
- 0.1 (green line) a bit more flexible but smoother, still overfitted.
- 1 (red line) balanced fit, smooth curvature but close enough to the data points.
- 10 (purple line) is over smoothed, which means that is losing finer patterns in the data, underfitted model.
- 100 (yellow line) almost flat, too rigid and it does not capture any meaningful trends.

Exercise 2: Regression stumps

Consider the following data set for the construction of a regression stump.

x	1	1	2	2	3	4	4	5	6	6
y	5	6	4	7	3	8	7	9	5	10

a) How many different split points t need to be regarded and analyzed in the search for the best split?

We know that a regression stump is a decision tree with only **ONE SPLIT,** a piecewise constant predictor of the form:

$$f(x) = \begin{cases} c_L \; if \; x \leq t \\ c_R \; if \; x > t \end{cases}$$

Where:

- t is the split point, also called the threshold.
- c_L and c_R are the predicted values in each region.
- The goal is to find the best split minimizing some loss.

So, we are considering splits **between distinct adjacent x-values**.

First, sort the x-values: $x = [1,1,2,2,3,4,4,5,6,6]$, the unique sorted x-values are: $[1,2,3,4,5,6]$.

Thus, possible split points t are the midpoints between them:

$$t \in \left\{\frac{1+2}{2}, \frac{2+3}{2}, \frac{3+4}{2}, \frac{4+5}{2}, \frac{5+6}{2}\right\} = \{1.5,2.5,3.5,4.5,5.5\}$$

Therefore, we have 5 different split points t that we need to analyzed.

b) Calculate the best split for a regression stump using the absolute loss function. Do the calculation by hand.

JUST THIS PART

c) Briefly point out which aspects of the calculation change if we use the quadratic loss instead.

If we switch to quadratic loss (the squared error), the optimal prediction in each region becomes the mean, not the median.

This will change:

- The values of c_L and c_R.
- The total loss values
- Possibly, the best split point (even though often it's the same or close).

So, in practice:

- Step will remain the same (looping over possible splits)
- There are only statistic changes from median to mean.
- Loss becomes: $\sum_{i:x_i \leq t}(y_i - \bar{y}_L)^2 + \sum_{i:x_i > t}(y_i - \bar{y}_R)^2$

b) split points / mid points are:

$t_1 = 1.5$

$t_2 = 2.5$

$t_3 = 3.5$

$t_4 = 4.5$

$t_5 = 5.5$

for split at $t_1 \rightarrow$

left ($x \leq 1$): $y = [5, 6]$; median = 5.5

abs. error = $|5-5.5| + |6-5.5| = 1.0$

right: $y = [4, 7, 3, 8, 7, 9, 5, 10]$; median = 7

abs. error = $|4-7| + |7-7| + |3-7| + |8-7| + |7-7|$
$+ |9-7| + |5-7| + |10-7| = 15.0$

Total error = $15 + 1 = 16$

for split at $t_2 \rightarrow$

left: $y = [5, 6, 7, 4]$; median = 5.5

abs. error = $|5-5.5| + \ldots + |4-5.5| = 4.0$

right: $y = [3, 8, 7, 9, 5, 10]$; median = 7.5

abs. error = $|3-7.5| + \ldots + |10-7.5| = 12.0$

total error = $4 + 12 = 16$

for split at $t_3 \rightarrow$

left: $y = [5, 6, 4, 7, 3]$; median = 5

abs. error = $|5-5| + \ldots + |3-5| = 6$

right: $y = [8, 7, 9, 5, 10]$; median = 8

abs. error = $|8-8| + \ldots + |10-8| = 7$

total error = $6 + 7 = 13$

Split at $t_4 \rightarrow$

left: $y = [5, 6, 4, 7, 3, 8, 7]$; median $= 6$

abs. error $= |5-6| + \ldots + |7-6| = 10$

right: $y = [9, 5, 10]$; median $= 9$

abs. error $= |9-9| + |5-9| + |10-9| = 5$

total error $= 10 + 5 = 15$

Split at $t_5 \rightarrow$

left: $y = [5, 6, 4, 7, 3, 8, 7, 9]$; median $= 6.5$

abs. error $= |5-6.5| + \ldots + |9-6.5| = 13.5$

right: $y = [5, 10]$; median $= 7.5$

abs. error $= |5-7.5| + |10-7.5| = 5$

total error $= 13.5 + 5 = 18.5$

* Split at $t_3 = 3.5$ gives the best split as it has the lowest total absolute loss of 13.