STAT 452/652: Statistical Learning and Prediction

Owen G. Ward
owen_ward@sfu.ca

Fall 2023

These notes are largely based on previous iterations of this course, taught by Prof. Tom Loughin and Prof. Haolun Shi.

8 Basis Functions and Splines

(Reading: ISLR 7.3–7.4)

Goals of this section

- We will first consolidate the many different kinds of variables into one general concept, BASIS FUNCTIONS
- Then we will show how very clever combination of different types of basis functions can *massively* increase the flexibility of linear regression models
- However, these methods are mainly useful for displaying a smooth mean trend of existing data
    - Not often used as prediction functions by themselves

8.1 Basis Functions

- We now have identified lots of features you can construct out of one variable
    - linear
    - polynomial of order $d = 2, 3, ...$
    - other transformations like logs and square roots
    - Indicator/step
- All of these can be thought of as "pieces" that could form the basis of a model.
- For this reason, all the different "nameable" functions of a variable are called BASIS FUNCTIONS.
    - You can make a regression model out of any combination of basis functions
- Often limit the basis functions used in one model to a particular type

  - This is called the LIBRARY of basis functions that is used to construct that model.
    - In usual linear regressions, the library is limited to $X_1, X_2, \ldots, X_p$
    - We a make a library out of polynomials up to some order, or the indicators for regions or anything else
    - Can combine function types in a library
    - Can make functions of more than one variable (e.g., cross-products).
  - Library can contain any number of basis functions on any number of variables

- Suppose we have a library of any K basis functions (we don't care what types)
  - Refer to the functions as $b_1(X), b_2(X), \ldots, b_K(X)$

- We can simply build a linear regression out of these basis functions,

$$f(X) = \beta_0 + \beta_1 b_1(X) + \beta_2 b_2(X) + \ldots + \beta_K b_K(X)$$

  - This allows us to estimate parameters using least squares
  - Other inference tools like significance tests for each parameter are available by linear regression theory

- But next we will see how we can cleverly combine certain basis functions into a smooth surface that *automatically adapts to the data!*

8.2 Regression (Basis, Cubic) Splines

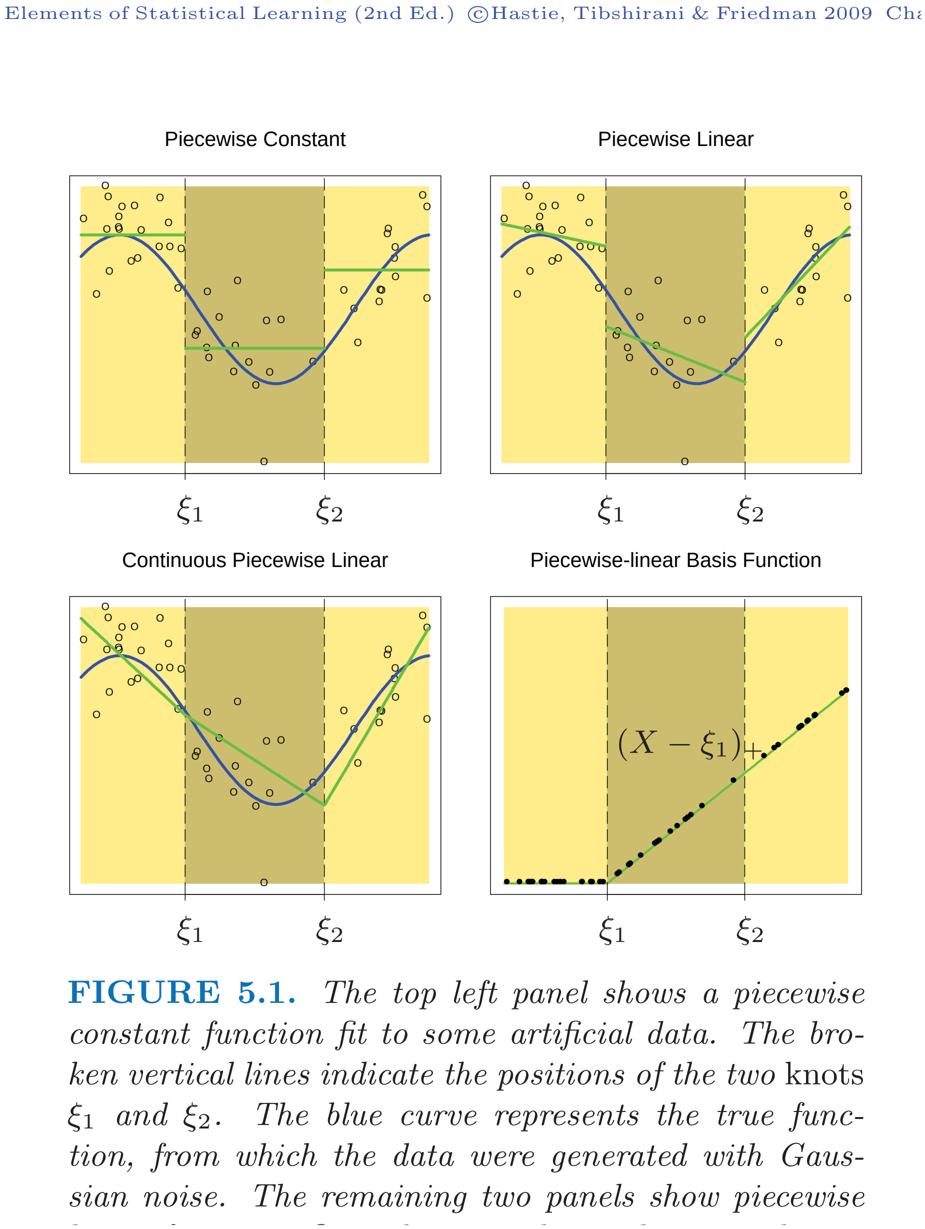
- So far, in every model we have created, we have had to specify a shape for the model
- We still seek models that have more flexible shapes and can adapt to the data
- The first of these methods that we will learn is a family of methods called SPLINES
  - "Spline" = "spliced line"
- They are actually more like spliced cubic polynomials

Piecewise polynomials and regression splines

Focus for now on a single explantory variable, X

- The key innovation is to **divide X into regions and fit simple functions in each piece!**
  - Specifically, create indicator functions that partition X
  - Fit a polynomial within each region (PIECEWISE POLYNOMIAL)
  - Add constraints on the model that force the pieces to join together smoothly (See Figure 1)
  - Details below.
- By fitting models separately in each region, you respect the shape suggested by the "local" data there

- Likely can fit a much simpler function in that region than what would fit the entire range of X
- By keeping functions simple within regions, they are more stable than, e.g., high-order polynomials
  - ∗ Local fits are not influenced by points far away
- Can possibly adapt to shapes that single models cannot



Elements of Statistical Learning (2nd Ed.) ©Hastie, Tibshirani & Friedman 2009 Cha

FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were generated with Gaussian noise. The remaining two panels show piecewise*

Figure 1: Piecewise polynomials, adapted from Hastie et al. 2009.

Details:

- The regions
  - Choose the number of cutpoints, K
    - Cutpoints are called KNOTS in this context because you "tie" the pieces from neighbouring regions together there.
    - Choice of K is a bias-variance tradeoff
    - Large K makes more complex model
      - May overfit and have high variance
    - Small K makes model less flexible
      - Potential for bias
    - Need enough data within each region to reliably estimate each local function.
  - Where to place the cutpoints?
    - Usually use percentiles of X or evenly spread across range of X
    - Again, want to make sure that there is enough data in each region
- The polynomials
  - *Could* use anything
  - For visual smoothness, cubic polynomial is best
    - Complex enough to model two curves in each region
    - Simple enough not to fly all over the place
    - Creates a CUBIC SPLINE when constraints below are added
- The constraints
  - We want "smoothness" at each knot
    - Want to ensure that pieces join together
    - Ensure that they have similar shapes leading up to the knot
    - In particular, same slope and curvature
  - Mathematically, need to set some parameters equal to combinations of others to make this happen
    - "Constraints"
  - Resulting shape is so smooth that human eye cannot detect knots. See Fig 2.

Tuning the fit to the data

- It turns out that this whole model can be written as a multiple linear regression using specially defined basis functions combining cubic polynomials and regions (details in book)
  - Parameter estimates are by least squares!
  - The number of parameters depends on the number of knots K.
    - Model degrees of freedom (DF) $= K + 3$

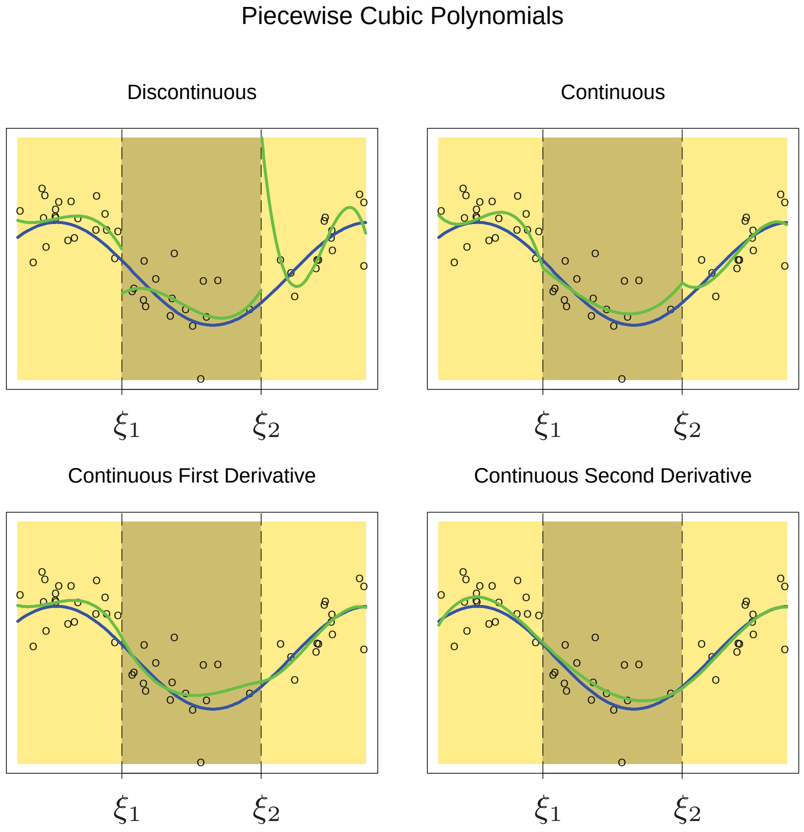


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

Figure 2: Cubic Splines (From HTF)

- Need to choose K
  - This is a tuning parameter like λ in LASSO and Ridge
  - Can estimate best value using MSPE via CV, for example
  - Some packages will compute GCV like in Ridge,
  - OR can choose just based on how it looks!

Natural Splines

- Often find that end segments are very variable, because there are no data on the other side to help place the curve.
  - Replacing cubic with linear at ends often eases this
    * Called a NATURAL CUBIC SPLINE
  - Reduces model DF by 2 to $K + 1$
    * Allows creation of 2 extra knots for same model DF
    * e.g., for 6 DF model, can use 5 knots instead of 3.

Example: Spline smoothing of Covid-19 cases in BC (`Sec8_Splines.R`) The Covid 19 pandemic has resulted in a huge amount of data being made available for analysis. For this example, we have data on the number of cases in BC on each date for the first 9 months or so of 2020. The data are in the spreadsheet `Covid-BC-2020-09-22.csv` and are shown in Figure 3. There are some obvious trends, but the data are also very noisy. We want to find some way to clearly display the mean trend so far. However, the overall pattern is not the shape of any common existing function. So we need to use something that has enough flexibility that it can adapt to what the pattern needs locally.

Polynomials Lets first fit polynomials to the data. We have no intuition what order of polynomial would be needed, but it clearly is not a low order. However, making the order too high might result in excessively variable results. So we can try orders 3, 6, 9, and 12. The results are in Figure 4

As we increase the order, curve comes closer to the middle of the shape. But all of these functions struggle to understand the flatness of the curve early or in June, and none adapt well to the two separate bumps in the first wave (roughly March vs. April). But the 12th order polynomial has problems at the ends, bending sharply and unrealistically down in late September. We could consider higher orders to try to do better with some of the missed features, but these might look even worse in other places, so let's move on.

Cubic regression basis splines Next lets fit cubic splines with DF to match the orders of the polynomials, so that the models would have identical complexity in the LS fit. That is, using $DF = K + 3$, choose $K = 0, 3, 6, 9$ knots. Zero knots is just the same cubic polynomial as above. The `splines` package has a function `bs()` that creates the basis functions that are needed to create cubic splines. You just feed it the X variable and the DF and use the results in the `formula` argument in `lm()`. See the program for details. The results are in Figure 5.

The cubic splines are pretty similar looking to the polynomials, with maybe even more curviness in the earliest flat region. If we were to match up the plots by DF, there would be little difference among the fits.

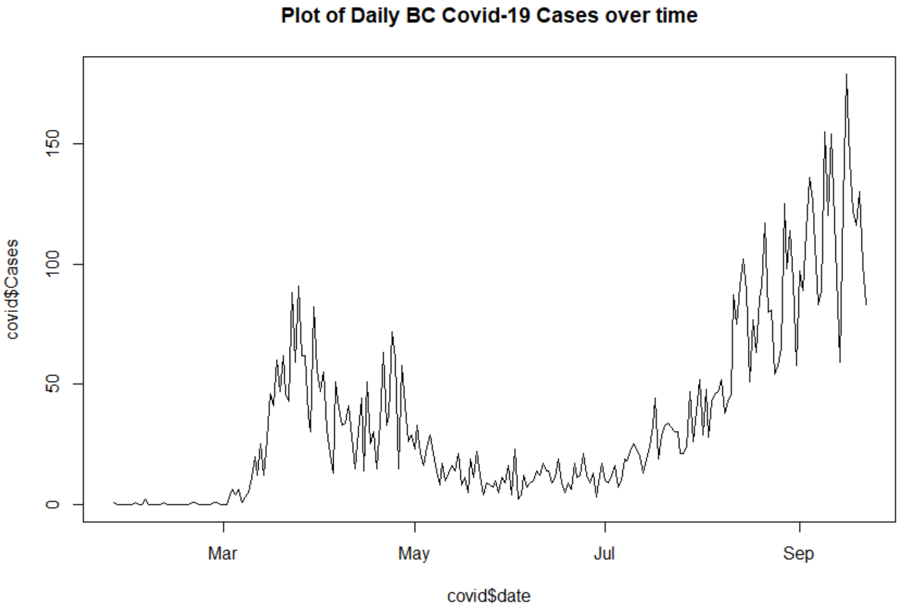


Figure 3: Daily Covid-19 cases in BC from Jan 26-Sept 22, 2020.

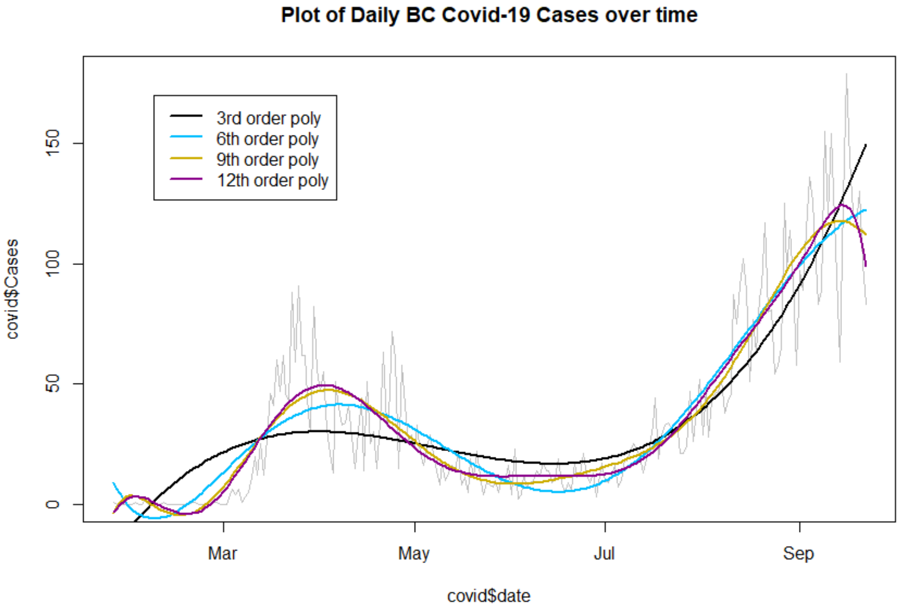


Figure 4: Polynomial fits for smoothing Covid-19 data.

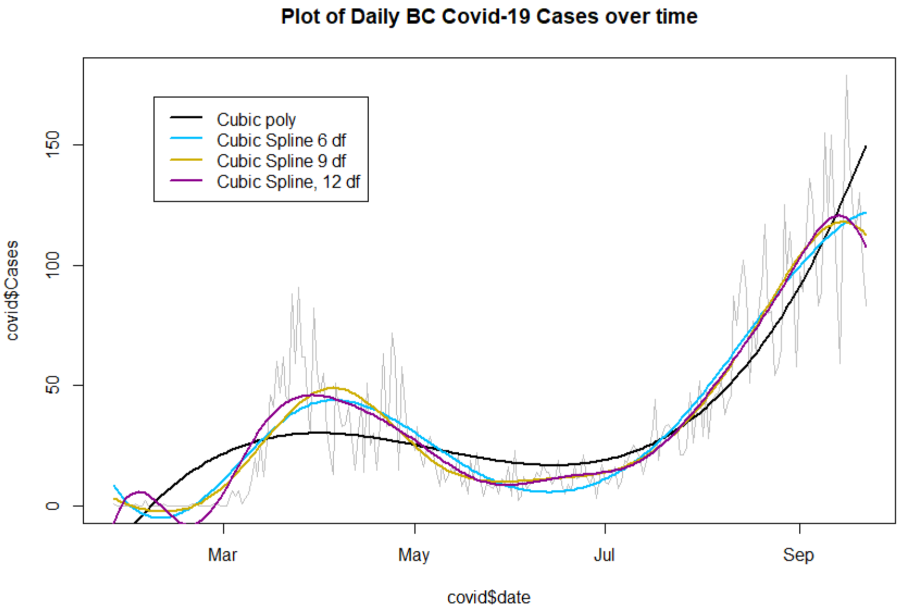


Figure 5: Cubic Regression Splines fit to the Covid-19 data

Natural Splines Natural splines add 2 extra knots and replace the end pieces with linear fits. This might be good here, because the problem we have had so far with the high-DF models is that they are poor in the ends.The `ns()` function creates the basis functions for natural splines and qorks the same way as the `bs()` function we just used. Using $DF = K + 1$ with the same DF as above yields the results in Figure 6.

Finally, the 12-DF fit seems to be able to capture some of the features that other fits could not. It is relatively flat early, thanks to the linearity in the outer regions; it doesn't bend down foolishly in the last part; and the extra knots allow it to sort of notice the second bump in the first wave. It's still not a *great* fit, but it seems better than anything else we have seen.

What to take away from this

- By focusing on data that are "local" to the point you want to predict, you may be able to get better predictions than by using remote data
  - Bias-variance tradeoff:
    - Less data to work with
    - More relevant data to work with
- Building functions by combining other functions is sometimes a winning strategy.
  - Natural splines, in particular, seem to me to squeeze the bias-variance tradeoff effectively.
- However, not useful for extrapolation
  - Nothing else is great at that, either
- Mostly used for presenting a visual smoother on the mean trend.

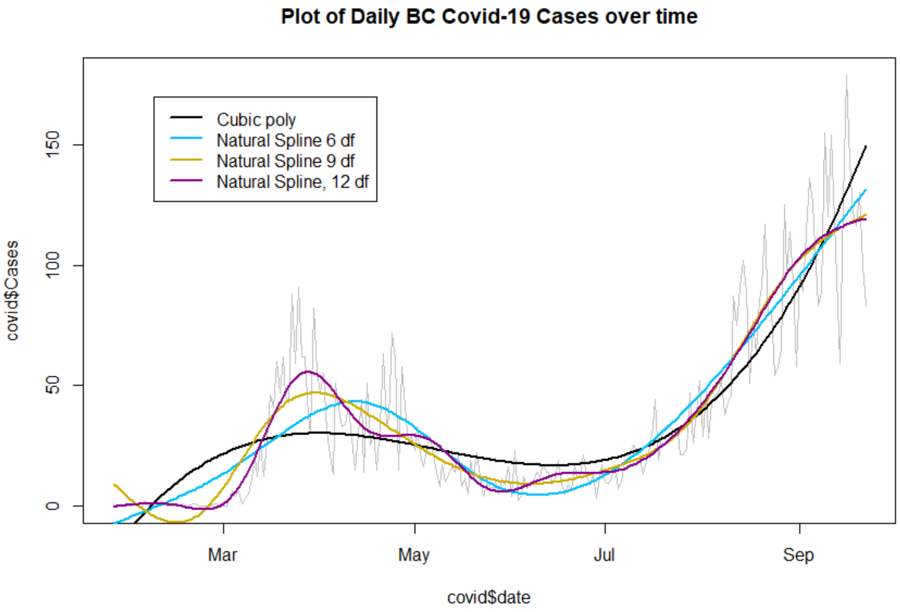


Figure 6: Natural Splines fit to the Covid-19 data

Problem Set 10: Basis Functions and Splines

Application

Refer to the Air Quality data described previously, and the analyses we have done with `Ozone` as the response variable, and the five explanatory variables (including the two engineered features).

1. Use cubic splines to model the relationship between `Ozone` and `Temp`:

   (a) On one graph, plot the data along with fits of

   i. Cubic regression

   ii. Cubic splines with 5, 7, 9, and 20 DF.

   Present the plot. Be sure to add a legend and use different colours for the different functions

   (b) Which model seems to have the most bias? (**Just report the name**)

   (c) Do any functions have a tendency to overfit? If so, **which one(s), and what do you see that causes you to think it/they overfit?**

   (d) If you had to choose one model, **which would it be? Why?**

2. Use natural splines to fit the same data.

   (a) Based on what you saw in the previous exercise, choose three values of DF that you might try here. **List them here.** You are not limited to the same DF values used for the regression splines, if you think that another value might be better than what you tried above.

   (b) **Show the plot** again, using the three models you suggest.

Note that, for these questions, I am less interested in seeing you achieve a perfect solution, and more interested in seeing you learn to appreciate the merits of different fits.