Polynomial regression: fit a nonlinear function to data

* Create new variables X1=X, X2=X^2, etc
* Treat as linear regression
* Not interested in coefficients; worried about the function values at any value x0
* Can find pointwise variances at any point x0
* Can find optimal degree d with cross-validation
* Bad for extrapolation

Step functions: cut a continuous variable into discrete subranges, fit a constant model on each region

* Piecewise constant
* Good if there are natural cutpoints that are of interest
* Advantage over polynomials: local (each point only affects the fit in its region)
* Easy to work with and to interpret
* Creates a set of dummy variables representing each group
* In R, can specify the cut points to create the step
* Cuts create nonlinearities: splines are a better choice
* Good choice for categorical features

Piecewise polynomials: cut the region into sections (add knots) and fit a different polynomial to each region

* Add constraints, like continuity

Spline: piecewise polynomial with continuity constraints

* Enforce continuity up to 1 degree less than the polynomial (so cubic spline is continuous up to 2nd derivative)
* Linear spline: transform the data to fit a series of basis functions
  + Basis function within each region takes some value, is 0 for all points to the left of the region
  + Enforces continuity at the knots
* Cubic spline: add 3 basis functions (x, x^2, x^3), then one basis function for each region
  + Same concept: third-order term starts at 0 at the knot (and first and second derivative are 0)
  + Therefore, function, first derivative, second derivative are continuous at the knot
* Natural cubic spline: linear extrapolation beyond the knots
* In R, use bs() for splines and ns() for natural cubic splines
* One strategy for fitting: find the appropriate number of knots, place them at the acceptable quantiles
* Cubic spline: K+4 degrees of freedom
  + R code ignores the constant: specify df=6 for a cubic spline with 3 knots
* Natural spline: K+d-2 degrees of freedom
  + Lose 2 at each boundary (cubic -> linear)

Smoothing splines: spline with a penalty term associated with the second derivative

* Minimize RSS + penalty
* Penalty determined by tuning parameter lambda
  + Small lambda: small penalty, allow a wiggly function
  + Large lambda: large penalty, function is forced to linear
* Solution: natural cubic spline with a knot at every unique x\_i
  + Avoids having to choose knot placement
* In R: smooth.spline()
* Effective degrees of freedom: sum of diagonal values of the coefficient matrix
  + In R, can specify desired degrees of freedom
* Find desired lambda/degrees of freedom with cross-validation
  + LOOCV works well

Local regression: fit separate linear fits over the range of X by sliding a window along the range

* Does better at boundaries
* Use weighted least squares
* Pretty similar results to smoothing splines
* Set the span: amount of data that’s included in each window

GAM: fit multiple nonlinear functions but retain additivity of linear models

* Fit a different nonlinear function for each feature, then add them together
  + Can include smoothing splines, local regression, any other function
* Not interested in coefficients; we care about the fitted function
* In R, use plot.gam() to visualize the models
* Compare models with anova() (test if a term should be linear or nonlinear)
* No interaction terms are inherently included, but you can add low-order interactions manually