HUDM 6026
Computational Statistics

Nonlinear Regression

References and reading

- **Reading:** Ch. 7 from ISLR
- *An Introduction to Statistical Learning, with Applications in R* (2013), by G. James, D. Witten, T. Hastie, and R. Tibshirani.
- *The Elements of Statistical Learning* (2009), by T. Hastie, R. Tibshirani, and J. Friedman.

Lesson Goals:

- Understand how to extend simple OLS linear regression in a flexible way using polynomial regression, step functions, regression splines, smoothing splines, local regression, and generalized additive models.

Moving Beyond Linearity

- The linearity assumption is good in many machine learning problems.

- However, there are other methods that offer a lot of flexibility, without losing the ease and interpretability of linear models:
  - Polynomial regression
  - Step functions
  - Regression and smoothing splines
  - Local regression
  - Generalized additive models (GAMs)

Polynomial Regression

Replace the standard linear model

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$$

with a polynomial function:

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \cdots + \beta_d x_i^d + \varepsilon_i .$$

Polynomial Regression (cont.)

- For large enough degree *d*, a polynomial regression allows us to produce an extremely non-linear curve.

- We do this by creating new variables $X_1 = X$, $X_2 = X^2$, etc. and then treat as multiple OLS linear regression.

- In general, we are not really interested in the coefficients, but instead the fitted function values at any value x_0:

$$\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0 + \hat{\beta}_2 x_0^2 + \hat{\beta}_3 x_0^3 + \hat{\beta}_4 x_0^4$$

Polynomial Regression (cont.)

- It is unusual to use d greater than 3 or 4 because for large values of d, the polynomial curve can become overly flexible and can take on very strange shapes.

- Note that we can also use cross-validation to choose d
- see R code for details

Step Functions

- Using polynomial functions of the variables as predictor in a linear model imposes a *global* structure on the non-linear function of X.
- To avoid imposing such a global structure, we can create transformations of a variable by cutting the variable into distinct regions.
- In particular, we use *step functions* to break the range of X into bins, where we fit a different constant in each bin.

Step Functions (cont.)

- This amounts to converting a continuous variable into an *ordered categorical variable*.
- In greater detail, we create cut points (or knots) c_1, c_2,…,c_K in the range of X and then construct $K + 1$ new variables:

$$\begin{aligned} C_0(X) &= I(X < c_1), \\ C_1(X) &= I(c_1 \leq X < c_2), \\ C_2(X) &= I(c_2 \leq X < c_3), \\ &\vdots \\ C_{K-1}(X) &= I(c_{K-1} \leq X < c_K), \\ C_K(X) &= I(c_K \leq X), \end{aligned}$$

where $I(.)$ is an *indicator function* that returns a 1 if the condition is true and 0 otherwise.

Step Functions (cont.)

- Note that for any value of X, $C_0(X) + C_1(X) + \ldots + C_K(X) = 1$, since X must be exactly in one of the $K + 1$ intervals.
- We then use OLS estimation to fit a linear model using these $K + 1$ new variables:
- For a given value of X, at most one of $C_1, C_2, \ldots, C_K$ can be non-zero.
- β_j represents the average increase in the response for X in $c_j \leq X \leq c_{j+1}$ relative to $X < c_1$.

Step Functions (cont.)

- Unless there are natural breakpoints in the predictors, piece-wise constant functions can miss the action.

- See R code for example

Basis Functions

- Polynomial and piece-wise constant regression models are special cases of a *basis function* approach.

- The idea is to have at hand a family of functions or transformations that can be applied to a variable X: $b_1(X)$, … , $b_K(X)$

- Instead of fitting a linear model in X, we fit the following model:

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + \ldots + \beta_K b_K(x_i) + \epsilon_i$$

- Note that the basis functions $b_1(.)$,….,$b_K(.)$ are fixed and known.

Basis Functions (cont.)

- For polynomial regression, the basis functions are $b_j(x_i) = x_i^j$
- For piece-wise constant functions, the basis functions are
$$b_j(x_i) = I(c_j \leq x_i \leq c_{j+1})$$
- Note that we can use OLS to estimate the unknown regression coefficients.
- Thus, all of the inference tools for linear models (standard errors, F-statistics, etc.) are available in this setting.

Regression Splines

- Regression splines are a flexible class of basis functions that extend upon the polynomial regressions and piece-wise constant regression approaches we just discussed.
- They involve dividing the range of X into K distinct regions; within each region, a polynomial function is fit to the data.
- These polynomials are constrained so that they join *smoothly* at the region boundaries (or *knots*).
- Provided that the interval is divided into enough regions, this can produce an extremely flexible it.

Piecewise Polynomials

- Instead of fitting a high-degree polynomial over the entire range of *X*, *piece-wise polynomial regression* involves fitting separate low-degree polynomials over different regions of *X*.
- Here, the beta coefficients differ in different parts of the range of *X*; the points where the coefficients change are called *knots*.
- Example: A piecewise cubic polynomial with a single knot at a point *c* takes the following form:

$$y_i = \begin{cases} \beta_{01} + \beta_{11} x_i + \beta_{21} x_i^2 + \beta_{31} x_i^3 + \epsilon_i & \text{if } x_i < c; \\ \beta_{02} + \beta_{12} x_i + \beta_{22} x_i^2 + \beta_{32} x_i^3 + \epsilon_i & \text{if } x_i \geq c. \end{cases}$$

Piecewise Polynomials (cont.)

- Each of the polynomial functions can be fit using OLS applied to simple functions of the original predictor.
- Using more knots leads to a more flexible piecewise polynomial.
- If general, if we place K different knots through the range of X, then we end up fitting $K + 1$ different polynomials.
- It is better to add *constraints* to the polynomials (e.g. continuity).
- *Splines* have the maximum amount of continuity

Piecewise Polynomials (cont.)

- Each constraint that we impose effectively frees up one degree of freedom, by reducing the complexity of the resulting piecewise polynomial fit.

- The general definition of a degree-d spline is that it is a piecewise degree-d polynomial, with continuity in derivatives up to degree d – 1 at each knot.

- Thus, a linear spline is obtained by fitting a line in each region of the predictor space defined by the knots, requiring continuity at each knot.

Linear Splines

A linear spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise linear polynomial continuous at each knot.

We can represent this model as

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

where the b_k are *basis functions*.

$$\begin{aligned} b_1(x_i) &= x_i \\ b_{k+1}(x_i) &= (x_i - \xi_k)_+, \quad k = 1, \ldots, K \end{aligned}$$

Here the $()_+$ means *positive part*; i.e.

$$(x_i - \xi_k)_+ = \begin{cases} x_i - \xi_k & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$

Cubic Splines

A cubic spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise cubic polynomial with continuous derivatives up to order 2 at each knot.

Again we can represent this model with truncated power basis functions

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$\begin{aligned} b_1(x_i) &= x_i \\ b_2(x_i) &= x_i^2 \\ b_3(x_i) &= x_i^3 \\ b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \ldots, K \end{aligned}$$

where

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$

Natural Splines

- A *natural spline* is a regression spline with additional boundary constraints.

- The function is required to be linear at the boundary (in the region where X is smaller than the smallest knot, or larger than the largest knot).

- This additional constraint means that natural splines generally produce more stable estimates at the boundaries.

Natural Splines (cont.)

- A natural cubic spline extrapolates linearly beyond the boundary knots. This adds 4 = 2 x 2 extra constraints, and allows us to put more internal knots for the same degrees of freedom as a regular cubic spline.

Choosing the Location of Knots

- The regression spline is most flexible in regions that contain a lot of knots, because in those regions the polynomial coefficients can change rapidly.

- One option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable.

- In practice, it is common to place knots in a uniform fashion. For example, one strategy is to decide K, the number of knots, and then place them at appropriate quantiles of the observed X.

Choosing the Number of Knots

- One option is to try out different numbers of knots and see which produces the best looking curve.
- However, a more objective approach is to use cross-validation.
- The procedure is repeated for different number of knots K; then the value of K giving the smallest RSS is chosen.
- Splines allow us to place more knots, and hence flexibility, over regions where the function f seems to be changing rapidly, and fewer knots where f appears more stable.

Smoothing Splines

- We create regression splines by specifying a set of knots, producing a sequence of basis functions, and then use OLS to estimate the spline coefficients.

- What we really want is a function g that makes RSS small and *smooth*. Thus, consider the following criterion for fitting a smooth function $g(x)$ to some data (known as a *smoothing spline*):

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^{n} (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

where λ is a nonnegative tuning parameter.

Smoothing Splines (cont.)

- The first term is a loss function (RSS), which tries to make $g(x)$ match the data at each x_i.
- The second term is a *roughness penalty* and controls how wiggly $g(x)$ is; this is modulated by the tuning parameter λ.
- The larger the value of λ, the smoother g will be. The smaller the value of λ, the more wiggly the function.
- As $\lambda \rightarrow \infty$, the function $g(x)$ becomes linear.

Smoothing Splines (cont.)

- It turns out that the solution is a natural cubic spline, with a knot at every unique value of x_i.
- The tuning parameter λ controls the level of roughness (i.e. the effective degrees of freedom).
- Smoothing splines avoid the knot-selection issue, leaving a single λ to be chosen.
- The vector of n fitted values (for a particular choice of λ) can be written as $\hat{\boldsymbol{g}}_\lambda = \boldsymbol{S}_\lambda \boldsymbol{y}$, where $\boldsymbol{S}_\lambda$ is the n x n smoother matrix.

Smoothing Splines (cont.)

- The *effective degrees of freedom* are trace($\boldsymbol{S}_\lambda$), which equals:

$$df_\lambda = \sum_{i=1}^{n} \{\mathbf{S}_\lambda\}_{ii}$$

which is the sum of the diagonal elements of the matrix $\boldsymbol{S}_\lambda$.

- We use LOOCV to find λ. The LOOCV error is given as:

$$\mathrm{RSS}_{cv}(\lambda) = \sum_{i=1}^{n} (y_i - \hat{g}_\lambda^{(-i)}(x_i))^2 = \sum_{i=1}^{n} \left[\frac{y_i - \hat{g}_\lambda(x_i)}{1 - \{\mathbf{S}_\lambda\}_{ii}} \right]^2$$

Local Regression

- *Local regression* is a different approach for fitting flexible non-linear functions, which involves computing the fit at a target point x_0 using only the nearby training observations.

- It is a *memory-based* procedure because we need all the training data each time we wish to compute a prediction.

- The *span* plays a role like that of the tuning parameter λ in smoothing splines; it controls the flexibility of the non-linear fit.

Local Regression (cont.)

- The smaller the value of the span *s*, the more *local* and wiggly will be our fit.
- A very large value of *s* will lead to a global fit to the data using all of the training observations.
- We can use cross-validation to choose *s* or specify it directly.
- Another choice to be made includes how to define the weighting function *K*, and whether to fit a linear, constant, or quadratic regression.

Local Regression (cont.)

Algorithm 7.1 *Local Regression At* $X = x_0$

1. Gather the fraction $s = k/n$ of training points whose x_i are closest to x_0.

2. Assign a weight $K_{i0} = K(x_i, x_0)$ to each point in this neighborhood, so that the point furthest from x_0 has weight zero, and the closest has the highest weight. All but these k nearest neighbors get weight zero.

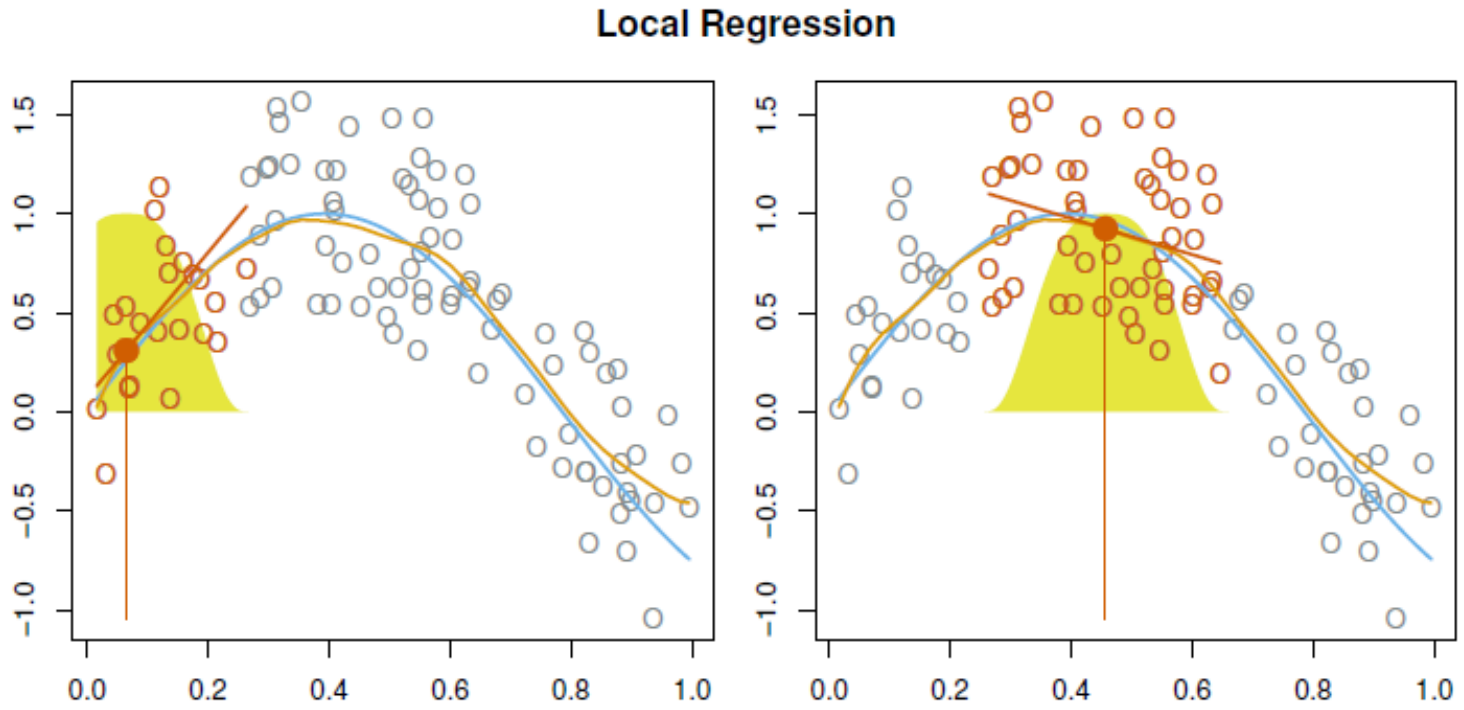
3. Fit a *weighted least squares regression* of the y_i on the x_i using the aforementioned weights, by finding $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize

$$\sum_{i=1}^{n} K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2. \tag{7.14}$$

4. The fitted value at x_0 is given by $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$.

Local Regression (cont.)

- With a sliding weight function, we fit separate linear fits over the range of X by weighted least squares.



Generalized Additive Models

- *Generalized additive models* (GAMs) allow for flexible nonlinearities in several variables, but retains the additive structure of linear models.

- GAMs can be applied with both quantitative and qualitative responses.

- In particular, we replace each linear component of the multiple linear regression model with a (smooth) non-linear function.

Generalized Additive Models (cont.)

$$\begin{aligned} y_i &= \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i \\ &= \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i. \end{aligned}$$

- It is called an *additive* model because we calculate a separate f_j for each X_j, and then add together all of their contributions.

- We can use any of the previously discussed methods (smoothing splines, local regression, polynomial regression, etc.) as building blocks for fitting an additive model.

Generalized Additive Models (cont.)

- GAMs allow us to fit a non-linear f_j to each X_j, so that we can automatically model non-linear relationships that standard linear regression will miss.

- This means that we do not need to manually try out many different transformations on each variable individually.

- The non-linear fits can potentially make more accurate predictions for the response Y.

Generalized Additive Models (cont.)

- Because the model is additive, we can still examine the effect of each X_j on Y individually while holding all of the other variables fixed.

- Thus, if we are interested in inference, GAMs provide a useful representation.

- The smoothness of the function f_j for the variable X_j can be summarized via degrees of freedom.

Generalized Additive Models (cont.)

- The main limitation of GAMs is that the model is restricted to be additive.

- With many variables, important interactions can be missed. However, we can manually add interaction terms to the GAM model by including additional predictors of the form X_j x X_k.

- Although we have not yet covered classification problems, note that GAMs can also be used in situations where Y is qualitative.

Summary

- Polynomial regression, step functions, regression splines, smoothing splines, local regression, and generalized additive models.