Statistics 8330: Data Analysis III
Beyond Linearity: Basis Function Representations and Splines

Suggested Reading: James, Witten, Hastie, and Tibshirani (JWHT), 2013, Chapter 7

Supplemental Reading: Hastie, Tibshirani, and Friedman (HTF), 2009, Chapter 5

Christopher K. Wikle

University of Missouri
Department of Statistics

Beyond Linearity

CKW

So far, we have considered $f(X) = \mathrm{E}(Y|X)$ to be well represented by a linear function in X. However, the linear assumption is almost always an approximation, and often it is not a good one!

We have seen in DA I that we can increase the flexibility of the linear assumption by transforming the variables (e.g., polynomial transformations, log transformations, square root transformations, etc.). We know that such transformations allow us to fit nonlinear curves and we retain the linearity in the parameters, which allows us to continue to use our standard regression machinery (e.g., least squares).

Here, we extend this notion further by considering first an arbitrary basis expansion (where the basis functions are dependent on X) and then some important special cases.

Basis Function Representations

Let the kth basis function of X be denoted by $b_k(X), k = 1, \ldots, K$, which is a function that maps from the p-dimensional real numbers to the real line. We then consider the linear functional representation of $f(X)$ given by these basis functions

$$f(X) = \beta_0 + \sum_{k=1}^{K} \beta_k b_k(X).$$

Clearly, we could then write

$$\mathbf{Y} = \mathbf{B}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where $\mathbf{B}$ is the matrix with columns given by $\mathbf{b}_k = (b_k(x_1), \ldots, b_k(x_n))'$, $k = 1, \ldots, K$. Thus, we can solve for $\boldsymbol{\beta}$ as usual using least squares:

$$\hat{\boldsymbol{\beta}} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{y}.$$

In some cases, when the basis functions are orthogonal, this simplifies even more, e.g., $\hat{\boldsymbol{\beta}} = \mathbf{B}^T\mathbf{y}$. So, the beauty of the linear basis expansion is that we can easily use our existing technology to get estimates.

Basis Functions

CKW

So, what do the basis functions look like?

Some are common functions that we have already considered such as

- *original linear model:* $b_k(X) = X_k$, $k = 1, \ldots, p$
- *polynomials:* $b_k(X) = X_j^2$ or $b_k(X) = X_j^3$ or $b_k(X) = X_j X_m$, etc.
- *nonlinear transformations:* $b_k(X) = \log(X_j)$, $b_k(X) = \sqrt{X_j}$ etc.
- *step-functions:* $b_k(X) = I(c_k \leq X \leq v_k)$, an indicator for a region of X, breaking the region up into non overlapping regions and step-function responses; e.g., for a single X, $b_0(X) = I(X < c_1)$, $b_1(X) = I(c_1 \leq X < c_2)$, ..., $b_{K-1}(X) = I(c_{K-1} \leq X < c_K)$, $b_K(X) = I(c_K \leq X)$, where $c_1, \ldots, c_K$ are called *cut points* in the range of X. (Note: this is just a fancy way to make dummy variables, and gives piecewise constant regression fits.)

We could also consider other functions such as sines and cosines, special functions, wavelets, and empirical basis functions. For now, we consider in a bit more detail various **spline basis functions**, which are very popular and useful.

Regression Splines

CKW

Regression splines build upon the notions of polynomial regression and piecewise constant regression that we have considered in the past.

The main idea behind **regression splines** is that we fit *piecewise polynomial regressions*, but have to add some continuity and smoothness constraints.

As a simple example, say we partition our X range into an area less than c and an area greater than or equal to c and we fit a cubic polynomial separately in each region:

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

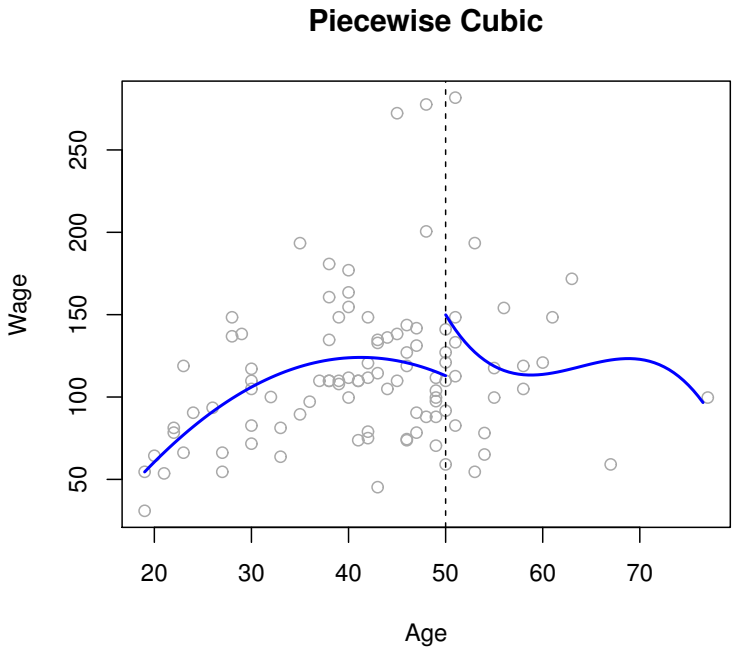
where we call the location c a **knot point**.

To obtain a more flexible model, we would add more knots. In general, if we have K knots through the range of X, we will fit $K + 1$ different cubic polynomials.

Regression Splines (cont.)

Obviously, we don't have to fit just cubic polynomials in these regions – we can fit any polynomial (or, constants). [Apparently, cubics are popular because the human eye cannot as easily detect the discontinuity at the knots. We will see another reason below.]

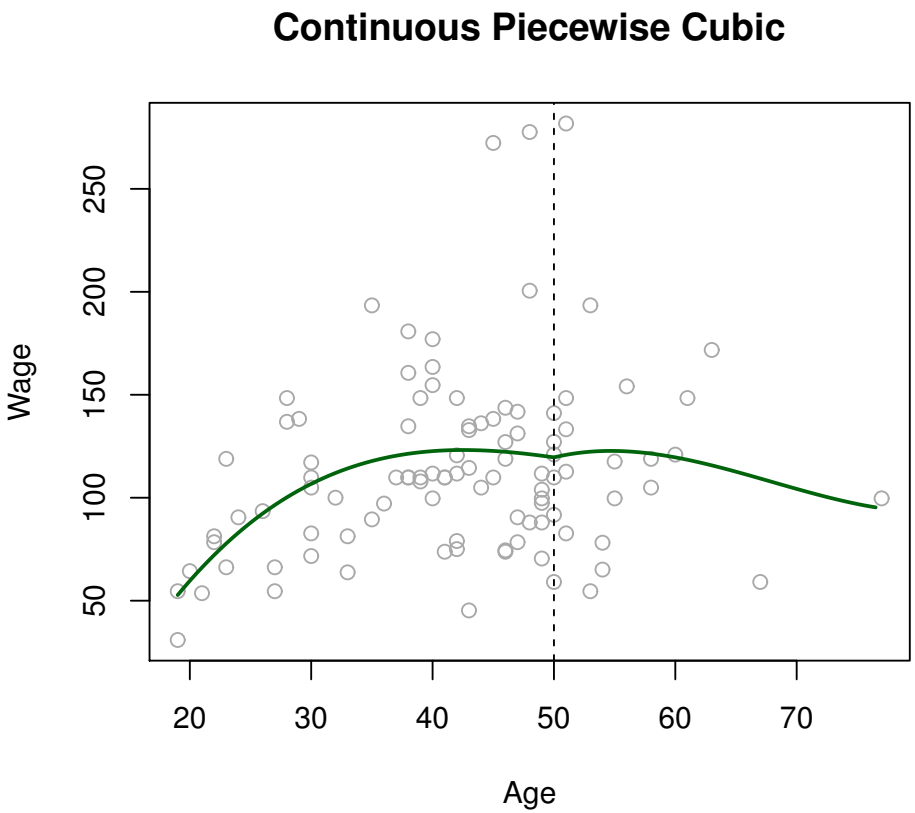
Regardless of the polynomial, there is an issue in that we have not restricted the pieces where they come together at the knot points. E.g., see the following figure from JWHT (2013, Ch. 7.4.1):

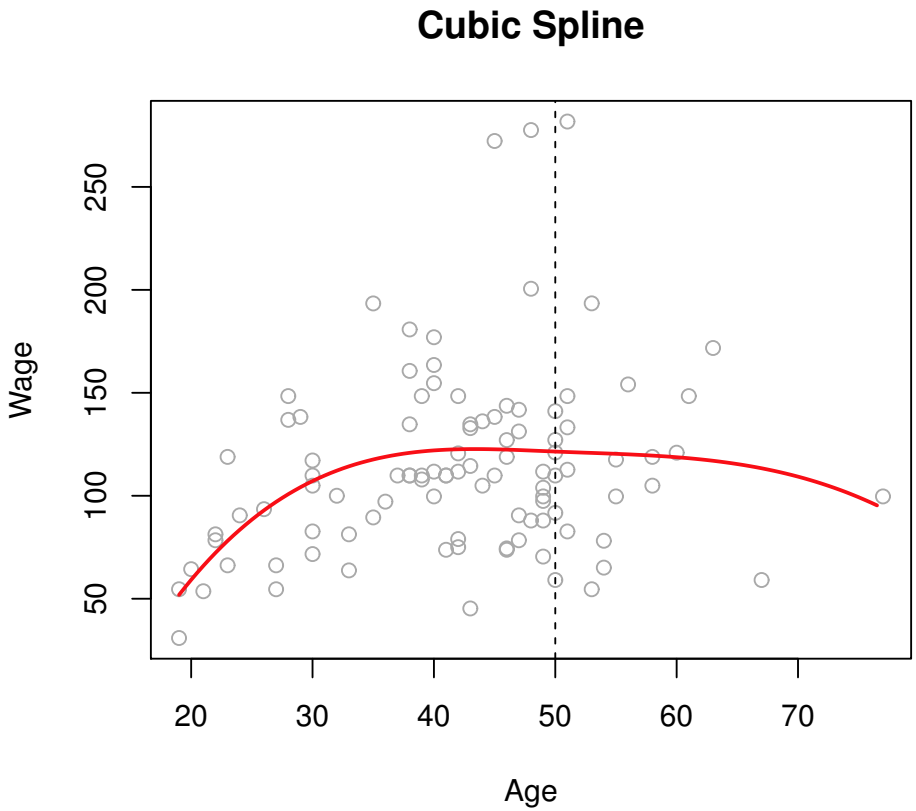


Regression Splines (cont.)

CKW

It would be much better to add the *constraint* that the fitted curve must be continuous. Even better, we typically prefer that it also be smooth, so we constrain the fit so that the first and second derivatives are continuous. In the case of the cubic polynomials, this is called a **cubic spline**. E.g., see the JWHT (2013, Ch. 7.4.1) example:





Regression Splines (cont.)

CKW

Also, consider the example in HTF (2009, Fig 5.2):

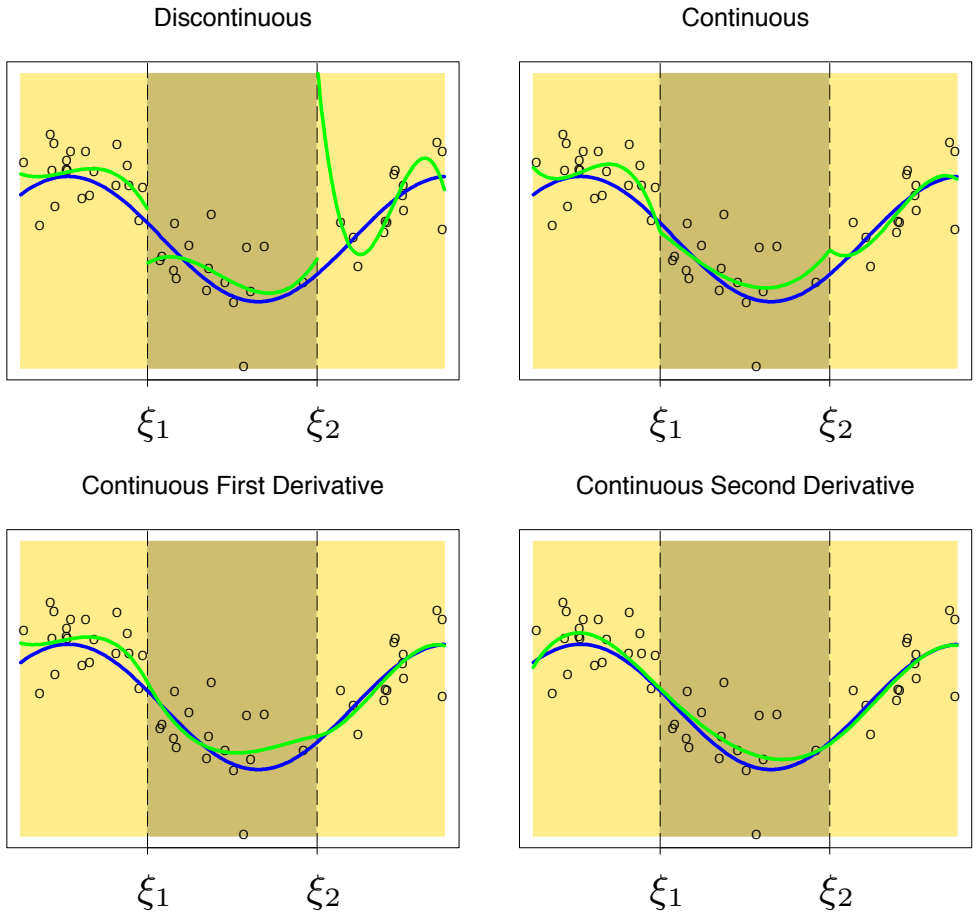


FIGURE 5.2. *A series of piecewise-cubic polynomials, with increasing orders of continuity.*

Regression Splines (cont.)

CKW

We note that each constraint that we add frees up degrees of freedom (i.e., less parameters need to be fit), which reduces the complexity of the fit. A cubic spline with K knots has $K + 4$ degrees of freedom.

Now, we didn't really write the cubic spline as a basis representation when we presented it above. Not surprisingly, we can in fact write them to fit into our general basis notation:

$$b_1(X) = X; b_2(X) = X^2, b_3(X) = X^3, b_4(X) = h(X, \xi_1),$$

$$b_5(X) = h(X, \xi_2), \ldots, b_{K+3}(X) = h(X, \xi_K),$$

where

$$h(x, \xi) = (x - \xi)^3_+ = \begin{cases} (x - \xi)^3 & \text{if } x > \xi \\ 0 & \text{otherwise,} \end{cases}$$

where ξ is the knot. So, fitting this model with an intercept requires estimating $K + 4$ regression coefficients (hence, why the *df* for a cubic spline with K knots is $K + 4$).

Regression Splines (cont.)

CKW

Splines typically have high variance at the outer range of the predictors. For this reason, **natural splines** are regression splines with extra constraints on the boundary that help mitigate this variance problem. In particular, these require the function to be linear in the regions where X is smaller than the smallest knot, and larger than the largest knot. E.g., consider JWHT (2013, Fig 7.4):

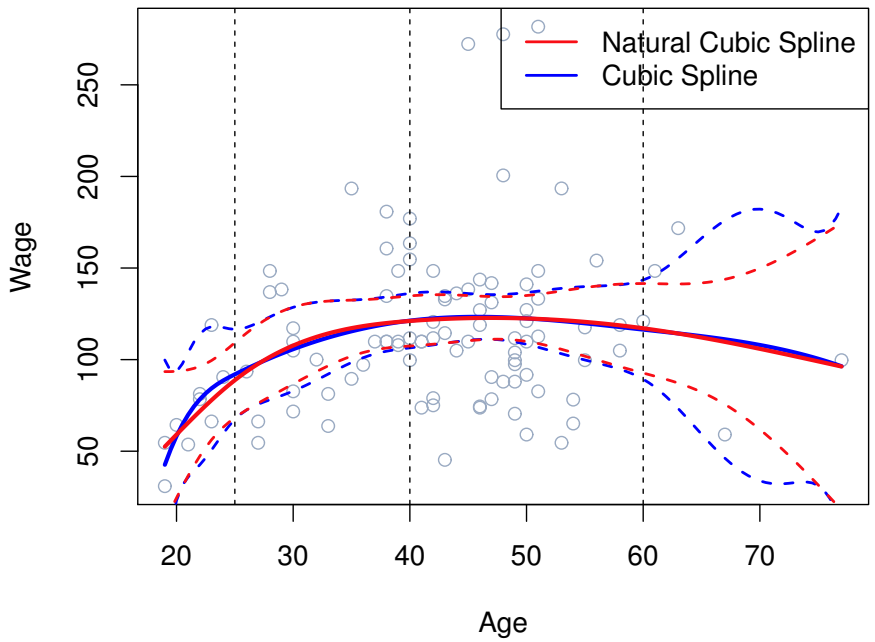


FIGURE 7.4. *A cubic spline and a natural cubic spline, with three knots, fit to a subset of the* `Wage` *data.*

Regression Splines

CKW

The obvious question is how do we choose the number of knots, K, and where do we place them?

Typically, given that we have decided on K knots, we tend to locate them uniformly in the quantile domain of X – that is, if $K = 3$, we would place a knot at the 25th, 50th, and 75th percentiles of the X variable. We note, however, that it is desireable in some cases to place knots less uniformly based on *local* data abundance.

With regards to the choice of the number of knots, a cross-validation procedure is typically considered the best approach. This can be facilitated in some software by a recognition of an equivalence between degrees of freedom and number of knots. In some cases, one simply chooses the number of knots that makes the fit most visually appealing relative to smoothness. This is not typically recommended due to the obvious subjectivity.

Smoothing Splines

Now, consider a different perspective for deciding on a smooth function to fit data. Say we are interested in such a function $g(x)$ and seek to minimize $RSS = \sum_{i=1}^{n}(y_i - g(x_i))^2$. Without any constraints, the $g(x)$ that minimizes this would interpolate all of the y_i, which would clearly over fit. Alternatively, we wish to find the $g(x)$ that minimizes RSS but that is also smooth. E.g., we can find the g that minimizes

$$\sum_{i=1}^{n}(y_i - g(x_i))^2 + \lambda \int g''(x)^2 dx,$$

where λ is a nonnegative smoothness parameter. We call the function g that minimizes this a **smoothing spline**.

Note that the RSS portion of this objective function is the *loss function* and the second term is a *smoothness penalty term*. This later term is critical as the second derivative controls the smoothness of the function (smaller in absolute value implies smoother) and the parameter λ controls the tradeoff between the smoothness and fit.

Smoothing Splines (cont.)

Clearly, when $\lambda = 0$ the function is not smooth at all (since it just seeks to fit the data as best as possible), where if $\lambda \to \infty$ then g will approach a straight line that passes through the training data (e.g., a simple linear regression fit). Thus, λ controls the bias-variance tradeoff of the smoothing spline.

So, how does this connect to the basis function formulation?

Amazingly, the smoothing spline function can be shown to correspond to a cubic polynomial basis with knots at the unique values of $x_1, \ldots, x_n$, with continuous first and second derivatives at each knot! Thus, $g(x)$ that minimizes the smoothing spline constraint is just a natural cubic spline with knots at the unique x observations.

Critically, the answer is not the same as applying the regression spline approach because λ controls the level of shrinkage for smoothing splines. (There is no equivalent shrinkage parameter in the regression spline formulation).

Smoothing Splines (cont.)

At first glance, we suspect that smoothing splines have too many degrees of freedom since they have knots at ALL unique data locations (i.e., as many as $K = n$). However, the parameter λ controls roughness and hence can make the *effective degrees of freedom* much smaller (e.g., as λ increases from 0 to ∞, df_λ decrease from n to 2.)

Note, if we let $\hat{\mathbf{g}}_\lambda$ be the n-vector containing the fitted values for the smoothing spline function for a given λ, then

$$\hat{\mathbf{g}}_\lambda = \mathbf{S}_\lambda \mathbf{y},$$

which shows that it is linear in $\mathbf{y}$. HTF (2009, 5.4) show that $\mathbf{S}_\lambda = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{\Omega}_N)^{-1}\mathbf{N}^T$, where $\{\mathbf{N}\}_{ij} = b_j(x_i)$ and $\{\mathbf{\Omega}_N\}_{jk} = \int b_j''(t)b_k''(t)dt$. As we saw before, we can then consider the effective degrees of freedom to be the trace of $\mathbf{S}_\lambda$:

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

Smoothing Splines (cont.)

The matrix $\mathbf{S}_\lambda$ is sometimes called a *smoother matrix* as it serves to smooth the responses in $\mathbf{y}$. In fact, this will be related to other types of smoothing we will consider shortly that are related to local regression and kernel smoothing. In particular, if we look at the structure of this matrix, we can see that it effectively smooths nearby observations with more weight. E.g., see HTF (2009, Fig 5.8).

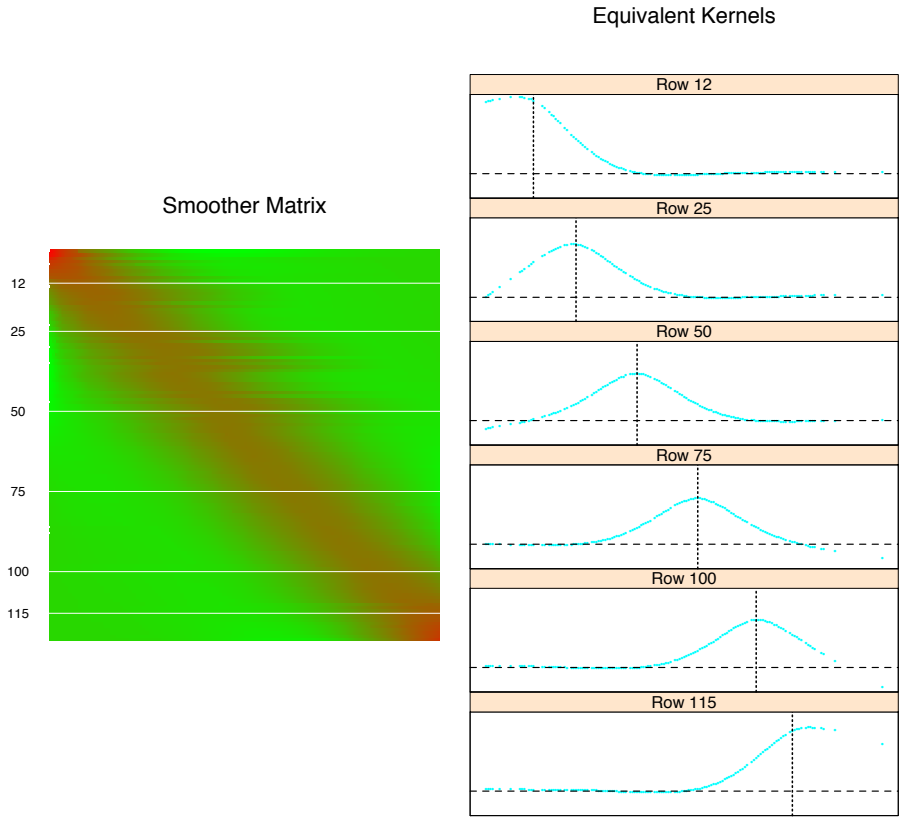


FIGURE 5.8. *The smoother matrix for a smoothing spline is nearly banded, indicating an equivalent kernel with local support. The left panel represents the elements of* $\mathbf{S}$ *as an image. The right panel shows the equivalent kernel or weighting function in detail for the indicated rows.*

Smoothing Splines (cont.)

CKW

To choose λ, we consider cross-validation – sometimes making use of the connection between λ and the effective degrees of freedom. In addition, as we mentioned previously, in the case where we can write a linear predictor we can facilitate cross-validation by using the simplified leave-one-out CV formula:

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

where it is clear that we only have to fit the model once for each λ.

Note, in some software packages, one does the CV on the effective degrees of freedom and then can work out the implied λ.

Smoothing Splines (cont.)

CKW

Consider the example from JWHT (2013, Fig 7.8)

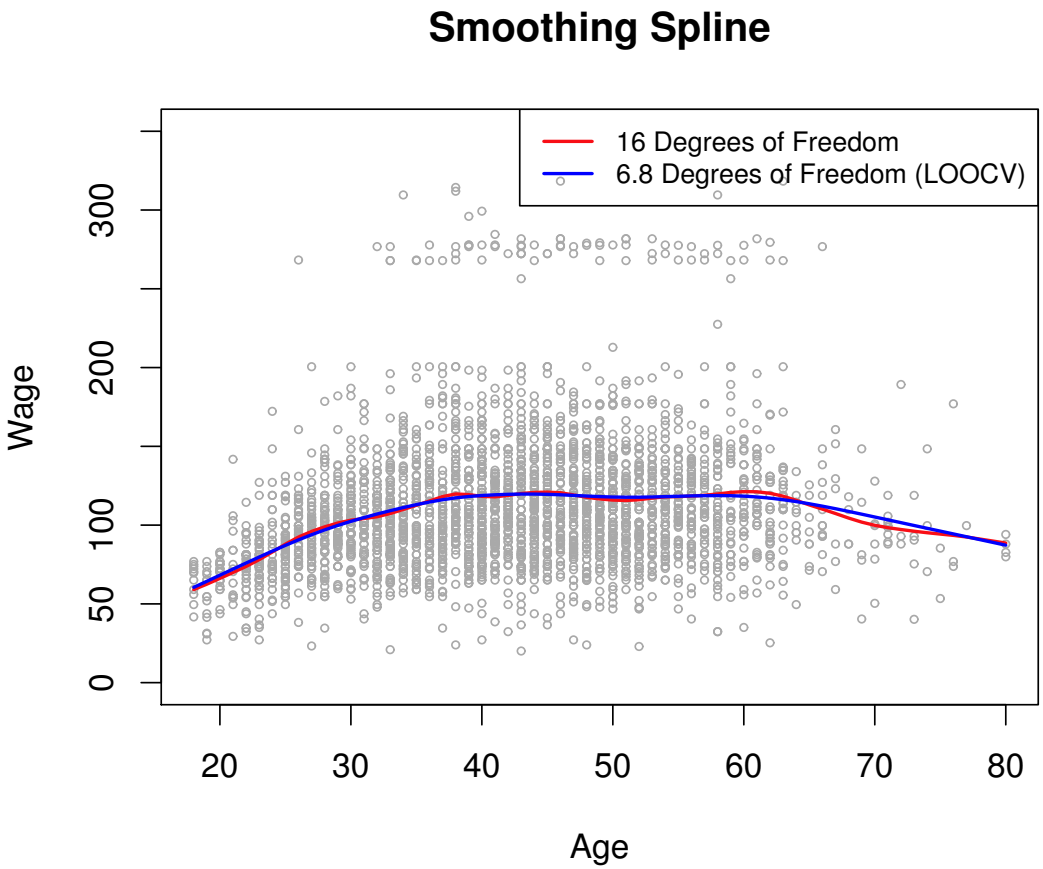


FIGURE 7.8. *Smoothing spline fits to the* `Wage` *data. The red curve results from specifying* 16 *effective degrees of freedom. For the blue curve,* λ *was found automatically by leave-one-out cross-validation, which resulted in* 6.8 *effective degrees of freedom.*

Multidimensional Splines

CKW

To this point, we have focused on one-dimensional spline models. The various spline methods we have considered have analogous forms for high dimensions. In general, one can form higher dimensional basis functions by the use of *tensor products*. For example, if $X \in \mathbb{R}^2$, and we have basis functions $b_{1j}(X_1), j = 1, \ldots, K_1$ for representing coordinate X_1 and $b_{2k}(X_2), k = 1, \ldots, K_2$ for coordinate X_2, then one can define the tensor product basis as:

$$g_{jk}(X) = b_{1j}(X_1)b_{2k}(X_2), \; j = 1, \ldots, K_1, \; k = 1, \ldots, K_2,$$

which can be used as

$$f(X) = \sum_{j=1}^{K_1} \sum_{k=1}^{K_2} \beta_{jk} g_{jk}(X).$$

Multidimensional Splines (cont.)

CKW

As an example, consider the tensor product basis set from HTF (2009):

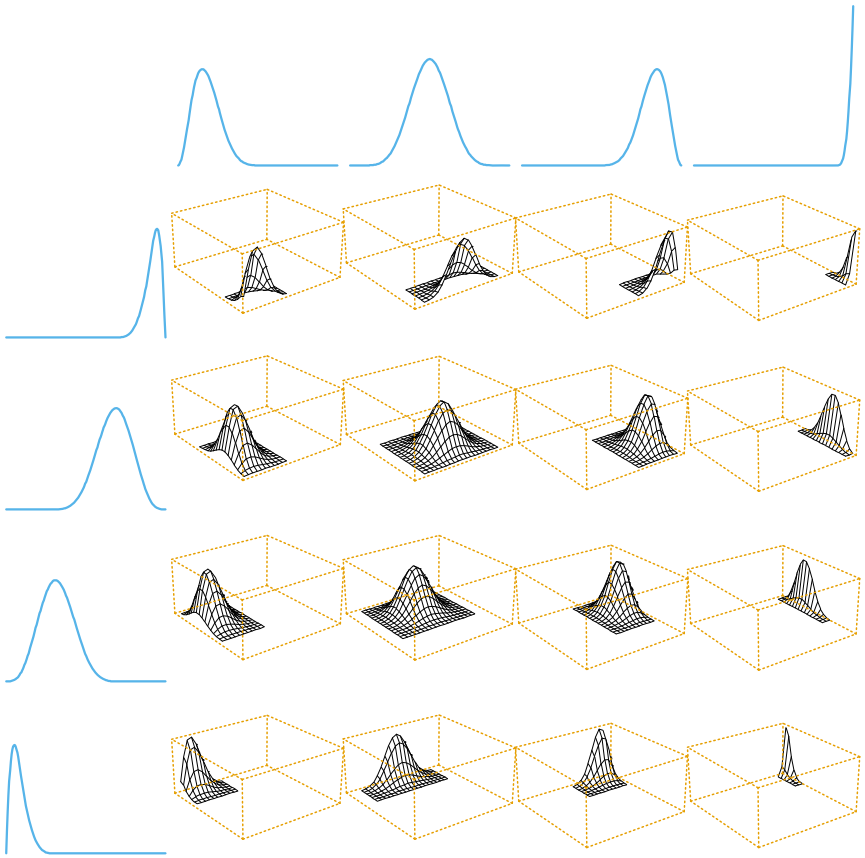


FIGURE 5.10. *A tensor product basis of B-splines, showing some selected pairs. Each two-dimensional function is the tensor product of the corresponding one dimensional marginals.*

Multidimensional Splines (cont.)

CKW

We can also extend the smoothing splines to higher dimensions. Generally, we seek to minimize:

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda J[f],$$

where J is the penalty function for the dimension of the problem. For example, in $\mathbb{R}^2$, we have

$$J[f] = \int\int \left[\left(\frac{\partial^2 f(x)}{\partial x_1^2} \right)^2 + 2 \left(\frac{\partial^2 f(x)}{\partial x_1 \partial x_2} \right)^2 + \left(\frac{\partial^2 f(x)}{\partial x_2^2} \right)^2 \right] dx_1 dx_2.$$

As with the one-dimensional case, $\lambda \to 0$ gives an interpolating function that goes through all of the points, $\lambda \to \infty$ gives a solution approaching the least squares plane, and we seek a λ that compromises between these extremes. These are called **thin-plate splines**.

Multidimensional Splines (cont.)

CKW

One can show that the solution to the thin plate spline minimization gives

$$f(x) = \beta_0 + \beta^T x + \sum_{j=1}^{n} \alpha_j b_j(x),$$

where $b_j(x) = ||x - x_j||^2 \log ||x - x_j||$. These basis functions are known as *radial basis functions* and are used in many types of applications.

Note that there are many other two-dimensional basis functions that could be considered.

We also note that there are additional spline basis functions, perhaps the most useful of which is the B-spline. These can be useful in terms of computing regression and smoothing spline bases more efficiently (see HTF, 2009, Chap 5 Appendix).

Connections

$\mathcal{CKW}$

We will see that methods associated with local regression and kernel regression are very much related to the spline basis regression formulations in this lecture. In addition, we are working toward a framework in which we can incorporate many of these different representations within a given model with multiple covariates.