STA 4241 Lecture, Week 11

Overview of what we will cover

- More flexible regression approaches
  - Smoothing splines
  - Local regression
- Extensions to multiple covariates
  - Generalized additive models

Review of last time

- Last time we learned about a range of different flexible approaches to estimating $f(X)$ in the model

$$E(Y|X) = f(X)$$

- Our goal was to flexibly estimate this function and avoid assuming linearity
- The overarching aim of these approaches is to allow our estimated function to be nonlinear while avoiding overfitting and including too many parameters
  - Highly variable
  - Bad out of sample predictions

Review of last time

- We saw that many approaches fall in the same class of methods based on basis functions
- We can fit the following model

$$E(Y|X) = \beta_0 + \sum_{j=1}^{K} \beta_j b_j(X)$$

- and estimate the parameters $\boldsymbol{\beta}$ using standard linear model techniques
- The main decision to be made was which basis functions to use

Review of last time

- Polynomial functions: $b_j(X) = X^j$
- Piecewise step functions: $b_j(X) = 1(c_j < X < c_{j+1})$
- Cubic splines: $b_1(X) = X$, $b_2(X) = X^2$, $b_3(X) = X^3$, $b_4(X) = (X - \xi_1)^3_+, \ldots, b_{K+3}(X) = (X - \xi_K)^3_+$
- We saw that splines provided a nice middle ground between piecewise step functions and polynomial regression
  - Local behavior of the function
  - Flexible fits from polynomial functions

Review of last time

- We also saw that these ideas can be coupled with other approaches in class
- When we have many basis functions we can use penalization to estimate β
  - Shrink towards linearity
  - Need to be careful about how we implement shrinkage
- Natural cubic splines provide a flexible alternative that requires a small number of parameters
  - Less need for penalization or dimension reduction

Smoothing splines

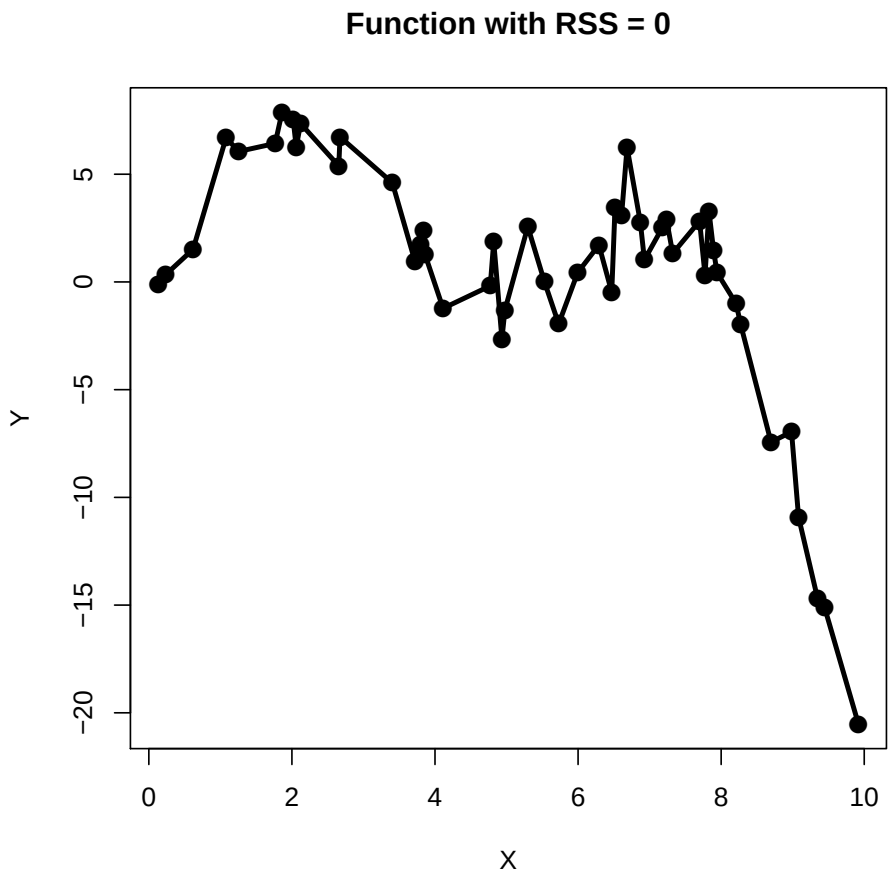
- Today we will learn about a different approach to estimating $f(X)$
  - Smoothing splines
- What is our goal when estimating nonlinear functions?
- First, we want a function such that

$$\text{RSS} = \sum_{i=1}^{n} (Y_i - f(X_i))^2$$

  is as small as possible

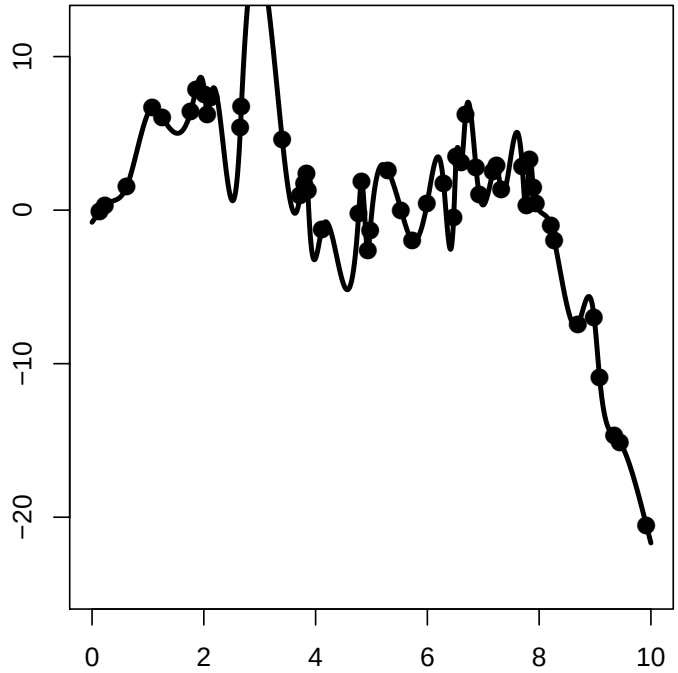
Smoothing splines

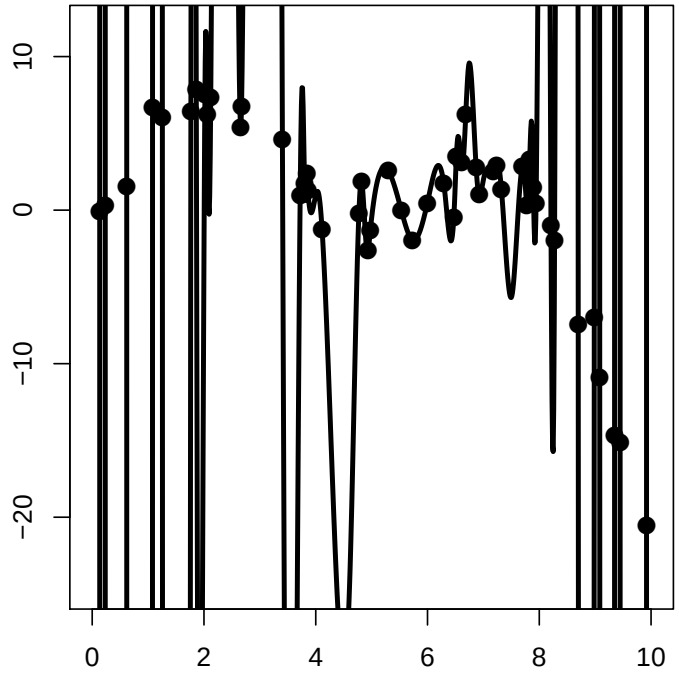
- Unfortunately, we can always find a function that makes this quantity zero
  - Simply interpolate between each point
  - Infinitely many curves that go through the observed data points



Smoothing splines

- Infinitely many curves that go through the observed data points
  - Some more reasonable than others





Smoothing splines

- Clearly minimizing RSS isn't the only goal
  - Need to constrain the function somehow
- Intuitively what we want is a function that reduces RSS but is also smooth
  - Smoothness keeps our function from drastically overfitting to the observed data and leads to better predictions
- How do we measure the smoothness of a function?
  - The second derivative of $f(\cdot)$
- How smooth do we want the function?

Smoothing splines

- The first derivative of the function at t is given by $f'(t)$
  - Measures the slope of the function at t
- The second derivative $f''(t)$ measures how quickly the slope is changing
  - Curvature of the function at t
- The second derivative can be used to measure smoothness of the function
  - Large values indicate a rough or wiggly fit
  - Small values indicate a smooth fit

Smoothing splines

- The smoothest function we have is the linear function, $f(t) = \beta_1 t$
- Second derivative of a linear function is zero
  - Perfectly smooth
- We will use $f''(t)^2$ to measure the smoothness of the function at t
- Interested in the overall smoothness of the function across the range of X
- Instead we use

$$\int_t f''(t)^2 dt$$

Smoothing splines

- We can think of this as being the overall or average smoothness of our function across the range of the variable we are interested in
- We can now formally write down our problem of minimizing RSS while maintaining smoothness
- We will minimize the following

$$\sum_{i=1}^{n}(Y_i - f(X_i))^2 + \lambda \int_t f''(t)^2 dt$$

Smoothing splines

- This looks very similar to the penalization approaches we have seen earlier
- Minimizing RSS + a penalty
- The penalty term penalizes overly flexible fits and favors smoother functions
  - λ controls the bias-variance trade-off
- When $\lambda = 0$, there is no penalty and we end up with a function that interpolates all of the observed data points
- When $\lambda \to \infty$ the penalty forces the function to be perfectly smooth
  - Estimated function is linear

Smoothing splines

- Amazingly the function that minimizes this quantity turns out to be a natural cubic spline with knots at the observed data locations $X_1, \ldots, X_n$
  - Continuous first and second derivatives
  - Linearity constraint outside of the outer knots
- Not the same natural cubic spline we would get if we used the natural spline construction of the previous lecture with knots at $X_1, \ldots, X_n$
  - Smoothing spline is a shrunken version of that
  - λ controls amount of shrinkage

Smoothing splines

- Due to this result, we can write our fitted function as

$$\hat{f}(X) = \sum_j b_j(X)\beta_j$$

  where $b_j(X)$ are the basis functions for the natural cubic splines with knots at each unique X_i

- But the parameters are estimated under a very specific penalty on the second derivative

Smoothing splines

- We estimate the coefficients by solving

$$\widehat{\boldsymbol{\beta}} = \underset{\beta \in \mathbb{R}^n}{\arg\min} \left\{ \sum_{i=1}^{n} \left(Y_i - \sum_j b_j(X)\beta_j \right)^2 + \lambda \boldsymbol{\beta}^T \boldsymbol{\Omega}_b \boldsymbol{\beta} \right\}$$

- Where $\boldsymbol{\Omega}_b$ is an $n \times n$ matrix whose (i, j) element is $\int b_j''(t) b_i''(t) dt$.
- This looks a lot like ridge regression!
  - Only difference is the introduction of $\boldsymbol{\Omega}_b$

Smoothing splines

- The estimated coefficients can be calculated analytically in the same way ridge regression estimates could

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{B}(X)^T \boldsymbol{B}(X) + \lambda \boldsymbol{\Omega}_b)^{-1} \boldsymbol{B}(X)^T \mathbf{Y},$$

  where $\boldsymbol{B}(X)$ is an $n \times n$ matrix with columns given by $b_j(X)$ for $j = 1, \ldots, n$

- It's quite remarkable that the estimator that minimizes the RSS subject to a penalty on the second derivatives has such a nice form

Smoothing splines

- λ controls the degree of smoothness of our resulting function estimate
- To better understand smoothness of the function we need to introduce the concept of *Effective degrees of freedom*
- We are more familiar with the standard degrees of freedom
  - Number of free parameters
  - Number of parameters in our model
- The standard notion does not apply when we utilize constraints or penalties
  - Parameters are not free to take any value

Smoothing splines

- Let df_λ be the effective degrees of freedom (edf)
- Indexed by λ because the edf depends on λ
  - As $\lambda \to \infty$, we have that $df_\lambda \to 2$
    - Linear model has 2 df
  - $\lambda = 0$ makes $df_\lambda = n$
- Larger values of df_λ represent more flexible model fits
  - Lower bias, higher variance

Smoothing splines

- Let our predicted values at the observed locations be given by

$$\begin{aligned}\widehat{\boldsymbol{f}}_\lambda &= \boldsymbol{B}(X)\widehat{\boldsymbol{\beta}} \\ &= \boldsymbol{B}(X)(\boldsymbol{B}(X)^T\boldsymbol{B}(X) + \lambda\boldsymbol{\Omega}_b)^{-1}\boldsymbol{B}(X)^T\mathbf{Y} \\ &= \boldsymbol{S}_\lambda \boldsymbol{Y}\end{aligned}$$

- $\boldsymbol{S}_\lambda$ is the smoothing spline equivalent of the hat matrix in linear regression

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$$

Smoothing splines

- The effective degrees of freedom is given by

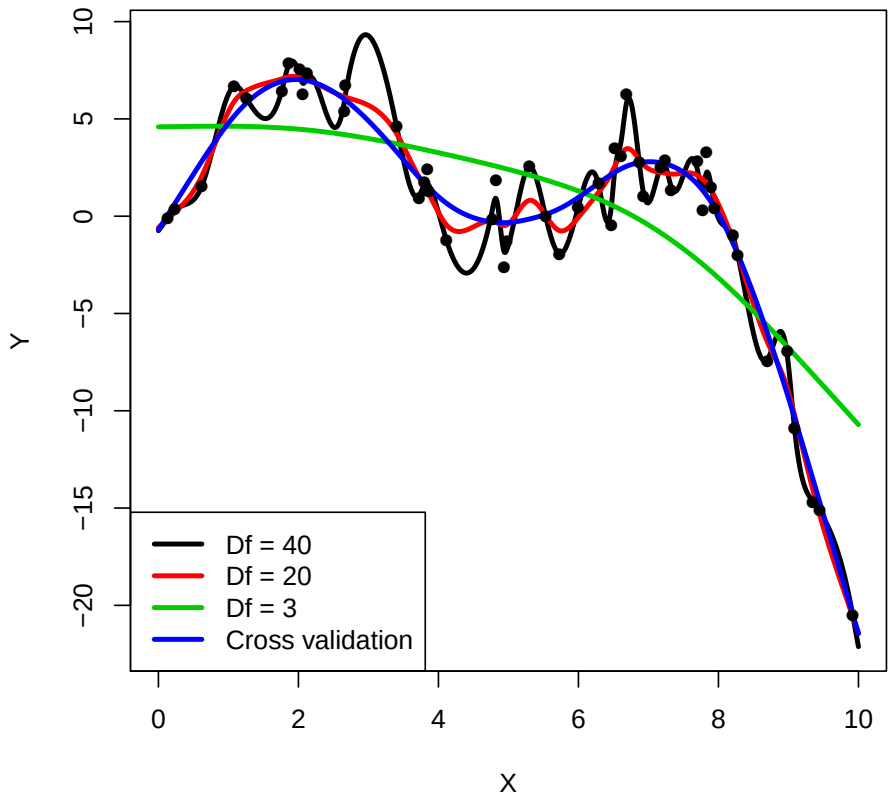
$$df_\lambda = Tr(\boldsymbol{S}_\lambda) = \sum_{i=1}^{n} \{\boldsymbol{S}_\lambda\}_{ii}$$

  the sum of the diagonal elements of $\boldsymbol{S}_\lambda$

- There is a one to one relationship between λ and df_λ
- Can choose a model based on the optimal λ or df_λ
  - Can use CV to find this optimal value

Smoothing splines

- We can see the fit of the smoothing spline for different df_λ values
  - $df_\lambda = 40$ almost perfectly interpolates the data
  - $df_\lambda = 3$ is approaching linearity



Smoothing splines

- Smoothing splines provide a natural way to find a function that fits the data well while maintaining smoothness of the function
- Solution to a very specific natural cubic spline with a penalty on the second derivative
  - High amount of flexibility
  - Shrinks towards smoothness
- We will now examine a slightly different solution that is not based on basis functions in any way

Kernel regression

- Kernel regression is a very flexible, nonparametric technique to estimating $f(X)$
- Similar in flavor to K-nearest neighbors (KNN) regression
- KNN regression amounts to estimating $f(X_0)$ via

$$\widehat{f}(X_0) = \frac{1}{k} \sum_{i \in \mathcal{N}(X_0)} Y_i$$

where $\mathcal{N}(X_0)$ is a set of indices for the k closest points to X_0

Kernel regression

- This can be written as a weighted average of all the data points

$$\widehat{f}(X_0) = \frac{\sum_{i=1}^{n} w(X_i) Y_i}{\sum_{i=1}^{n} w(X_i)}$$

  where the weights are given by

$$\begin{cases} w(X_i) = 1/k & i \in \mathcal{N}(X_0) \\ w(X_i) = 0 & i \notin \mathcal{N}(X_0) \end{cases}$$

- So each observation in the neighborhood is given equal observation and the rest are dropped

Kernel regression

- This is very flexible and doesn't require the selection of any basis functions
- Only decision to be made is the number of neighbors, k
  - Controls bias-variance trade-off
  - Chosen via cross validation
- May not work well in situations with small sample sizes
  - k nearest neighbors aren't very close
  - Generally true for fully nonparametric approaches

Kernel regression

- Kernel regression is an extension of this idea that allows for differing weights for each observation
- We can estimate $f(X_0)$ with

$$\widehat{f}(X_0) = \frac{\sum_{i=1}^{n} K(X_0 - X_i) Y_i}{\sum_{i=1}^{n} K(X_0 - X_i)}$$

- This looks very similar to before but now our weight function $w(X_i)$ has been replaced with a kernel function $K(X_0 - X_i)$

Kernel regression

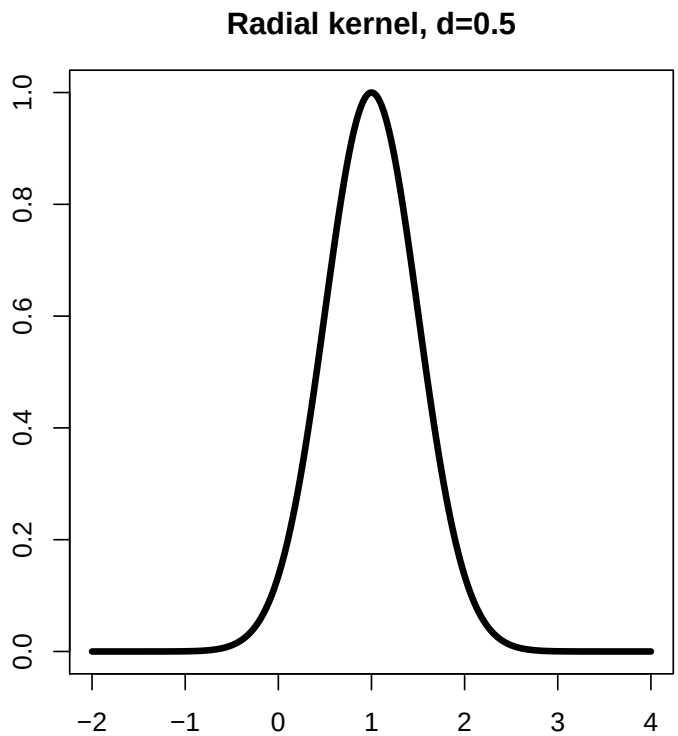
- The kernel is intended to reflect the distance between X_0 and X_i
  - Larger distances lead to small values of $K(X_0 - X_i)$
- Many commonly used kernel functions, some of which we saw when we studied support vector machines
- Radial kernel

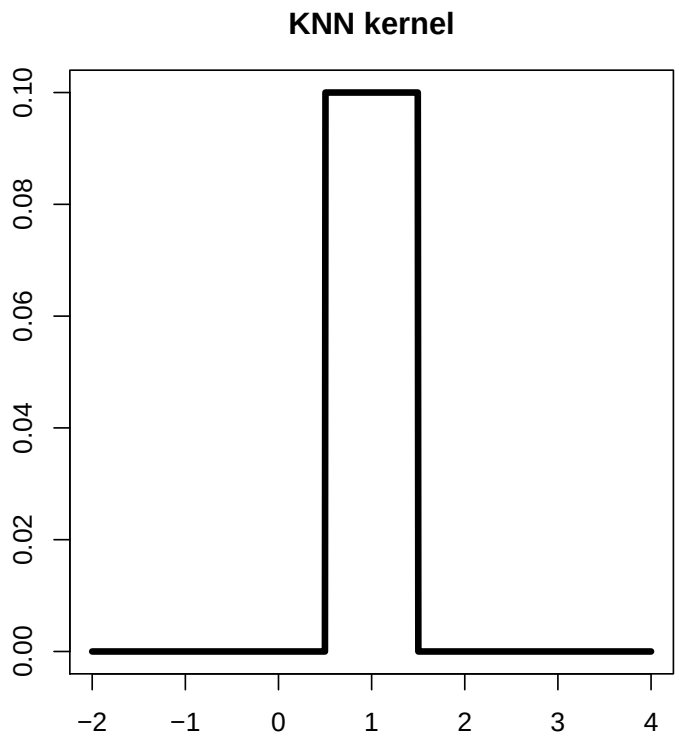
$$K(X_0 - X_i) = \exp\left(\frac{-(X_0 - X_i)^2}{d}\right)$$

- d is a tuning parameter that dictates how the weights vary by distance
  - Analogous to k in the KNN algorithm

Kernel regression

- Below are the radial kernel and KNN kernels for estimating $f(X_0)$ at $X_0 = 1$



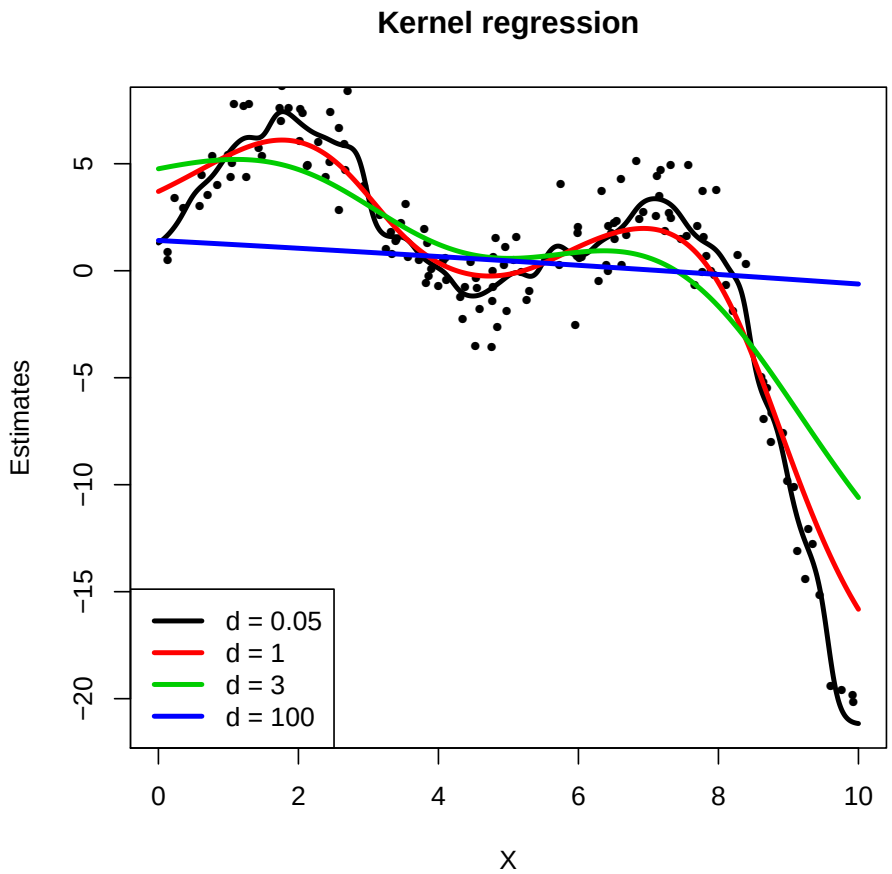


Kernel regression

- Unlike KNN, the kernel assigns positive probability to every single observation
- Increasingly smaller weights as we go farther from X_0
  - Effectively zero at certain distances
  - This distance is controlled by d
- Again d controls the tuning parameter
  - All of these flexible, nonparametric estimators have a tuning parameter that effectively controls the bandwidth of observations we are willing to use in our weighted average

Kernel regression

- Let's apply to our example from before with a few values of d and a radial kernel



Local regression

- An extension of these ideas is called local regression
- Instead of taking a weighted average of the other data points, we fit a weighted regression model
- The idea is still based on kernels as our previous estimator, but now we find (β_0, β_1) that minimize

$$\sum_{i=1}^{n} K(X_0 - X_i)(Y_i - \beta_0 - \beta_1 X_i)^2$$

- This is simply a weighted least squares estimate
  - Different estimates for different X_0 values

Local regression

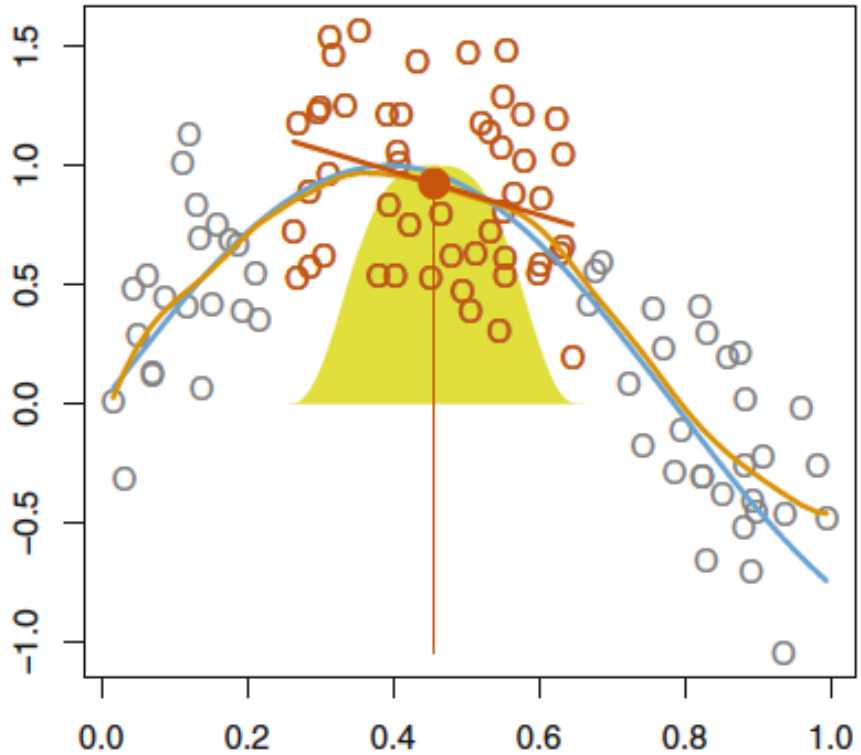
- Once we have these estimated coefficients, we simply estimate

$$\widehat{f}(X_0) = \widehat{\beta}_0 + \widehat{\beta}_1 X_0$$

- Kernel regression is a special case of this procedure where β_1 is set to zero
- Again the most important decision is the choice of the kernel and the bandwidth parameter
  - Bandwidth parameter is ultimately what drives the success/failure of this approach

Local regression

- Here is an example from the book that uses a kernel that only assigns weight to observations within a certain range around X_0



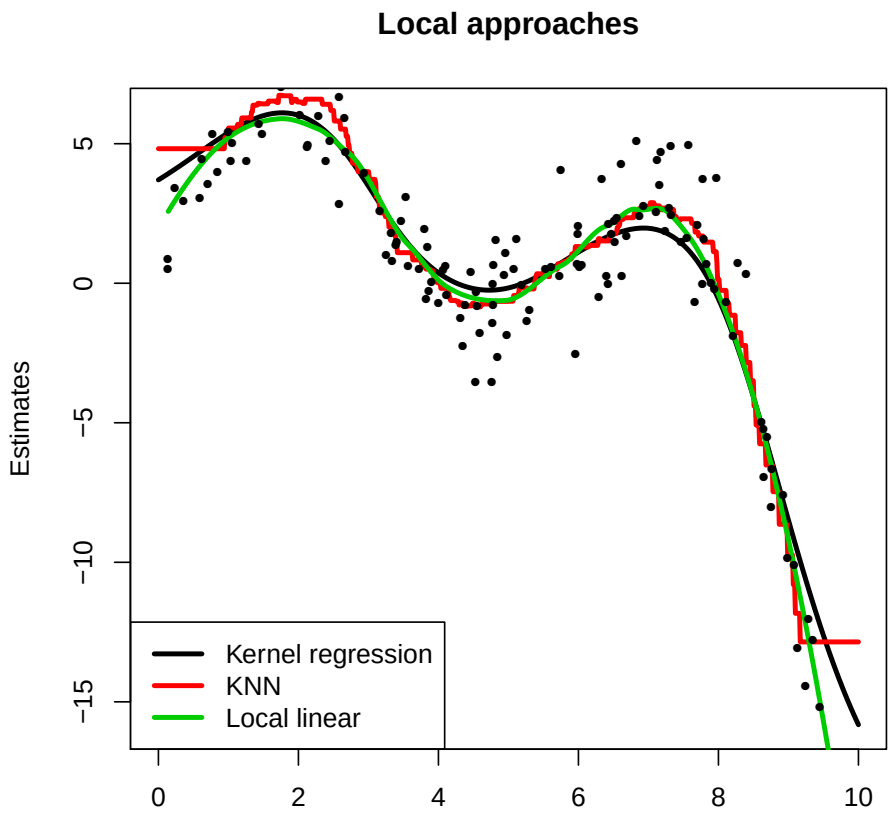
James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). An introduction to statistical learning. New York: springer.

Local regression

- The pros and cons of local regression are the same as for the kernel estimator or KNN
- Pros
  - Highly flexible, can capture nonlinear curves
  - Don't need to specify basis functions
- Cons
  - Computationally heavy because we need to re-do the procedure for every X_0 value
  - Needs larger data sets

Local regression

- Now let's look at all of the localized approaches we have considered



Local regression

- One nice feature of these local approaches is that they can easily be applied with more than one covariate
- Suppose we have two covariates and are interested in estimating

$$E(Y|X_1, X_2) = f(X_1, X_2)$$

- This function allows for both nonlinear terms of X_1 and X_2, but also the possibility of an interaction between the two

Local regression

- The main reason that these approaches apply in the two covariate setting is that the kernel function readily extends to two covariates
  One covariate:

$$K(X_0 - X_i) = \exp\left(\frac{-(X_0 - X_i)^2}{d}\right)$$

  Two covariates:

$$K(X_0 - X_i) = \exp\left(\frac{-||X_0 - X_i||_2^2}{d}\right)$$

  and then all other aspects remain unchanged

Local regression

- The KNN and kernel estimators remain unchanged and are defined as KNN:

$$\widehat{f}(X_0) = \frac{1}{k} \sum_{i \in \mathcal{N}(X_0)} Y_i$$

Kernel regression:

$$\widehat{f}(X_0) = \frac{\sum_{i=1}^{n} K(X_0 - X_i) Y_i}{\sum_{i=1}^{n} K(X_0 - X_i)}$$

except now the neighborhood and kernels are defined with respect to a bivariate X

Local regression

- The main idea of local regression remains unchanged, but can be altered slightly to incorporate both covariates into the linear function
- Now we minimize

$$\sum_{i=1}^{n} K(X_0 - X_i)(Y_i - \beta_0 - \beta_1 X_{1i} - \beta_2 X_{2i})^2$$

and our predictions are

$$\widehat{f}(X_0) = \widehat{\beta}_0 + \widehat{\beta}_1 X_{01} + \widehat{\beta}_2 X_{02}$$

Local regression

- While these approaches readily extend to more than one covariate, we can only utilize them when p is relatively small
- We saw earlier in a lab on KNN that these types of approaches are heavily impacted by the curse of dimensionality
- As the number of covariates grows, it becomes increasingly unlikely to have observations in the data that are close to X_0
  - Need massive data sets in these cases
- When p is larger, we can utilize parametric approaches that are not as affected by the curse of dimensionality

Two-dimensional basis functions

- The approaches that we considered earlier can also be extended to two dimensions
- Any approach based on basis functions can be extended to $p = 2$ settings
- The main idea is to use tensor product basis functions
- Suppose we have K_1 basis functions for X_1 and K_2 for X_2 defined by

$$\{b_j^1(X_1)\}_{j=1,\ldots,K_1} \qquad \{b_j^2(X_2)\}_{j=1,\ldots,K_2}$$

Two-dimensional basis functions

- We can define tensor products as

$$b_k(X_1, X_2) = b^1_{j_1}(X_1) b^2_{j_2}(X_2) \quad \text{for } k = 1, \ldots, K_1 K_2$$

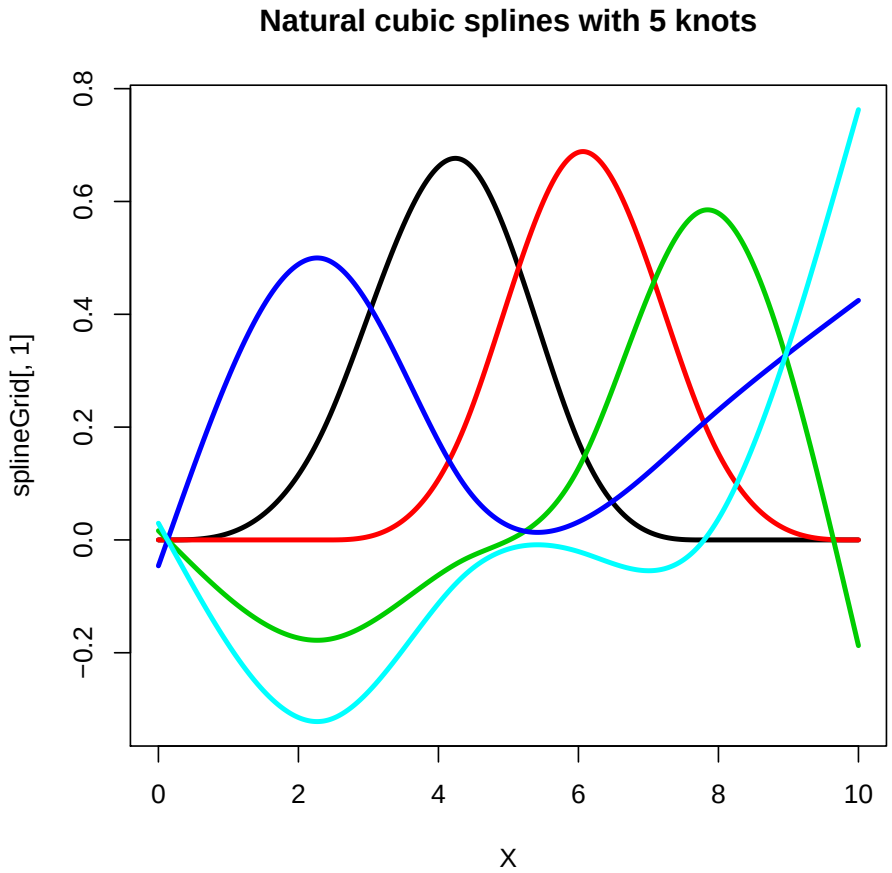
- And we can estimate our two-dimensional function using

$$f(X_1, X_2) = \sum_{k=1}^{K} b_k(X_1, X_2)\beta_k$$

- And estimate $\boldsymbol{\beta}$ any way we like
  - Least squares
  - Penalization

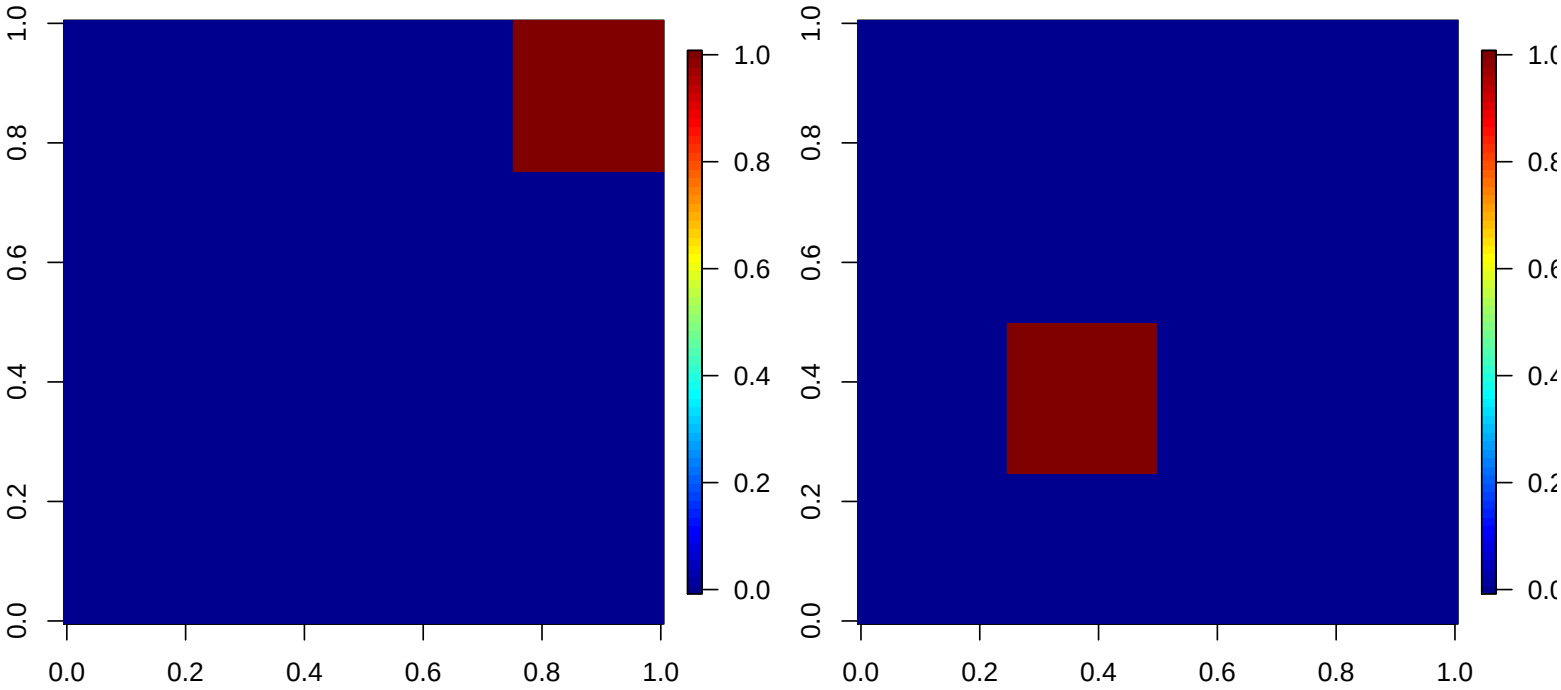
Two-dimensional basis functions

- In one dimension, our basis functions are curves and we aim to represent our function as a linear combination of these curves



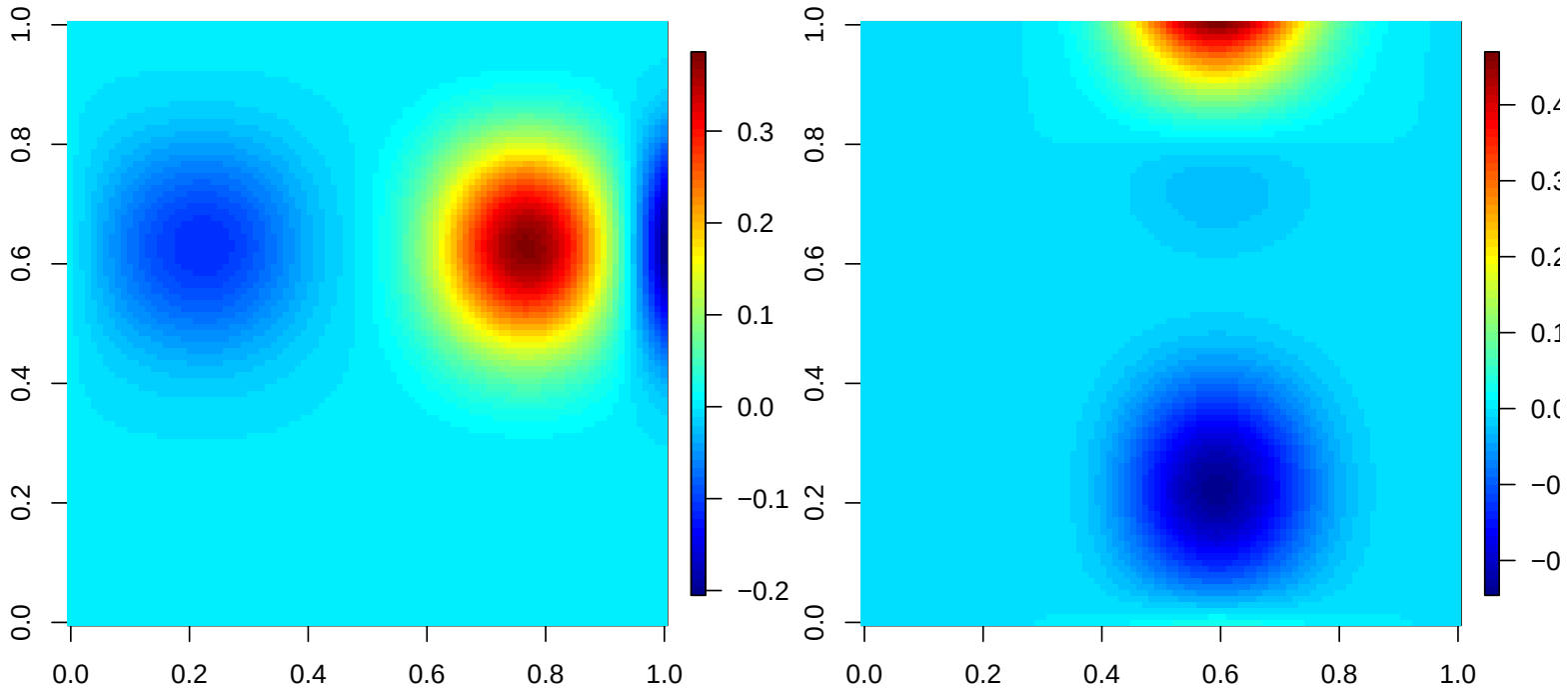
Two-dimensional basis functions

- In two dimensions, our basis functions are surfaces
- Here is a visualization of two such basis functions when we use step functions



Two-dimensional basis functions

- Here is a two-dimensional basis function constructed from tensor products of natural splines



Two-dimensional basis functions

- While this provides a solution to the two-dimensional problem, it is not as flexible as the local approaches such as KNN or kernel approaches
- Estimated function is constrained by the tensor product formulation
- Not flexible enough to capture any 2d function
  - Adding more basis functions helps

Two-dimensional basis functions

- This also gets increasingly difficult as p grows
- In principle could construct any order function via

$$b(X_1, \ldots, X_p) = b_{j_1}(X_1) b_{j_2}(X_2) \ldots b_{j_p}(X_p)$$

- This has two major problems
  - Leads to a massive number of basis functions
  - Only can capture very specific functions
- Typically as p grows we need to make some additional modeling assumptions

Generalized additive models

- The main assumption made as p grows is that of additivity
- In the traditional linear model, we assume both additivity and linearity

$$E(Y|\boldsymbol{X}) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

- The effect of X_j on Y is linear and it is the same regardless of the values of the other covariates (additivity)
- Generalized additive models (GAMs) relax the linearity assumption while maintaining additivity

Generalized additive models

- A typical GAM takes the following form

$$E(Y|\boldsymbol{X}) = \beta_0 + \sum_{j=1}^{p} f(X_j)$$

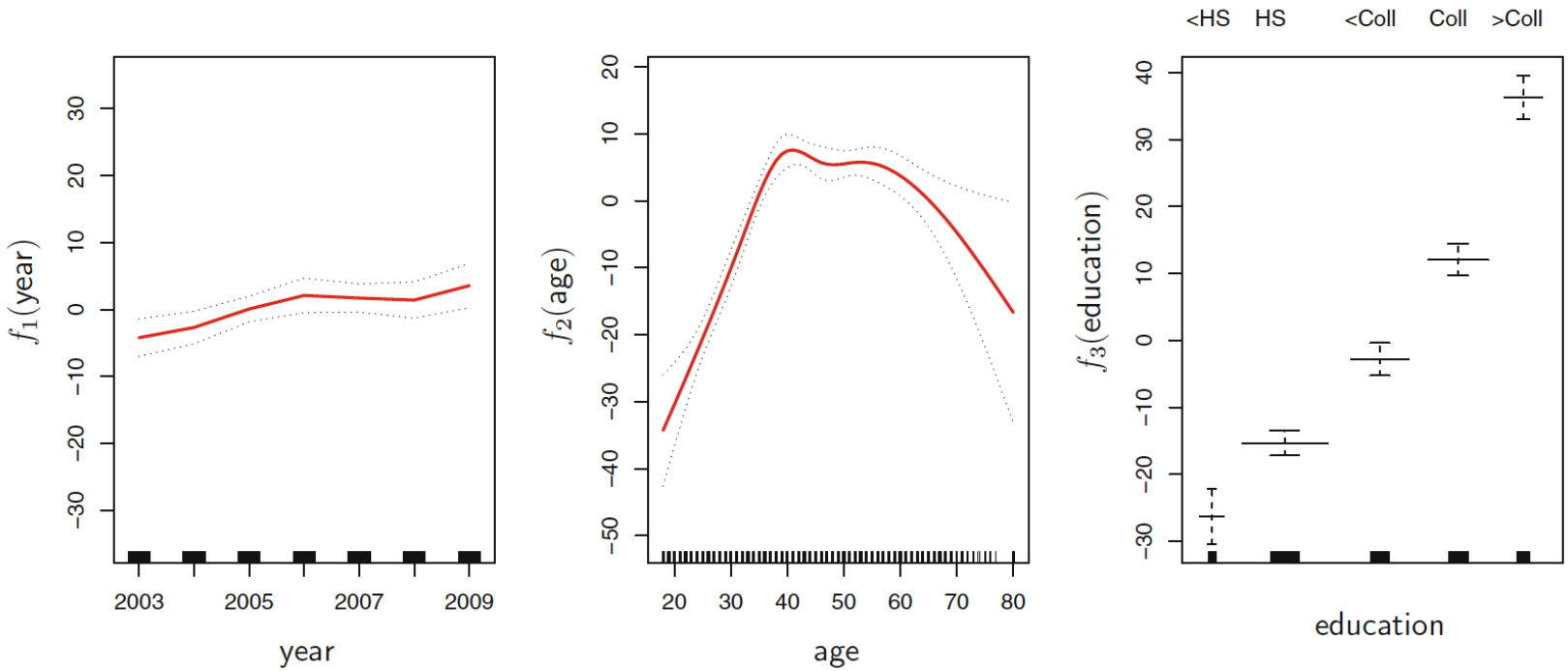
- Each covariate has its own nonlinear function $f_j(X_j)$
- Additivity holds as the effect of X_j on Y still does not depend on other covariates
  - No interactions between predictors

Generalized additive models

- One desirable feature of GAMs is that nearly all of the approaches over the last two lectures we have considered for nonlinear function estimation can be applied
- Each $f_j(X_j)$ can be modeled using your favorite nonlinear approach
  - Natural splines
  - Smoothing splines
  - Local regression
- If X_j is categorical, we can simply use indicator functions as we normally would

Generalized additive models

- Here is the wage example from the textbook with age, year, and education as predictors
  - Natural splines used for year and age
  - Highly nonlinear age effect



James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). An introduction to statistical learning. New York: springer.

Generalized additive models

- One nice feature of GAMs is that we can still easily examine the impact of one covariate on the outcome
- We can easily test if a predictor is important
- Suppose that $\boldsymbol{\beta}_j$ represents the coefficients used to define $f_j(X_j)$
  - Some approaches don't have coefficients, such as local regression
  - Assume for now we are using splines or polynomial regression
- We can test the following hypothesis

$$H_0 : \boldsymbol{\beta}_j = \mathbf{0} \quad H_a : \boldsymbol{\beta}_j \neq \mathbf{0}$$

- If any coefficient in $\boldsymbol{\beta}_j$ is nonzero, the predictor is important

Generalized additive models

- The main drawback of GAMs is the additivity assumption
- When p is large, it is difficult to model $E(Y|X)$ without some level of additivity
- One relaxation of additivity is to include all second order functions

$$E(Y|X) = \beta_0 + \sum_{j=1}^{p} f_j(X_j) + \sum_{j<k} \sum_{k=2}^{p} f_{jk}(X_j, X_k)$$

- This still falls in the additive model framework and all of our approaches to flexible function estimation can be used

Generalized additive models

- This quickly leads to a large number of parameters
  - Depends on how flexibly these functions are estimated
- As in any other linear model framework we have seen, penalization can be used to improve estimation
- Suppose we can write our model in the following fashion

$$E(Y|\boldsymbol{X}) = \beta_0 + \sum_{j=1}^{p} \widetilde{\boldsymbol{X}}_j \boldsymbol{\beta}_j$$

where $\widetilde{\boldsymbol{X}}_j$ are basis functions corresponding to covariate j

Generalized additive models

- The group lasso has been proposed, which minimizes the following

$$\sum_{i=1}^{n}\left(Y_i - \beta_0 - \sum_{j=1}^{p}\widetilde{\boldsymbol{X}}_j\boldsymbol{\beta}_j\right)^2 + \lambda\sum_{j=1}^{p}||\boldsymbol{\beta}_j||_2$$

- Penalizes the magnitude of the groups of coefficients
  - Groups correspond to each individual covariate's parameters
- Remarkably, this leads to certain groups of parameters being zeroed out completely
  - Effectively performing variable selection

Generalized additive models

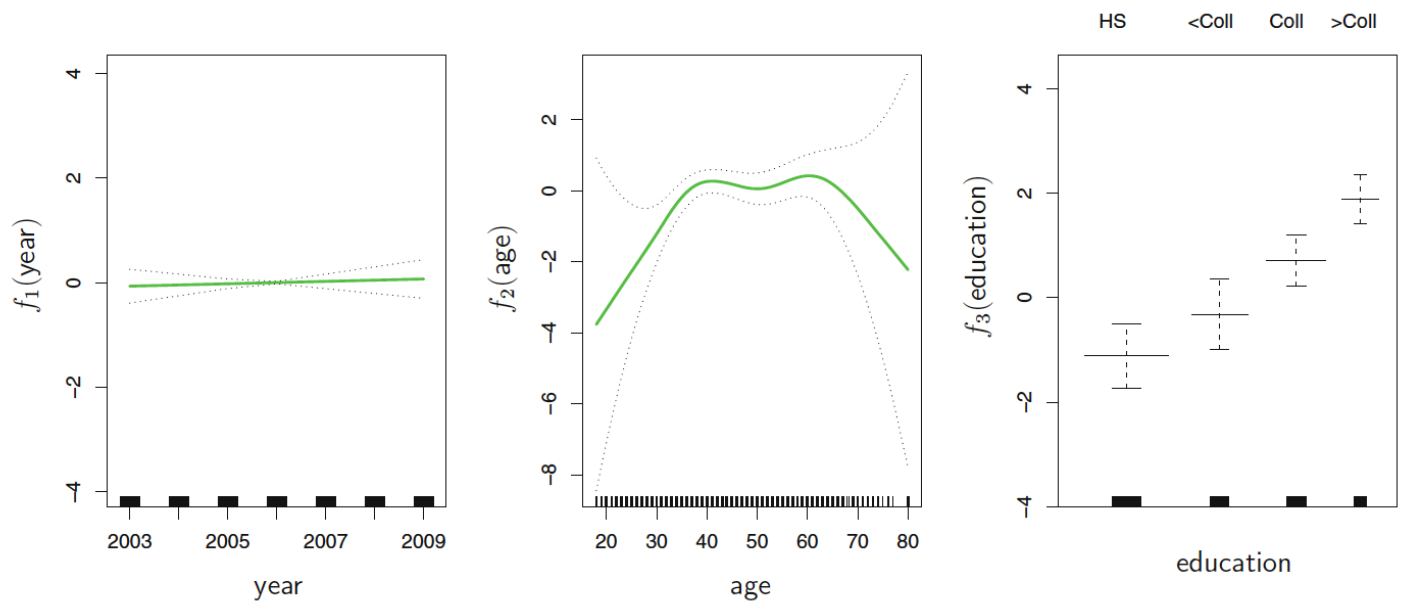
- GAMs directly extend to classification problems in the same manner
- Supposing Y is binary, we can model

$$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \beta_0 + \sum_{j=1}^{p} f_j(X_j)$$

- We can still use any of the same flexible or nonparametric approaches for estimating nonlinear $f_j(\cdot)$ functions

Generalized additive models

- Here is the wage example again from the textbook with age, year, and education as predictors
  - Now the outcome is a binary indicator of whether wage is above $250,000



James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). An introduction to statistical learning. New York: springer.