Advanced Computational Econometrics: Machine Learning

Chapter 4: Splines and Generalized Additive Models

Thierry Denœux

Spring 2023

Overview

Usefulness of linear models

- Linear models are widely used and very useful in practice. In particular, linear regression, linear discriminant analysis, logistic regression all rely on a linear model.
- In regression problems, $f(X) = \mathbb{E}(Y \mid X)$ will typically be nonlinear and nonadditive in X. However, representing $f(X)$ by a linear model is usually a convenient, and sometimes a necessary, approximation:
    - Convenient because a linear model is easy to interpret, and is the first-order Taylor approximation to $f(X)$.
    - Sometimes necessary, because with n small and/or p large, a linear model might be all we are able to fit to the data without overfitting.
- Likewise in classification, it is usually assumed that some monotone transformation of $\mathbb{P}(Y = k \mid X)$ is linear in X. This is inevitably an approximation.

Moving beyond linearity

- The core idea in this chapter is to augment/replace the vector of inputs X with additional variables, which are transformations of X, and then use linear models in this new space of derived input features.
- Denote by $h_m(X) : \mathbb{R}^p \to \mathbb{R}$ the m-th transformation of X, $m = 1, \ldots, M$. We then have the following model:

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X),$$

  a linear basis expansion in X.
- Once the basis functions h_m have been determined, the models are linear in these new variables, and the fitting proceeds as for linear models.

Popular choices for basis functions h_m

Some simple and widely used examples of the h_m are the following:

- $h_m(X) = X_m,\ m = 1, \ldots, p$ recovers the original linear model.
- $h_m(X) = X_j^2$ or $h_m(X) = X_j X_k$ allows us to augment the inputs with polynomial terms to achieve higher-order Taylor expansions. However, the number of variables grows exponentially in the degree of the polynomial. A full quadratic model in p variables requires $O(p^2)$ square and cross-product terms, or more generally $O(p^d)$ for a degree-d polynomial.

Popular choices for basis functions h_m (continued)

- $h_m(X) = \log(X_j)$, $\sqrt{X_j}$, ... permits other nonlinear transformations of single inputs. More generally one can use similar functions involving several inputs, such as $h_m(X) = \|X\|$.
- $h_m(X) = I(L_m \leq X_j < U_m)$, an indicator for a region of X_j. Breaking the range of X_j up into M_j such non-overlapping regions results in a model with a piecewise constant contribution for X_j.
- Remark: Sometimes the problem at hand will call for particular basis functions h_m, such as logarithms or power functions. More often, however, we use the basis expansions as a device to achieve more flexible representations for $f(X)$.

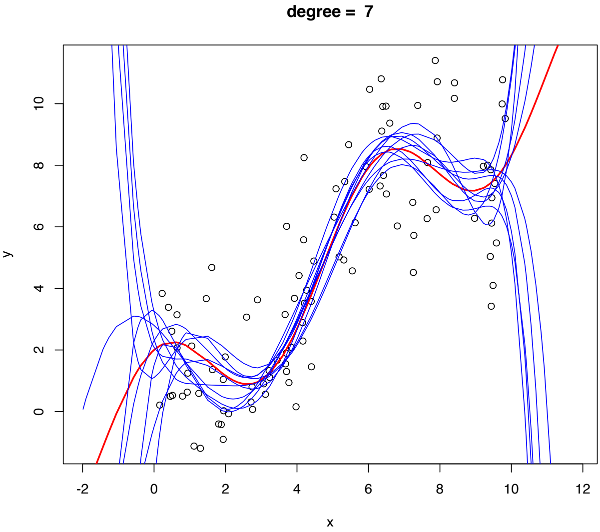
Overview

Overview

Fitting polynomials

- In most parts of this lecture, we assume $p = 1$.
- Create new variables $h_1(X) = X$, $h_2(X) = X^2$, $h_3(X) = X^3$, etc. and then do multiple linear regression on the transformed variables.
- We either fix the degree d at some reasonably low value, else use cross-validation to choose d.
- Polynomials are limited by their global nature – tuning the coefficients to achieve a functional form in one region can strongly influence the shape of the function in remote regions. As a consequence, polynomials have unpredictable tail behavior – very bad for extrapolation.

Example: fitting data with a polynomial with $d = 7$



Overview

Step Functions

- Another way of creating transformations of a variable is to cut the variable into distinct regions:

$$\begin{aligned} h_1(X) &= I(X < \xi_1), \\ h_2(X) &= I(\xi_1 \leq X < \xi_2), \\ &\vdots \\ h_M(X) &= I(X \geq \xi_{M-1}) \end{aligned}$$

- The RSS for the model $f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$ is

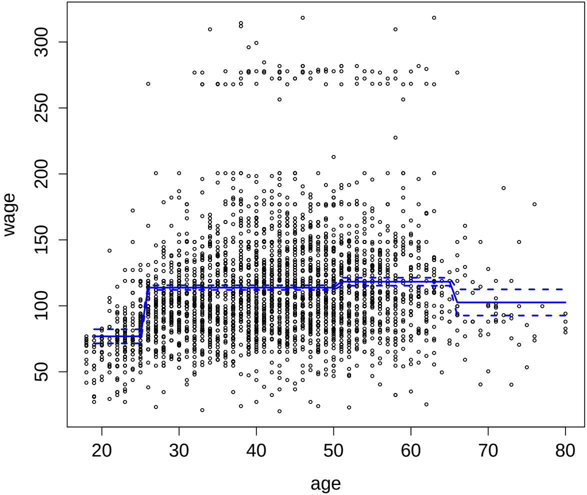
$$\text{RSS}(\beta) = \sum_{i=1}^{n} (f(x_i) - y_i)^2 = \sum_{m=1}^{M} \sum_{\{i: h_m(x_i)=1\}} (\beta_m - y_i)^2.$$

The LS estimates are $\widehat{\beta}_m = \overline{y}_m$, the means of Y in the m-th region.

Example in R

```
library("ISLR")

reg<-lm(wage ~ cut(age, c(18, 25, 50, 65, 90)),data=Wage)
ypred<-predict(reg,newdata=data.frame(age=18:80),interval="c")

plot(Wage$age,Wage$wage,cex=0.5,xlab="age",ylab="wage")
lines(18:80,ypred[,"fit"],lty=1,col="blue",lwd=2)
lines(18:80,ypred[,"lwr"],lty=2,col="blue",lwd=2)
lines(18:80,ypred[,"upr"],lty=2,col="blue",lwd=2)
```

Result



Step functions – continued

- Easy to work with. Creates a series of dummy variables representing each group.
- Useful way of creating interactions that are easy to interpret. For example, interaction effect of Year and Age:

$$I(Year < 2005) \cdot Age, \quad I(Year \geq 2005) \cdot Age$$

  would allow for different linear functions in each period.
- Choice of cutpoints or knots can be problematic. For creating nonlinearities, smoother alternatives such as splines are available.

Overview

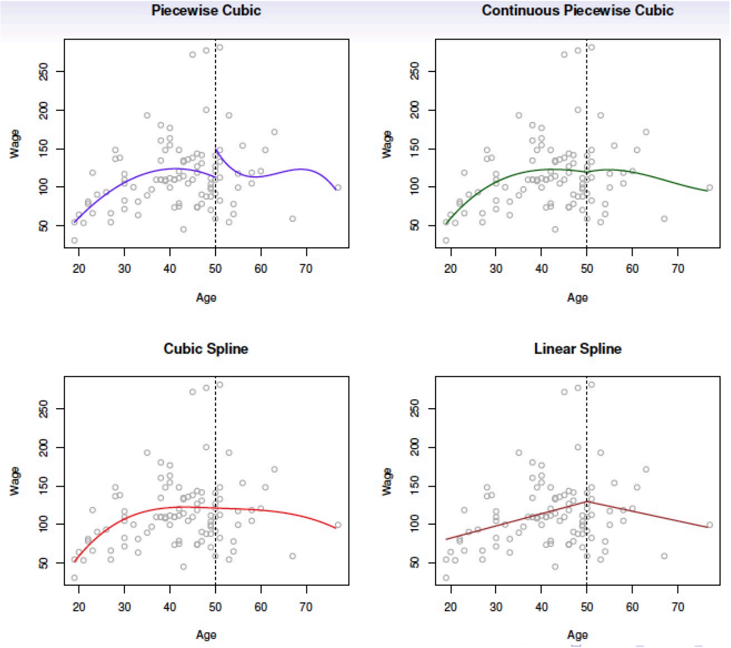
Overview

Piecewise Polynomials

- Instead of a single polynomial in X over its whole domain, we can rather use different polynomials in regions defined by knots. E.g. (see figure)

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

- Better to add constraints to the polynomials, e.g. continuity.
- Splines have the "maximum" amount of continuity.


Piecewise Cubic
Continuous Piecewise Cubic
Cubic Spline
Linear Spline
Wage
Age
50
100
150
200
250
20
30
40
50
60
70

Linear Splines

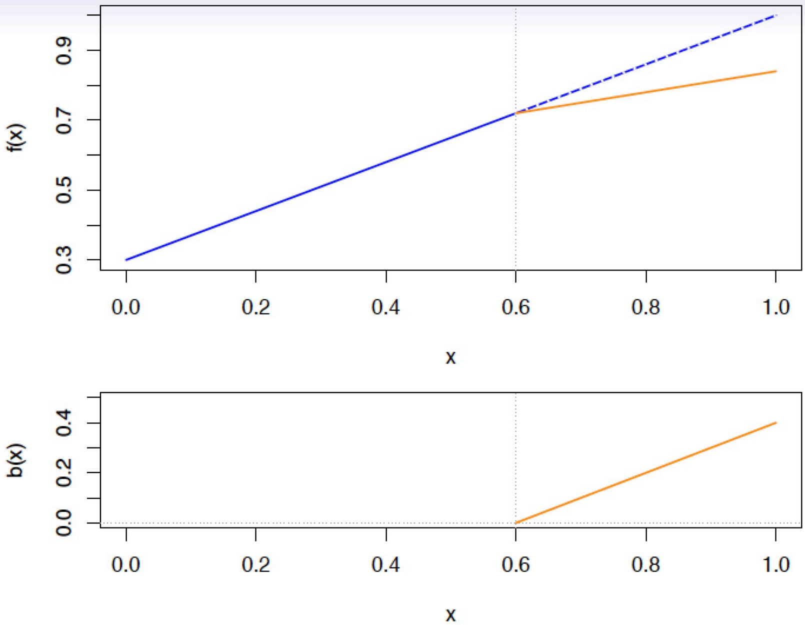
- A linear spline with knots at ξ_k, $k = 1, \ldots, K$ is a piecewise linear polynomial continuous at each knot.
- The set of linear splines with fixed knots is a vector space.
- The number of degrees of freedom is $2(K+1) - K = K + 2$. We can thus decompose linear splines on a basis of $K + 2$ basis functions,

$$y = \sum_{m=1}^{K+2} \beta_m h_m(x) + \epsilon.$$

- The basis functions can be chosen as

$$\begin{aligned} h_1(x) &= 1 \\ h_2(x) &= x \\ h_{k+2}(x) &= (x - \xi_k)_+, \quad k = 1, \ldots, K, \end{aligned}$$

where $(\cdot)_+$ denotes the positive part, i.e., $(x - \xi_k)_+ = x - \xi_k$ if $x > \xi_k$ and $(x - \xi_k)_+ = 0$ otherwise.


f(x)
0.3
0.5
0.7
0.9
0.0
0.2
0.4
0.6
0.8
1.0
x
b(x)
0.0
0.2
0.4
0.0
0.2
0.4
0.6
0.8
1.0
x

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.
- Enforcing one more order of continuity would lead to a global cubic polynomial.
- Again, the set of cubic splines with fixed knots is a vector space, and the number of degrees of freedom is $4(K+1) - 3K = K + 4$. We can thus decompose cubic splines on a basis of $K + 4$ basis functions,

$$y = \sum_{m=1}^{K+4} \beta_m h_m(x) + \epsilon.$$

- We can choose truncated power basis functions,

$$\begin{aligned} h_k(x) &= x^{k-1}, \quad k = 1, \ldots, 4, \\ h_{k+4}(x) &= (x - \xi_k)_+^3, \quad k = 1, \ldots, K. \end{aligned}$$

Order-M splines

- More generally, an order-M spline with knots ξ_k, $k = 1, \ldots, K$ is a piecewise-polynomial of degree $M-1$, which has continuous derivatives up to order $M-2$.
- A cubic spline has $M = 4$. A piecewise-constant function is an order-1 spline, while a continuous piecewise linear function is an order-2 spline.
- The general form for the truncated-power basis set is

$$
\begin{aligned}
h_k(x) &= x^{k-1}, \quad k = 1, \ldots, M, \\
h_{k+M}(x) &= (x - \xi_k)_+^{M-1}, \quad k = 1, \ldots, K.
\end{aligned}
$$

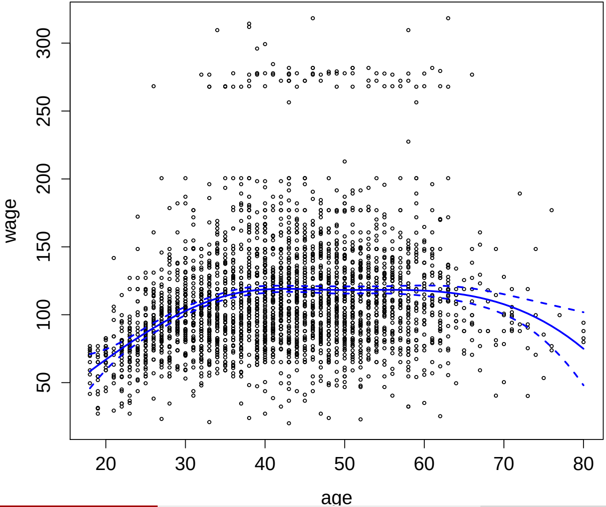
- In practice the most widely used orders are $M = 1, 2$ and 4. There is seldom any good reason to go beyond cubic-splines.

Splines in R

```
library('splines')
fit<-lm(wage~bs(age,df=5),data=Wage)

ypred<-predict(fit,newdata=data.frame(age=18:80),interval="c")

plot(Wage$age,Wage$wage,cex=0.5,xlab="age",ylab="wage")
lines(18:80,ypred[,"fit"],lty=1,col="blue",lwd=2)
lines(18:80,ypred[,"lwr"],lty=2,col="blue",lwd=2)
lines(18:80,ypred[,"upr"],lty=2,col="blue",lwd=2)
```

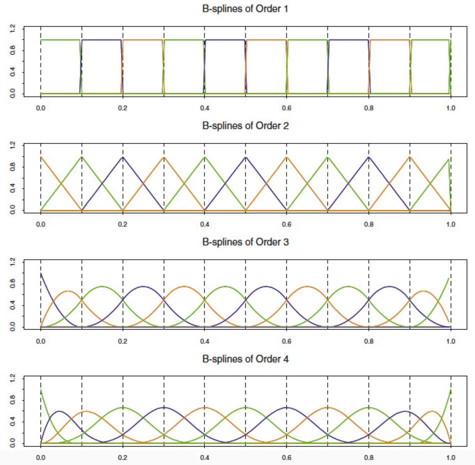
- By default, `degree=3` (cubic splines), and the intercept is not included in the basis functions. (It is added by function `lm`.)
- The number of knots, if not specified, is `df-degree`; the knots are then placed at quantiles.
- The actual number of degrees of freedom is `df`+1 (taking into account the intercept).

Result



B-splines

- Since the space of spline functions of a particular order and knot sequence is a vector space, there are many equivalent bases for representing them (just as there are for ordinary polynomials.)
- While the truncated power basis is conceptually simple, it is not too attractive numerically: powers of large numbers can lead to severe rounding problems.
- In practice, we often use another basis: the B-spline basis, which allows for efficient computations even when the number of knots K is large (each basis function has a local support).



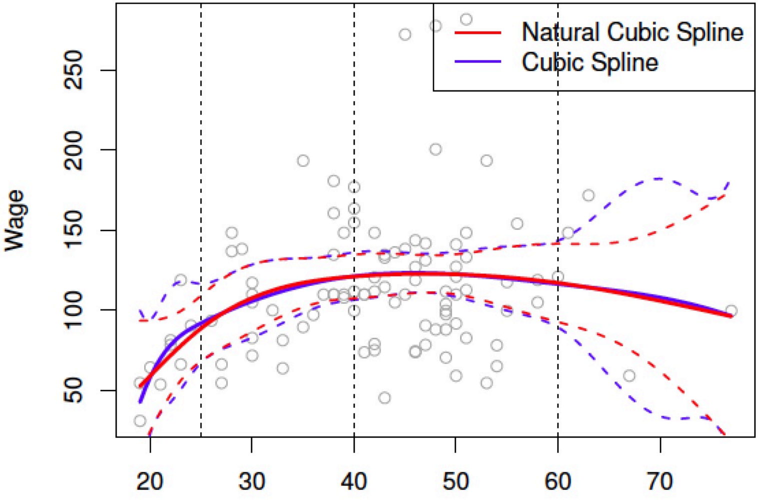
Sequence of B-splines up to order 4 with 9 knots evenly spaced from 0 to 1. The B-splines have local support; they are nonzero on an interval spanned by $M + 1$ knots.

Overview

Natural cubic spline

- We know that the behavior of polynomials fit to data tends to be erratic near the boundaries, and extrapolation can be dangerous. These problems still exist with splines.
- A natural cubic spline adds additional constraints, namely that the function is linear beyond the boundary knots.
- This frees up four degrees of freedom (two constraints each in both boundary regions), which can be spent more profitably by putting more knots in the interior region.
- There will be a price paid in bias near the boundaries, but assuming the function is linear near the boundaries (where we have less information anyway) is often considered reasonable.

Example


Natural Cubic Spline
Cubic Spline
Wage
50
100
150
200
250
20
30
40
50
60
70

Natural cubic spline basis

- A natural cubic spline with K knots has K degrees of freedom: it can be represented by K basis functions.
- One can start from a basis for cubic splines, and derive the reduced basis by imposing the boundary constraints.
- For example, starting from the truncated power series basis, we can show that we arrive at

$$N_1(X) = 1, \quad N_2(X) = X,$$

$$N_{k+2}(X) = d_k(X) - d_{K-1}(X), \quad k = 1, \ldots, K-2$$

with

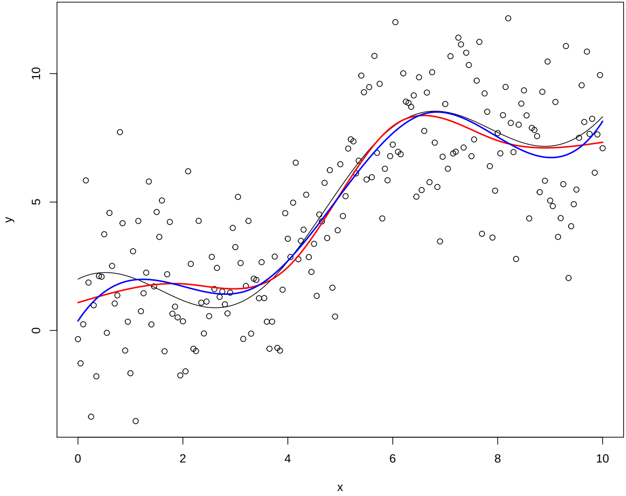
$$d_k(X) = \frac{(X - \xi_k)_+^3 - (X - \xi_K)_+^3}{\xi_K - \xi_k}$$

(Sketch of proof in appendix).

Example in R

```
fit1<-lm(y ~ ns(x,df=5))
fit2<-lm(y ~ bs(x,df=5))

ypred1<-predict(fit1,newdata=data.frame(x=xtest),interval="c")
ypred2<-predict(fit2,newdata=data.frame(x=xtest),interval="c")

plot(x,y,xlim=range(xtest))
lines(xtest,ftest)
lines(xtest,ypred1[,"fit"],lty=1,col="red",lwd=2)
lines(xtest,ypred2[,"fit"],lty=1,col="blue",lwd=2)
```

Result



Using splines with logistic regression

- Until now, we have discussed regression problems. However, splines can also be used for classification.
- Consider, for instance, natural splines with K knots. For binary classification, we can fit the binomial logistic regression model,

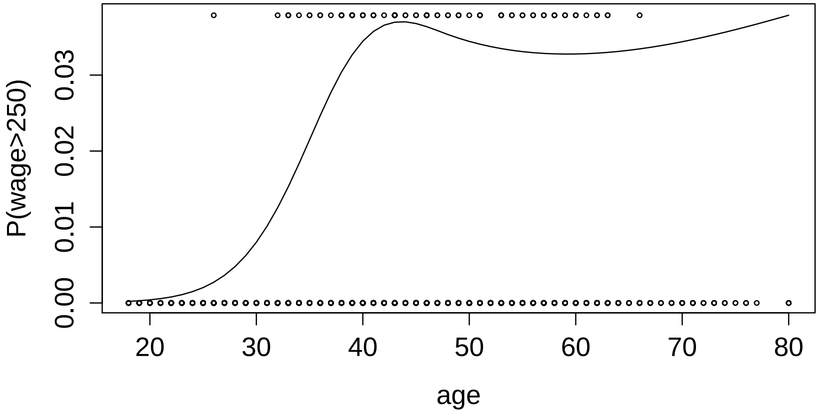
$$\log \frac{\mathbb{P}(Y=1 \mid X=x)}{\mathbb{P}(Y=0 \mid X=x)} = f(x)$$

  with $f(x) = \sum_{k=1}^{K} \beta_k N_k(x)$.
- Once the basis functions have been defined, we just need to estimate coefficients β_k using a standard logistic regression procedure.
- A smooth estimate of the conditional probability $\mathbb{P}(Y=1 \mid x)$ can then be used for classification or risk scoring.

Example in R

```
class<-glm(I(wage>250) ~ ns(age,3),data=Wage,family='binomial')
proba<-predict(class,newdata=data.frame(age=18:80),type='response')

plot(18:80,proba,xlab="age",ylab="P(wage>250)",type="l")
ii<-which(Wage$wage>250)
points(Wage$age[ii],rep(max(proba),length(ii)),cex=0.5)
points(Wage$age[-ii],rep(0,nrow(Wage)-length(ii)),cex=0.5)
```

Result



Overview

Problem formulation

- Here we discuss a spline basis method that avoids the knot selection problem completely by using a maximal set of knots. The complexity of the fit is controlled by regularization.
- Problem: among all functions $f(x)$ with two continuous derivatives, find one that minimizes the penalized residual sum of squares

$$RSS(f,\lambda) = \sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda \int [f''(t)]^2 dt,$$

where λ is a fixed smoothing parameter.

- The first term measures closeness to the data, while the second term penalizes curvature in the function, and λ establishes a tradeoff between the two. Special cases: $\lambda = 0$ (no constraint on f) and $\lambda = \infty$ (f has to be linear).

Solution

- It can be shown that this problem has an explicit, finite-dimensional, unique minimizer which is a natural cubic spline with knots at the unique values of the x_i, $i = 1, \ldots, n$.
- Although there are as many as n knots, the penalty term decreases the number of degrees of freedom by shrinking the spline coefficients toward the linear fit.
- The solution is thus of the form

$$f(x) = \sum_{j=1}^{n} N_j(x)\theta_j,$$

where the $N_j(x)$ are an n-dimensional set of basis functions for representing this family of natural splines.

Computation

- The criterion can be written as

$$RSS(\theta, \lambda) = (\mathbf{y} - \mathbf{N}\theta)^T(\mathbf{y} - \mathbf{N}\theta) + \lambda\theta^T\boldsymbol{\Omega}_n\theta,$$

  where $\mathbf{N}_{ij} = N_j(x_i)$ and $(\boldsymbol{\Omega}_n)_{jk} = \int N_j''(t)N_k''(t)dt$.
- The solution is

$$\widehat{\theta} = (\mathbf{N}^T\mathbf{N} + \lambda\boldsymbol{\Omega}_n)^{-1}\mathbf{N}^T\mathbf{y},$$

  a generalized ridge regression.
- The fitted smoothing spline is given by

$$\widehat{f}(x) = \sum_{j=1}^{n} N_j(x)\widehat{\theta}_j.$$

- In practice, when n is large, we can use only a subset of the n interior knots (rule of thumb: number of knots proportional to $\log n$).

Degrees of freedom

- Denote by $\widehat{\mathbf{f}}$ the n-vector of fitted values $\widehat{f}(x_i)$ at the training predictors x_i. Then,

$$\widehat{\mathbf{f}} = \mathbf{N}\widehat{\theta} = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\boldsymbol{\Omega}_n)^{-1}\mathbf{N}^T\mathbf{y} = \mathbf{S}_\lambda\mathbf{y}$$

- As matrix $\mathbf{S}_\lambda$ does not depend on $\mathbf{y}$, the smoothing spline is a linear smoother.
- As in ridge regression, we define the effective degrees of freedom of a smoothing spline as

$$\mathrm{df}_\lambda = \mathrm{tr}(\mathbf{S}_\lambda)$$

Selection of smoothing parameters

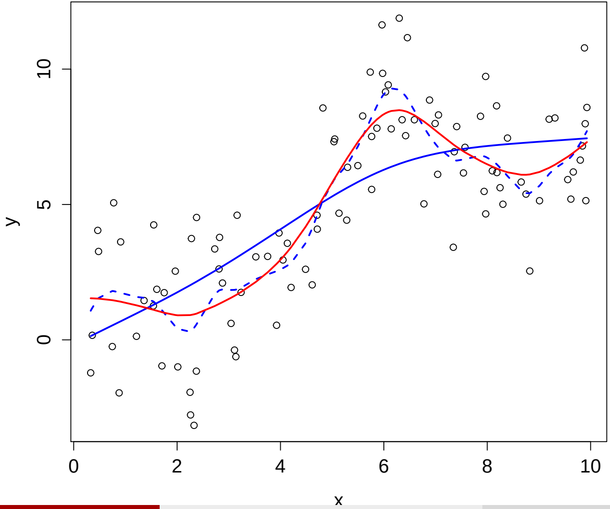
- As $\lambda \to 0$, $\mathrm{df}_\lambda \to n$ and $\mathbf{S}_\lambda \to \mathbf{I}$. As $\lambda \to \infty$, $\mathrm{df}_\lambda \to 2$ and $\mathbf{S}_\lambda \to \mathbf{H}$, the hat matrix for linear regression on $\mathbf{x}$.
- Since df_λ is monotone in λ, we can invert the relationship and specify λ by fixing df_λ (this can be achieved by simple numerical methods). Using df in this way provides a uniform approach to compare many different smoothing methods.
- The leave-one-out (LOO) cross-validated error is given by

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

Smoothing splines in R

```
ss1<-smooth.spline(x,y,df=3)
ss2<-smooth.spline(x,y,df=15)
ss<-smooth.spline(x,y)

plot(x,y)
lines(x,ss1$y,col="blue",lwd=2)
lines(x,ss2$y,col="blue",lwd=2,lty=2)
lines(x,ss$y,col="red",lwd=2)

> ss$df
7.459728
```

Result



Application to logistic regression

- The smoothing spline problem has been posed in a regression setting, but it is typically easy to transfer this technology to other domains.
- Here we consider logistic regression with a single quantitative input X. The model is

$$\log \frac{\mathbb{P}(Y = 1 \mid X = x)}{\mathbb{P}(Y = 0 \mid X = x)} = f(x),$$

which implies

$$\mathbb{P}(Y = 1 \mid X = x) = \frac{e^{f(x)}}{1 + e^{f(x)}} = P(x).$$

Penalized log-likelihood

- We construct the penalized log-likelihood criterion

$$\ell(f;\lambda) = \sum_{i=1}^{n} [y_i \log P(x_i) + (1-y_i)\log(1-P(x_i))] - \frac{1}{2}\lambda \int \{f''(t)\}^2 dt$$
$$= \sum_{i=1}^{n} [y_i f(x_i) - \log(1+e^{f(x)})] - \frac{1}{2}\lambda \int \{f''(t)\}^2 dt$$

- As before, the optimal f is a finite-dimensional natural spline with knots at the unique values of x. We can represent f as

$$f(x) = \sum_{j=1}^{n} N_j(x)\theta_j.$$

Optimization

- We compute the first and second derivatives

$$\frac{\partial \ell(\theta)}{\partial \theta} = \mathbf{N}^T(\mathbf{y}-\mathbf{p}) - \lambda \boldsymbol{\Omega}_n \theta$$

$$\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta^T} = -\mathbf{N}^T\mathbf{W}\mathbf{N} - \lambda \boldsymbol{\Omega}_n,$$

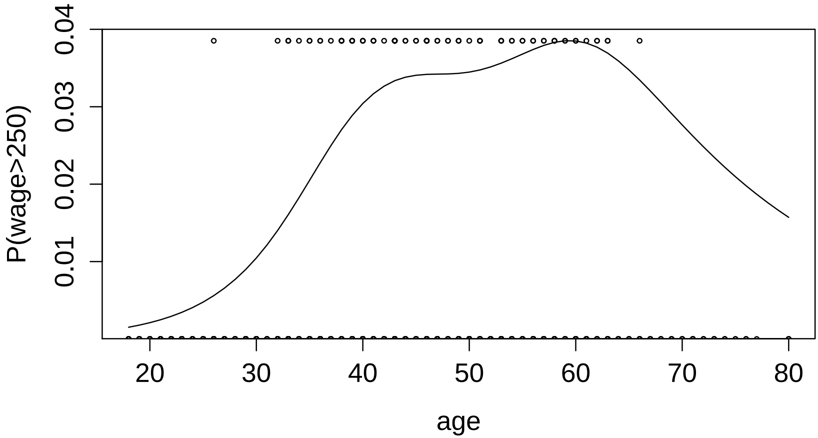
  where $\mathbf{p}$ is the n-vector with elements $P(x_i)$, and $\mathbf{W}$ is a diagonal matrix of weights $P(x_i)(1-P(x_i))$.
- Parameters θ_j can be estimated using the Newton method,

$$\theta^{new} = \theta^{old} - \left(\frac{\partial^2 \ell(\theta^{old})}{\partial \theta \partial \theta^T} \right)^{-1} \frac{\partial \ell(\theta^{old})}{\partial \theta}$$

Nonparametric logistic regression in R

```
library(gam)
class<-gam(I(wage>250) ~ s(age,df=3),data=Wage,family='binomial')
proba<-predict(class,newdata=data.frame(age=18:80),type='response')

plot(18:80,proba,xlab="age",ylab="P(wage>250)",type="l")
ii<-which(Wage$wage>250)
points(Wage$age[ii],rep(max(proba),length(ii)),cex=0.5)
points(Wage$age[-ii],rep(0,nrow(Wage)-length(ii)),cex=0.5)
```

Result



Overview

Multidimensional extension

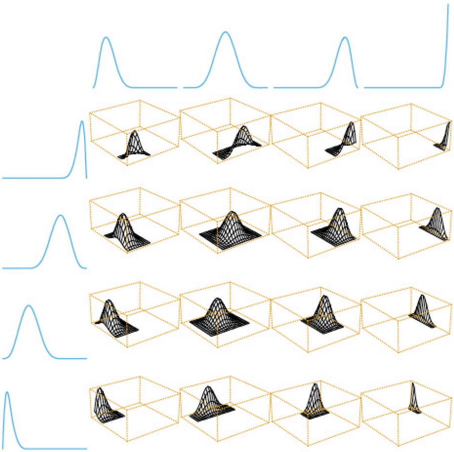
- So far we have focused on one-dimensional spline models. Each of these approaches have multidimensional analogs.
- Suppose $X \in \mathbb{R}^2$, and we have a basis of functions $h_{1k}(X_1)$, $k = 1, \ldots, M_1$ for representing functions of coordinate X_1, and likewise a set of M_2 functions $h_{2k}(X_2)$ for coordinate X_2.
- Then the $M_1 \times M_2$ dimensional tensor product basis defined by

$$g_{jk}(X) = h_{1j}(X_1)h_{2k}(X_2), \quad j = 1, \ldots, M_1, \; k = 1, \ldots, M_2$$

can be used for representing a two-dimensional function

$$g(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{jk} g_{jk}(X)$$

- The coefficients can be fit by least squares, as before.

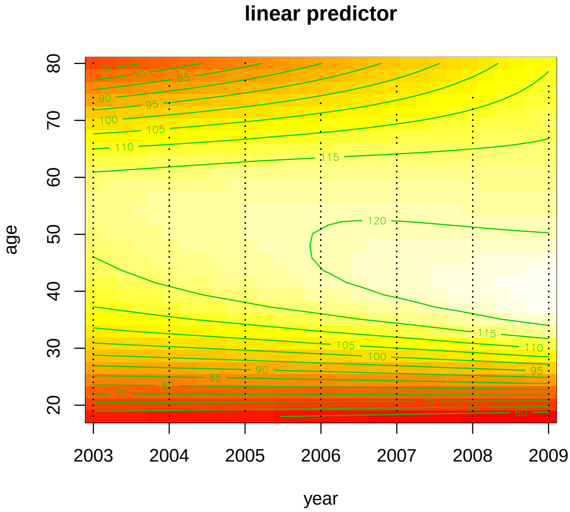


A tensor product basis of B-splines, showing some selected pairs. Each two-dimensional function is the tensor product of the corresponding one dimensional marginals.

Tensor product splines in R

```
library(mgcv)

# k=5 basis functions along each dimension
# (cubic regression splines by default)
fit1<-gam(wage~ te(year,age,k=5),data=Wage)

vis.gam(fit1,plot.type = "contour",color="heat")
points(Wage$year,Wage$age,pch=".")
```

Result


linear predictor
age
year
80
70
60
50
40
30
20
2003
2004
2005
2006
2007
2008
2009
75
80
85
90
95
100
105
110
115
120
115
110
105
100
95
90
85
80
75
70
65
60

Curse of dimensionality

- The tensor product approach can be generalized to p dimensions, but the dimension of the basis grows exponentially fast – yet another manifestation of the curse of dimensionality.
- A more parsimonious approach will be described in the next section.

Multidimensional smoothing splines

- One-dimensional smoothing splines (via regularization) generalize to higher dimensions as well.
- Suppose we have pairs (x_i, y_i) with $x_i \in \mathbb{R}^p$, and we seek a p-dimensional regression function $f(x)$. The idea is to set up the problem

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

  where J is an appropriate penalty functional for stabilizing a function f in $\mathbb{R}^p$.

Thin-plate splines

- A natural choice for J when $p = 2$ is

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

- Optimizing the cost function with this penalty leads to a smooth two-dimensional surface, known as a thin-plate spline.
- It shares many properties with the one-dimensional cubic smoothing spline:
  - As $\lambda \to 0$, the solution approaches an interpolating function (the one with smallest penalty)
  - As $\lambda \to \infty$, the solution approaches the least squares plane
  - For intermediate values of λ, the solution can be represented as a linear expansion of basis functions, whose coefficients are obtained by a form of generalized ridge regression.

Radial basis functions

- The general solution has the form

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

  where $h_j(x) = \|x - x_j\|^2 \log \|x - x_j\|$.
- These h_j are examples of radial basis functions, which will be discussed later.
- The coefficients are found by plugging the expression of $f(x)$ in the cost function, which reduces to a finite-dimensional penalized least squares problem.

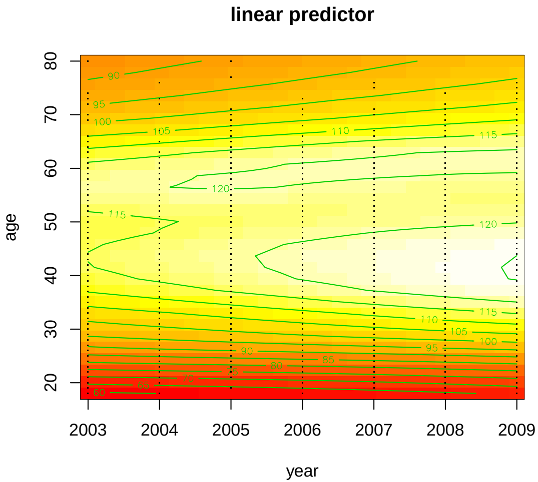
High-dimensional extension

- Thin-plate splines are defined more generally for arbitrary dimension p, for which an appropriately more general J is used.
- The computational complexity for thin-plate splines is $O(n^3)$. However, as with univariate smoothing splines, we can get away with substantially less than the n knots.
- In practice, it is usually sufficient to work with a lattice of knots covering the domain.
- With K knots, the complexity is reduced to $O(nK^2 + K^3)$.

Thin plate splines in R

```
library(mgcv)

# The smoothing coefficient is automatically determines by CV
fit2<-gam(wage~ s(year,age),data=Wage)

vis.gam(fit2,plot.type = "contour",color="heat")
points(Wage$year,Wage$age,pch=".")
```

Result


linear predictor
age
year
2003
2004
2005
2006
2007
2008
2009
20
30
40
50
60
70
80
90
95
100
105
110
115
120
60
65
70
80
85

Overview

Overview

Motivation

- In general, extending the linear basis function approach to learning problems with a large number p of inputs poses two main problems:
  1. Curse of dimensionality: we have seen that, with the tensor product spline basis with M basis functions in each dimension, we have M^p dimensions; for instance, with $M = 5$ and $p = 10$, we have almost $5^{10} = 9,765,625$ parameters to estimate.
  2. Poor interpretability: for $p > 2$ it becomes impossible to understand the effect of each input variable on the response variable.
- In this section, we study flexible, yet interpretable models based on the assumption that the effects of each input variables are additive: in this way, we replace the problem of estimating a p-dimensional function by that of simultaneously estimating p one-dimensional functions.
- These methods are called generalized additive models (GAMs).

GAM for regression

- In the regression setting, a generalized additive model has the form

$$\mathbb{E}(Y \mid X_1, X_2, \ldots, X_p) = \alpha + f_1(X_1) + f_2(X_2) + \ldots + f_p(X_p)$$

- As usual $X_1, X_2, \ldots, X_p$ represent predictors and Y is the outcome.
- The f_j's are unspecified smooth (nonparametric) functions.

GAM for binary classification

- For two-class classification, recall the logistic regression model for binary data discussed previously:

$$\log \frac{P(X)}{1 - P(X)} = \alpha + \beta_1 X_1 + \ldots + \beta_p X_p,$$

with $P(X) = \mathbb{P}(Y = 1 \mid X)$.

- The additive logistic regression model replaces each linear term by a more general functional form

$$\log \frac{P(X)}{1 - P(X)} = \alpha + f_1(X_1) + \ldots + f_p(X_p)$$

where again each f_j is an unspecified smooth function.

- While the nonparametric form for the functions f_j makes the model more flexible, the additivity is retained and allows us to interpret the model in much the same way as before.

Mixing linear and nonlinear effects

- We can easily mix in linear and other parametric forms with the nonlinear terms, a necessity when some of the inputs are qualitative variables (factors).
- For instance, we can have a regression model of the form

$$Y = X^T\beta + \sum_k \alpha_k I(V = k) + f(Z) + \epsilon.$$

This is a semiparametric model, where

  - X is a vector of predictors to be modeled linearly,
  - α_k the effect for the k-th level of a qualitative input V, and
  - the effect of predictor Z is modeled nonparametrically.

Overview

GAMs with natural splines

- If we model each function f_j as a natural spline, then we can fit the resulting model using simple least square (regression) or likelihood maximization algorithm (classification).
- For instance, with natural cubic splines, we have the following GAM:

$$Y = \sum_{j=1}^{p} \underbrace{\sum_{k=1}^{K(j)} \beta_{jk} N_{jk}(X_j)}_{f_j(X_j)} + \epsilon,$$

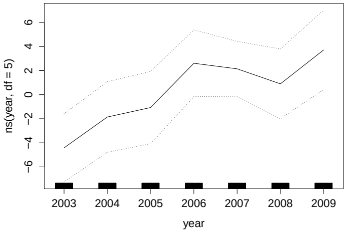
where $K(j)$ is the number of knots for variable j.

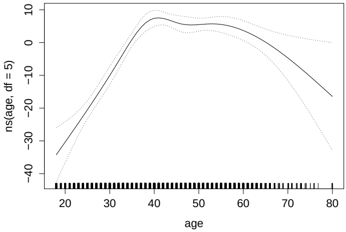
- Remark: if $K(j) = K$ is constant for all j, the number of basis functions remains linear in p.

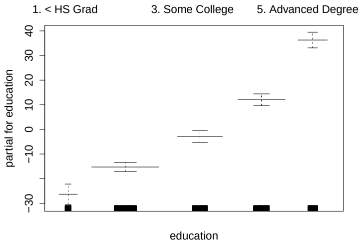
Example in R

```
library("ISLR") # For the Wage data
library("splines")

fit1<-lm(wage ~ ns(year,df=5)+ns(age,df=5)+education,data=Wage)

library("gam")
fit2<-gam(wage ~ ns(year,df=5)+ns(age,df=5)+education,data=Wage)
plot(fit2,se=TRUE)
```

Result


ns(year, df = 5)
−6
−4
−2
0
2
4
6
2003
2004
2005
2006
2007
2008
2009
year


ns(age, df = 5)
−40
−30
−20
−10
0
10
20
30
40
50
60
70
80
age


1. < HS Grad
3. Some College
5. Advanced Degree
partial for education
−30
−10
0
10
20
30
40
education

GAMs with smoothing splines

- Consider an additive model of the form

$$Y = \alpha + f_1(X_1) + f_2(X_2) + \ldots + f_p(X_p) + \epsilon,$$

where the error term ϵ has mean zero.

- We can specify a penalized sum of squares for this problem,

$$SS(\alpha, f_1, \ldots, f_p) = \sum_{i=1}^{n} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int f_j''(t_j)^2 dt_j,$$

where the $\lambda_j \geq 0$ are tuning parameters.

- It can be shown that the minimizer of SS is an additive cubic spline model: each of the functions f_j is a cubic spline in the component X_j, with knots at each of the unique values of $x_{ij}, i = 1, \ldots, n$.

Unicity of the solution

- Without further restrictions on the model, the solution is not unique.
- The constant α is not identifiable, since we can add or subtract any constants to each of the functions f_j, and adjust α accordingly.
- The standard convention is to assume that $\sum_{i=1}^{n} f_j(x_{ij}) = 0$ for all j – the functions average zero over the data. It is easily seen that $\widehat{\alpha} = \overline{y}$ in this case.
- If in addition to this restriction, the matrix of input values (having ij-th entry x_{ij}) has full column rank, then SS is a strictly convex criterion and the minimizer is unique.
- A simple iterative procedure exists for finding the solution: the backfitting algorithm

Backfitting algorithm

- We set $\widehat{\alpha} = \overline{y}$, and it never changes.
- We fit a cubic smoothing spline $\mathcal{S}_j$ to the targets $\left\{y_i - \widehat{\alpha} - \sum_{k \neq j} \widehat{f}_k(x_{ik})\right\}_{i=1}^{n}$, as a function of x_{ij} to obtain a new estimate $\widehat{f}_j$.
- This is done for each predictor in turn, using the current estimates of the other functions $\widehat{f}_k$ when computing $y - \widehat{\alpha} - \sum_{k \neq j} \widehat{f}_k(x_{ik})$.
- The process is continued until the estimates $\widehat{f}_j$ stabilize.
- This procedure (known as backfitting) is a grouped cyclic coordinate descent algorithm.

Backfitting algorithm

1. Initialize: $\widehat{\alpha} = \overline{y}$, $\widehat{f}_j = 0$, $\forall j$.
2. Cycle: $j = 1, 2, \ldots, p, 1, 2, \ldots, p, \ldots,$

$$\widehat{f}_j \leftarrow \mathcal{S}_j \left[\{y_i - \widehat{\alpha} - \sum_{k \neq j} \widehat{f}_k(x_{ik})\}_{i=1}^{n} \right]$$

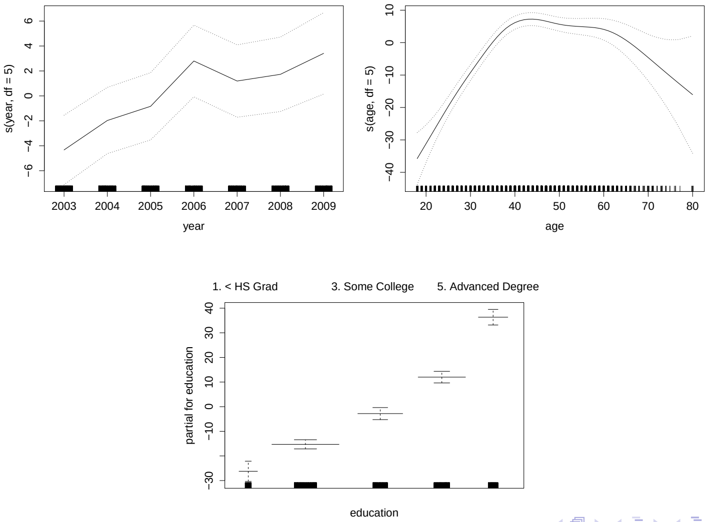
$$\widehat{f}_j \leftarrow \widehat{f}_j - \frac{1}{n} \sum_{i=1}^{n} \widehat{f}_j(x_{ij})$$

until the functions $\widehat{f}_j$ change less than a prespecified threshold.

Example in R

```
library("gam")

fit3<-gam(wage ~ s(year,df=5)+s(age,df=5)+education,data=Wage)
plot(fit3,se=TRUE)
```

Result


s(year, df = 5)
−6 −4 −2 0 2 4 6
2003 2004 2005 2006 2007 2008 2009
year
s(age, df = 5)
−40 −30 −20 −10 0 10
20 30 40 50 60 70 80
age
1. < HS Grad
3. Some College
5. Advanced Degree
partial for education
−30 −10 0 10 20 30 40
education

Natural cubic spline bases I

- From

$$f(X) = \sum_{j=0}^{3} \beta_j X^j + \sum_{k=1}^{K} \theta_k (X - \xi_k)_+^3 \qquad (1)$$

and the boundary conditions $f''(X) = 0$ and $f^{(3)}(X) = 0$ for $X < \xi_1$ and $X > \xi_K$, we get

$$\beta_2 = \beta_3 = 0, \quad \sum_{k=1}^{K} \theta_k = 0, \quad \sum_{k=1}^{K} \xi_k \theta_k = 0$$

- This gives us the following relations between the coefficients:

$$\theta_K = -\sum_{k=1}^{K-1} \theta_k \quad \text{and} \quad \theta_{K-1} = -\sum_{k=1}^{K-2} \theta_k \frac{\xi_K - \xi_k}{\xi_K - \xi_{K-1}} \qquad (2)$$

Natural cubic spline bases II

- Using the first relation in (1) we get

$$f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K-1} \theta_k \underbrace{\left[(X-\xi_k)_+^3 - (X-\xi_K)_+^3\right]}_{(\xi_K-\xi_k)d_k(X)}.$$

- Using the second relation, we get

$$f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta_k(\xi_K - \xi_k) \underbrace{\left[d_k(X) - d_{K-1}(X)\right]}_{N_{k+2}(X)}$$

- Denoting $\theta_k(\xi_K - \xi_k)$ as θ'_k, we finally get

$$f(X) = \beta_0 + \beta_1 X + \sum_{k=1}^{K-2} \theta'_k N_{k+2}(X). \quad \text{QED}$$

Back