STAT 542: Statistical Learning

Splines

Ruoqing Zhu, Ph.D. <rqzhu@illinois.edu>

Course Website: https://teazrq.github.io/stat542/

February 20, 2022

Department of Statistics
University of Illinois at Urbana-Champaign

Outline

- From Linear to Nonlinear Methods
- Piecewise Polynomials and Splines
- Smoothing Splines

Linear vs. Nonlinear Models

- For most of our lectures up to now, we focused on linear models. Why?
  - Convenient and easy to fit
  - Easy to interpret
  - A relatively good approximation to the underlying truth
  - When n is small and/or p is large, linear models tend not to overfit
- Nonlinear models are more flexible and may lead to better fitting to reduce bias
- The concept in this lecture is mainly about nonlinear model fitting of a univariate function.

Linear vs. Nonlinear Models

- Univariate functions has many applications. They can be assembled to approximate multivariate functions.
- Example: Additive Model assumes that a model has the form

$$f(x) = \sum_{j=1}^{p} f_j(x_j)$$

- This allows some more flexibility since f_j does not need to be $\beta_j x_j$ — a linear function of x_j
- For the most part in this lecture, we will focus on how to estimate the functions f_j's, which are univariate functions of x_j's.

Linear vs. Nonlinear Models

- In particular, we consider a linear basis expansion of f_j, i.e.,

$$f_j(x) = \sum_{m=1}^{M_j} \beta_{jm} h_{mj}(x_j)$$

- h_{mj} are called the basis functions
- h_{mj} could be different for each covariate x_j.
- For simplicity, since we only deal with one covariate, we drop the index j, and focus on

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

Linear vs. Nonlinear Models

- Once we have determined the basis functions h_m, the model is again linear (just not in the original covariates)
- Some typical choices of h
    - $h_m(x) = x$: the original linear model
    - $h_m(x) = x^2, x^3, \ldots$: polynomials
    - $h_m(x) = \log(x), \sqrt{x}, \ldots$: other nonlinear transformations
    - $h_m(x) = \mathbf{1}\{L_m < x < U_m\}$: indicator for a region of X

Linear vs. Nonlinear Models

- The approach is straight forward:
  - Find a collection of M basis functions (we will introduce several choices), and calculate the $h_m(x_i)$ values of each subject i on these basis.
  - Then, for each observation i, treat $\big(h_1(x_i), h_2(x_i), \ldots, h_M(x_i)\big)^\mathsf{T}$ as the observed covariate of x_i
  - This allows us to construct a new design matrix with dimension $n \times M$.
  - We then fit a linear regression based on these M variables

Piecewise Polynomials and Splines

Piecewise Polynomials

- For example, consider the piecewise constant:

$$h_1(x) = \mathbf{1}\{x < \xi_1\}, \quad h_2(x) = \mathbf{1}\{\xi_1 \leq x < \xi_2\}, \quad h_3(x) = \mathbf{1}\{\xi_2 \leq x\}$$

- ξ_1 and ξ_2 are called knots
- Hence the model becomes

$$f(x) = \sum_{i=1}^{3} \beta_m h_m(x)$$

- This is essentially fitting a constant function at each region, so $\beta_m = \overline{Y}_m$, where $\overline{Y}_m$ is just the mean of the mth region.
- This is similar to a regression tree model (introduced later).

Piecewise Polynomials

- We can also fit a linear function at each region by considering three additional basis functions:

$$h_4(x) = x\mathbf{1}\{x < \xi_1\}$$
$$h_5(x) = x\mathbf{1}\{\xi_1 \leq x < \xi_2\}$$
$$h_6(x) = x\mathbf{1}\{\xi_2 \leq x\}$$

- This leads to piecewise linear models

Piecewise Polynomials

Example: birthrate data, 1917 to 2003

Continuous Piecewise Polynomials

- However, the fitted functions are not continuous.
- We might want some restrictions on the parameter estimates to force continuity.
- For example, a continuous piecewise linear function requires

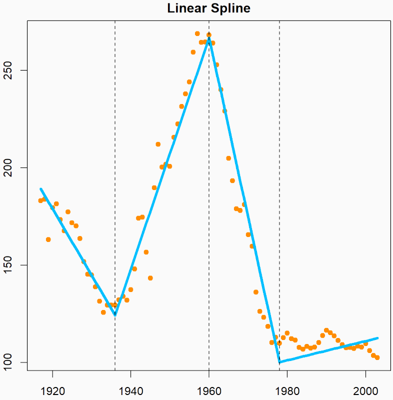
$$f(\xi_k^-) = f(\xi_k^+)$$

  at all knots ξ_k.

- For our previous example, this implies

$$\begin{aligned} & \beta_1 + \xi_1\beta_4 = \beta_2 + \xi_1\beta_5 \\ \text{and} \quad & \beta_2 + \xi_2\beta_5 = \beta_3 + \xi_2\beta_6 \end{aligned}$$

Continuous Piecewise Polynomials



Continuous Piecewise Polynomials

- For the piecewise linear model, we have a total of 6 basis. Hence the degrees of freedom is 6 (keep in mind that we fit a linear model once these basis are constructed).
- Because of the two constrains, for the continuous piecewise linear, there are only 4 degrees of freedom
- However, fitting constrained linear regression is “complicated”, hence not preferred.

Continuous Piecewise Polynomials

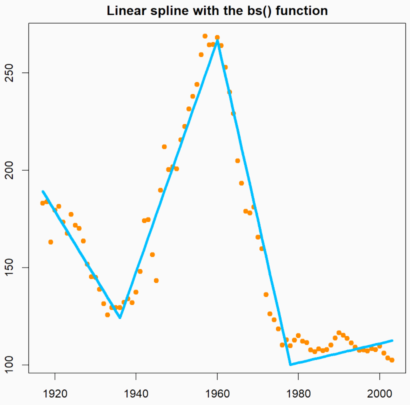
- A trick is to incorporate the constrains into the basis functions (we define an equivalent set of basis that achieves the same property):

$$
\begin{aligned}
h_1(x) &= 1 \\
h_2(x) &= x \\
h_3(x) &= (x - \xi_1)_+ \\
h_4(x) &= (x - \xi_2)_+
\end{aligned}
$$

  where $(\cdot)^+$ denotes the positive part.

- Note that this definition of basis has only 4 elements.
- All we need is to properly define the basis.

Continuous Piecewise Polynomials



Continuous Piecewise Polynomials

- We can then check that any linear combination of these four functions lead to
  - Continuous everywhere
  - Linear everywhere except the knots
  - Has a different slope for each region
- This can be easily done using R function bs in the package splines.

Cubic Splines

- We can extend this idea to obtain higher order of smoothness
- A common choice is the cubic spline, which uses cubic functions within each region
- However, continuities of the first and second order at the knots are forced
- This can be done again using the tricks, similarly to the previous example

Cubic Splines

- Cubic spline function with K knots:

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \sum_{k=1}^{K} b_k (x - \xi_k)_+^3$$

- This leads to a total of $(4 + \#$ knots) degrees of freedom
- The (third order) knot discontinuity is not really visible

Degrees of Freedom

- For cubic spline, we initially requires 4 degrees of freedom to describe each region (1, x, x^2 and x^3)
- With the constrains, we require the two regional functions that joint at a knot has the same value up to the second derivative:

$$
\begin{aligned}
f(\xi^-) &= f(\xi^+) \\
f'(\xi^-) &= f'(\xi^+) \\
f''(\xi^-) &= f''(\xi^+)
\end{aligned}
$$

- Hence, the degrees of freedom for a cubic spline:

  $(\#\,\text{regions}) \times (4\,\text{per region}) - (\#\,\text{knots}) \times (3\,\text{constraints per knot})$

- Note that $\#\,\text{regions} = \#\,\text{knots} + 1$, this becomes $(4 + \#\,\text{knots})$

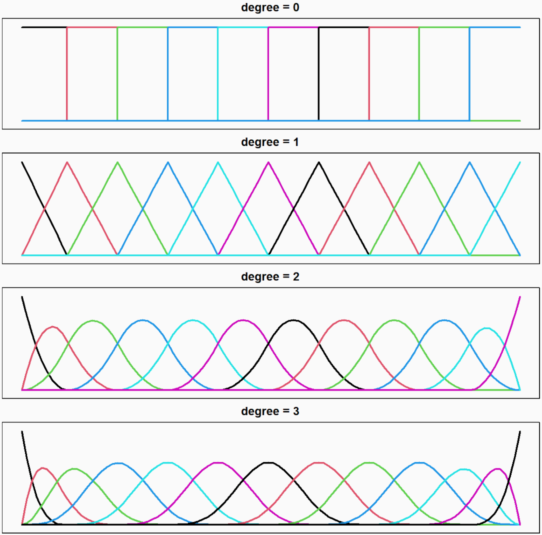
B-spline Basis

- Previous definitions of splines are known as regression splines
- An alternative definition (computationally more efficient) is proposed by de Boor (1978)
- Each basis function is nonzero over at most

$$\text{degree of the polynomial} + 1$$

  consecutive intervals.
- The resulting design matrix is banded

B-spline Basis



Define B-spline Basis

- Create augmented knot sequence τ:

$$\tau_1 = \cdots = \tau_M = \xi_0$$
$$\tau_{M+j} = \xi_j, \quad j = 1, \ldots, K$$
$$\tau_{M+K+1} = \cdots = \tau_{2M+K+1} = \xi_{K+1}$$

  - where ξ_j's, $j = 1, \ldots, K$ are the knots
  - ξ_0 and ξ_{K+1} are the left and right boundary points

Define B-spline Basis

- Denote $B_{i,m}(x)$ the ith B-spline basis function of order m for the knot sequence τ, $m \leq M$. We recursively calculate them as follows:

$$B_{i,1}(x) = \begin{cases} 1 & \text{if} \quad \tau_i \leq x < \tau_{i+1} \\ 0 & \text{o.w.} \end{cases}$$

$$B_{i,m}(x) = \frac{x - \tau_i}{\tau_{i+m-1} - \tau_i} B_{i,m-1}(x) + \frac{\tau_{i+m} - x}{\tau_{i+m} - \tau_{i+1}} B_{i+1,m-1}(x)$$

Generating B-spline Basis in R

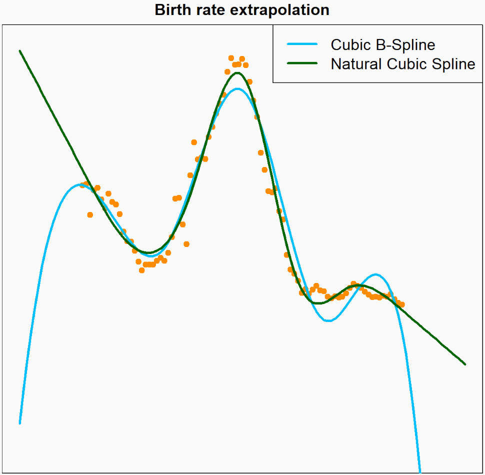
```
> library(splines)
> bs(x, df = NULL, knots = NULL, degree = 3, intercept = FALSE)
```

- df : degrees of freedom (the total number of basis)
- knots : specify knots. By default, these will be the quantiles of x
- degree : degree of piecewise polynomial, default 3 (cubic splines)
- intercept : if TRUE , an intercept is included, default FALSE
- Always return a matrix of dimension $n\times$ df (this may force a change of other parameters)

Natural Cubic Splines

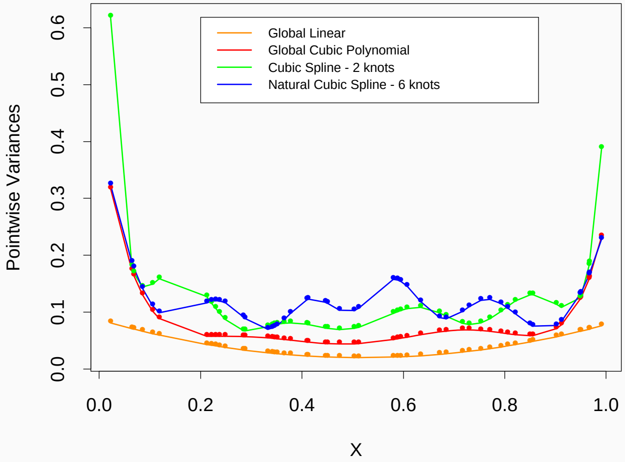
- Polynomials fit to data tend to be erratic near the boundaries. Extrapolation can be dangerous.
- Natural Cubic Splines (NCS) forces the second and third derivatives to be zero at the boundaries, i.e., $\min(x)$ and $\max(x)$
- Hence, the fitted model is linear beyond the two extreme knots $(-\infty, \xi_1]$ and $[\xi_K, \infty)$
- The constraints frees up 4 degrees of freedom. The degrees of freedom of NCS is just the number of knots K.
- Assuming linearity near the boundary is reasonable since there is less information available

Extrapolating beyond the boundaries



Example: Birthrate data, 1917-2003

Comparing Variation of Different Choices



Natural Cubic Splines

- Basis function construction for natural cubic splines
- Starting with a basis for cubic splines, and derive the reduced bases by imposing the boundary constraint, we obtain the basis functions

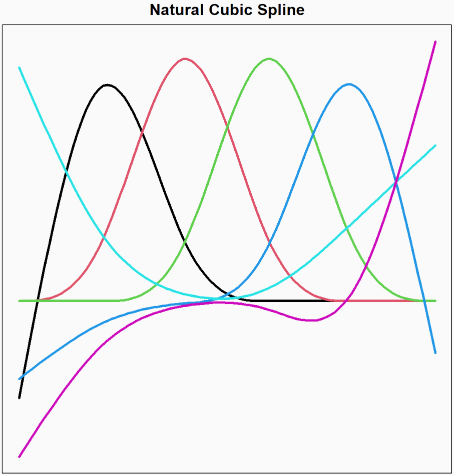
$$N_1(x) = 1,\ N_2(x) = x,\ N_{k+2}(x) = d_k(x) - d_{K-1}(x)$$

where

$$d_k(x) = \frac{(x-\xi_k)_+^3 - (x-\xi_K)_+^3}{\xi_K - \xi_k}, \quad k = 1, \dots, K-2$$

- We can check that each of the basis functions has zero second and third derivatives for $x \leq \xi_1$ and $x \geq \xi_K$

NCS Basis



Generating Natural Cubic Spline basis in R

```
> library(splines)
> ns(x, df = NULL, knots = NULL, intercept = FALSE)
```

- df : degrees of freedom (the total number of basis)
- knots : specify knots. By default, these will be the quantiles of x
- intercept : if TRUE , an intercept is included, default FALSE
- Return a matrix of dimension $n\times$ df

Smoothing Splines

Smoothing Splines

- B-splines and NCS are both methods that construct a $p \times M$ basis matrix $\mathbf{F}$ (p is the number of variables; $p = 1$ in our previous examples), and then model the outcome using a linear regression on $\mathbf{F}$.
- Inevitably, we need to select the order of the spline, the number of knots (AIC, BIC, CV) and even the location of knots (difficult)
- Is there a method that we can select the number and location of knots automatically?

Smoothing Splines

- Smoothing Spline: Let's start with an easy but "horrible" solution, by putting knots at all the observed data points $(x_1, \ldots x_n)$:

$$\mathbf{y}_{n\times 1} = \mathbf{F}_{n\times n}\boldsymbol{\beta}_{n\times 1}$$

Instead of selecting knots, let's use ridge-type shrinkage

$$\text{minimize}_{\boldsymbol{\beta}} \ \|\mathbf{y} - \mathbf{F}\boldsymbol{\beta}\|^2 + \lambda\boldsymbol{\beta}^\mathsf{T}\Omega\boldsymbol{\beta}$$

where Ω will be defined later and λ can be chosen by CV or GCV.

- In fact, the solution can be derived from a different aspect

Roughness Penalty Approach

- Let $W_2[a, b]$ be the space of all smooth functions defined on $[a, b]$
- Second order Sobolev space

$$W_2[a,b] = \left\{ g : g, g' \text{ are absolutely continuous and } \int_a^b [g'']^2 dx < \infty \right\}$$

- Global polynomial functions and cubic spline functions belong to $W_2[a, b]$.
- $W_2[a, b]$ is an infinite-dimension function space
- Find the “best” function in $W_2[a, b]$ to approximate f

Roughness Penalty Approach

- Penalized residual sum of squares

$$\text{RSS}(g, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \big(y_i - g(x_i)\big)^2 + \lambda \int_a^b [g''(x)]^2 dx$$

- First term measure the closeness of the model to the data
- Second term penalizes the roughness/curvature of the function
- Avoid the knot selection problem
- $\int_a^b [g''(x)]^2 dx$ is called the roughness penalty

Roughness Penalty Approach

- λ is the smoothing parameter that controls the bias-variance trade-off
- $\lambda = 0$: interpolate the data, overfitting
- $\lambda = \infty$: linear least-squares regression
- It turns out that the solution to the penalized residual sum of squares has to be a NCS

Theorem

$\widehat{g} = \arg\min$ *RSS*(g, λ) *is a NCS with knots at the* n *data points* $x_1, \ldots, x_n$

Proof

Intuition: Let g be a function on $[a, b]$ and $\widetilde{g}$ be a NCS with

$$g(x_i) = \widetilde{g}(x_i), \quad i = 1, \ldots, n.$$

We can always find such $\widetilde{g}$ since it consists of n basis. Then we can show

$$\int g''^2 dx \geq \int \widetilde{g}''^2 dx,$$

meaning that we will always prefer the $\widetilde{g}$, the NCS “representation” of g, since the penalty is smaller, and the loss doesn’t change.

Proof

Its only left to show that

$$\int g''^2 dx \geq \int \widetilde{g}''^2 dx.$$

We define $h(x) = g(x) - \widetilde{g}(x)$. So $h(x_i) = 0$ for $i = 1, \ldots, n$. Then

$$\int g''^2 dx = \int \widetilde{g}''^2 dx + \int h''^2 dx + 2 \int \widetilde{g}'' h'' dx$$

and (WLOG assuming x_i's are ordered)

$$\begin{aligned} \int \widetilde{g}'' h'' dx &= \widetilde{g}'' h' \Big|_a^b - \int_a^b h' \widetilde{g}^{(3)} dx \\ &= -\sum_{i=1}^{n-1} \widetilde{g}^{(3)}(x_j^+) \int_{x_j}^{x_{j+1}} h' dx \quad (\widetilde{g}^{(3)} \text{constant piecewise}) \\ &= -\sum_{i=1}^{n-1} \widetilde{g}^{(3)}(x_j^+) \big(h(x_{j+1}) - h(x_j)\big) \end{aligned}$$

Proof

- Hence the solution has to have a finite representation

$$\widehat{g}(x) = \sum_{j=1}^{n} \beta_j N_j(x)$$

where N_j's are a set of natural cubic spline basis functions with knots at each of the unique x values

- We can then rewrite

$$\sum_{i=1}^{n} \big(y_i - g(x_i)\big)^2 = (\mathbf{y} - \mathbf{F}\boldsymbol{\beta})^\mathsf{T}(\mathbf{y} - \mathbf{F}\boldsymbol{\beta})$$

where $\mathbf{F}$ is an $n \times n$ matrix with $\mathbf{F}_{ij} = N_j(x_i)$

Proof

- The penalty function

$$\int_a^b g''^2 dx = \int \Big(\sum_i \beta_i N_i''(x)\Big)^2 dx$$
$$= \sum_{i,j} \beta_i \beta_j \int N_i''(x) N_j''(x) dx$$
$$= \boldsymbol{\beta}^\mathsf{T} \Omega \boldsymbol{\beta}$$

where Ω is an $n \times n$ matrix with $\Omega_{ij} = \int N_i''(x) N_j''(x) dx$.

Proof

- Hence our goal is to find $\boldsymbol{\beta}$ that minimizes

$$\text{RSS}(\boldsymbol{\beta}, \lambda) = \|\mathbf{y} - \mathbf{F}\boldsymbol{\beta}\|^2 + \lambda \boldsymbol{\beta}^\mathsf{T} \Omega \boldsymbol{\beta}$$

- This is a ridge penalized function and the solution is

$$\begin{aligned} \widehat{\boldsymbol{\beta}} &= \arg\min_{\boldsymbol{\beta}} \text{RSS}(\boldsymbol{\beta}, \lambda) \\ &= (\mathbf{F}^\mathsf{T}\mathbf{F} + \lambda\Omega)^{-1}\mathbf{F}^\mathsf{T}\mathbf{y} \end{aligned}$$

- The smoothing spline version of the "hat" matrix is called the smoother matrix

$$\begin{aligned} \widehat{f} &= \mathbf{F}(\mathbf{F}^\mathsf{T}\mathbf{F} + \lambda\Omega)^{-1}\mathbf{F}^\mathsf{T}\mathbf{y} \\ &= \mathbf{S}_\lambda \mathbf{y} \end{aligned}$$

Remark

- We have done the analysis of degrees of freedom for ridge type regression. The degrees of freedom of a smoothing spline is

$$\text{df} = \text{Trace}(\mathbf{S}_\lambda)$$

  which ranges between 0 and n.

- Under some special constructions (Demmler and Reinsch, 1975), a basis with double orthogonality property, can lead to

$$\mathbf{F}^\mathsf{T}\mathbf{F} = \mathbf{I}, \quad \text{and } \Omega \text{ is diagonal}$$

  which gives exact solution of $\boldsymbol{\beta}$ (see our ridge lecture notes).

Remark

- Choosing the penalty λ is the same as ridge regression
- Leave-one-out CV:

$$\text{CV} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{g}(x_i)}{1 - \mathbf{S}_\lambda(i,i)}\right)^2$$

- Generalized CV:

$$\text{GCV} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{g}(x_i)}{1 - \frac{1}{n}\text{Trace}(\mathbf{S}_\lambda)}\right)^2$$

Smoothing Splines in R

```
> library(splines)
> smooth.spline(x, y = NULL, w = NULL, df, cv = FALSE)
```

- cv : FALSE uses GCV, TRUE uses Leave-one-out CV
- df : degrees of freedom between 1 and n, let GCV decide it automatically
- w : can be used if x has replicates

Generalized Additive Models

Generalized Additive Models

- Additive models assume that the conational expectation of Y is

$$f(x) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

- This can be extended to modeling Y's with other distributions. Such as binary, counts, positive values, etc.
- Generalized Additive Models (GAM) assume that

$$\mathsf{E}(Y|X) = g^{-1}(\alpha + \sum_{j=1}^{p} f_j(x_j))$$

- In logistic regression, we used logit link for g
- We fit each f_j using a cubic smoothing spline or kernel smoother

$$g(\mathsf{E}(Y|X)) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

Generalized Additive Models

- Logistic regression

$$\text{logit}(\mathsf{P}(Y = 1|X)) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

- Poisson regression

$$\log(E(Y|X)) = \alpha + \sum_{j=1}^{p} f_j(x_j)$$

- ...

Fitting Additive Models

- $y = \alpha + \sum_{j=1}^{p} f_j(x_j) + \epsilon$
- Initialize
  - Constant α = average response
  - All $f_j = 0$
- Cycle
  - Fit one component at a time to residuals from the other components using a smoother
  - Normalize most recent component to average to 0
  - Stop when all components average within desired accuracy

Backfitting Algorithm

- Consider using general smoothers as building blocks to fit the model
- Initialize $\widehat{\alpha} = \overline{y}, \widehat{f}_j = 0$
- Iterate (backfitting) until $\widehat{f}_j$'s stabilize:

$$\widehat{f}_j = S_j(y - \widehat{\alpha} - \sum_{j' \neq j} \widehat{f}_{j'})$$

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

where S_j denotes a smoothing spline fit.

Fitting Generalized Additive Models

- We cannot update the response values by subtract the current fitting values
- Use Iteratively Reweighted Least Squares (see previous lecture on logistic regression)
- Initialize $\widehat{\alpha} = \log[\overline{y}/(1-\overline{y})], \widehat{f_j} = 0$
- Define $\widehat{\eta}_i = \widehat{\alpha} + \sum_j \widehat{f_j}(x_{ij})$ and $\widehat{p}_i = 1/[1+\exp(-\widehat{\eta}_i)]$
  - Calculate the working target variable $z_i = \widehat{\eta}_i + \frac{y_i - \widehat{p}_i}{\widehat{p}_i(1-\widehat{p}_i)}$
  - Construct weights $w_i = \widehat{p}_i(1-\widehat{p}_i)$
  - Fit additive model to targets z_i with weight w_i using a weighted backfitting algorithm
- Stop when converged within specified accuracy

R implementation

- Package: mgcv , function gam
- The gam solves the smoothing parameter estimation problem by using the Generalized Cross Validation (GCV) criterion

```
> library(gam)
> form = formula("chd ~ ns(sbp,df=4) + ns(tobacco,df=4) +
>                        ns(ldl,df=4) + famhist + ns(obesity,df=4) +
>                        ns(alcohol,df=4) + ns(age,df=4)")
> m = gam(form, data=SAheart, family=binomial)
> summary(m)
> par(mfrow = c(3, 3), mar = c(5, 5, 2, 0))
> plot(m, se = TRUE, residuals = TRUE, pch = 19, col = "darkorange")
```

Example: South African Heart Disease