CEE 697M: Probabilistic Machine Learning

M2 Linear Methods: Splines, GAMs and GLMs

Jimi Oke

UMass Amherst

College of Engineering

Wed, Mar 29, 2023

Outline

Flexible linear methods

Flexible linear methods

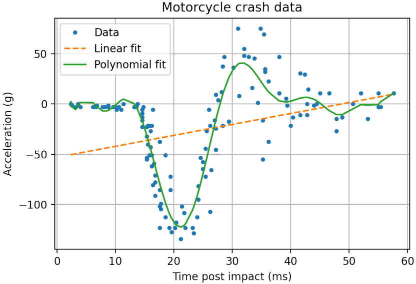
In many cases, **flexibility** is required to obtain useful **linear** models:

Flexible linear methods

In many cases, **flexibility** is required to obtain useful **linear** models:

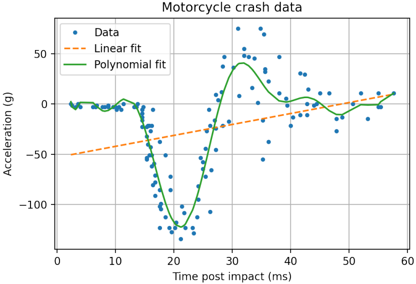
Flexible linear methods

In many cases, **flexibility** is required to obtain useful **linear** models:



Flexible linear methods

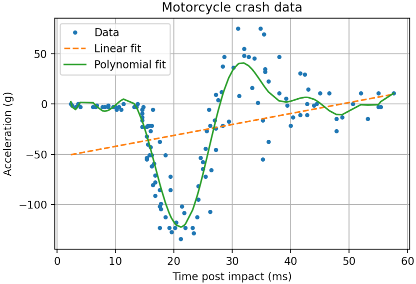
In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs

Flexible linear methods

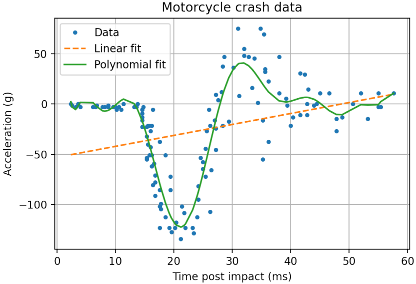
In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs
- models are still linear in the parameters $\boldsymbol{w}$, e.g.

Flexible linear methods

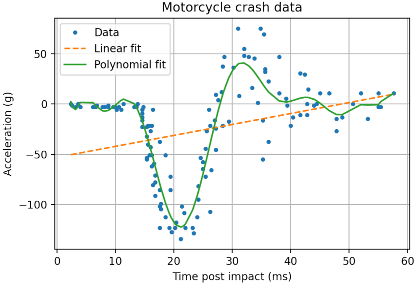
In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs
- models are still linear in the parameters $\boldsymbol{w}$, e.g.
  - polynomial regression

Flexible linear methods

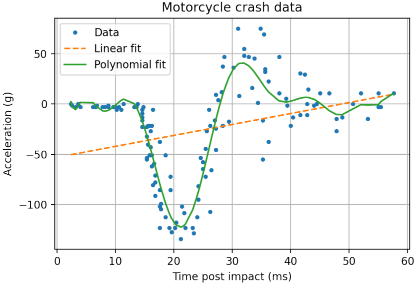
In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs
- models are still linear in the parameters $\boldsymbol{w}$, e.g.
  - polynomial regression
  - step functions: fitting a piecewise constant function

Flexible linear methods

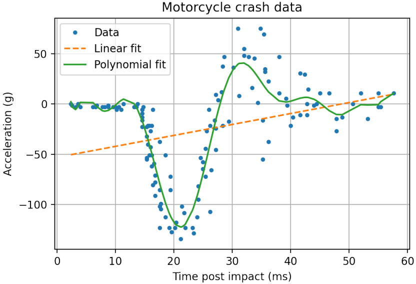
In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs
- models are still linear in the parameters $\boldsymbol{w}$, e.g.
  - polynomial regression
  - step functions: fitting a piecewise constant function
  - regression splines: piecewise polynomial fit

Flexible linear methods

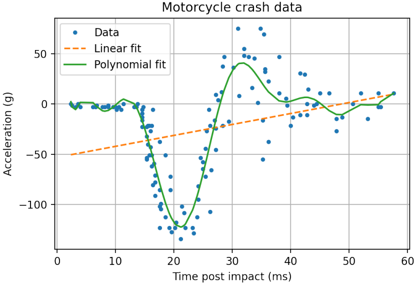
In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs
- models are still linear in the parameters $\boldsymbol{w}$, e.g.
  - polynomial regression
  - step functions: fitting a piecewise constant function
  - regression splines: piecewise polynomial fit
  - smoothing splines: regularization

Flexible linear methods

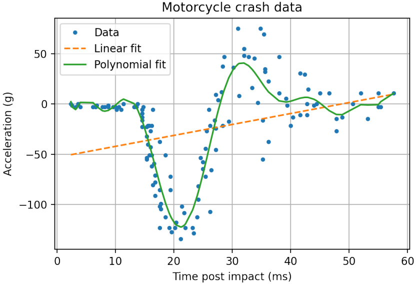
In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs
- models are still linear in the parameters $\boldsymbol{w}$, e.g.
  - polynomial regression
  - step functions: fitting a piecewise constant function
  - regression splines: piecewise polynomial fit
  - smoothing splines: regularization
  - generalized additive models (GAMs): deal with multiple predictors

Flexible linear methods

In many cases, **flexibility** is required to obtain useful **linear** models:



- achieved via nonlinear transformations of the inputs
- models are still linear in the parameters $\boldsymbol{w}$, e.g.
  - polynomial regression
  - step functions: fitting a piecewise constant function
  - regression splines: piecewise polynomial fit
  - smoothing splines: regularization
  - generalized additive models (GAMs): deal with multiple predictors
  - local regression: allow overlap

Flexible linear methods: topics

Flexible linear methods: topics

Topics covered in this module:

Flexible linear methods: topics

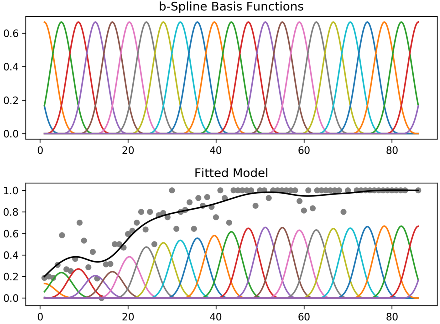
Topics covered in this module:

- Polynomial regression and basis functions (handout)
- Splines, Generalized Additive Models and Generalized Linear Models (this lecture)
- Kernel Methods and Local Regression (Lecture 4a)

Flexible linear methods: topics

Topics covered in this module:

- Polynomial regression and basis functions (handout)
- Splines, Generalized Additive Models and Generalized Linear Models (this lecture)
- Kernel Methods and Local Regression (Lecture 4a)



Source: `https://pygam.readthedocs.io/en/latest/notebooks/tour_of_pygam.html`

Splines

A spline is a piecewise polynomial function defined as a linear combination of **basis functions** in X:

$$f(X) = \sum_p w_p b_p(x) \tag{1}$$

Generally, the **order M (regression) spline** is given by the **truncated power basis**:

$$\begin{aligned} b_p(x) &= x^m, \quad m = 0, \ldots, M &(2)\\ b_{M+k}(x) &= (X - \xi_k)_+^M, k = 1, \ldots, K &(3) \end{aligned}$$

Splines

A spline is a piecewise polynomial function defined as a linear combination of **basis functions** in X:

$$f(X) = \sum_p w_p b_p(x) \tag{1}$$

Generally, the **order M (regression) spline** is given by the **truncated power basis**:

$$\begin{aligned} b_p(x) &= x^m, \quad m = 0, \ldots, M \qquad (2) \\ b_{M+k}(x) &= (X - \xi_k)_+^M, k = 1, \ldots, K \qquad (3) \end{aligned}$$

where

$$(x - \xi_k)_+ = \begin{cases} x - \xi_k, & x \geq \xi_k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Splines

A spline is a piecewise polynomial function defined as a linear combination of **basis functions** in X:

$$f(X) = \sum_p w_p b_p(x) \tag{1}$$

Generally, the **order M (regression) spline** is given by the **truncated power basis**:

$$\begin{aligned} b_p(x) &= x^m, \quad m = 0, \ldots, M & (2) \\ b_{M+k}(x) &= (X - \xi_k)_+^M, k = 1, \ldots, K & (3) \end{aligned}$$

where

$$(x - \xi_k)_+ = \begin{cases} x - \xi_k, & x \geq \xi_k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and ξ_k are the knots.

Splines

A spline is a piecewise polynomial function defined as a linear combination of **basis functions** in X:

$$f(X) = \sum_p w_p b_p(x) \tag{1}$$

Generally, the **order** M (**regression**) **spline** is given by the **truncated power basis**:

$$\begin{aligned} b_p(x) &= x^m, \quad m = 0, \ldots, M \qquad (2) \\ b_{M+k}(x) &= (X - \xi_k)_+^M, k = 1, \ldots, K \qquad (3) \end{aligned}$$

where

$$(x - \xi_k)_+ = \begin{cases} x - \xi_k, & x \geq \xi_k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

and ξ_k are the knots.
In most cases, M is taken as 0 (piecewise-constant), 1 (piecewise-linear) or 3 (cubic spline).

Natural cubic splines

Natural cubic splines

- **Natural cubic splines** presume linearity at the boundaries and thus have the same number of basis functions as knots.

Natural cubic splines

- **Natural cubic splines** presume linearity at the boundaries and thus have the same number of basis functions as knots.
- Basis functions are constrained at **knots** (ξ_k) for continuity

Natural cubic splines

- **Natural cubic splines** presume linearity at the boundaries and thus have the same number of basis functions as knots.
- Basis functions are constrained at **knots** (ξ_k) for continuity
- Their first derivatives are further constrained to be equal for smoothness

Natural cubic splines

- **Natural cubic splines** presume linearity at the boundaries and thus have the same number of basis functions as knots.
- Basis functions are constrained at **knots** (ξ_k) for continuity
- Their first derivatives are further constrained to be equal for smoothness

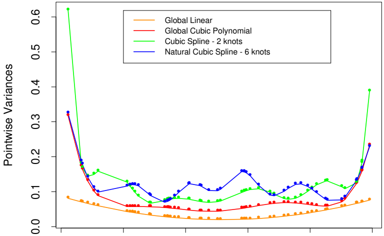


Figure: Pointwise variance curves for 4 different models. $x \sim \mathcal{U}(0,1)$; error assumed constant

Matrix representation of basis functions

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M$$

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k (X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$,

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k(X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$,

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k (X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k (X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

$$f(x)_{N\times 1} = \boldsymbol{B}_{N\times P}\boldsymbol{w}_{P\times 1} \tag{6}$$

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k(X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

$$f(x)_{N\times 1} = \boldsymbol{B}_{N\times P}\boldsymbol{w}_{P\times 1} \tag{6}$$

where $\boldsymbol{B}$ is an $N \times P$

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k (X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

$$f(x)_{N\times 1} = \boldsymbol{B}_{N\times P}\boldsymbol{w}_{P\times 1} \tag{6}$$

where $\boldsymbol{B}$ is an $N \times P$ or $N \times (M + K + 1)$ design matrix

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k (X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

$$f(x)_{N\times 1} = \boldsymbol{B}_{N\times P}\boldsymbol{w}_{P\times 1} \tag{6}$$

where $\boldsymbol{B}$ is an $N \times P$ or $N \times (M + K + 1)$ design matrix

$$B_{np} \quad = \quad b_p(x_n),$$

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k (X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

$$f(x)_{N\times 1} = \boldsymbol{B}_{N\times P}\boldsymbol{w}_{P\times 1} \tag{6}$$

where $\boldsymbol{B}$ is an $N \times P$ or $N \times (M + K + 1)$ design matrix

$$B_{np} \quad = \quad b_p(x_n), \quad n = 1, \ldots, N \tag{7}$$

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k(X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

$$f(x)_{N\times 1} = \boldsymbol{B}_{N\times P}\boldsymbol{w}_{P\times 1} \tag{6}$$

where $\boldsymbol{B}$ is an $N \times P$ or $N \times (M + K + 1)$ design matrix

$$\begin{aligned} B_{np} &= b_p(x_n), \quad n = 1, \ldots, N && (7) \\ p &= (0,0), \ldots, (0, M), 1, \ldots, K && (8) \end{aligned}$$

Matrix representation of basis functions

A more explicit way to write the M-th order **truncated power basis** expansion is:

$$f(x) = w_{0,0} + w_{0,1}x + \cdots + w_{0,1}x^M + \sum_{k=1}^{K} w_k(X - \xi_k)_+^M \tag{5}$$

If we let $b_{0,k}(x) = x^k$ for $k = 0, \ldots, M$, and $b_k(x) = (x - \xi_k)_+^M$ for $k = 1, \ldots, K$, then we can write in matrix form:

$$f(x)_{N\times 1} = \boldsymbol{B}_{N\times P}\boldsymbol{w}_{P\times 1} \tag{6}$$

where $\boldsymbol{B}$ is an $N \times P$ or $N \times (M + K + 1)$ design matrix

$$B_{np} = b_p(x_n), \quad n = 1, \ldots, N \tag{7}$$
$$p = (0, 0), \ldots, (0, M), 1, \ldots, K \tag{8}$$

In this form, it can be easily seen that $\hat{\boldsymbol{w}}$ can be found via least squares.

Challenges in fitting a spline

Challenges in fitting a spline

Decisions

- How many knots do you need?

Challenges in fitting a spline

Decisions

- How many knots do you need?
- Where should you put the knots?

Challenges in fitting a spline

Decisions

- How many knots do you need?
- Where should you put the knots?

Challenges in fitting a spline

Decisions

- How many knots do you need?
- Where should you put the knots?

Challenge

- Risk of overfitting. Why?

Challenges in fitting a spline

Decisions

- How many knots do you need?
- Where should you put the knots?

Challenge

- Risk of overfitting. Why?
- How do we mitigate for this?

Challenges in fitting a spline

Decisions

- How many knots do you need?
- Where should you put the knots?

Challenge

- Risk of overfitting. Why?
- How do we mitigate for this?

Challenges in fitting a spline

Decisions

- How many knots do you need?
- Where should you put the knots?

Challenge

- Risk of overfitting. Why?
- How do we mitigate for this? **Regularization!**

Smoothing splines

These avoid the problem of knot selection

- Smoothing splines use the *maximal* set of knots
- Obtained via regularization

Smoothing splines

These avoid the problem of knot selection

- Smoothing splines use the *maximal* set of knots
- Obtained via regularization

We define a penalized loss function:

Smoothing splines

These avoid the problem of knot selection

- Smoothing splines use the *maximal* set of knots
- Obtained via regularization

We define a penalized loss function:

$$RSS(f, \lambda) = \sum_{n=1}^{N} \{y_n - f(x_n)\} + \lambda \int \{f''(t)\}^2 dt \tag{9}$$

Then the **smoothing spline** is given by the function that minimizes this:

Smoothing splines

These avoid the problem of knot selection

- Smoothing splines use the *maximal* set of knots
- Obtained via regularization

We define a penalized loss function:

$$RSS(f, \lambda) = \sum_{n=1}^{N} \{y_n - f(x_n)\} + \lambda \int \{f''(t)\}^2 dt \tag{9}$$

Then the **smoothing spline** is given by the function that minimizes this:

$$\hat{f}(x) = \arg\min_f RSS(f, \lambda) \tag{10}$$

Smoothing splines

These avoid the problem of knot selection

- Smoothing splines use the *maximal* set of knots
- Obtained via regularization

We define a penalized loss function:

$$RSS(f,\lambda) = \sum_{n=1}^{N} \{y_n - f(x_n)\} + \lambda \int \{f''(t)\}^2 dt \tag{9}$$

Then the **smoothing spline** is given by the function that minimizes this:

$$\hat{f}(x) = \arg\min_f RSS(f,\lambda) \tag{10}$$

where λ is a nonnegative tuning parameter (or hyperparameter)

Smoothing splines

These avoid the problem of knot selection

- Smoothing splines use the *maximal* set of knots
- Obtained via regularization

We define a penalized loss function:

$$RSS(f, \lambda) = \sum_{n=1}^{N} \{y_n - f(x_n)\} + \lambda \int \{f''(t)\}^2 dt \tag{9}$$

Then the **smoothing spline** is given by the function that minimizes this:

$$\hat{f}(x) = \arg\min_{f} RSS(f, \lambda) \tag{10}$$

where λ is a nonnegative tuning parameter (or hyperparameter)—roughness penalty.

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{11}$$

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{11}$$

where the $S_k(x)$ are the basis functions for the natural cubic splines:

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{11}$$

where the $S_k(x)$ are the basis functions for the natural cubic splines:

$$S_1(x) = 1$$

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{11}$$

where the $S_k(x)$ are the basis functions for the natural cubic splines:

$$S_1(x) = 1$$

$$S_2(x) = x$$

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{11}$$

where the $S_k(x)$ are the basis functions for the natural cubic splines:

$$S_1(x) = 1$$

$$S_2(x) = x$$

$$S_{k+2}(x) = d_k(x) - d_{K-1}(x);$$

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{11}$$

where the $S_k(x)$ are the basis functions for the natural cubic splines:

$$\begin{aligned} S_1(x) &= 1 \\ S_2(x) &= x \\ S_{k+2}(x) &= d_k(x) - d_{K-1}(x); \quad d_k(x) = \frac{(x-\xi_k)_+^3 - (x-\xi_K)_+^3}{\xi_K - \xi_k} \end{aligned} \tag{12}$$

Properties of smoothing splines

- Unique minimizer to $RSS(f, \lambda)$
- Formulated as **natural cubic spline** with knots ξ_k at unique values of x_n

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{11}$$

where the $S_k(x)$ are the basis functions for the natural cubic splines:

$$\begin{aligned} S_1(x) &= 1 \\ S_2(x) &= x \\ S_{k+2}(x) &= d_k(x) - d_{K-1}(x); \quad d_k(x) = \frac{(x-\xi_k)_+^3 - (x-\xi_K)_+^3}{\xi_K - \xi_k} \end{aligned} \tag{12}$$

- If N unique values in dataset, then $K = N$

Smoothing spline estimation

Smoothing spline estimation

We can then write the penalized loss function as:

Smoothing spline estimation

We can then write the penalized loss function as:

$$RSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{13}$$

Smoothing spline estimation

We can then write the penalized loss function as:

$$RSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{13}$$

where

Smoothing spline estimation

We can then write the penalized loss function as:

$$RSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{13}$$

where

$$S_{nk} = S_k(x_n) \tag{14}$$

Smoothing spline estimation

We can then write the penalized loss function as:

$$RSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{13}$$

where

$$S_{nk} = S_k(x_n) \tag{14}$$

$$\Omega_{k',k} = \int S''_{k'}(t) S''_k(t) dt \tag{15}$$

Smoothing spline estimation

We can then write the penalized loss function as:

$$RSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{13}$$

where

$$S_{nk} = S_k(x_n) \tag{14}$$

$$\Omega_{k',k} = \int S''_{k'}(t) S''_k(t) dt \tag{15}$$

As $\boldsymbol{S}$ and $\boldsymbol{\Omega}$ are constant in $\boldsymbol{w}$, we take the derivative and set to zero to obtain the smoothing spline estimate:

Smoothing spline estimation

We can then write the penalized loss function as:

$$RSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{13}$$

where

$$S_{nk} = S_k(x_n) \tag{14}$$

$$\Omega_{k',k} = \int S''_{k'}(t) S''_k(t) dt \tag{15}$$

As $\boldsymbol{S}$ and $\boldsymbol{\Omega}$ are constant in $\boldsymbol{w}$, we take the derivative and set to zero to obtain the smoothing spline estimate:

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{S}^T \boldsymbol{S} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{S}^T \boldsymbol{y}} \tag{16}$$

Smoothing spline estimation

We can then write the penalized loss function as:

$$RSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{13}$$

where

$$S_{nk} = S_k(x_n) \tag{14}$$

$$\Omega_{k',k} = \int S''_{k'}(t) S''_k(t) dt \tag{15}$$

As $\boldsymbol{S}$ and $\boldsymbol{\Omega}$ are constant in $\boldsymbol{w}$, we take the derivative and set to zero to obtain the smoothing spline estimate:

$$\boxed{\hat{\boldsymbol{w}} = (\boldsymbol{S}^T \boldsymbol{S} + \lambda \Omega)^{-1} \boldsymbol{S}^T \boldsymbol{y}} \tag{16}$$

This is simply a generalized ridge regression!

Smoother matrix

Smoother matrix

We can compactly write the *fitted smoothing spline* as:

Smoother matrix

We can compactly write the *fitted smoothing spline* as:

$$\hat{\boldsymbol{w}} = \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S} + \lambda\Omega)^{-1}\boldsymbol{S}^T\boldsymbol{y} = \boldsymbol{S}_\lambda\boldsymbol{y} \tag{17}$$

Smoother matrix

We can compactly write the *fitted smoothing spline* as:

$$\hat{\boldsymbol{w}} = \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S} + \lambda\Omega)^{-1}\boldsymbol{S}^T\boldsymbol{y} = \boldsymbol{S}_\lambda\boldsymbol{y} \tag{17}$$

where $\boldsymbol{S}_\lambda$ is known as the **smoother matrix**.

Smoother matrix

We can compactly write the *fitted smoothing spline* as:

$$\hat{\boldsymbol{w}} = \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S} + \lambda\Omega)^{-1}\boldsymbol{S}^T\boldsymbol{y} = \boldsymbol{S}_\lambda \boldsymbol{y} \tag{17}$$

where $\boldsymbol{S}_\lambda$ is known as the **smoother matrix**.

The number of effective degrees of freedom of a smoothing spline is given by:

Smoother matrix

We can compactly write the *fitted smoothing spline* as:

$$\hat{\boldsymbol{w}} = \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S} + \lambda\Omega)^{-1}\boldsymbol{S}^T\boldsymbol{y} = \boldsymbol{S}_\lambda\boldsymbol{y} \tag{17}$$

where $\boldsymbol{S}_\lambda$ is known as the **smoother matrix**.

The number of effective degrees of freedom of a smoothing spline is given by:

$$df_\lambda = \text{trace}(\boldsymbol{S}_\lambda) \tag{18}$$

Smoother matrix

We can compactly write the *fitted smoothing spline* as:

$$\hat{\boldsymbol{w}} = \boldsymbol{S}(\boldsymbol{S}^T\boldsymbol{S} + \lambda\Omega)^{-1}\boldsymbol{S}^T\boldsymbol{y} = \boldsymbol{S}_\lambda\boldsymbol{y} \tag{17}$$

where $\boldsymbol{S}_\lambda$ is known as the **smoother matrix**.

The number of effective degrees of freedom of a smoothing spline is given by:

$$df_\lambda = \text{trace}(\boldsymbol{S}_\lambda) \tag{18}$$

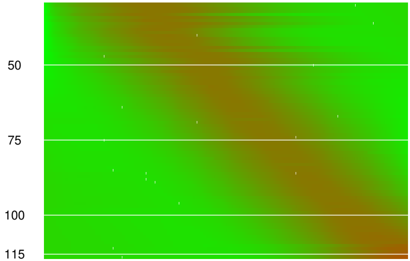


Figure: Visualization of a smoother matrix. Reddish hues indicate higher values.

Eigendecomposition of smoother matrix

Eigendecomposition of smoother matrix

$\boldsymbol{S}_\lambda$ is symmetric and positive semidefinite, and can be factored as:

Eigendecomposition of smoother matrix

$\boldsymbol{S}_\lambda$ is symmetric and positive semidefinite, and can be factored as:

$$\boldsymbol{S}_\lambda = \sum_{k=1}^{K} \frac{1}{1 + \lambda d_k} \boldsymbol{u}_k \boldsymbol{u}_k^T = \sum_{k=1}^{K} \rho_k(\lambda) \boldsymbol{u}_k \boldsymbol{u}_k^T \tag{19}$$

Eigendecomposition of smoother matrix

$\boldsymbol{S}_\lambda$ is symmetric and positive semidefinite, and can be factored as:

$$\boldsymbol{S}_\lambda = \sum_{k=1}^{K} \frac{1}{1+\lambda d_k} \boldsymbol{u}_k \boldsymbol{u}_k^T = \sum_{k=1}^{K} \rho_k(\lambda) \boldsymbol{u}_k \boldsymbol{u}_k^T \tag{19}$$

where d_k are the eigenvalues of the penalty matrix $\boldsymbol{K}$, such that:

Eigendecomposition of smoother matrix

$\boldsymbol{S}_\lambda$ is symmetric and positive semidefinite, and can be factored as:

$$\boldsymbol{S}_\lambda = \sum_{k=1}^{K} \frac{1}{1+\lambda d_k} \boldsymbol{u}_k \boldsymbol{u}_k^T = \sum_{k=1}^{K} \rho_k(\lambda) \boldsymbol{u}_k \boldsymbol{u}_k^T \tag{19}$$

where d_k are the eigenvalues of the penalty matrix $\boldsymbol{K}$, such that:

$$\boldsymbol{S}_\lambda = (\boldsymbol{I} + \lambda \boldsymbol{K})^{-1} \tag{20}$$

Eigendecomposition of smoother matrix

$\boldsymbol{S}_\lambda$ is symmetric and positive semidefinite, and can be factored as:

$$\boldsymbol{S}_\lambda = \sum_{k=1}^{K} \frac{1}{1+\lambda d_k} \boldsymbol{u}_k \boldsymbol{u}_k^T = \sum_{k=1}^{K} \rho_k(\lambda) \boldsymbol{u}_k \boldsymbol{u}_k^T \tag{19}$$

where d_k are the eigenvalues of the penalty matrix $\boldsymbol{K}$, such that:

$$\boldsymbol{S}_\lambda = (\boldsymbol{I} + \lambda \boldsymbol{K})^{-1} \tag{20}$$

and ρ_k are the eigenvalues of $\boldsymbol{S}_\lambda$.

Properties of the smoother matrix

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

$$\rho_1(\lambda) = \rho_2(\lambda) = \frac{1}{1 + \lambda(0)} =$$

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

$$\rho_1(\lambda) = \rho_2(\lambda) = \frac{1}{1 + \lambda(0)} = 1 \tag{21}$$

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

$$\rho_1(\lambda) = \rho_2(\lambda) = \frac{1}{1 + \lambda(0)} = 1 \tag{21}$$

The linear basis functions are thus not penalized.

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

$$\rho_1(\lambda) = \rho_2(\lambda) = \frac{1}{1 + \lambda(0)} = 1 \tag{21}$$

  The linear basis functions are thus not penalized.
- For the corresponding smoother matrices of regression splines, all the eigenvalues are 1, i.e. no shrinking.

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

$$\rho_1(\lambda) = \rho_2(\lambda) = \frac{1}{1 + \lambda(0)} = 1 \tag{21}$$

  The linear basis functions are thus not penalized.
- For the corresponding smoother matrices of regression splines, all the eigenvalues are 1, i.e. no shrinking.

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

$$\rho_1(\lambda) = \rho_2(\lambda) = \frac{1}{1 + \lambda(0)} = 1 \tag{21}$$

  The linear basis functions are thus not penalized.
- For the corresponding smoother matrices of regression splines, all the eigenvalues are 1, i.e. no shrinking.

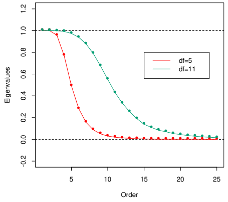


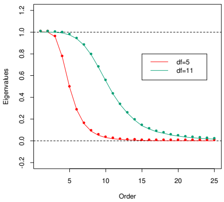
Figure: (Left) Eigenvalues of the smoother matrix for a smoothing spline fit. (Right) Eigenvectors $\boldsymbol{u}_k$ are plotted versus $\boldsymbol{x}$ for $k = 3, 4, 5, 6$.

Properties of the smoother matrix

- The first two eigenvalues of penalty matrix $\boldsymbol{K}$ are $0 = d_1 = d_2$.
- This implies that the first two eigenvalues of $\boldsymbol{S}_\lambda$ (corresponding to the 2-D eigenspace of functions linear in x) are always 1:

$$\rho_1(\lambda) = \rho_2(\lambda) = \frac{1}{1 + \lambda(0)} = 1 \tag{21}$$

  The linear basis functions are thus not penalized.
- For the corresponding smoother matrices of regression splines, all the eigenvalues are 1, i.e. no shrinking.



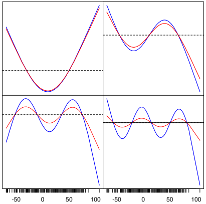


Figure: (Left) Eigenvalues of the smoother matrix for a smoothing spline fit. (Right) Eigenvectors $\boldsymbol{u}_k$ are plotted versus $\boldsymbol{x}$ for $k = 3, 4, 5, 6$.

Example: Estimating a response across different groups

Example: Estimating a response across different groups

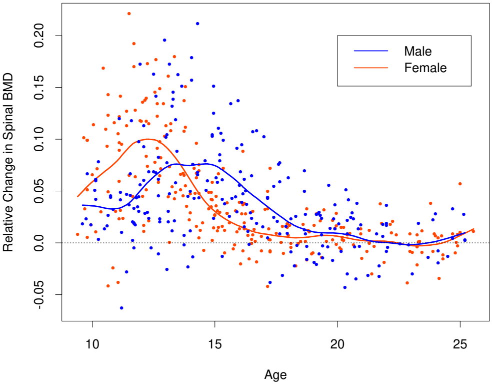


Figure: Estimating a smoothing spline for bone mineral density for male and female adolescents. $df_\lambda \approx 12$. The fitted splines shows that changes in BMD occur earlier in females than males.

Example: Estimating blood pressure (two predictors)

Example: Estimating blood pressure (two predictors)

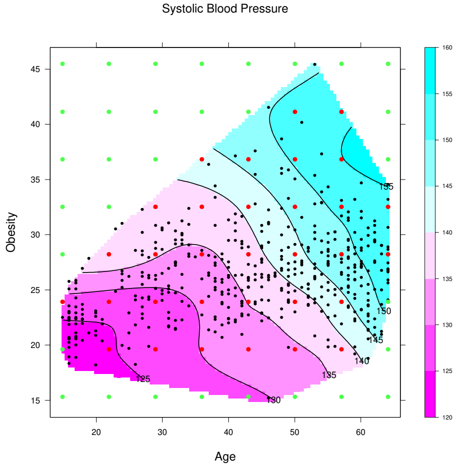


Figure: Estimating systolic blood pressure (response) as a function of obesity and age. Knots used are shown in red.

Generalized additive models

The **generalized additive model** has the form:

$$\mathbb{E}(y|x_1, x_2, \ldots, x_d) = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_D(x_D) \tag{22}$$

Generalized additive models

The **generalized additive model** has the form:

$$\mathbb{E}(y|x_1, x_2, \ldots, x_d) = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_D(x_D) \tag{22}$$

where:

$x_1, x_2, \cdots, x_D$

Generalized additive models

The **generalized additive model** has the form:

$$\mathbb{E}(y|x_1, x_2, \ldots, x_d) = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_D(x_D) \tag{22}$$

where:

$x_1, x_2, \cdots, x_D$ — predictors

Generalized additive models

The **generalized additive model** has the form:

$$\mathbb{E}(y|x_1, x_2, \ldots, x_d) = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_D(x_D) \tag{22}$$

where:

$x_1, x_2, \cdots, x_D$ — predictors

y

Generalized additive models

The **generalized additive model** has the form:

$$\mathbb{E}(y|x_1, x_2, \ldots, x_d) = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_D(x_D) \tag{22}$$

where:

$x_1, x_2, \cdots, x_D$ — predictors

y — outcome

Generalized additive models

The **generalized additive model** has the form:

$$\mathbb{E}(y|x_1, x_2, \ldots, x_d) = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_D(x_D) \tag{22}$$

where:

$x_1, x_2, \cdots, x_D$ — predictors

y — outcome

f_d — unspecified smooth/nonparametric functions

Generalized additive models

The **generalized additive model** has the form:

$$\mathbb{E}(y|x_1, x_2, \ldots, x_d) = \alpha + f_1(x_1) + f_2(x_2) + \cdots + f_D(x_D) \tag{22}$$

where:

$x_1, x_2, \cdots, x_D$	— predictors
y	— outcome
f_d	— unspecified smooth/nonparametric functions e.g. basis functions

GAMs: regression setting

GAMs: regression setting

The multiple linear regression model is given by:

GAMs: regression setting

The multiple linear regression model is given by:

$$y =$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots +$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

$$y =$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

$$y = \alpha +$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

$$y = \alpha + f_1(x_1)$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

$$y = \alpha + f_1(x_1) + \cdots +$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

$$y = \alpha + f_1(x_1) + \cdots + f_D(x_D) + \epsilon$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

$$y = \alpha + f_1(x_1) + \cdots + f_D(x_D) + \epsilon$$

$$=$$

GAMs: regression setting

The multiple linear regression model is given by:

$$y = w_0 + w_1 x_1 + \cdots + w_D x_D + \epsilon \tag{23}$$

The generalized additive movel replaces each linear term by a smooth/nonlinear function $f_d(x_d)$:

$$\begin{aligned} y &= \alpha + f_1(x_1) + \cdots + f_D(x_D) + \epsilon \\ &= \alpha + \sum_{d=1}^{D} f_d(x_d) + \epsilon \end{aligned} \tag{24}$$

Fitting additive models

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) =$$

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2$$

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for *PRSS* form an **additive cubic spline**:

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \dots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for *PRSS* form an **additive cubic spline**:

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for $PRSS$ form an **additive cubic spline**: each $\hat{f}_d$ is a cubic spline in x_d with knots at each unique $x_{nd} = 1, \ldots, N$
- To ensure uniqueness we must constrain the objective.

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for $PRSS$ form an **additive cubic spline**: each $\hat{f}_d$ is a cubic spline in x_d with knots at each unique $x_{nd} = 1, \ldots, N$
- To ensure uniqueness we must constrain the objective.

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for $PRSS$ form an **additive cubic spline**: each $\hat{f}_d$ is a cubic spline in x_d with knots at each unique $x_{nd} = 1, \ldots, N$
- To ensure uniqueness we must constrain the objective. One possibility:

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for $PRSS$ form an **additive cubic spline**: each $\hat{f}_d$ is a cubic spline in x_d with knots at each unique $x_{nd} = 1, \ldots, N$
- To ensure uniqueness we must constrain the objective. One possibility:

$$\sum_{n=1}^{N} f_d(x_{nd}) = 0 \quad \forall d \in \{1, \ldots, D\} \tag{26}$$

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for $PRSS$ form an **additive cubic spline**: each $\hat{f}_d$ is a cubic spline in x_d with knots at each unique $x_{nd} = 1, \ldots, N$
- To ensure uniqueness we must constrain the objective. One possibility:

$$\sum_{n=1}^{N} f_d(x_{nd}) = 0 \quad \forall d \in \{1, \ldots, D\} \tag{26}$$

  In this case, then the intercept estimate is the mean response:

Fitting additive models

Additive models can be estimated in a similar fashion as smoothing splines.

The **penalized loss function** (PRSS) is given by:

$$PRSS(\alpha, f_1, f_2, \ldots, f_D) = \sum_{n=1}^{n} \left(y_n - \alpha - \sum_{d=1}^{D} f_d(x_{nd}) \right)^2 + \sum_{d=1}^{D} \lambda_d \int f_d''(t_d)^2 dt_d \tag{25}$$

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

- The minimizing functions for $PRSS$ form an **additive cubic spline**: each $\hat{f}_d$ is a cubic spline in x_d with knots at each unique $x_{nd} = 1, \ldots, N$
- To ensure uniqueness we must constrain the objective. One possibility:

$$\sum_{n=1}^{N} f_d(x_{nd}) = 0 \quad \forall d \in \{1, \ldots, D\} \tag{26}$$

In this case, then the intercept estimate is the mean response:

$$\hat{\alpha} = \frac{1}{N} \sum_{n=1}^{N} y_n \tag{27}$$

Backfitting algorithm for additive model estimation

Backfitting algorithm for additive model estimation

1. Initialize:

Backfitting algorithm for additive model estimation

1. Initialize:

Backfitting algorithm for additive model estimation

1. Initialize:

$$\hat{\alpha} = \frac{1}{N}\sum_{n=1}^{N} y_n$$

Backfitting algorithm for additive model estimation

1. Initialize:

$$\begin{aligned} \hat{\alpha} &= \frac{1}{N}\sum_{n=1}^{N} y_n \\ \hat{f}_d &= 0, \quad \forall i, j \end{aligned}$$

2. Iterate q times for $j = 1, 2, \ldots, p$:

Backfitting algorithm for additive model estimation

1. Initialize:

$$\begin{aligned} \hat{\alpha} &= \frac{1}{N}\sum_{n=1}^{N} y_n \\ \hat{f}_d &= 0, \quad \forall i, j \end{aligned}$$

2. Iterate q times for $j = 1, 2, \ldots, p$:

Backfitting algorithm for additive model estimation

1. Initialize:

$$\begin{aligned} \hat{\alpha} &= \frac{1}{N}\sum_{n=1}^{N} y_n \\ \hat{f}_d &= 0, \quad \forall i, j \end{aligned}$$

2. Iterate q times for $j = 1, 2, \ldots, p$:

$$\hat{f}_d \leftarrow \boldsymbol{S}_d \left[\left\{ y_n - \hat{\alpha} - \sum_{d' \neq j} \hat{f}_{d'}(x_{id'}) \right\}_1^N \right]$$

Backfitting algorithm for additive model estimation

1. Initialize:

$$\begin{aligned} \hat{\alpha} &= \frac{1}{N}\sum_{n=1}^{N} y_n \\ \hat{f}_d &= 0, \quad \forall i, j \end{aligned}$$

2. Iterate q times for $j = 1, 2, \ldots, p$:

$$\begin{aligned} \hat{f}_d &\leftarrow \boldsymbol{S}_d \left[\left\{ y_n - \hat{\alpha} - \sum_{d' \neq j} \hat{f}_{d'}(x_{id'}) \right\}_1^N \right] \\ \hat{f}_d &\leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{ij}) \end{aligned}$$

Backfitting algorithm for additive model estimation

1. Initialize:

$$\begin{aligned}\hat{\alpha} &= \frac{1}{N}\sum_{n=1}^{N} y_n \\ \hat{f}_d &= 0, \quad \forall i, j\end{aligned}$$

2. Iterate q times for $j = 1, 2, \ldots, p$:

$$\begin{aligned}\hat{f}_d &\leftarrow \boldsymbol{S}_d\left[\left\{y_n - \hat{\alpha} - \sum_{d' \neq j} \hat{f}_{d'}(x_{id'})\right\}_1^N\right] \\ \hat{f}_d &\leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{ij})\end{aligned}$$

until $|\hat{f}_d^{(q)} - \hat{f}_d^{(q-1)}| \leq \texttt{tol}$ (tolerance/stopping threshold).

Backfitting algorithm for additive model estimation

1. Initialize:

$$\begin{aligned}\hat{\alpha} &= \frac{1}{N}\sum_{n=1}^{N} y_n \\ \hat{f}_d &= 0, \quad \forall i, j\end{aligned}$$

2. Iterate q times for $j = 1, 2, \ldots, p$:

$$\begin{aligned}\hat{f}_d &\leftarrow \boldsymbol{S}_d \left[\left\{ y_n - \hat{\alpha} - \sum_{d' \neq j} \hat{f}_{d'}(x_{id'}) \right\}_1^N \right] \\ \hat{f}_d &\leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{ij})\end{aligned}$$

until $|\hat{f}_d^{(q)} - \hat{f}_d^{(q-1)}| \leq \texttt{tol}$ (tolerance/stopping threshold).

Note: $\boldsymbol{S}_d$ is the natural cubic spline smoother matrix

Notes on backfitting algorithm

- Analogous to multiple regression for linear models

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
    - local polynomial regression

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
  - local polynomial regression
  - kernel

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
  - local polynomial regression
  - kernel
  - periodic

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
    - local polynomial regression
    - kernel
    - periodic
    - D-order basis function projection

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
  - local polynomial regression
  - kernel
  - periodic
  - D-order basis function projection
- $\boldsymbol{S}_d$ is specified as the natural cubic spline smoother matrix, and is given by:

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
  - local polynomial regression
  - kernel
  - periodic
  - D-order basis function projection
- $\boldsymbol{S}_d$ is specified as the natural cubic spline smoother matrix, and is given by:

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
  - local polynomial regression
  - kernel
  - periodic
  - D-order basis function projection
- $\boldsymbol{S}_d$ is specified as the natural cubic spline smoother matrix, and is given by:

$$\boldsymbol{S}_d = \boldsymbol{N}_d(\boldsymbol{N}_d^\top \boldsymbol{N}_d - \lambda\boldsymbol{\Omega}_d)^{-1}\boldsymbol{N}_d^\top \tag{28}$$

Notes on backfitting algorithm

- Analogous to multiple regression for linear models
- The idea is to estimate a nonlinear function for each feature x_d while keeping the other features constant
- The initial value of $\hat{\alpha} = \overline{\boldsymbol{y}}$ remains unchanged during iterations
- The update $\hat{f}_d \leftarrow \hat{f}_d - \frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd})$ is only required due to computational rounding, as otherwise, $\frac{1}{N}\sum_{n=1}^{N} \hat{f}_d(x_{nd}) \sim 0$
- The smoothing operator $\boldsymbol{S}_d$ can have several specifications, e.g.
  - local polynomial regression
  - kernel
  - periodic
  - D-order basis function projection
- $\boldsymbol{S}_d$ is specified as the natural cubic spline smoother matrix, and is given by:

$$\boldsymbol{S}_d = \boldsymbol{N}_d(\boldsymbol{N}_d^\top \boldsymbol{N}_d - \lambda\Omega_d)^{-1}\boldsymbol{N}_d^\top \tag{28}$$

where $\boldsymbol{N}_d$ is the matrix of the natural cubic spline basis functions for feature j

Example: fitting a GAM to predict wages

Example: fitting a GAM to predict wages

Three predictors: 2 continuous, 1 factor

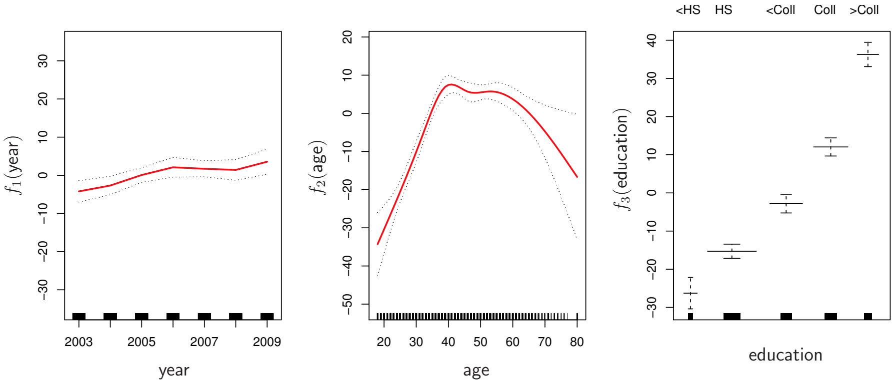


Figure: GAM estimate for Wage on year, age and education.

Example: fitting a GAM to predict wages

Three predictors: 2 continuous, 1 factor

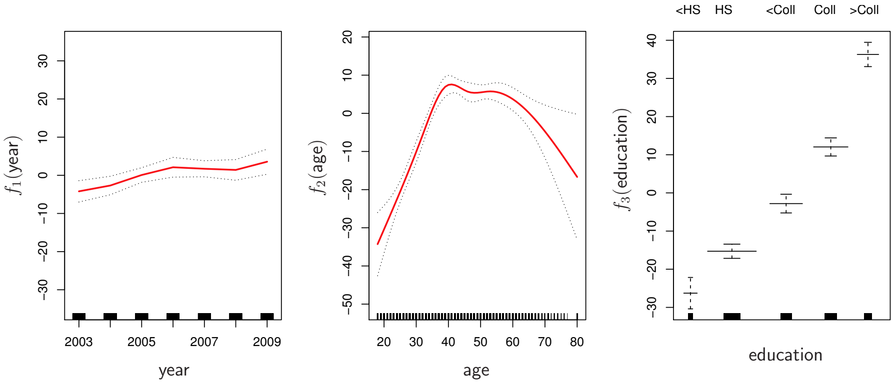


Figure: GAM estimate for Wage on year, age and education.

GAMs: classification setting

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

and:

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

and:

$$p =$$

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

and:

$$p = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_p x_p)}} \tag{31}$$

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

and:

$$p = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_p x_p)}} \tag{31}$$

p represents the **mean of response** y_n, and thus can be expressed as μ_n
The logit function is used as the *link* between input $\boldsymbol{x}$ and p:

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

and:

$$p = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_p x_p)}} \tag{31}$$

p represents the **mean of response** y_n, and thus can be expressed as μ_n
The logit function is used as the *link* between input $\boldsymbol{x}$ and p:

$$\text{logit(p)} = \log\left(\frac{p}{1-p}\right) =$$

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

and:

$$p = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_p x_p)}} \tag{31}$$

p represents the **mean of response** y_n, and thus can be expressed as μ_n
The logit function is used as the *link* between input $\boldsymbol{x}$ and p:

$$\text{logit(p)} = \log\left(\frac{p}{1-p}\right) = w_0 + w_1 x_1 + \cdots + w_p x_p \tag{32}$$

GAMs: classification setting

In logistic regression, we address classification by estimating the probability p, where:

$$p = p(y = 1|\boldsymbol{x}), \quad \text{(in the 0/1 binary case)} \tag{29}$$

Recall that:

$$1 - p = p(y = 0|\boldsymbol{x}) \tag{30}$$

and:

$$p = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_p x_p)}} \tag{31}$$

p represents the **mean of response** y_n, and thus can be expressed as μ_n
The logit function is used as the *link* between input $\boldsymbol{x}$ and p:

$$\text{logit(p)} = \log\left(\frac{p}{1-p}\right) = w_0 + w_1 x_1 + \cdots + w_p x_p \tag{32}$$

Additive logistic regression model

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu)$$

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\mathrm{logit}(\mu) = \mathrm{logit}(\mathrm{p})$$

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\mathrm{logit}(\mu) = \mathrm{logit}(\mathrm{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

$$\mathrm{logit}(\mu) = \alpha + w^\top \boldsymbol{x} + f(\boldsymbol{z}) \tag{34}$$

- Interaction effects:

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

$$\text{logit}(\mu) = \alpha + w^\top \boldsymbol{x} + f(\boldsymbol{z}) \tag{34}$$

- Interaction effects:

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

$$\text{logit}(\mu) = \alpha + w^\top \boldsymbol{x} + f(\boldsymbol{z}) \tag{34}$$

- Interaction effects:

$$\text{logit}(\mu) = f(\boldsymbol{x}) + g_k(\boldsymbol{z}) \tag{35}$$

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

$$\text{logit}(\mu) = \alpha + w^\top \boldsymbol{x} + f(\boldsymbol{z}) \tag{34}$$

- Interaction effects:

$$\text{logit}(\mu) = f(\boldsymbol{x}) + g_k(\boldsymbol{z}) \tag{35}$$

  k levels of input $\boldsymbol{x}$ against $\boldsymbol{z}$

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

$$\text{logit}(\mu) = \alpha + w^\top \boldsymbol{x} + f(\boldsymbol{z}) \tag{34}$$

- Interaction effects:

$$\text{logit}(\mu) = f(\boldsymbol{x}) + g_k(\boldsymbol{z}) \tag{35}$$

k levels of input $\boldsymbol{x}$ against $\boldsymbol{z}$

Model estimation

Additive logistic regression model

In the additive case, we replace each linear term by a general functional form to obtain the **logistic regression GAM**:

$$\text{logit}(\mu) = \text{logit}(\text{p}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \tag{33}$$

The additive model can be specified in a variety of ways, e.g:

- Semiparametric:

$$\text{logit}(\mu) = \alpha + w^\top \boldsymbol{x} + f(\boldsymbol{z}) \tag{34}$$

- Interaction effects:

$$\text{logit}(\mu) = f(\boldsymbol{x}) + g_k(\boldsymbol{z}) \tag{35}$$

  k levels of input $\boldsymbol{x}$ against $\boldsymbol{z}$

Model estimation

The additive logistic regression model can also be solved via backfitting using iteratively reweighted least squares for the model estimation in the second step (Newton-Raphson for maximum likelihood)[a]

[a]See ESL page 300 for a description of this algorithm.

Piecewise-linear fit

Piecewise-linear fit

Piecewise-linear basis functions (2 knots); no continuity:

$$\begin{pmatrix} h_{0,0}(x) = 1_{|x \leq \xi_1|} & h_{1,0}(x) = 1_{|\xi_1 \leq x \leq \xi_2|} & h_{2,0}(x) = 1_{|x \geq \xi_2|} \\ h_{0,1}(x) = 1_{|x \leq \xi_1|}x & h_{1,1}(x) = 1_{|\xi_1 \leq x \leq \xi_2|}x & h_{2,1}(x) = 1_{|x \geq \xi_2|}x \end{pmatrix} \tag{36}$$

Piecewise-linear fit

Piecewise-linear basis functions (2 knots); no continuity:

$$\begin{pmatrix} h_{0,0}(x) = 1_{|x \leq \xi_1|} & h_{1,0}(x) = 1_{|\xi_1 \leq x \leq \xi_2|} & h_{2,0}(x) = 1_{|x \geq \xi_2|} \\ h_{0,1}(x) = 1_{|x \leq \xi_1|}x & h_{1,1}(x) = 1_{|\xi_1 \leq x \leq \xi_2|}x & h_{2,1}(x) = 1_{|x \geq \xi_2|}x \end{pmatrix} \tag{36}$$

Generalizing to D-order polynomial with K knots, this gives $(K+1) \times (D+1)$ parameters

- Achieve continuity by imposing equality at the knots:

Piecewise-linear fit

Piecewise-linear basis functions (2 knots); no continuity:

$$\begin{pmatrix} h_{0,0}(x) = 1_{|x \le \xi_1|} & h_{1,0}(x) = 1_{|\xi_1 \le x \le \xi_2|} & h_{2,0}(x) = 1_{|x \ge \xi_2|} \\ h_{0,1}(x) = 1_{|x \le \xi_1|}x & h_{1,1}(x) = 1_{|\xi_1 \le x \le \xi_2|}x & h_{2,1}(x) = 1_{|x \ge \xi_2|}x \end{pmatrix} \tag{36}$$

Generalizing to D-order polynomial with K knots, this gives $(K+1) \times (D+1)$ parameters

- Achieve continuity by imposing equality at the knots:

Piecewise-linear fit

Piecewise-linear basis functions (2 knots); no continuity:

$$\begin{pmatrix} h_{0,0}(x) = 1_{|x \leq \xi_1|} & h_{1,0}(x) = 1_{|\xi_1 \leq x \leq \xi_2|} & h_{2,0}(x) = 1_{|x \geq \xi_2|} \\ h_{0,1}(x) = 1_{|x \leq \xi_1|} x & h_{1,1}(x) = 1_{|\xi_1 \leq x \leq \xi_2|} x & h_{2,1}(x) = 1_{|x \geq \xi_2|} x \end{pmatrix} \tag{36}$$

Generalizing to D-order polynomial with K knots, this gives $(K+1) \times (D+1)$ parameters

- Achieve continuity by imposing equality at the knots:

$$\xi_1 : w_{0,0} + \xi_1 w_{0,1} = w_{1,0} + \xi_1 w_{1,1} \tag{37}$$

$$\xi_2 : w_{1,0} + \xi_1 w_{1,1} = w_{1,2} + \xi_2 w_{1,1} \tag{38}$$

- This frees up 2 degrees of freedom

Piecewise-linear fit

Piecewise-linear basis functions (2 knots); no continuity:

$$\begin{pmatrix} h_{0,0}(x) = 1_{|x \leq \xi_1|} & h_{1,0}(x) = 1_{|\xi_1 \leq x \leq \xi_2|} & h_{2,0}(x) = 1_{|x \geq \xi_2|} \\ h_{0,1}(x) = 1_{|x \leq \xi_1|}x & h_{1,1}(x) = 1_{|\xi_1 \leq x \leq \xi_2|}x & h_{2,1}(x) = 1_{|x \geq \xi_2|}x \end{pmatrix} \tag{36}$$

Generalizing to D-order polynomial with K knots, this gives $(K+1) \times (D+1)$ parameters

- Achieve continuity by imposing equality at the knots:

$$\xi_1 : w_{0,0} + \xi_1 w_{0,1} = w_{1,0} + \xi_1 w_{1,1} \tag{37}$$

$$\xi_2 : w_{1,0} + \xi_1 w_{1,1} = w_{1,2} + \xi_2 w_{1,1} \tag{38}$$

- This frees up 2 degrees of freedom
- Only 4 basis functions are then needed

Truncated-power basis

Truncated-power basis

- For a piecewise-linear fit wtih 2 knots, continuity can be specified using the truncated-power basis:

Truncated-power basis

- For a piecewise-linear fit wtih 2 knots, continuity can be specified using the truncated-power basis:

Truncated-power basis

- For a piecewise-linear fit wtih 2 knots, continuity can be specified using the truncated-power basis:

$$\begin{pmatrix} h_0(x) = 1 & h_1(x) = x \\ h_2(x) = (x - \xi_1)_+ & h_3(x) = (x - \xi_2)_+ \end{pmatrix} \tag{39}$$

Truncated-power basis

- For a piecewise-linear fit wtih 2 knots, continuity can be specified using the truncated-power basis:

$$\begin{pmatrix} h_0(x) = 1 & h_1(x) = x \\ h_2(x) = (x - \xi_1)_+ & h_3(x) = (x - \xi_2)_+ \end{pmatrix} \tag{39}$$

where $(x - \xi_k)_+ = \begin{cases} x - \xi_k, & x \geq \xi_k \\ 0, & \text{otherwise} \end{cases}$

- For an order M polynomial, the number of basis functions required is thus:

Truncated-power basis

- For a piecewise-linear fit wtih 2 knots, continuity can be specified using the truncated-power basis:

$$\begin{pmatrix} h_0(x) = 1 & h_1(x) = x \\ h_2(x) = (x - \xi_1)_+ & h_3(x) = (x - \xi_2)_+ \end{pmatrix} \tag{39}$$

where $(x - \xi_k)_+ = \begin{cases} x - \xi_k, & x \geq \xi_k \\ 0, & \text{otherwise} \end{cases}$

- For an order M polynomial, the number of basis functions required is thus:

Truncated-power basis

- For a piecewise-linear fit wtih 2 knots, continuity can be specified using the truncated-power basis:

$$\begin{pmatrix} h_0(x) = 1 & h_1(x) = x \\ h_2(x) = (x - \xi_1)_+ & h_3(x) = (x - \xi_2)_+ \end{pmatrix} \tag{39}$$

where $(x - \xi_k)_+ = \begin{cases} x - \xi_k, & x \geq \xi_k \\ 0, & \text{otherwise} \end{cases}$

- For an order M polynomial, the number of basis functions required is thus:

$$(K + 1) \times (M + 1) - K \times M = K + M + 1 \tag{40}$$

Regression spline (truncated-power basis)

Regression spline (truncated-power basis)

- General form of truncated-power basis:

Regression spline (truncated-power basis)

- General form of truncated-power basis:

Regression spline (truncated-power basis)

- General form of truncated-power basis:

$$h_m(x) = x^m, \quad m = 0, \ldots, M \tag{41}$$

$$h_{D+k}(x) = (x - \xi_k)_+^D, \quad k = 1, \ldots, K \tag{42}$$

Regression spline (truncated-power basis)

- General form of truncated-power basis:

$$h_m(x) = x^m, \quad m = 0, \ldots, M \tag{41}$$

$$k = 1, \ldots, K \tag{42}$$

Regression spline (truncated-power basis)

- General form of truncated-power basis:

$$h_m(x) = x^m, \quad m = 0, \ldots, M \tag{41}$$

$$h_{D+k}(x) = (x - \xi_k)_+^D, \quad k = 1, \ldots, K \tag{42}$$

- Achieves both continuity and smoothness

Regression spline (truncated-power basis)

- General form of truncated-power basis:

$$h_m(x) = x^m, \quad m = 0, \ldots, M \tag{41}$$

$$h_{D+k}(x) = (x - \xi_k)_+^D, \quad k = 1, \ldots, K \tag{42}$$

- Achieves both continuity and smoothness
- Degrees of freedom (number of parameters):

$$(K+1) \times (M+1) - K \times M = K + M + 1 \tag{43}$$

Natural cubic spline

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$

$$S_1(x) = x \tag{45}$$

$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

where

$$d_k(x) = \frac{(x - \xi_k)_+^3 - (x - \xi_K)_+^3}{\xi_K - \xi_k} \tag{47}$$

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$

$$S_1(x) = x \tag{45}$$

$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

where

$$d_k(x) = \frac{(x-\xi_k)_+^3 - (x-\xi_K)_+^3}{\xi_K - \xi_k} \tag{47}$$

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$

$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

where

$$d_k(x) = \frac{(x - \xi_k)^3_+ - (x - \xi_K)^3_+}{\xi_K - \xi_k} \tag{47}$$

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$

$$S_1(x) = x \tag{45}$$

$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$
$$S_1(x) = x \tag{45}$$
$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

where

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$

$$S_1(x) = x \tag{45}$$

$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

where

$$d_k(x) = \frac{(x - \xi_k)^3_+ - (x - \xi_K)^3_+}{\xi_K - \xi_k} \tag{47}$$

- Recall: regression spline has $K + M + 1$ basis functions

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$

$$S_1(x) = x \tag{45}$$

$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

where

$$d_k(x) = \frac{(x - \xi_k)_+^3 - (x - \xi_K)_+^3}{\xi_K - \xi_k} \tag{47}$$

- Recall: regression spline has $K + M + 1$ basis functions
- Here, $M = 3$, and since 4 are freed up, then $df = K + 3 + 1 - 4 = K$.

Natural cubic spline

- Reduces variance by ensuring linearity at the boundary knots
- Two constraints at each boundary knot: $f''(x) = f'''(x) = 0$
- Frees up 4 degrees of freedom
- General representation:

$$S_0(x) = 1 \tag{44}$$

$$S_1(x) = x \tag{45}$$

$$S_{1+k}(x) = d_k(x) - d_{K-1}(x) \tag{46}$$

where

$$d_k(x) = \frac{(x - \xi_k)^3_+ - (x - \xi_K)^3_+}{\xi_K - \xi_k} \tag{47}$$

- Recall: regression spline has $K + M + 1$ basis functions
- Here, $M = 3$, and since 4 are freed up, then $df = K + 3 + 1 - 4 = K$.
- So, same number of basis functions as there are knots.

Smoothing spline

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x) \hat{w}_k \tag{49}$$

- We can also write:

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x)\hat{w}_k \tag{49}$$

- We can also write:

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x) \hat{w}_k \tag{49}$$

- We can also write:

$$PRSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{50}$$

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x) \hat{w}_k \tag{49}$$

- We can also write:

$$PRSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{50}$$

where $S_{nk} = S_k(x_n)$ and $\Omega_{k,k'} = \int S''_k(t) S''_{k'}(t) dt$
- $\hat{\boldsymbol{w}} = (\boldsymbol{S}^T \boldsymbol{S} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{N}^T \boldsymbol{y}$

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x) \hat{w}_k \tag{49}$$

- We can also write:

$$PRSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{50}$$

where $S_{nk} = S_k(x_n)$ and $\Omega_{k,k'} = \int S''_k(t) S''_{k'}(t) dt$
- $\hat{\boldsymbol{w}} = (\boldsymbol{S}^T \boldsymbol{S} + \lambda \Omega)^{-1} \boldsymbol{N}^T \boldsymbol{y}$
- The smoothing spline is thus:

$$\hat{\boldsymbol{f}} = \boldsymbol{S}(\boldsymbol{S}^T \boldsymbol{S} + \lambda \Omega)^{-1} \boldsymbol{N}^T \boldsymbol{y} = \boldsymbol{S}_\lambda y \tag{51}$$

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x) \hat{w}_k \tag{49}$$

- We can also write:

$$PRSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{Sw})^T (\boldsymbol{y} - \boldsymbol{Sw}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{50}$$

  where $S_{nk} = S_k(x_n)$ and $\Omega_{k,k'} = \int S_k''(t) S_{k'}''(t) dt$
- $\hat{\boldsymbol{w}} = (\boldsymbol{S}^T \boldsymbol{S} + \lambda \Omega)^{-1} \boldsymbol{N}^T \boldsymbol{y}$
- The smoothing spline is thus:

$$\hat{\boldsymbol{f}} = \boldsymbol{S}(\boldsymbol{S}^T \boldsymbol{S} + \lambda \Omega)^{-1} \boldsymbol{N}^T \boldsymbol{y} = \boldsymbol{S}_\lambda y \tag{51}$$

Smoothing spline

- Introduce roughness penalty:

$$PRSS(\boldsymbol{w}, \lambda) = \sum_{i=1}^{n} [y_n - f(x)]^2 + \lambda \int [f''(t)]^2 \, dt \tag{48}$$

- Solution is a shrunken (damped) natural cubic spline:

$$\hat{f}(x) = \sum_{k=1}^{K} S_k(x) \hat{w}_k \tag{49}$$

- We can also write:

$$PRSS(\boldsymbol{w}, \lambda) = (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{S}\boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{\Omega} \boldsymbol{w} \tag{50}$$

  where $S_{nk} = S_k(x_n)$ and $\Omega_{k,k'} = \int S_k''(t) S_{k'}''(t) dt$
- $\hat{\boldsymbol{w}} = (\boldsymbol{S}^T \boldsymbol{S} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{N}^T \boldsymbol{y}$
- The smoothing spline is thus:

$$\hat{\boldsymbol{f}} = \boldsymbol{S}(\boldsymbol{S}^T \boldsymbol{S} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{N}^T \boldsymbol{y} = \boldsymbol{S}_\lambda y \tag{51}$$

  where $\boldsymbol{S}_\lambda$ is the **shrinking smoother**.

Reading

- **PMLI** 11.5
- **ESL** 5.2-5

Piecewise functions: constant and linear formulations

Piecewise functions: constant and linear formulations

- One constant per region:

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) =$$

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) = \sum_{m=0}^{M-1} w_m h_m(X)$$

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) = \sum_{m=0}^{M-1} w_m h_m(X) = \sum_{m=0}^{M-1} w_m \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{52}$$

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) = \sum_{m=0}^{M-1} w_m h_m(X) = \sum_{m=0}^{M-1} w_m \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{52}$$

This is **piecewise constant**

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) = \sum_{m=0}^{M-1} w_m h_m(X) = \sum_{m=0}^{M-1} w_m \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{52}$$

  This is **piecewise constant**

- One linear model per region:

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) = \sum_{m=0}^{M-1} w_m h_m(X) = \sum_{m=0}^{M-1} w_m \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{52}$$

  This is **piecewise constant**

- One linear model per region:

$$f(X) =$$

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) = \sum_{m=0}^{M-1} w_m h_m(X) = \sum_{m=0}^{M-1} w_m \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{52}$$

  This is **piecewise constant**

- One linear model per region:

$$f(X) = \sum_{m=0}^{M-1} \left[w_{0,m} + w_{1,m} X \right] \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{53}$$

Piecewise functions: constant and linear formulations

- One constant per region:

$$f(X) = \sum_{m=0}^{M-1} w_m h_m(X) = \sum_{m=0}^{M-1} w_m \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{52}$$

  This is **piecewise constant**

- One linear model per region:

$$f(X) = \sum_{m=0}^{M-1} \left[w_{0,m} + w_{1,m} X\right] \mathbb{I}(\xi_m \leq X < \xi_{m+1}) \tag{53}$$

  This is **piecewise linear**

Piecewise functions (cont.)

Piecewise functions (cont.)

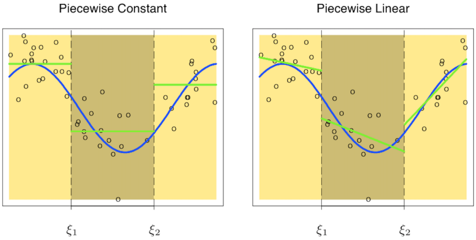
- Individual models follow same principles of *linear regression*

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*

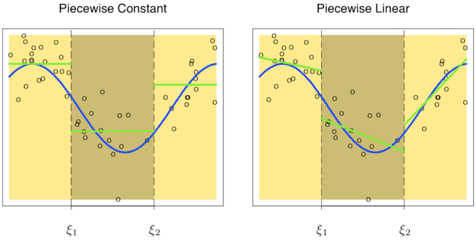


(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

$$h_0(X) = \mathbb{I}(X < \xi_1)$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*

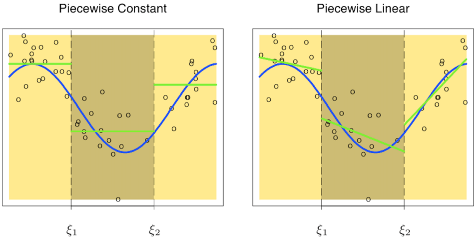


(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

$$h_0(X) = \mathbb{I}(X < \xi_1)$$
$$h_1(X) = \mathbb{I}(\xi_1 \leq X \leq \xi_2)$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*

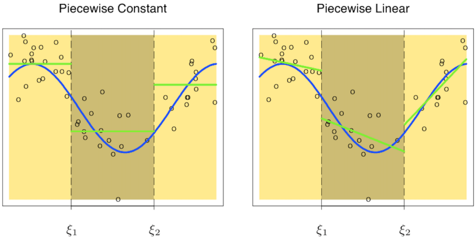


(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

$$\begin{aligned} h_0(X) &= \mathbb{I}(X < \xi_1) \\ h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\ h_2(X) &= \mathbb{I}(\xi_2 \leq X) \end{aligned}$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*



(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

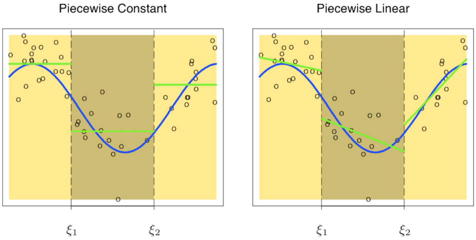
$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \leq X)
\end{aligned}
$$

(Right) Piecewise linear function (green) fitted to simulated dataset whose true function is shown in blue: Basis functions:

$$
h_0(X) = \mathbb{I}(X < \xi_1)
$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*



(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

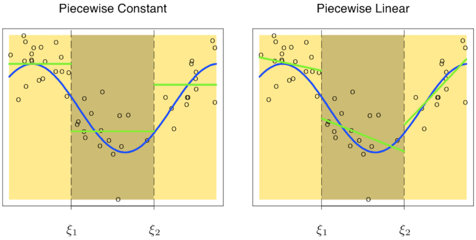
$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \leq X)
\end{aligned}
$$

(Right) Piecewise linear function (green) fitted to simulated dataset whose true function is shown in blue: Basis functions:

$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2)
\end{aligned}
$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*



(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

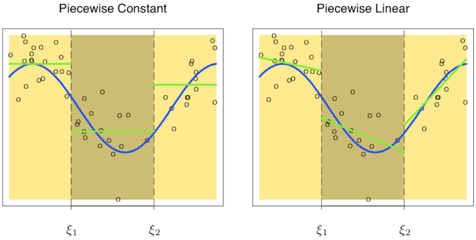
$$\begin{aligned} h_0(X) &= \mathbb{I}(X < \xi_1) \\ h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\ h_2(X) &= \mathbb{I}(\xi_2 \leq X) \end{aligned}$$

(Right) Piecewise linear function (green) fitted to simulated dataset whose true function is shown in blue: Basis functions:

$$\begin{aligned} h_0(X) &= \mathbb{I}(X < \xi_1) \\ h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\ h_2(X) &= \mathbb{I}(\xi_2 \leq X) \end{aligned}$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*



(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

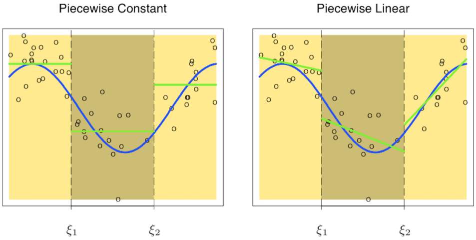
$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \leq X)
\end{aligned}
$$

(Right) Piecewise linear function (green) fitted to simulated dataset whose true function is shown in blue: Basis functions:

$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \leq X) \\
h_3(X) &= X\mathbb{I}(X < \xi_1)
\end{aligned}
$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*



(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

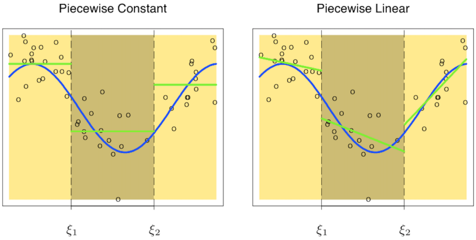
$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \leq X)
\end{aligned}
$$

(Right) Piecewise linear function (green) fitted to simulated dataset whose true function is shown in blue: Basis functions:

$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \leq X \leq \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \leq X) \\
h_3(X) &= X\mathbb{I}(X < \xi_1) \\
h_4(X) &= X\mathbb{I}(\xi_1 \leq X \leq \xi_2)
\end{aligned}
$$

Piecewise functions (cont.)

- Individual models follow same principles of *linear regression*



(Left) Piecewise constant function (green) fitted to a simulated dataset whose true function is shown in blue. Basis functions:

$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \le X \le \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \le X)
\end{aligned}
$$

(Right) Piecewise linear function (green) fitted to simulated dataset whose true function is shown in blue: Basis functions:

$$
\begin{aligned}
h_0(X) &= \mathbb{I}(X < \xi_1) \\
h_1(X) &= \mathbb{I}(\xi_1 \le X \le \xi_2) \\
h_2(X) &= \mathbb{I}(\xi_2 \le X) \\
h_3(X) &= X\mathbb{I}(X < \xi_1) \\
h_4(X) &= X\mathbb{I}(\xi_1 \le X \le \xi_2) \\
h_5(X) &= X\mathbb{I}(\xi_2 \le X)
\end{aligned}
$$

Piecewise-linear basis functions

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$h_0(X) = 1 \tag{54}$$

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$h_0(X) = 1 \tag{54}$$

$$h_1(X) = X \tag{55}$$

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$h_0(X) = 1 \tag{54}$$

$$h_1(X) = X \tag{55}$$

$$h_2(X) = (X - \xi_1)_+ \tag{56}$$

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$h_0(X) = 1 \tag{54}$$

$$h_1(X) = X \tag{55}$$

$$h_2(X) = (X - \xi_1)_+ \tag{56}$$

$$h_3(X) = (X - \xi_2)_+ \tag{57}$$

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$h_0(X) = 1 \tag{54}$$

$$h_1(X) = X \tag{55}$$

$$h_2(X) = (X - \xi_1)_+ \tag{56}$$

$$h_3(X) = (X - \xi_2)_+ \tag{57}$$

where,

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$h_0(X) = 1 \quad (54)$$

$$h_1(X) = X \quad (55)$$

$$h_2(X) = (X - \xi_1)_+ \quad (56)$$

$$h_3(X) = (X - \xi_2)_+ \quad (57)$$

where,

$$(X - \xi_1)_+ = \begin{cases} X - \xi_1, & X \geq \xi_1 \\ & \end{cases}$$

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$\begin{aligned} h_0(X) &= 1 && (54) \\ h_1(X) &= X && (55) \\ h_2(X) &= (X - \xi_1)_+ && (56) \\ h_3(X) &= (X - \xi_2)_+ && (57) \end{aligned}$$

where,

$$(X - \xi_1)_+ = \begin{cases} X - \xi_1, & X \geq \xi_1 \\ 0, & \text{otherwise} \end{cases} \tag{58}$$

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$\begin{aligned} h_0(X) &= 1 && (54) \\ h_1(X) &= X && (55) \\ h_2(X) &= (X - \xi_1)_+ && (56) \\ h_3(X) &= (X - \xi_2)_+ && (57) \end{aligned}$$

where,

$$(X - \xi_1)_+ = \begin{cases} X - \xi_1, & X \geq \xi_1 \\ 0, & \text{otherwise} \end{cases} \tag{58}$$

The model can then be written as:

Piecewise-linear basis functions

- To ensure continuity at the knots, we need to constrain the basis functions to be equal at both ξ_1 and ξ_2.
- This means we lose 2 degrees of freedom and only need to fit $6 - 2 = 4$ basis functions.

The basis functions for a continuous piecewise-linear fit with two knots are thus:

$$\begin{aligned} h_0(X) &= 1 && (54) \\ h_1(X) &= X && (55) \\ h_2(X) &= (X - \xi_1)_+ && (56) \\ h_3(X) &= (X - \xi_2)_+ && (57) \end{aligned}$$

where,

$$(X - \xi_1)_+ = \begin{cases} X - \xi_1, & X \geq \xi_1 \\ 0, & \text{otherwise} \end{cases} \tag{58}$$

The model can then be written as:

$$y_i = w_0 + w_1 x_i + w_2(x_i - \xi_1)_+ + w_3(x_i - \xi_2)_+ + \epsilon_i \tag{59}$$

Piecewise-linear basis functions (cont.)

Piecewise-linear basis functions (cont.)

- When entering a new region, the added linear component alters the slope

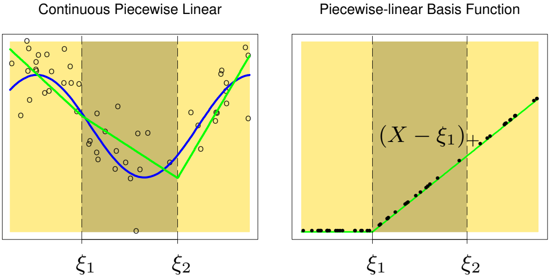


Figure: (Left): Piecewise-linear fit with continuity (but not smoothness) at the knots. $f(X) = w_0 + w_1 X + w_2(X - \xi_1)_+ + w_3(X - \xi_2)_+$.
(Right): Piecewise-linear fit with continuity at ξ_1. $f(X) = w_2(X - \xi_1)_+$

Piecewise polynomials

Piecewise polynomials

- To achieve smoothness, we increase the degree of the polynomial, adding one **truncated power basis function** per knot:

$$h(X, \xi) =$$

Piecewise polynomials

- To achieve smoothness, we increase the degree of the polynomial, adding one **truncated power basis function** per knot:

$$h(X, \xi) = (X - \xi)_+^D =$$

Piecewise polynomials

- To achieve smoothness, we increase the degree of the polynomial, adding one **truncated power basis function** per knot:

$$h(X,\xi) = (X-\xi)_+^D = \begin{cases} (X-\xi)^D, & X > \xi \\ & \end{cases}$$

Piecewise polynomials

- To achieve smoothness, we increase the degree of the polynomial, adding one **truncated power basis function** per knot:

$$h(X,\xi) = (X-\xi)_+^D = \begin{cases} (X-\xi)^D, & X > \xi \\ 0, & \text{otherwise} \end{cases} \tag{60}$$

Piecewise polynomials

- To achieve smoothness, we increase the degree of the polynomial, adding one **truncated power basis function** per knot:

$$h(X, \xi) = (X - \xi)_+^D = \begin{cases} (X - \xi)^D, & X > \xi \\ 0, & \text{otherwise} \end{cases} \tag{60}$$

  where D is the degree of the piecewise polynomial

Piecewise polynomials

- To achieve smoothness, we increase the degree of the polynomial, adding one **truncated power basis function** per knot:

$$h(X,\xi) = (X-\xi)_+^D = \begin{cases}(X-\xi)^D, & X > \xi \\ 0, & \text{otherwise}\end{cases} \tag{60}$$

where D is the degree of the piecewise polynomial

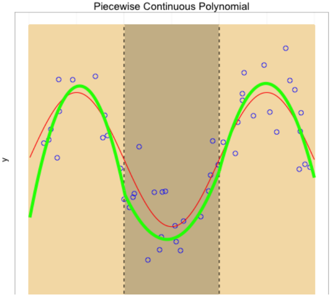


Figure: Piecewise-quadratic fit: $f(X) = w_0 + w_1 X + w_2 X^2 + w_3 (X-\xi_1)_+^2 + w_4 (X-\xi_2)_+^2$

Piecewise polynomials (cont.)

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$f(X) =$$

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$f(X) = \qquad w_0 + w_1 X + w_2 X^2 + \dots$$

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$\begin{aligned} f(X) = & \quad w_0 + w_1 X + w_2 X^2 + \dots \\ = & \quad + w_D X^D + w_{D+1}(X - \xi_1)_+^D + \dots + w_{D+K}(X - \xi_K)_+^D \end{aligned} \tag{61}$$

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$\begin{aligned} f(X) &= \qquad\qquad w_0 + w_1 X + w_2 X^2 + \dots \\ &= \quad + w_D X^D + w_{D+1}(X - \xi_1)_+^D + \dots + w_{D+K}(X - \xi_K)_+^D \end{aligned} \tag{61}$$

- For K knots and degree D polynomial:

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$\begin{aligned} f(X) &= w_0 + w_1 X + w_2 X^2 + \ldots \\ &= + w_D X^D + w_{D+1}(X - \xi_1)_+^D + \ldots + w_{D+K}(X - \xi_K)_+^D \end{aligned} \tag{61}$$

- For K knots and degree D polynomial:

$$h_m(X) = X^m, \quad m = 0, \ldots, D \tag{62}$$

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$\begin{aligned} f(X) = & \quad w_0 + w_1 X + w_2 X^2 + \ldots \\ = & \quad + w_D X^D + w_{D+1}(X - \xi_1)_+^D + \ldots + w_{D+K}(X - \xi_K)_+^D \end{aligned} \tag{61}$$

- For K knots and degree D polynomial:

$$h_m(X) = X^m, \quad m = 0, \ldots, D \tag{62}$$

$$h_{D+k}(X) = (X - \xi_k)_+^D, \quad k = 1, \ldots, K \tag{63}$$

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$\begin{aligned} f(X) &= w_0 + w_1 X + w_2 X^2 + \dots \\ &= + w_D X^D + w_{D+1}(X - \xi_1)_+^D + \dots + w_{D+K}(X - \xi_K)_+^D \end{aligned} \tag{61}$$

- For K knots and degree D polynomial:

$$h_m(X) = X^m, \quad m = 0, \dots, D \tag{62}$$

$$h_{D+k}(X) = (X - \xi_k)_+^D, \quad k = 1, \dots, K \tag{63}$$

- Degree $D = 3$ is a **cubic spline**

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$\begin{aligned} f(X) &= w_0 + w_1 X + w_2 X^2 + \dots \\ &= + w_D X^D + w_{D+1}(X-\xi_1)_+^D + \dots + w_{D+K}(X-\xi_K)_+^D \end{aligned} \tag{61}$$

- For K knots and degree D polynomial:

$$h_m(X) = X^m, \quad m = 0, \dots, D \tag{62}$$

$$h_{D+k}(X) = (X-\xi_k)_+^D, \quad k = 1, \dots, K \tag{63}$$

- Degree $D = 3$ is a **cubic spline**
  - Continuous first and second derivatives (smooth)

Piecewise polynomials (cont.)

Generally, we can write the basis functions for a degree D piecewise polynomial fit using the basis for a Dth order polynomial and then adding K truncated power basis functions for each knot.

- Model for degree D:

$$\begin{aligned} f(X) &= w_0 + w_1 X + w_2 X^2 + \dots \\ &= + w_D X^D + w_{D+1}(X - \xi_1)_+^D + \dots + w_{D+K}(X - \xi_K)_+^D \end{aligned} \tag{61}$$

- For K knots and degree D polynomial:

$$h_m(X) = X^m, \quad m = 0, \dots, D \tag{62}$$

$$h_{D+k}(X) = (X - \xi_k)_+^D, \quad k = 1, \dots, K \tag{63}$$

- Degree $D = 3$ is a **cubic spline**
  - Continuous first and second derivatives (smooth)
  - The most common spline degree

B-spline basis representation

B-spline basis representation

- For larger orders, the truncated power basis can become unstable

Degree-D B-spline

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence:

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence:

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [\xi_i, \xi_{i+1}) \\ & \end{cases}$$

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [\xi_i, \xi_{i+1}) \\ 0, & \text{otherwise} \end{cases} \tag{64}$$

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [\xi_i, \xi_{i+1}) \\ 0, & \text{otherwise} \end{cases} \tag{64}$$

   and by convention $B_{i,0}(x) = 0$ when $\xi_i = \xi_{i+1}$
3. The ith B-spline basis function of degree $j, j = 1, \ldots, D$ is:

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [\xi_i, \xi_{i+1}) \\ 0, & \text{otherwise} \end{cases} \tag{64}$$

   and by convention $B_{i,0}(x) = 0$ when $\xi_i = \xi_{i+1}$
3. The ith B-spline basis function of degree $j, j = 1, \ldots, D$ is:

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [\xi_i, \xi_{i+1}) \\ 0, & \text{otherwise} \end{cases} \tag{64}$$

and by convention $B_{i,0}(x) = 0$ when $\xi_i = \xi_{i+1}$

3. The ith B-spline basis function of degree $j, j = 1, \ldots, D$ is:

$$B_{i,j}(x) = \frac{x - \xi_i}{\xi_{i+j} - \xi_i} B_{i,j-1}(x) + \frac{\xi_{i+j+1} - x}{\xi_{i+j+1} - \xi_{i+1}} B_{i+1,j-1}(x) \tag{65}$$

B-spline basis representation

- For larger orders, the truncated power basis can become unstable
- B-splines are a computationally convenient alternative

Degree-D B-spline

1. Define augmented knot sequence: $\boldsymbol{\xi}^* = (\xi_{-D}, \ldots, \xi_0, \boldsymbol{\xi}, \xi_{K+1}, \ldots, \xi_{K+D+1})$
2. For $i = -D, \ldots, K + D$, let:

$$B_{i,0}(x) = \begin{cases} 1, & x \in [\xi_i, \xi_{i+1}) \\ 0, & \text{otherwise} \end{cases} \tag{64}$$

   and by convention $B_{i,0}(x) = 0$ when $\xi_i = \xi_{i+1}$
3. The ith B-spline basis function of degree $j, j = 1, \ldots, D$ is:

$$B_{i,j}(x) = \frac{x - \xi_i}{\xi_{i+j} - \xi_i} B_{i,j-1}(x) + \frac{\xi_{i+j+1} - x}{\xi_{i+j+1} - \xi_{i+1}} B_{i+1,j-1}(x) \tag{65}$$

   for $i = -D, \ldots, K + D - j$

The generalized linear model (GLM)

The generalized linear model (GLM)

- Conventional linear models have the form:

$$y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \tag{66}$$

The generalized linear model (GLM)

- Conventional linear models have the form:

$$y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \tag{66}$$

where

The generalized linear model (GLM)

- Conventional linear models have the form:

$$y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \tag{66}$$

where

- y_i is a continuous response
- $\boldsymbol{x}_i$ is a vector of quantitative and/or qualitative explanatory variables

The generalized linear model (GLM)

- Conventional linear models have the form:

$$y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \tag{66}$$

where
  - y_i is a continuous response
  - $\boldsymbol{x}_i$ is a vector of quantitative and/or qualitative explanatory variables
- Generalized linear models (GLMs) were introduced to extend this framework to allow y_i to be modeled by other exponential family distributions besides the normal/Gaussian, e.g.
  - exponential
  - binomial/multinomial (with fixed number of trials)
  - Poisson

The generalized linear model (GLM)

- Conventional linear models have the form:

$$y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \tag{66}$$

  where
  - y_i is a continuous response
  - $\boldsymbol{x}_i$ is a vector of quantitative and/or qualitative explanatory variables
- Generalized linear models (GLMs) were introduced to extend this framework to allow y_i to be modeled by other exponential family distributions besides the normal/Gaussian, e.g.
  - exponential
  - binomial/multinomial (with fixed number of trials)
  - Poisson
- In the GLM framework:

The generalized linear model (GLM)

- Conventional linear models have the form:

$$y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \tag{66}$$

where

- y_i is a continuous response
- $\boldsymbol{x}_i$ is a vector of quantitative and/or qualitative explanatory variables

- Generalized linear models (GLMs) were introduced to extend this framework to allow y_i to be modeled by other exponential family distributions besides the normal/Gaussian, e.g.
  - exponential
  - binomial/multinomial (with fixed number of trials)
  - Poisson
- In the GLM framework:
  - The mean of y_i is given by μ_i
  - μ_i can be specified by a nonlinear function of $\boldsymbol{x}_i^\top \boldsymbol{\beta}$
  - Note that the simple linear regression is a special case of GLM in which $\mu_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}$ and y_i follows a Gaussian distribution

The generalized linear model (GLM)

- Conventional linear models have the form:

$$y_i \sim N(\boldsymbol{x}_i^\top \boldsymbol{\beta}, \sigma^2) \tag{66}$$

where

  - y_i is a continuous response
  - $\boldsymbol{x}_i$ is a vector of quantitative and/or qualitative explanatory variables

- Generalized linear models (GLMs) were introduced to extend this framework to allow y_i to be modeled by other exponential family distributions besides the normal/Gaussian, e.g.
  - exponential
  - binomial/multinomial (with fixed number of trials)
  - Poisson
- In the GLM framework:
  - The mean of y_i is given by μ_i
  - μ_i can be specified by a nonlinear function of $\boldsymbol{x}_i^\top \boldsymbol{\beta}$
  - Note that the simple linear regression is a special case of GLM in which $\mu_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}$ and y_i follows a Gaussian distribution

Back to Recap

GLM components

GLM components

A GLM consists of three parts:

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $\boldsymbol{g(\mu)}$:** defines the relationship between the random and systematic components:

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $\boldsymbol{g(\mu)}$:** defines the relationship between the random and systematic components:
    - Simple linear regression (identity link function):

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $\boldsymbol{g(\mu)}$:** defines the relationship between the random and systematic components:
    - Simple linear regression (identity link function):

$$g(\mu_i) = g(E(y_i)) = \boldsymbol{x}_i^\top \boldsymbol{w} \tag{67}$$

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $\boldsymbol{g(\mu)}$:** defines the relationship between the random and systematic components:
  - Simple linear regression (identity link function):

$$g(\mu_i) = g(E(y_i)) = \boldsymbol{x}_i^\top \boldsymbol{w} \tag{67}$$

  - Binary logistic regression (logit link function):

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $g(\mu)$:** defines the relationship between the random and systematic components:
  - Simple linear regression (identity link function):

$$g(\mu_i) = g(E(y_i)) = \boldsymbol{x}_i^\top \boldsymbol{w} \tag{67}$$

  - Binary logistic regression (logit link function):

$$g(\mu_i) = g(p(\boldsymbol{x}_i)) =$$

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $\boldsymbol{g(\mu)}$:** defines the relationship between the random and systematic components:
  - Simple linear regression (identity link function):

$$g(\mu_i) = g(E(y_i)) = \boldsymbol{x}_i^\top \boldsymbol{w} \tag{67}$$

  - Binary logistic regression (logit link function):

$$g(\mu_i) = g(p(\boldsymbol{x}_i)) = \text{logit}(p(\boldsymbol{x}_i)) =$$

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $\boldsymbol{g(\mu)}$:** defines the relationship between the random and systematic components:
  - Simple linear regression (identity link function):

$$g(\mu_i) = g(E(y_i)) = \boldsymbol{x}_i^\top \boldsymbol{w} \tag{67}$$

  - Binary logistic regression (logit link function):

$$g(\mu_i) = g(p(\boldsymbol{x}_i)) = \text{logit}(p(\boldsymbol{x}_i)) = \ln\left(\frac{p(\boldsymbol{x}_i)}{1 - p(\boldsymbol{x}_i)}\right) =$$

GLM components

A GLM consists of three parts:

- **Random component:** this is the probability distribution of the response variable
- **Systematic component:** specifies the explanatory variables within the linear combination of their coefficients ($\boldsymbol{Xw}$)
- **Link function $\boldsymbol{g(\mu)}$:** defines the relationship between the random and systematic components:
  - Simple linear regression (identity link function):

$$g(\mu_i) = g(E(y_i)) = \boldsymbol{x}_i^\top \boldsymbol{w} \tag{67}$$

  - Binary logistic regression (logit link function):

$$g(\mu_i) = g(p(\boldsymbol{x}_i)) = \text{logit}(p(\boldsymbol{x}_i)) = \ln\left(\frac{p(\boldsymbol{x}_i)}{1 - p(\boldsymbol{x}_i)}\right) = \boldsymbol{x}_i^\top \boldsymbol{w} \tag{68}$$

Assumptions of GLM

Assumptions of GLM

- The observations of the response variable $\boldsymbol{y}$ are i.i.d.

Assumptions of GLM

- The observations of the response variable $\boldsymbol{y}$ are i.i.d.
- Response variable y_i is typically exponentially distributed (not restricted to being normally distributed)

Assumptions of GLM

- The observations of the response variable $\boldsymbol{y}$ are i.i.d.
- Response variable y_i is typically exponentially distributed (not restricted to being normally distributed)
  - Implies that errors need not be normally distributed (but should be independent)
- Link function is linear with respect to the coefficients (w_j)

Assumptions of GLM

- The observations of the response variable $\boldsymbol{y}$ are i.i.d.
- Response variable y_i is typically exponentially distributed (not restricted to being normally distributed)
  - Implies that errors need not be normally distributed (but should be independent)
- Link function is linear with respect to the coefficients (w_j)
  - Relationship between response and explanatory variables does not have to be linear

Assumptions of GLM

- The observations of the response variable $\boldsymbol{y}$ are i.i.d.
- Response variable y_i is typically exponentially distributed (not restricted to being normally distributed)
  - Implies that errors need not be normally distributed (but should be independent)
- Link function is linear with respect to the coefficients (w_j)
  - Relationship between response and explanatory variables does not have to be linear
  - Explanatory variables can be nonlinear transformations of original values (as in simple linear regression)

Assumptions of GLM

- The observations of the response variable $\boldsymbol{y}$ are i.i.d.
- Response variable y_i is typically exponentially distributed (not restricted to being normally distributed)
    - Implies that errors need not be normally distributed (but should be independent)
- Link function is linear with respect to the coefficients (w_j)
    - Relationship between response and explanatory variables does not have to be linear
    - Explanatory variables can be nonlinear transformations of original values (as in simple linear regression)
- Variance may not homogeneous (i.e. homoscedasticity is not a requirement)

Assumptions of GLM

- The observations of the response variable $\boldsymbol{y}$ are i.i.d.
- Response variable y_i is typically exponentially distributed (not restricted to being normally distributed)
  - Implies that errors need not be normally distributed (but should be independent)
- Link function is linear with respect to the coefficients (w_j)
  - Relationship between response and explanatory variables does not have to be linear
  - Explanatory variables can be nonlinear transformations of original values (as in simple linear regression)
- Variance may not homogeneous (i.e. homoscedasticity is not a requirement)
- Parameters are estimated via MLE

Commonly used GLM models and their components

Commonly used GLM models and their components

Model	**Random component**	**Link function**
Linear regression	Gaussian	Identity: $g(\mu_i) = \mu_i = w^\top x_i$
Binary logistic regression	Bernoulli	Logit: $g(\mu_i) = \ln\left(\frac{\mu_i}{1-\mu_i}\right)$
Probit regression	Bernoulli	Probit: $g(\mu_i) = \Phi^{-1}(\mu_i)$
Multinomial logit/logistic	Categorical	Multinomial logit: $g(\mu_{ic}) = \ln\left(\frac{\mu_{ic}}{\mu_{iC}}\right)$
Poisson regression	Poisson	Log: $g(\mu_i) = \ln(\mu_i)$

Commonly used GLM models and their components

Model	**Random component**	**Link function**
Linear regression	Gaussian	Identity: $g(\mu_i) = \mu_i = w^\top x_i$
Binary logistic regression	Bernoulli	Logit: $g(\mu_i) = \ln\left(\frac{\mu_i}{1-\mu_i}\right)$
Probit regression	Bernoulli	Probit: $g(\mu_i) = \Phi^{-1}(\mu_i)$
Multinomial logit/logistic	Categorical	Multinomial logit: $g(\mu_{ic}) = \ln\left(\frac{\mu_{ic}}{\mu_{iC}}\right)$
Poisson regression	Poisson	Log: $g(\mu_i) = \ln(\mu_i)$

Note that in all cases, the link function always results in:

$$g(\mu_i) = w^\top x_i \tag{69}$$

Commonly used GLM models and their components

Model	**Random component**	**Link function**
Linear regression	Gaussian	Identity: $g(\mu_i) = \mu_i = w^\top x_i$
Binary logistic regression	Bernoulli	Logit: $g(\mu_i) = \ln\left(\frac{\mu_i}{1-\mu_i}\right)$
Probit regression	Bernoulli	Probit: $g(\mu_i) = \Phi^{-1}(\mu_i)$
Multinomial logit/logistic	Categorical	Multinomial logit: $g(\mu_{ic}) = \ln\left(\frac{\mu_{ic}}{\mu_{iC}}\right)$
Poisson regression	Poisson	Log: $g(\mu_i) = \ln(\mu_i)$

Note that in all cases, the link function always results in:

$$g(\mu_i) = w^\top x_i \tag{69}$$

Its job is to "link" the response to the systematic component via a suitable transformation that results in a linear function of the w's.

Further reading on GLMs

Further reading on GLMs

- German Rodriguez's lecture notes on GLMs: `https://data.princeton.edu/wws509/notes/`
- Penn State: `https://online.stat.psu.edu/stat504/lesson/6/6.1` (Including more on logistic and multinomial logistic)