Generalized Additive Models

Creighton Heaukulani

Department of Statistics and Data Science
National University of Singapore

Spring 2025

Outline

- Review of nonlinear and nonparametric models, which we mostly used for a univariate predictor X.
- Introduce Generalized Additive Models (GAMs), which are a very useful models to analyze multivariate $X_1, \ldots, X_p$:

$$f(X) = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) \qquad (0.1)$$

- We will derive an inference algorithm, the backfitting algorithm, which is modular.
- Show you how to implement and explore these models in PyGAM.
- Briefly discuss extensions to classification, etc.
- Unify most models we've studied so far through linear smoothers.

Table of Contents

Regression Splines

- Recall that a cubic regression spline with K knots $c_1, \ldots, c_K$ is represented by K natural basis functions

$$f(X) = N(X)^T\theta = \sum_{k=1}^{K} \theta_k N_k(X), \tag{1.1}$$

where

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

$$d_k(X) = \frac{(X - c_k)_+^3 - (X - c_K)_+^3}{c_K - c_k}. \tag{1.2}$$

- Given a dataset (X_i, Y_i), we fit $\hat{\theta}$ by least squares

$$\hat{\theta} = (\mathbf{N}^T\mathbf{N})^{-1}\mathbf{N}^T\mathbf{Y}, \qquad \hat{f}(X) = N(X)^T\hat{\theta} \tag{1.3}$$

$$N_{i,j} = N_j(X_i). \tag{1.4}$$

- You can choose the number of knots by cross-validation (and place them evenly throughout the dataset, for example).

Smoothing Splines

- Alternatively, you can use n knots placed at the training locations X_i, and regularize (or smooth). Let

$$f(X) = N(X)^T\theta = \sum_{i=1}^{n} \theta_i N_i(X), \tag{1.5}$$

where

$$N_1(X) = 1, \qquad N_2(X) = X, \qquad N_{i+2}(X) = d_i(X) - d_{n-1}(X),$$
$$d_i(X) = \frac{(X - X_i)_+^3 - (X - X_n)_+^3}{X_n - X_i}. \tag{1.6}$$

- Then given a dataset (X_i, Y_i), we fit $\hat{\theta}$ by *penalized* least squares:

$$\hat{\theta} = (\mathbf{N}^T\mathbf{N} + \lambda\mathbf{R})^{-1}\mathbf{N}^T\mathbf{Y}, \qquad \hat{f}(X) = N(X)^T\hat{\theta}, \tag{1.7}$$
$$R_{j,k} = \int N_j''(t)N_k''(t)\mathrm{d}t. \tag{1.8}$$

for a penalty or smoothing parameter $\lambda > 0$, selected by cross-validation, say.

Smoothing Splines

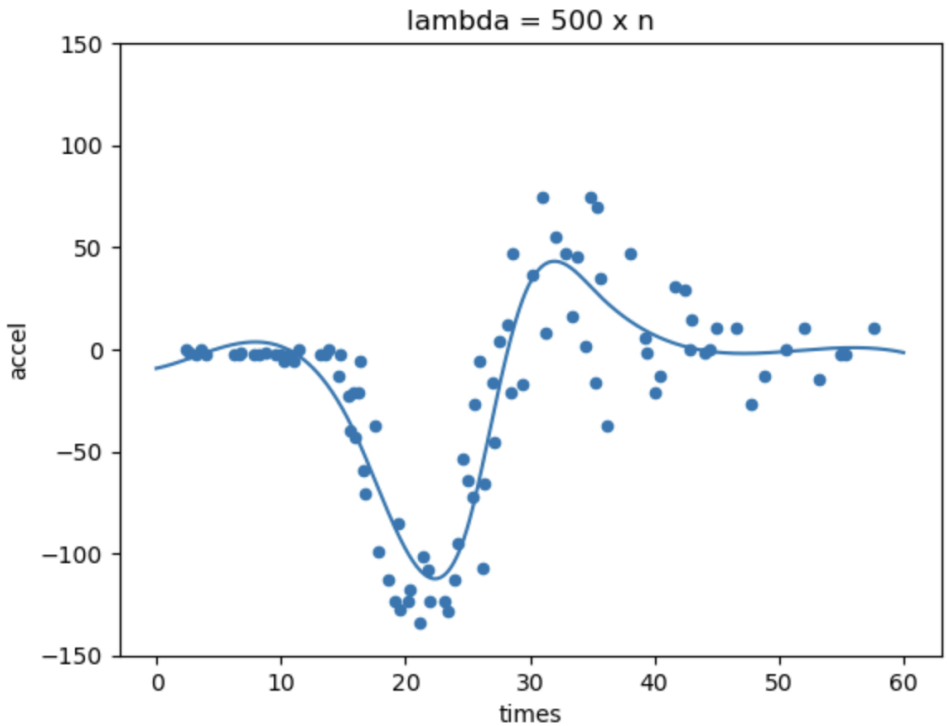

Figure: Smoothing spline with $\lambda = 500n$.

Nonparametric Methods

With nonparametric methods, there are no longer parameters to fit, and instead "fitting" occurs at prediction time.

- k-nearest neighbors - discontinuous functions:

$$\hat{f}(X) = \frac{1}{k} \sum_{i \in N_k(X)} Y_i, \tag{1.9}$$

- Nadaraya-Watson kernel smoothing - continuous functions:

$$\hat{f}(X) = \sum_{i=1}^{n} \frac{K(\frac{X - X_i}{h})}{\sum_{j=1}^{n} K(\frac{X - X_j}{h})} Y_i, \tag{1.10}$$

- Local (polynomial) regression models - reduce bias at the cost of some variance

$$\hat{f}(X) = b(X)^T (\mathbf{B}^T \mathbf{W}(X) \mathbf{B})^{-1} \mathbf{B}^T \mathbf{W}(X) \mathbf{Y}, \tag{1.11}$$

Local polynomial regression

Recall that in local regression

$$\hat{f}(X) = b(X)^T(\mathbf{B}^T\mathbf{W}(X)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(X)\mathbf{Y}, \qquad (1.12)$$

we had

- $b(x) = (1, x, x^2, \ldots, x^d)^T$,
- $\mathbf{B}$ be the $n \times d+1$ matrix with i-th row $b(X_i)^T = (1, X_i, X_i^2, \ldots, X_i^d)$,
- $\mathbf{W}(X)$ be the $n \times n$ diagonal matrix with i-th element $K(X, X_i)$.

Multi-dimensional Splines

Suppose $X \in \mathbb{R}^2$ and we have M_1 and M_2 basis functions for predictors X_1 and X_2, respectively:

$$h_{1,k}(X_1), \qquad k = 1, \ldots, M_1, \tag{1.13}$$

$$h_{2,k}(X_2), \qquad k = 1, \ldots, M_2. \tag{1.14}$$

Then the $M_1 \times M_2$ dimensional tensor product basis defined by

$$g_{j,k}(X) = h_{1,j}(X_1)h_{2,k}(X_2), \qquad j \leq M_1,\ k \leq M_2 \tag{1.15}$$

can be used as the basis expansion for f:

$$f(X) = \sum_{j=1}^{M_1} \sum_{k=1}^{M_2} \theta_{j,k} g_{j,k}(X), \tag{1.16}$$

and we proceed analogously to the univariate case.

Multi-dimensional Splines

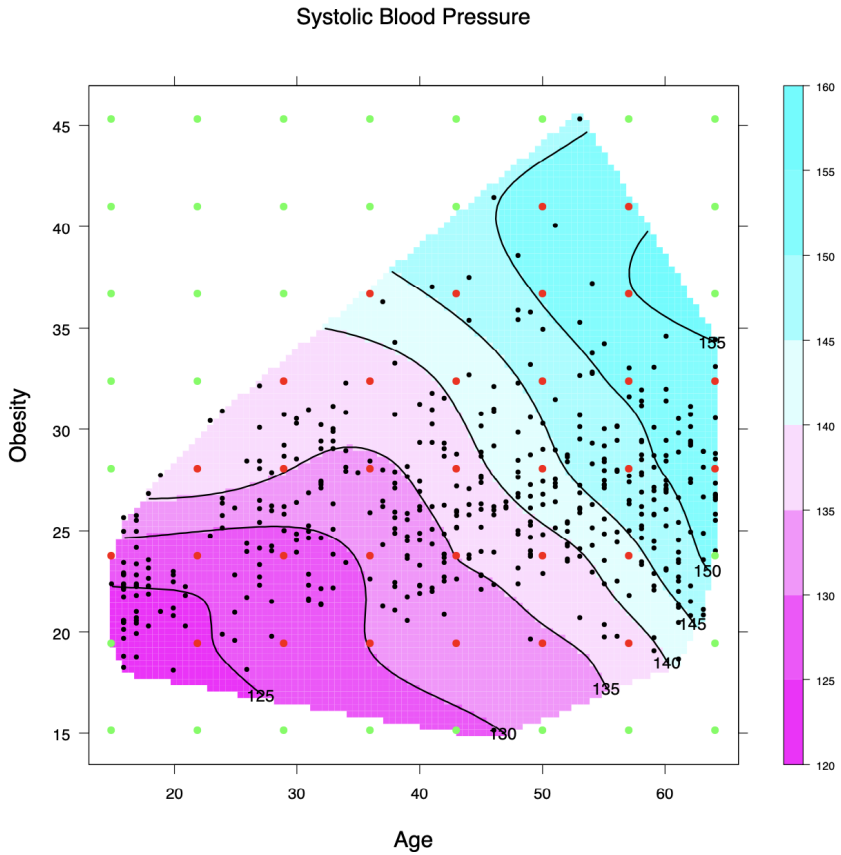

Figure: (ESL2 Fig. 5.12) A thin-plate spline fit to a dataset predicting systolic blood pressure from age and an obesity score. The lattice of points are the regularly gridded knots, and note that we only use knots within the convex hull of the data (red).

Multivariate kernel regression

- We now consider generalizing X beyond the univariate case.
- Let $b(X)$ be a vector of polynomial terms of a multivariate X, for example,

$$b(X) = (1, X_1, X_2, X_1^2, X_2^2, X_1 X_2)^T, \tag{1.17}$$

  which is like a set of basis functions.
- Then at each X, we solve

$$\hat{\beta} := \arg\min_{\beta} \Big\{ \sum_{i=1}^{n} K\Big(\frac{||X - X_i||}{h}\Big)(Y_i - b(X_i)^T\beta)^2 \Big\} \tag{1.18}$$

  where the distance is now given by the vector norm.
- We then produce the fit

$$\hat{f}(X) = b(X)^T\hat{\beta}. \tag{1.19}$$

Table of Contents

Generalized Additive Models

- **Large** p is a major focus of machine learning, motivating much of what comes later in ST5227 (random forests, etc.).
- **Generalized Additive Models (GAMs)** are a useful tool to model multivariate $X_1, \ldots, X_p$, which retain much interpretability and are modular to implement.
- Generally, we will model each X_j with a nonlinear function (a spline or kernel smoother) and assume they additively contribute to the overall function $f(X)$:

$$f(X) = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) \qquad (2.1)$$

Generalized Additive Models

- Recall the construction of nonlinear models using basis expansions

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X), \tag{2.2}$$

for some basis functions h_m.

- For multivariate $X_1, \ldots, X_p$, we suggested the following

$$h_m(X) = h_{m,1}(X_1) \cdots h_{m,p}(X_p) \tag{2.3}$$

a set of basis functions on each predictor independently.

- **Or alternatively**, we could instead make the construction additive

$$f(X) = h_1(X_1) + h_2(X_2) + h_{1,2}(X_1, X_2) + \cdots \tag{2.4}$$

Generalized Additive Models

- More generally,

$$f(X) = \sum_{m=1}^{M} \sum_{j=1}^{p} h_{m,j}(X_j) + \sum_{m=1}^{M} \sum_{k \leq \ell} h_{m,k,\ell}(X_k, X_\ell) + \cdots \quad (2.5)$$

  where we call

  - $h_{m,j}$ are *main effects*,
  - $h_{m,k,\ell}(X_k, X_\ell)$ are *interaction terms*.

- We will restrict attention to models of only the main effects

$$f(X) = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) \quad (2.6)$$

  where in general we will let the f_j be nonlinear smooth functions.

- We call these Generalized Additive Models or GAMs.

Generalized Additive Models

- You can use all sorts of models for the f_j.
- It should not surprise you that minimizing the penalized sum of squared residuals

$$\sum_{i=1}^{n}\left(Y_i-\alpha-\sum_{j=1}^{p}f_j(X_{i,j})\right)^2+\sum_{j=1}^{p}\lambda_j\int\left(f_j''(t_j)\right)^2\mathrm{d}t_j, \quad (2.7)$$

  over all smooth f_j, for some regularization parameters $\lambda_j \geq 0$...
- is achieved by a natural cubic spline for each f_j with knots at each of the unique values of $X_{i,j}$ for $i = 1, \ldots, n$.

Nonlinear logistic regression

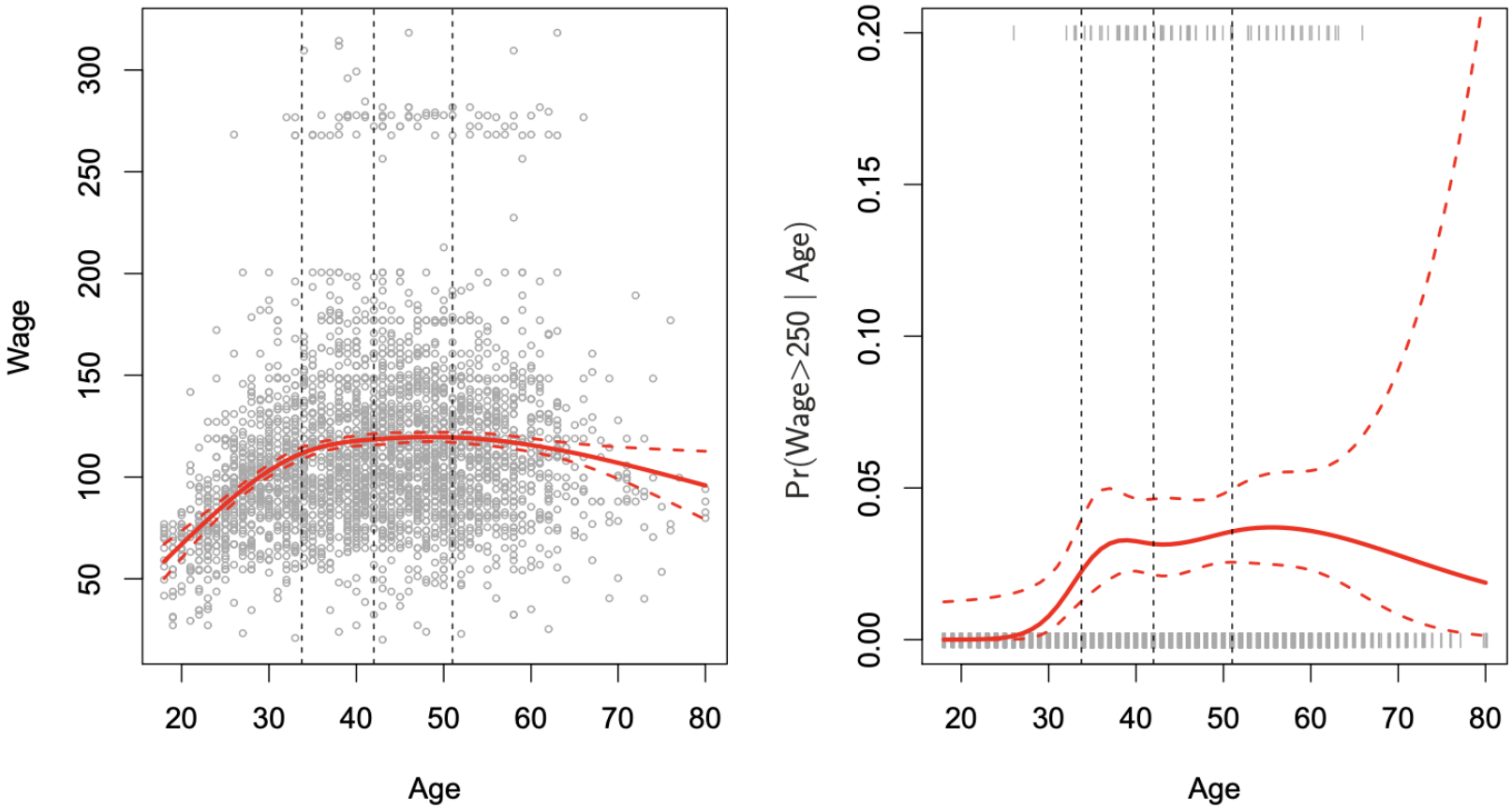

Figure: (ISL2 Fig. 7.5) A natural cubic spline and confidence intervals on the wage dataset, where wage is predicted from age (left). On the right, we predict the binary event Wage $>$ 250 with logistic regression.

Fitting Generalized Additive Models

If each f_j is a smoothing spline:

$$f(X) = \alpha + \sum_{j=1}^{p} f_j(X_j), \tag{2.8}$$

then we could plot partial contributions of each predictor on wage:

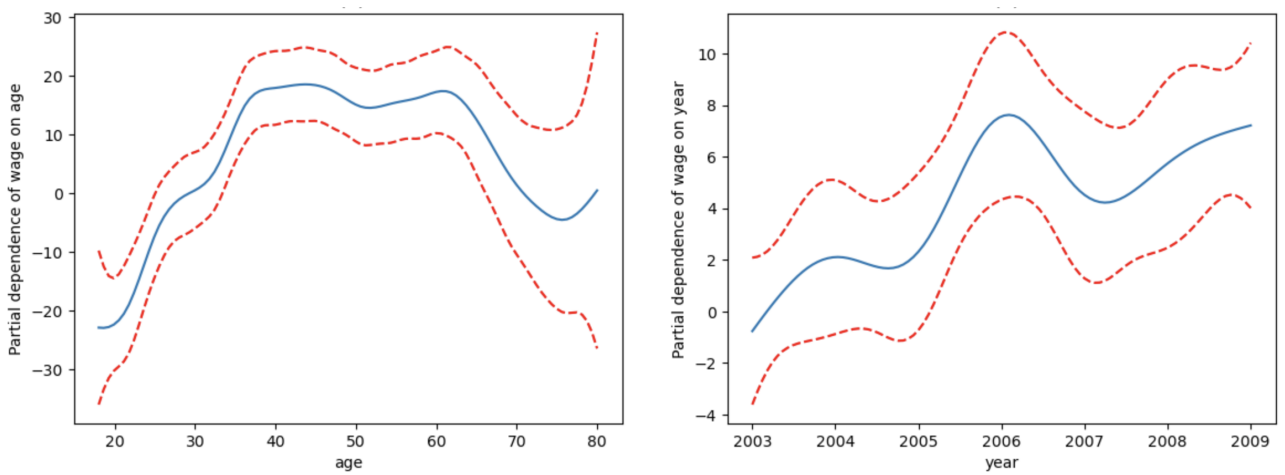

Figure: Partial dependence plots of age and year on wage in the Wage dataset.

Table of Contents

Fitting Generalized Additive Models

- So if each f_j is a smoothing spline,

$$Y = f(X) + \varepsilon \tag{3.1}$$

$$f(X) = \alpha + \sum_{j=1}^{p} f_j(X_j), . \tag{3.2}$$

- then we have to fit p functions f_j, which are interlocking because they all contribute to the prediction of Y.
- We will iteratively update the splines f_j one at a time.

The Backfitting Algorithm

- Our model *assumes* that

$$\mathbb{E}[Y] = f(X) = \alpha + \sum_{j=1}^{p} f_j(X_j) \tag{3.3}$$

- Then for each $j = 1, \ldots, p$,

$$f_j(X_j) = \mathbb{E}[Y] - \alpha - \sum_{k \neq j} f_k(X_k) \tag{3.4}$$

$$= \mathbb{E}\left[Y - \alpha - \sum_{k \neq j} f_k(X_k)\right] \tag{3.5}$$

- So at each step, we fit a spline for f_j on the "remainder" term

$$z_j = Y - \alpha - \sum_{k \neq j} f_k(X_k). \tag{3.6}$$

The Backfitting Algorithm

For each $j = 1, \ldots, p$,

$$f_j(X_j) = \mathbb{E}\Big[Y - \alpha - \sum_{k \neq j} f_k(X_k)\Big] \tag{3.7}$$

Given a dataset (X_i, Y_i), we "fit" the values

$$\mathbf{z}_j = \mathbf{Y} - \hat{\alpha} - \sum_{k \neq j} \hat{\mathbf{f}}_k \tag{3.8}$$

$$\hat{\mathbf{f}}_j = \mathbf{N}_j (\mathbf{N}_j^T \mathbf{N}_j + \lambda_j \mathbf{R}_j)^{-1} \mathbf{N}_j^T \mathbf{z}_j, \tag{3.9}$$

where

- $\hat{\mathbf{f}}_j$ is the n-vector with i-th element $\hat{f}_j(X_{i,j})$;
- $\mathbf{N}_j$ is the $n \times n$ matrix with (i,k)-th entry $N_k(X_{i,j})$;
- $\mathbf{R}_j$ is the $n \times n$ matrix of usual penalty terms.

The Backfitting Algorithm

Given a dataset (X_i, Y_i), we "fit" the values

$$\hat{\mathbf{f}}_j = \mathbf{N}_j(\mathbf{N}_j^T\mathbf{N}_j + \lambda_j\mathbf{R}_j)^{-1}\mathbf{N}_j^T\mathbf{z}_j, \tag{3.10}$$

$$\mathbf{z}_j = \mathbf{Y} - \hat{\alpha} - \sum_{k\neq j}\hat{\mathbf{f}}_k, \tag{3.11}$$

For simplicity, we write

$$\hat{\mathbf{f}}_j \leftarrow \mathcal{S}_j(\mathbf{z}_j, \mathbf{X}). \tag{3.12}$$

And the intercept term is estimated as usual

$$\hat{\alpha} = \frac{1}{n}\sum_{i=1}^{n} Y_i \tag{3.13}$$

Fitting Generalized Additive Models

Algorithm The Backfitting Algorithm for Additive Models

1. Initialize $\hat{\alpha} = \frac{1}{n}\sum_{i=1}^{n} Y_i$, and $\hat{\mathbf{f}}_j = 0$, for all $j \leq p$.
2. Iterate for $j = 1, \ldots, p$ until convergence:
   2.1 Compute the new targets

$$\mathbf{z}_j = \mathbf{Y} - \hat{\alpha} - \sum_{k \neq j} \hat{\mathbf{f}}_k, \tag{3.14}$$

   2.2 Update $\hat{\mathbf{f}}_j$ by fitting a smoothing spline from the n-vector $\mathbf{z}$ to the predictors $\mathbf{X}$:

$$\hat{\mathbf{f}}_j \leftarrow \mathcal{S}_j(\mathbf{z}_j, \mathbf{X}). \tag{3.15}$$

The Gauss-Seidel Algorithm

- How do we know this will converge? Because it is a form of Gauss-Seidel algorithm.
- The Gauss-Seidel algorithm solves a system of linear equations $Az = b$ for a known $k \times k$ matrix A, a known k-vector b, and an unknown k-vector z.
- It works by successively solving for element z_j in the j-th equation, fixing all other z_j's at their current guesses.
- The process is iterated through $\ell = 1, 2, \ldots, k$ repeatedly and will converge to the solution of the system.

The Gauss-Seidel Algorithm

To see this, note for each $j = 1, \ldots, p$, we are updating:

$$\hat{\mathbf{f}}_j = \mathcal{S}_j\Big(\mathbf{Y} - \hat{\alpha} - \sum_{k \neq j} \hat{\mathbf{f}}_k, \mathbf{X}\Big) \tag{3.16}$$

$$= \mathbf{N}_j^T \mathbf{N}_j (\mathbf{N}_j^T \mathbf{N}_j + \lambda_j \mathbf{R}_j)^{-1} \mathbf{N}_j^T \mathbf{z}_j \tag{3.17}$$

$$= \mathbf{S}_j \mathbf{z}_j \tag{3.18}$$

$$= \mathbf{S}_j \tilde{\mathbf{Y}} - \sum_{k \neq j} \mathbf{S}_j \hat{\mathbf{f}}_k, \tag{3.19}$$

where $\tilde{\mathbf{Y}} = \mathbf{Y} - \hat{\alpha}$ is the centered targets.

This is the j-th row in the following linear system:

$$\begin{pmatrix} I & \mathbf{S}_1 & \mathbf{S}_1 & \cdots & \mathbf{S}_1 \\ \mathbf{S}_2 & I & \mathbf{S}_2 & \cdots & \mathbf{S}_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{S}_p & \mathbf{S}_p & \mathbf{S}_p & \cdots & I \end{pmatrix} \begin{pmatrix} \hat{\mathbf{f}}_1 \\ \hat{\mathbf{f}}_2 \\ \vdots \\ \hat{\mathbf{f}}_p \end{pmatrix} = \begin{pmatrix} \mathbf{S}_1 \tilde{\mathbf{Y}} \\ \mathbf{S}_2 \tilde{\mathbf{Y}} \\ \vdots \\ \mathbf{S}_p \tilde{\mathbf{Y}} \end{pmatrix} \tag{3.20}$$

Modularity

- Note that our derivations work for any other model that can be fit with a linear estimate $\mathbf{S}_j\mathbf{z}_j$:

$$\text{Smoothing spline:} \quad \hat{\mathbf{f}} = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{R})^{-1}\mathbf{N}^T\mathbf{Y}$$

$$\text{Local regression:} \quad \hat{\mathbf{f}} = \mathbf{B}(\mathbf{B}^T\mathbf{W}(X)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(X)\mathbf{Y}$$

- These models are called linear smoothers (more later).
- The backfitting algorithm is modular. Simply drop your fitting method $\hat{\mathbf{f}}_j = \mathbf{S}_j\mathbf{z}_j$.

Summary

A summary so far:

- GAMs are useful for fitting interpretable nonlinear models for multivariate $X_1, \ldots, X_p$:

$$f(X) = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) \qquad (3.21)$$

- The backfitting algorithm iteratively fits each f_j in turn. As a Gauss-Seidel algorithm, we know it will converge.
- If the f_j are natural cubic splines with knots at each of the training $X_{i,j}$, then the solution minimizes the penalized sum of squared residuals.
- Otherwise the backfitting algorithm is modular and works for any fit for f_j that is a linear smoother.

Demo

Demo on fitting GAMs with PyGAM.

Table of Contents

Classification with GAMs

Additive logistic regression for $Y \in \{0, 1\}$ simply puts

$$\mathbb{P}\{Y = 1\} = g\Big(\alpha + \sum_{j=1}^{p} f_j(X_j)\Big) \tag{4.1}$$

for g the logistic sigmoid.

- **Recall:** How should we perform inference?
- Maximize the log-likelihood with a Newton-Rhapson algorithm.
- We need to iteratively update the f_j as before.

Fitting Additive Logistic Regression Models

Here is a rough sketch of the inference algorithm (we will not derive it).

Algorithm Local Scoring Algorithm for GAM Logistic Regression

1. Initialize $\hat{\alpha} = \log(\frac{\bar{y}}{1-\bar{y}})$ and $\hat{f}_j = 0$, for all $j \leq p$.
2. Define $\hat{\eta}_i = \hat{\alpha} + \sum_j \hat{f}_j(X_{i,j})$ and $\hat{p}_i = 1/(1 + \exp(-\hat{\eta}_i))$.
3. Iteratively perform the following updates until convergence:
   - 3.1 Construct the targets $z_i = \hat{\eta}_i + \frac{Y_i - \hat{p}_i}{\hat{p}_i(1-\hat{p}_i)}$.
   - 3.2 Construct the weights $w_i = \hat{p}_i(1 - \hat{p}_i)$.
   - 3.3 Fit an additive model to the targets z_i with weights w_i (using a weighted backfitting algorithm).
   - 3.4 This produces new $\hat{\alpha}$ and $\hat{f}_j$.

Do the steps look familiar? (Recall iteratively reweighted least squares.)

Multiclass classification and other data with GAMs

- For multiclass classification, we define a categorical distribution with parameters

$$\mathbb{P}\{Y = k\} \propto \exp\Big(\alpha_k + \sum_{j=1}^{p} f_{j,k}(X_j)\Big), \tag{4.2}$$

  for class-specific α_k and $f_{1,k}, \ldots, f_{p,k}$.

- For count regression, we let

$$Y \sim \text{Poisson}\Big(\exp\Big(\alpha + \sum_{j=1}^{p} f_j(X_j)\Big)\Big) \tag{4.3}$$

- Maximum likelihood inference can proceed as before, but these algorithms do get complicated.

Benefits and Limitations of GAMs

- Main benefits of additive models:
  - Interpretability: We can analyze the flexible model for each X_j separately.
  - Modularity: The backfitting algorithm is simple and modular for any linear smoothing model choice for f_j.
- Main limitation:
  - The additive assumption is strong and may hold back predictive performance.
  - Consider other methods like bagging and boosting instead (later in ST5227).

Poll: Would you consider these parametric or nonparametric models?

Table of Contents

Unification of the supervised learning models so far

- Ordinary least squares:

$$\hat{\beta}^{\text{ols}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{5.1}$$

- Ridge regression:

$$\hat{\beta}^{\text{ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{Y}, \qquad \text{for } \lambda > 0. \tag{5.2}$$

- Note that both are linear in $\mathbf{Y}$.

Unification of the supervised learning models so far

- For regression splines, we formed the estimate

$$\hat{f}(X) = \sum_{m=1}^{M} \hat{\theta}_m^{\text{rs}} h_m(X) \tag{5.3}$$

for basis functions $h_m(X)$ (usually cubic splines), and we found

$$\hat{\theta}^{\text{rs}} = (\mathbf{B}_c^T \mathbf{B}_c)^{-1} \mathbf{B}_c^T \mathbf{Y} \tag{5.4}$$

where we emphasize the matrix of basis functions $\mathbf{B}_c$ evaluated at the training points X_i depend on the knot locations c.

- Note linearity in $\mathbf{Y}$.

Unification of the supervised learning models so far

- For smoothing splines, we formed the estimate

$$\hat{f}(X) = \sum_{i=1}^{n} \hat{\theta}_j^{\text{ss}} N_j(X) \tag{5.5}$$

for the natural basis functions $N_j(X)$, and we found

$$\hat{\theta}^{\text{ss}} = (\mathbf{N}^T\mathbf{N} + \lambda\mathbf{R})^{-1}\mathbf{N}^T\mathbf{Y} \tag{5.6}$$

for $\lambda > 0$ and $\mathbf{N}$ and $\mathbf{R}$ have elements

$$N_{i,j} = N_j(X_i), \tag{5.7}$$

$$R_{j,k} = \int N_j''(t)N_k''(t)\mathrm{d}t. \tag{5.8}$$

- Note linearity in $\mathbf{Y}$.

Unification of the supervised learning models so far

- For kernel regression and local polynomial regression, we used the estimate

$$\hat{f}(X) = b(X)^T \hat{\theta}^{\mathsf{ks}} \tag{5.9}$$

where $b(X) = (1, X, \dots)^T$ and

$$\hat{\theta}^{\mathsf{ks}} = (\mathbf{B}^T \mathbf{W}(X) \mathbf{B})^{-1} \mathbf{B}^T \mathbf{W}(X) \mathbf{Y} \tag{5.10}$$

where $\mathbf{B}$ has $b(X_i)^T$ as its rows and $\mathbf{W}(X)$ is diagonal with elements $\mathbf{W}_{i,i}(X) = K(\frac{X - X_i}{h})$ for $h > 0$.

- Again, note linearity in $\mathbf{Y}$.

Unification of the supervised learning models so far

For all the models we have studied, write the n-vector $\hat{\mathbf{f}}$ of fitted values $\hat{f}(X_i)$ at the training data as:

$$\hat{\mathbf{f}} = \mathbf{X}^T\hat{\beta}^{\text{ols}} = \mathbf{X}^T(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{5.11}$$

$$\hat{\mathbf{f}} = \mathbf{X}^T\hat{\beta}^{\text{ridge}} = \mathbf{X}^T(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{Y} \tag{5.12}$$

$$\hat{\mathbf{f}} = \mathbf{B}_c\hat{\theta}^{\text{rs}} = \mathbf{B}_c(\mathbf{B}_c^T\mathbf{B}_c)^{-1}\mathbf{B}_c^T\mathbf{Y} \tag{5.13}$$

$$\hat{\mathbf{f}} = \mathbf{N}\hat{\theta}^{\text{ss}} = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{R})^{-1}\mathbf{N}^T\mathbf{Y} \tag{5.14}$$

$$\hat{\mathbf{f}} = \mathbf{B}\hat{\theta}^{\text{ks}} = \mathbf{B}(\mathbf{B}^T\mathbf{W}(X)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(X)\mathbf{Y} \tag{5.15}$$

- These are all linear in $\mathbf{Y}$!
- Let $\mathbf{S}$ be the matrix such that the estimator is $\mathbf{SY}$.
- $\mathbf{S}$ is called a linear smoother, or smoothing matrix, or sometimes the "hat" matrix.

Linear Smoothers

- For ridge regression

$$\hat{\mathbf{f}} = \mathbf{X}\hat{\beta}^{\text{ridge}} = \mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T\mathbf{Y} \tag{5.16}$$

$$= \mathbf{S}_\lambda \mathbf{Y}. \tag{5.17}$$

- The trace of the smoother matrix $\mathbf{S}_\lambda$ is of great interest:

$$\text{tr}[\mathbf{S}_\lambda] = \text{tr}[\mathbf{X}(\mathbf{X}^T\mathbf{X} + \lambda I)^{-1}\mathbf{X}^T] \tag{5.18}$$

$$= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}, \tag{5.19}$$

- where the d_j are the singular values from the singular value decomposition of X.

Linear Smoothers

- We call

$$\mathrm{df}(\lambda) := \mathrm{tr}[\mathbf{S}_\lambda] = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}, \tag{5.20}$$

the effective degrees of freedom, which describes how flexible the model is.

- The coefficient β_j of X_j is shrunk by a multiplicative factor

$$\frac{d_j^2}{d_j^2 + \lambda}$$

- As λ increases, the amount of shrinkage increases.
- If $\lambda \to 0$, then there is no shrinkage and we recover ordinary least squares.

Linear Smoothers

For regression splines, we have

$$\hat{\mathbf{f}} = \mathbf{B}_c \hat{\theta}^{\text{rs}} = \mathbf{B}_c (\mathbf{B}_c^T \mathbf{B}_c)^{-1} \mathbf{B}_c^T \mathbf{Y} \tag{5.21}$$

$$= \mathbf{S}_c \mathbf{Y}, \tag{5.22}$$

and we could derive

$$\text{tr}[\mathbf{S}_c] = M, \tag{5.23}$$

which is the number of basis functions (and hence the number of parameters).

So this is still interpreted as a the effective degrees of freedom of the model.

Linear Smoothers

For smoothing splines, we first rewrite $\mathbf{S}_\lambda$ in the Reinsch form:

$$\hat{\mathbf{f}} = \mathbf{N}\hat{\theta}^{\text{ss}} = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{R})^{-1}\mathbf{N}^T\mathbf{Y} \tag{5.24}$$

$$= \mathbf{N}(\mathbf{N}^T(I + \lambda(\mathbf{N}^T)^{-1}\mathbf{R}\mathbf{N}^{-1})\mathbf{N})^{-1}\mathbf{N}^T\mathbf{Y} \tag{5.25}$$

$$= (I + \lambda\mathbf{K})^{-1}\mathbf{Y}, \tag{5.26}$$

where $\mathbf{K} := (\mathbf{N}^T)^{-1}\mathbf{R}\mathbf{N}^{-1}$.

- So $\mathbf{S}_\lambda = \mathbf{N}(\mathbf{N}^T\mathbf{N} + \lambda\mathbf{R})^{-1}\mathbf{N}^T\mathbf{Y} = (I + \lambda\mathbf{K})^{-1}$.
- Taking the singular value decomposition of $\mathbf{K}$ would reveal

$$\text{df}(\lambda) := \text{tr}[\mathbf{S}_\lambda] = \sum_{j=1}^{n} \frac{1}{1 + \lambda d_j}, \tag{5.27}$$

  for d_j the singular values of $\mathbf{K}$.
- We could set λ by analyzing this effective degrees of freedom.

Linear Smoothers

Finally, for kernel smoothing and local polynomial regression,

$$\hat{\mathbf{f}} = \mathbf{B}\hat{\theta}^{\mathsf{ks}} = \mathbf{B}(\mathbf{B}^T\mathbf{W}(X)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(X)\mathbf{Y} \tag{5.28}$$

$$= \mathbf{S}_h\mathbf{Y}, \tag{5.29}$$

the effective degrees of freedom $\mathrm{tr}[\mathbf{S}_h]$ does not have any further derivation/simpler form, but can be computed and interpreted in the same way.

Summary

- All the models we have studied these 6 weeks (except Lasso) are linear smoothers.
- Each attempts to add some flexibility (add more predictors, nonlinearity, higher order polynomials, nonparametric relationships, etc.)...
- ... but due to the bias-variance tradeoff, we must regularize to keep variance low and generalize to new data well.
- The effective degrees of freedom used to be (mathematically) analyzed for model selection...
- ... but now we have modern computers that can do experimental cross-validation easily.

Further Topics

- GAMs have one more necessary assumption that I did not mention: we assume $\mathbb{E}[f_j(X_j)] = 0$ for each $j \leq n$.
- Additionally, the backfitting algorithm needs some extra steps for computational stability. See ESL2 § 9.1 for details.
- Generalized additive models are in statsmodels: https://www.statsmodels.org/dev/gam.html
- It seems PyGAM is popular (with a lot of Youtube videos on it): https://pygam.readthedocs.io/en/latest/
- Linear smoothers and effective degrees of freedom are discussed throughout ESL2, but see in particular Sec 3.4.1 (the end of the section) and Sec. 5.4.1.