STAT3612 Lecture 6

Generalized Additive Models

Dr. Aijun Zhang

6 October 2020

Department of 統計及精算學系
Statistics & Actuarial Science

Table of Contents

Generalized Additive Models (GAM)

- Given features $\boldsymbol{x} \in \mathbb{R}^p$, the GAM takes the form

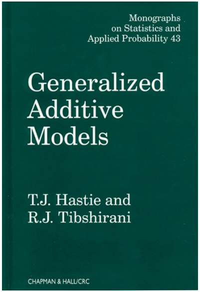
$$g(\mathbb{E}(Y)) = \mu + f_1(x_1) + \cdots + f_p(x_p)$$

  where $g(\cdot)$ is the link function, μ is the overall mean, and $f_j(\cdot)$ is the feature function for x_j.

- $f_j(\cdot)$ can be specified via parametric functions or via feature engineering.

- We consider the nonparametric estimation of $f_j(\cdot)$ subject to certain interpretability constraints.

- GAM dates back to Trevor Hastie and Robert Tibshirani (1990). See also Wikipedia.



Backfitting Algorithm

In statistics, the backfitting algorithm is a particularly useful procedure for fitting GAMs iteratively. See Wikipedia for details. It provides a greedy sub-optimal solution though.

For regression case with $g(y) = y$, the backfitting algorithm is as simple as

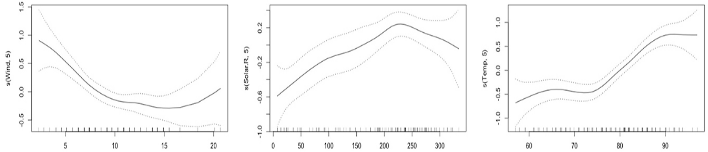
1. Initialize $\bar{\mu} = \bar{y}$ and $\hat{f}_j \equiv 0 \; \forall j$
2. Cycle through $j = 1, \ldots, p$, perform univariate smoothing

$$\hat{f}_j(x_{ij} \leftarrow S_j\left(\left\{y_i - \hat{\mu} - \sum_{k \neq j} \hat{f}_k(x_{ik})\right\}_{i=1}^{n}\right)$$

   where $S_j(\cdot)$ is a smoothing operator to be discussed in this chapter.

3. Continue Step 2 until the individual functions do not change.

Feature Representation by Nonparametric Smoothing



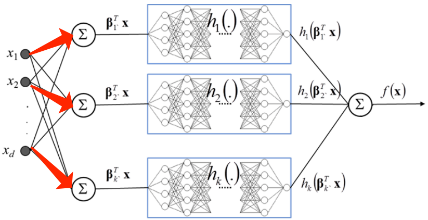
Each univariate $f_j(x_j)$ in a GAM is data-driven, subject to the following interpretability constraints:

- **Homogeneously Smooth:** classical nonparametric regression
  $\Rightarrow$ Kernel/Scatterplot smoothing: loess, local linear regression
  $\Rightarrow$ Smoothing splines, Hodrick-Prescott filter (ℓ_2-penalty)
- **Inhomogeneously Smooth:** e.g. piecewise-constant, piecewise-linear
  $\Rightarrow$ ℓ_1/ℓ_0-trend filtering with automatic knot detection
  $\Rightarrow$ $\ell_2/\ell_1/\ell_0$-penalized B-Splines
- **Shape Constraints:** e.g. increasing/decreasing, convex/concave
  $\Rightarrow$ Monotone/Isotonic regression, Least concave majorant

GAM for Interpretable Machine Learning

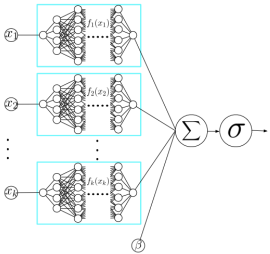
- The classical GAM (Hastie and Tibshirani, 1990) provides an important class of interpretable machine learning today.
- It can take advantages of deep learning for automated sub-modular feature representation, resulting in optimized solution via SGD network training.

GAM-Net (Special case of xNN)



Vaughan, Sudjianto, Brahimi, Chen, and Nair (2018)
Yang, Zhang and Sudjianto (2019)

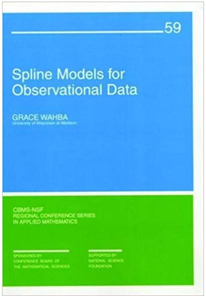
NAM (Neural Additive Model)



Agarwal, Frosst, Zhang, Caruana, and Hinton (2020)

Table of Contents

Smoothing Spline



Grace Wabha: Spline Models for Observational Data (1990)

Smoothing Spline

- Smoothing spline is a basic tool for nonparametric regression. It controls the degree of smoothness through the roughness penalty:

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} [y_i - f(x)]^2 + \lambda \int |f''(u)|^2 du$$

  where $\mathcal{H}$ denotes the 2nd-order Sobolev space.

- When $\lambda = 0$, there is no smoothing effect, but only interpolating.

- When $\lambda = \infty$, $|f''(x)| = 0$ for all x, which results in a line.

B-Spline Representation

- The unique minimizer is a cubic spline with knots at the unique x_i.
- By expressing $f(x) = \boldsymbol{\beta}^T \boldsymbol{\phi}(x)$ through use of B-spline bases, we can solve

$$\min_{\boldsymbol{\beta}} (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta})^T (\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}^T \boldsymbol{\Omega} \boldsymbol{\beta}$$

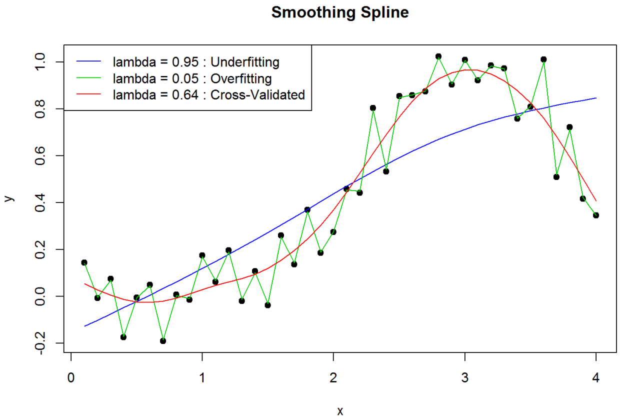
  where $\Omega_{ij} = \int \ddot{\phi}_i(x) \ddot{\phi}_j(x) dx$. It leads to the generalized ridge estimator:

$$\hat{\boldsymbol{\beta}}_\lambda = (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{\Phi}^T \mathbf{y}$$

- The smooth curve is given by $\hat{\boldsymbol{y}} = \mathbf{S}_\lambda \mathbf{y}$, where the smoothing matrix is

$$\mathbf{S}_\lambda = \boldsymbol{\Phi}(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{\Omega})^{-1} \boldsymbol{\Phi}^T$$

Smoothing Spline Fits

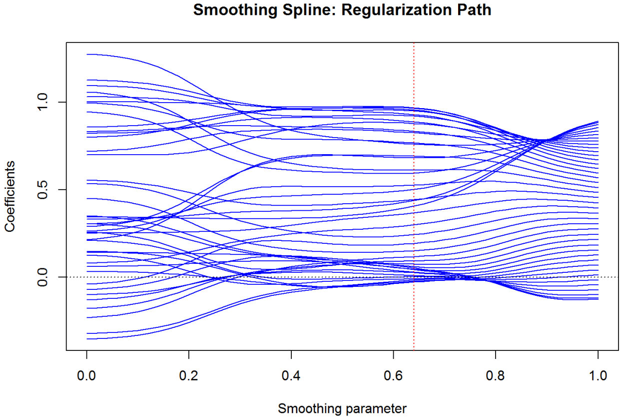


Regularization Paths

R code:

```
ss = seq(0, 1, by=0.02)
bb = NULL
for (k in 1:length(ss)) {
  tmp = smooth.spline(x, y, spar=ss[k])
  bb = cbind(bb, tmp$fit$coef)
}
matplot(ss, t(bb), type='l', lty=1, lwd = 1, col=4,
        xlab = "Smoothing parameter", ylab = "Coefficients",
        main = 'Smoothing Spline: Regularization Path')
abline(v= smooth.spline(x, y, cv = TRUE)$spar, col=2, lty=3,lwd=1)
abline(h=0, lty=3, col=1)
```

Regularization Paths



Cross-Validation

1. Split the re-shuffled data into K (e.g. 5, 10, n) folds
2. For each fold $k = 1, \ldots, K$:
   - Fit model based on the remaining K-1 folds of data
   - Evaluate the fitted model on the left-out fold
3. Take the average risk (i.e. MSE) as the cross-validation score

Cross-Validation

- Smoothing spline usually adopts the **GCV** (generalized cross-validation) based on the leave-one-out scheme (i.e. n-fold).

- For $i = 1, \ldots, n$, let $\hat{f}^{[i]}(x_i)$ denote the prediction at x_i based on the leave-one-out sample $\{(x_j, y_j)\}_{j \neq i}$, define

$$\mathrm{LOOCV}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}^{[i]}(x_i) \right)^2$$

- Upon some relaxation, the LOOCV score can be approximated by the following GCV score:

$$\mathrm{GCV}(\lambda) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{f}(x_i)}{1 - \mathrm{tr}(\mathbf{S}_\lambda)/n} \right)^2$$

Cross-Validation

R code:

```
ss = seq(0, 1, by=0.02)
gcv = NULL
for (k in 1:length(ss)) {
  tmp = smooth.spline(x, y, spar=ss[k])
  gcv = c(gcv, tmp$cv.crit)
}
plot(ss, gcv, type='b',
     xlab="lambda", ylab="GCV",
     main="GCV Selection of Smoothing Parameter")
abline(v=ss[which.min(gcv)],col=2,lty=3,lwd=1)
```

Cross-Validation

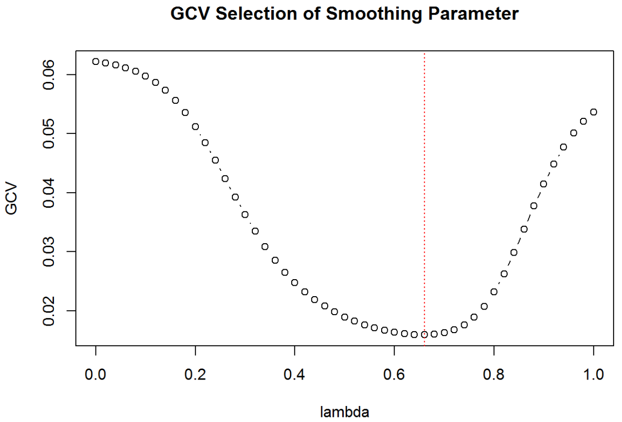


Table of Contents

HP Trend Filtering

- Let $\{y_i\}_{i\in[n]}$ be the sequence data observed regularly (with equal spacing).
- Assume $y_i = \alpha_i + \varepsilon_i$, with α_i representing the underlying signal/trend.
- HP ℓ_2-trend filtering by Hodrick and Prescott (1997):

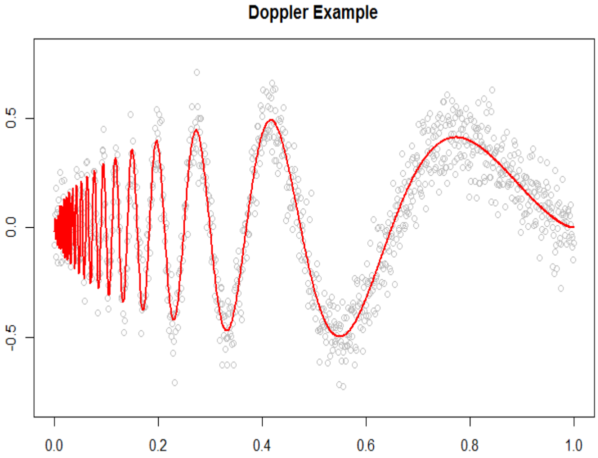
$$\min_{\{\alpha_i\}} \frac{1}{2}\sum_{i=1}^{n}(y_i - \alpha_i)^2 + \lambda \sum_{i=2}^{n-1}(\alpha_{i-1} - 2\alpha_i + \alpha_{i+1})^2$$

- It can be viewed as the smoothing spline under the discrete setting:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{D}^{(2)}\boldsymbol{\alpha}\|_{\ell_2}^2,$$

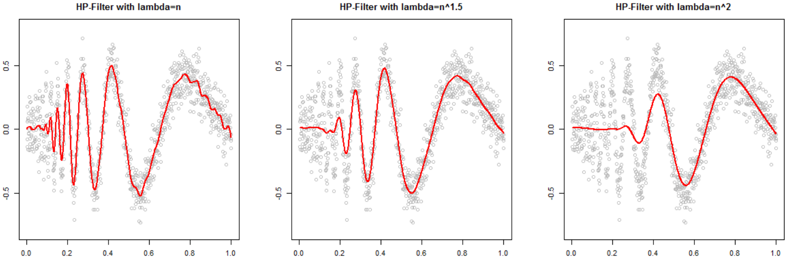
- $\boldsymbol{D}^{(2)} = [\cdots;0\ldots0,1,-2,1,0\ldots0;\cdots]$ is the 2nd-order difference matrix

Illustrative Examples



R:hpfilter Results

- R package: https://cran.r-project.org/package=mFilter
- Use `hpfilter(y, type="lambda", freq)` with λ-specification



Trend Filtering: ℓ_1 approach

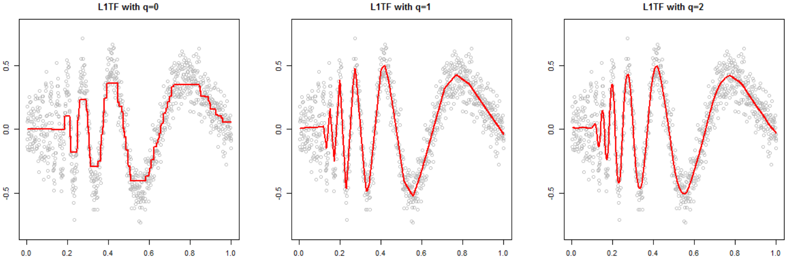
- ℓ_1-trend filtering by Kim, et al. (2009) and Tibshiran (2014):

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{D}^{(q+1)} \boldsymbol{\alpha}\|_{\ell_1}, \quad q = 0, 1, 2, \ldots$$

- Extended to different orders of finite differences.
- The ℓ_1-penalty induces the piecewise smoothness.
- Hyperparameter can be determined by the BIC criterion.

ℓ_1-Trend Filtering Results

- R package at https://github.com/glmgen/glmgen
- Use `trendfilter(y, k=q)` plus BIC parameter tuning



Research on ℓ_0-Trend Filtering

- The ℓ_0-regularized trend filtering problem is formulated by

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{D}^{(q+1)}\boldsymbol{\alpha}\|_{\ell_0}, \quad q = 0, 1, 2, \ldots$$

- Much more promising results, but challenging with ℓ_0-optimization
- R:AMIAS Package: https://cran.r-project.org/package=AMIAS

Table of Contents

Penalzied B-Splines

- Initialize B-Spline bases (degree q) with dense knots (equal spaced)
- Run the ℓ_2-penalized regression:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{D}^{(q+1)}\boldsymbol{\alpha}\|_{\ell_2}^2,$$

  where $\boldsymbol{\Phi}$ represents the design matrix generated by B-Spline bases.

- It leads to the closed-form solution (generalized ridge estimator):

$$\hat{\boldsymbol{y}} = \boldsymbol{\Phi}\left(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda (\boldsymbol{D}^{(q+1)})^T \boldsymbol{D}^{(q+1)}\right)^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}$$

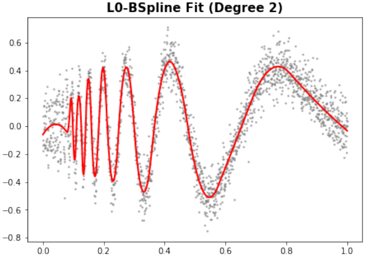
- Note that this is used by the pyGAM package (to be discussed).

Research on ℓ_0-penalized B-Splines

- Ongoing investigation by switching ℓ_2-penalty to ℓ_0-penalty:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\alpha}\|_2^2 + \lambda \|\boldsymbol{D}^{(q+1)}\boldsymbol{\alpha}\|_{\ell_0},$$

- An iterative reweighing solution being developed, with promising results:



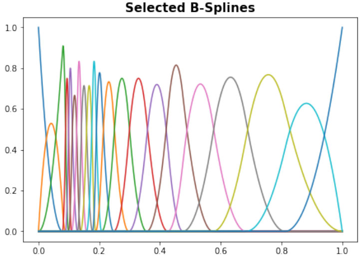


Table of Contents

The pyGAM Package

- A Python package for GAM: https://github.com/dswah/pyGAM
- `pip install pygam`
- The pyGAM package adopts the ℓ_2-penalized B-splines.
- It supports increasing/decreasing, convex/concave constraints.
- It comes with "partial dependency plot" for visualizing feature functions.
- See the supplementary Python code/notebook for demonstration with examples ...

Thank You!

Q&A or Email ajzhang@umich.edu