

GAM fitting & Big Data methods

Matteo Fasiolo (University of Bristol, UK)

matteo.fasiolo@bristol.ac.uk

Material available at:

https://github.com/mfasiolo/workshop_BOZEN19

These slides cover:

- 1 GAM model fitting
- 2 Big Data GAM methods
- 3 R demonstration

GAM model fitting

Recall the GAM model structure:

$$y|\mathbf{x} \sim \text{Distr}\{y|\mu(\mathbf{x}), \theta\}$$

where $\mu(\mathbf{x}) = \mathbb{E}(y|\mathbf{x}) = g^{-1}\left\{\sum_{j=1}^m f_j(\mathbf{x})\right\}$.

The f_j 's can be

- parametric e.g. $f_j(\mathbf{x}) = \beta_1 x_j + \beta_2 x_j^2$
- random effects
- spline-based smooths such as

$$f_j(x_j) = \sum_{i=1}^r \beta_{ji} b_{ji}(x_j)$$

where β_{ji} are coefficients and $b_{ji}(x_j)$ are known spline basis functions.

NB: we call $\sum_{j=1}^m f_j(\mathbf{x})$ **linear predictor** because it is linear in β .

GAM model fitting

$\hat{\beta}$ is the maximizer of **penalized** log-likelihood

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \operatorname{PenLogLik}(\beta|\gamma) = \underset{\beta}{\operatorname{argmax}} \left\{ \overbrace{\log p(\mathbf{y}|\beta)}^{\text{goodness of fit}} - \underbrace{\operatorname{Pen}(\beta|\gamma)}_{\text{penalize complexity}} \right\}$$

where:

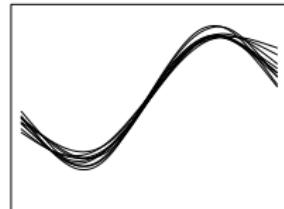
- $\log p(\mathbf{y}|\beta) = \sum_i \log p(y_i|\beta)$ is log-likelihood (i.i.d. case)
- $\operatorname{Pen}(\beta|\gamma)$ penalizes the complexity of the f_j 's
- $\gamma > 0$ smoothing parameters ($\uparrow \gamma \uparrow$ smoothness)

GAM model fitting

`mgcv` uses a hierarchical fitting framework:

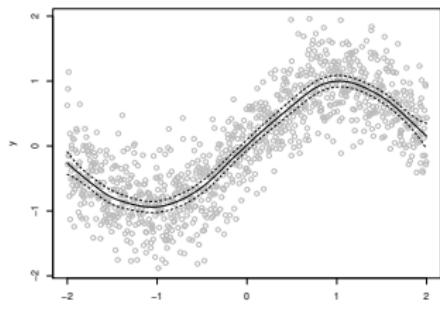
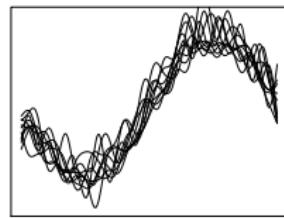
- ① Select γ to determine smoothness

$$\hat{\gamma} = \underset{\gamma}{\operatorname{argmax}} \text{LAML}(\gamma).$$



- ② For fixed γ , estimate β to determine actual fit

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \text{PenLogLik}(\beta|\gamma).$$



GAM model fitting

Assume smoothing parameter γ are known, so

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \left\{ \log p(\mathbf{y}|\beta) - \text{Pen}(\beta|\gamma) \right\}$$

Concrete example:

- $\mathbb{E}(y|x) = f(x)$
- $f(x) = \sum \beta_j b_j(x) = \beta^T \mathbf{b}(x)$
- $f''(x) = \beta^T \mathbf{b}''(x)$
- a cubic spline penalty is

$$\int f''(x)^2 dx = \int \beta^T \mathbf{b}''(x) \underbrace{\beta^T \mathbf{b}''(x)}_{= \mathbf{b}''(x)\beta^T} dx = \beta^T \left[\int \mathbf{b}''(x) \mathbf{b}''(x)^T dx \right] \beta$$

Define $\mathbf{S} = \int \mathbf{b}''(x) \mathbf{b}''(x)^T dx$

GAM model fitting

Hence

$$\hat{\beta} = \underset{\beta}{\operatorname{argmax}} \left\{ \log p(\mathbf{y}|\beta) - \frac{1}{2}\gamma \underbrace{\beta^T \mathbf{S} \beta}_{\text{Pen}(\beta|\gamma)} \right\}$$

In general $\mathbb{E}(y|\mathbf{x}) = \sum_j f_j(\mathbf{x})$ and penalty matrix is $\mathbf{S}_\gamma = \sum_j \gamma_j \mathbf{S}_j$.

Bayesian view: consider *smoothing prior* $\beta \sim N(\mathbf{0}, \mathbf{S}_\gamma^-)$ call it $p(\beta)$.

By Bayes theorem

$$p(\beta|\mathbf{y}) = \frac{p(\mathbf{y}|\beta)p(\beta)}{p(\mathbf{y})}$$

or

$$\log p(\beta|\mathbf{y}) = \log p(\mathbf{y}|\beta) + \log p(\beta) - \log p(\mathbf{y}).$$

But $\log p(\beta) = -\frac{1}{2}\beta^T \mathbf{S}_\gamma \beta + \text{const}$, so

$$\log p(\beta|\mathbf{y}) = \log p(\mathbf{y}|\beta) - \frac{1}{2}\beta^T \mathbf{S}_\gamma \beta + \text{const.}$$

GAM model fitting

So $\text{PenLogLik}(\beta|\gamma) \propto \log p(\beta|y)$: we are doing **Maximum a Posteriori** (MAP) estimation!

How to select smoothing parameters γ ?

Recall

$$p(\beta|y) = \frac{p(y|\beta)p(\beta)}{p(y)}$$

where

$$p(y) = \int p(y|\beta) \underbrace{\frac{p(\beta)}{p(\beta|\gamma)}}_{p(\beta|\gamma)} d\beta = p(y|\gamma).$$

We want to maximize $p(y|\gamma)$ wrt γ .

Integral is intractable \rightarrow use Laplace Approximate Marginal Likelihood

$$\hat{\gamma} = \operatorname*{argmax}_{\gamma} \text{LAML}(\gamma).$$

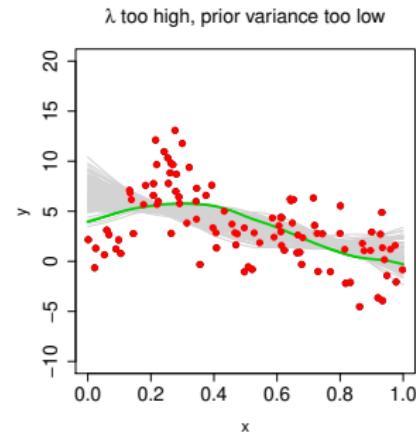
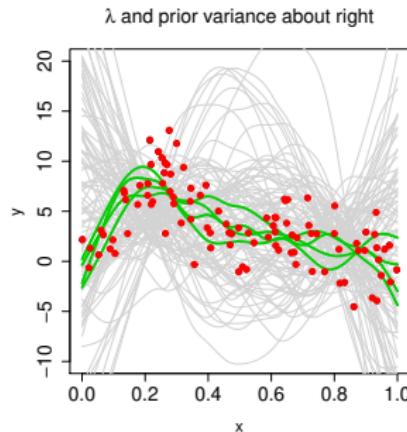
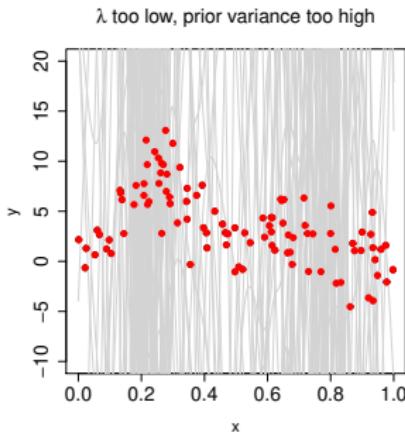
GAM model fitting

Why do we want to maximize

$$\text{LAML}(\gamma) \approx p(\mathbf{y}|\gamma) = \int p(\mathbf{y}|\beta)p(\beta|\gamma)d\beta$$

wrt γ ?

Let $\lambda = 1/\gamma$



GAM model fitting

Alternatives LAML for γ selection:

- Generalized Cross-Validation (GCV)
- Akaike Information Criterion (AIC)

but LAML is most widely applicable in `mgcv`.

Variance parameters of random effects can be included in γ and estimated by LAML.

To choose γ estimation method in `mgcv`

```
fit <- gam(y ~ ... , method = "REML")
```

see `?gam`.

These slides cover:

- 1 GAM model fitting
- 2 Big Data GAM methods
- 3 R demonstration

GAMs for Big Data

Example: a Gaussian GAM for expected load is

$$\begin{aligned}\mathbb{E}(\text{Load}_i) &= \sum_{j=1}^7 \beta_j w_{d(i)}^j \cdot \text{Day-of-week factor} \\ &+ \beta_8 \text{Load}_{i-48} \cdot \text{Lagged load} \\ &+ f_1(t_i) \cdot \text{Long-term trend} \\ &+ f_2(T_i) \cdot \text{Temperature} \\ &+ f_3(T_i^s) \cdot \text{Smoothed temperature} \\ &+ f_4(\text{toy}_i), \quad \cdot \text{Time-of-year}\end{aligned}$$

where $T_i^s = \alpha T_i + (1 - \alpha) T_{i-1}^s$, with $\alpha = 0.05$.

It is standard practice to model the 48 30min slots separately.

So we need to fit 48 models.

GAMs for Big Data

Example: a more ambitious model is

$$\begin{aligned}\mathbb{E}(\text{Load}_i) &= \sum_{j=1}^7 \beta_j w_{d(i)}^j \quad \cdot \text{Day-of-week factor} \\ &+ f(\text{tod}_i) \text{Load}_{i-48} \quad \cdot \text{Lagged load} \\ &+ \text{te}_1(t_i, \text{tod}_i) \quad \cdot \text{Long-term trend} \\ &+ \text{te}_2(T_i, \text{tod}_i) \quad \cdot \text{Temperature} \\ &+ \text{te}_3(T_i^s, \text{tod}_i) \quad \cdot \text{Smoothed temperature} \\ &+ \text{te}_4(\text{toy}_i, \text{tod}_i), \quad \cdot \text{Time-of-year}\end{aligned}$$

where

- tod is time of day $1, \dots, 48$
- te 's are 2D tensor product smooths
- $f(\text{tod}_i) \text{Load}_{i-48}$ is varying coefficient effect

GAMs for Big Data

Why is this useful? Some answers:

- statistical efficiency → share information across time-of-day
- ease of use and interpretation

Do we need Big Data methods? Notice that:

- n is 48 times bigger than a 30min model
- tensor product can have large number of basis functions

$$te(T, tod) = \sum_{j=1}^J \sum_{k=1}^K \beta_{ij} b_j(T) b_k(tod) = \sum_{j=1}^J \sum_{k=1}^K \beta_{ij} \tilde{b}_{jk}(T, tod)$$

so tensor effect has $J \times K$ coefficients.

GAMs for Big Data

Recall that $\mathbb{E}(\text{load}|\mathbf{x}_i) = g^{-1}(\mathbf{x}_i^\top \boldsymbol{\beta})$, where \mathbf{x}_i^\top row of

$$\mathbf{X} = \begin{bmatrix} 1 & \mathbb{1}(\text{dow}_1 = \text{Mon}) & \cdots & b_{11}(\text{T}_1, \text{tod}_1) & \cdots & b_{JK}(\text{T}_1, \text{tod}_1) & \cdots \\ 1 & \mathbb{1}(\text{dow}_2 = \text{Mon}) & \cdots & b_{11}(\text{T}_2, \text{tod}_2) & \cdots & b_{JK}(\text{T}_2, \text{tod}_2) & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \mathbb{1}(\text{dow}_n = \text{Mon}) & \cdots & b_{11}(\text{T}_n, \text{tod}_n) & \cdots & b_{JK}(\text{T}_n, \text{tod}_n) & \cdots \end{bmatrix}$$

with n rows and

$$d = p + J \times K + \cdots,$$

columns.

Bottom line: \mathbf{X} can get very big, which causes problems:

- storing \mathbf{X} takes too much memory
- computing with \mathbf{X} (e.g. $\mathbf{X}^\top \mathbf{X}$ or $\text{QR}(\mathbf{X})$) takes time

GAMs for Big Data

`bam()` implements memory-saving methods of Wood et al. (2015):

- do not create \mathbf{X} but only sub-blocks:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} \\ \mathbf{X}_{21} & \mathbf{X}_{22} \\ \vdots & \vdots \\ \mathbf{X}_{B1} & \mathbf{X}_{B2} \end{bmatrix}$$

do not store them either, but build them when needed

- any computation involving \mathbf{X} is based on the blocks

Block-oriented methods can be used also to perform fast model updates:

```
fit <- bam.update(fit, data = newData, chunk.size = 1e4)
```

GAMs for Big Data

Faster computation and memory savings using Wood et al. (2017).

Simple observation is that many variables are discrete in nature:

- time of day (tod) $\in \{1, \dots, 48\}$
- time of year (toy) $\in \{1, \dots, 365\}$
- temperature (T) $\in \{\dots, -0.1, 0, 0.1, 0.2, \dots\}$

There is room for data compression, example:

- we have 10 years of data and 48×365 obs per year
- effect of toy is

$$s(\text{toy}) = \sum_{i=1}^p \beta_i b_i(\text{toy}).$$

so model matrix part \mathbf{X} of toy is $(10 * 48 * 365) \times p$

- compressed model matrix $\bar{\mathbf{X}}$ is $365 \times p$
- saving factor $\# \text{elem}(\mathbf{X}) / \# \text{elem}(\bar{\mathbf{X}}) = 10 * 48$

GAMs for Big Data

Discretization can be applied to variables that are not “naturally” discrete.

Sampling variability is $O(n^{-\frac{1}{2}})$, so discretizing in $m = O(n^{\frac{1}{2}})$ bins is ok.

Wood et al. (2017) use discretization to fit UK black smoke pollution data from 2000 stations, with $n = 10^8$ and $p = 10^4$.

With latest `mgcv` version, the model

$$\begin{aligned}\log(\text{bs}_i) = & f_1(y_i) + f_2(\text{doy}_i) + f_3(\text{dow}_i) + f_4(y_i, \text{doy}_i) + f_5(y_i, \text{dow}_i) \\ & + f_6(\text{doy}_i, \text{dow}_i) + f_7(n_i, e_i) + f_8(n_i, e_i, y_i) + f_9(n_i, e_i, \text{doy}_i) \\ & + f_{10}(n_i, e_i, \text{dow}_i) + f_{11}(h_i) + f_{12}(T_i^0, T_i^1) + f_{13}(\bar{T}_1, \bar{T}_2) \\ & + f_{14}(r_i) + \alpha_{k(i)} + b_{\text{id}(i)} + e_i\end{aligned}$$

can be fitted in 5min on 8 cores (Li and Wood, 2019).

These slides cover:

- 1 GAM model fitting
- 2 Big Data GAM methods
- 3 R demonstration

References I

- Li, Z. and S. N. Wood (2019). Faster model matrix crossproducts for large generalized linear models with discretized covariates. *Statistics and Computing*, 1–7.
- Wood, S. N., Y. Goude, and S. Shaw (2015). Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 64(1), 139–155.
- Wood, S. N., Z. Li, G. Shaddick, and N. H. Augustin (2017). Generalized additive models for gigadata: modeling the uk black smoke network daily data. *Journal of the American Statistical Association* 112(519), 1199–1210.