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:

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

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

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 $\boldsymbol{\beta}$.

GAM model fitting

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

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

where:

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

GAM model fitting

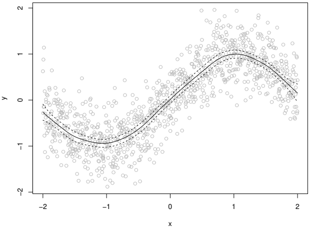
`mgcv` uses a hierarchical fitting framework:

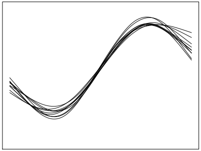
1. Select γ to determine smoothness

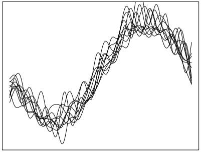
$$\hat{\gamma} = \underset{\gamma}{\text{argmax}}\ \text{LAML}(\gamma).$$

2. For fixed γ, estimate $\boldsymbol{\beta}$ to determine actual fit

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







GAM model fitting

Assume smoothing parameter γ are known, so

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

Concrete example:

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

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

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

GAM model fitting

Hence

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

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* $\boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{S}_\gamma^-)$ call it $p(\boldsymbol{\beta})$.

By Bayes theorem

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

or

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

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

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

GAM model fitting

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

How to select smoothing parameters $\boldsymbol{\gamma}$?

Recall

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

where

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

We want to maximize $p(\mathbf{y}|\boldsymbol{\gamma})$ wrt $\boldsymbol{\gamma}$.

Integral is intractable → use Laplace Approximate Marginal Likelihood

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

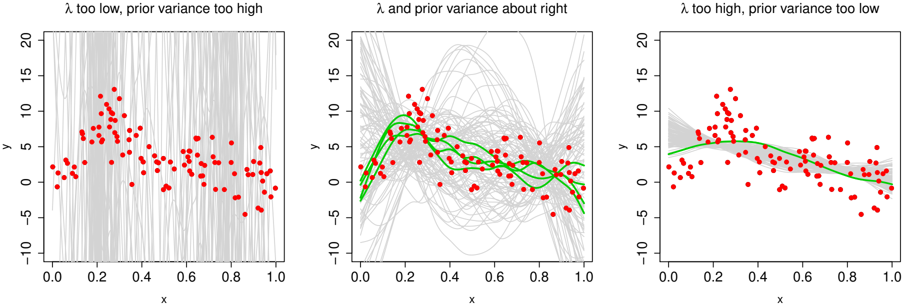
GAM model fitting

Why do we want to maximize

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

wrt γ?

Let $\boldsymbol{\lambda} = 1/\gamma$



GAM model fitting

Alternatives LAML for γ selection:

- Generalized Cross-Validation (GCV)
- Akaike Infomation 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

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

where $T^s_i = \alpha T_i + (1-\alpha) T^s_{i-1}$, 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

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

where

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

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

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

GAMs for Big Data

Recall that $\mathbb{E}(\text{load}|\mathbf{x}_i) = g^{-1}(\mathbf{X}_i^\mathsf{T}\beta)$, where $\mathbf{X}_i^\mathsf{T}$ 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. $\mathbf{X}^\mathsf{T}\mathbf{X}$ or $\text{QR}(\mathbf{X})$) takes time

GAMs for Big Data

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

- do not create **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 **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 year 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 $\#elem(\mathbf{X}) / \#elem(\bar{\mathbf{X}}) = 10 * 48$

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

$$\begin{aligned}\log(\mathtt{bs}_i) &= f_1(\mathtt{y}_i) + f_2(\mathtt{doy}_i) + f_3(\mathtt{dow}_i) + f_4(\mathtt{y}_i, \mathtt{doy}_i) + f_5(\mathtt{y}_i, \mathtt{dow}_i) \\ &\quad + f_6(\mathtt{doy}_i, \mathtt{dow}_i) + f_7(\mathtt{n}_i, \mathtt{e}_i) + f_8(\mathtt{n}_i, \mathtt{e}_i, \mathtt{y}_i) + f_9(\mathtt{n}_i, \mathtt{e}_i, \mathtt{doy}_i) \\ &\quad + f_{10}(\mathtt{n}_i, \mathtt{e}_i, \mathtt{dow}_i) + f_{11}(\mathtt{h}_i) + f_{12}(\mathtt{T}^0_i, \mathtt{T}^1_i) + f_{13}(\bar{\mathtt{T}}\mathtt{1}_i, \bar{\mathtt{T}}\mathtt{2}_i) \\ &\quad + f_{14}(\mathtt{r}_i) + \alpha_{k(i)} + b_{\text{id}(i)} + e_i\end{aligned}$$

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.