Big Data GAM methods

Matteo Fasiolo (University of Bristol, UK)

matteo.fasiolo@bristol.ac.uk

Material available at:

`https://github.com/mfasiolo/GAM_Workshop_Enbis_EDF_23`

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.

Example: a more ambitious model is

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

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

Why is this useful? Some answers:

- statistical efficiency $\rightarrow$ 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

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

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

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

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

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

`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)
```

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 $\#elem(\mathbf{X}) / \#elem(\bar{\mathbf{X}}) = 10 * 48$

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

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

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

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.