Tree-based estimation of point process intensity functions

Changqing Lu, Yongtao Guan
Marie-Colette van Lieshout and Ganggang Xu

colette@cwi.nl

CWI & University of Twente
The Netherlands

Intensity function

Let X be a **point process** on the plane and write $N(A)$ for the number of points X places in (Borel) set A. Its **intensity function** is the Radon–Nikodym derivative of the moment measure

$$M(A) = \mathbb{E}N(A) = \int_A \lambda(x)dx$$

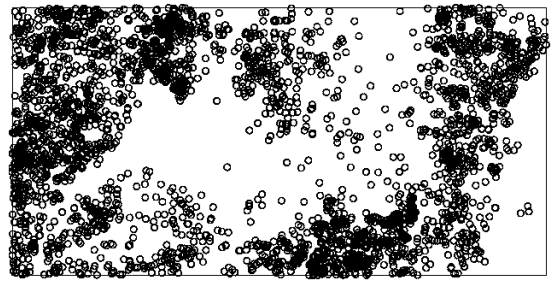
so that

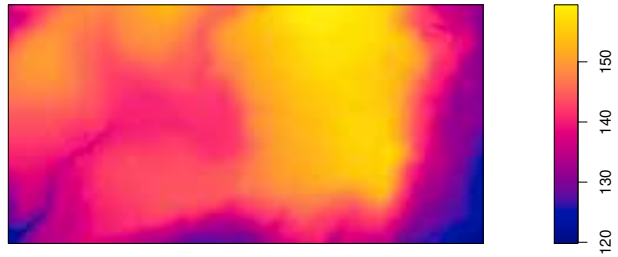
$$\lambda(x)dx = \mathbb{P}(X \text{ has a point in } dx).$$

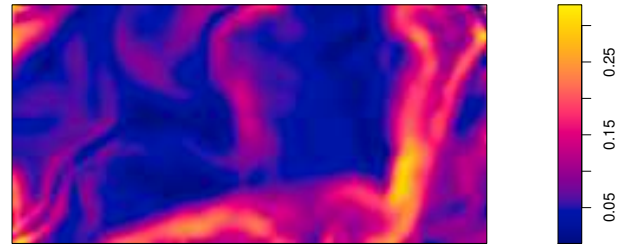
Often, λ is a function of **covariates** $z = (z_1, \ldots, z_p)$ where $z_i : \mathbb{R}^2 \to \mathbb{R}$.

Example: the prevalence of trees depends on terrain characteristics and availability of nutrients.

Point pattern of trees and two (of 8) covariates







Non-parametric estimators

Kernel estimators can be defined in the spatial or the covariate domain. For the latter

$$\hat{\lambda}(x_0) = \frac{c(x_0, \sigma)}{\sqrt{2\pi}\sigma} \sum_{x \in X \cap W} \exp(-||z(x) - z(x_0)||^2/(2\sigma^2)),$$

where $c(x_0, \sigma)$ is an edge correction factor and W the observation window.

The method is **computationally fast** but **depends crucially** on σ^2

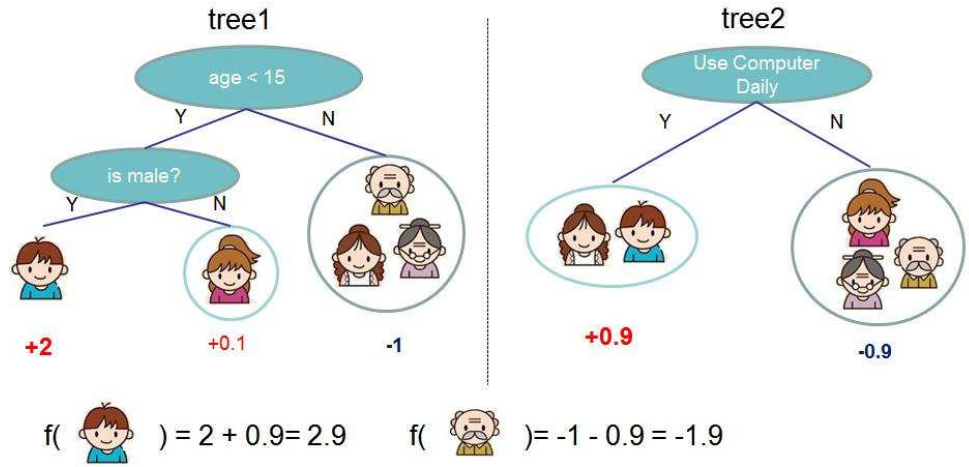
Goals:

- develop a tree-based approach that can deal with many covariates,
- using an appropriate loss function,
- that outperforms kernel estimators in terms of MAE.

The XGBoostPP model

Write

$$\log \lambda(x) = \sum_{k=1}^{K} f_k(z(x)).$$



Chen and Guestrin, 2016.

Each tree k is uniquely defined by its structure q_k and leaf score vector $\theta_{k,v}$ over leaves v.

Loss function

We strive to minimise the **weighted Poisson likelihood loss**

$$L((f_k)_{k=1}^K) = \gamma \sum_{k=1}^{K} \sum_{v} |\theta_{k,v}| - \sum_{x \in X} w(x; (f_k)_k)) \log \lambda(x; (f_k)_k)$$

$$+ \int_W w(x; (f_k)_k)) \lambda(x; (f_k)_k) dx$$

for regularisation parameter $\gamma > 0$ to avoid over-fitting.

Weights $w \equiv 1$ for Poisson and regular point processes. For clustered point processes, set

$$w(x; (f_k)_k)) = \frac{1}{1 + \lambda(x; (f_k)_k) \int_W (g(x - w) - 1) dw}$$

to account for the spatial dependence, where g is the **pair correlation function** of X.

Additive greedy optimisation

The space of tree structures and associated leaf values is **too large** to allow exhaustive searching.

Practical solution:

- iteratively add a single tree as follows:
  - for each tree structure, calculate the optimal leaf scores based on second order Taylor approximation, also updating the weights if necessary;
  - find the best tree structure by iteratively adding branches.

Leaf scores

Given $(\hat{f}_1, \ldots, \hat{f}_{k-1})$, add f_k having tree structure q_k with leaves v. Write

$$I_{k,v}(z) = \{x \in W : q_k(z(x)) = v\}.$$

Also update $\hat{w}_k(x) = w(x; \hat{f}_1, \ldots, \hat{f}_k)$ in the clustered case.

Leaf v contributes approximately (up to second order)

$$L(k, I_{k,v}(z), \theta_{k,v}) =$$

$$\gamma|\theta_{k,v}| - \theta_{k,v} \sum_{x \in X \cap I_{k,v}} \hat{w}_k(x) + \left(\theta_{k,v} + \frac{1}{2}\theta_{k,v}^2\right) \int_{I_{k,v}} \hat{w}_k(x)\lambda(x; \hat{f}_1, \ldots, \hat{f}_{k-1})dx$$

to the loss, which can be minimised to give leaf score $\hat{\theta}_{k,v}$.

Leaf scores (ctd)

$$\hat{\theta}_{k,v} = \frac{\operatorname{sgn}(R_{k,v} - T_{k,v}) \max(|R_{k,v} - T_{k,v}| - \gamma, 0)}{T_{k,v}},$$

where

$$R_{k,v} = \sum_{x \in X \cap I_{k,v}} \hat{w}_k(x)$$

$$T_{k,v} = \int_{I_{k,v}} \hat{w}_k(x) \hat{\lambda}(x; \hat{f}_1, \ldots, \hat{f}_{k-1}) dx.$$

Node splits

Idea: Start from a single leaf and iteratively add branches.

Given current tree q_k, consider **splitting** leaf v based on **threshold** value M of **covariate** component z_i. Set

$$I_{L_{k,v}} = \{x \in W : q_k(z) = v; z_i(x) \leq M\}$$

$$I_{R_{k,v}} = \{x \in W : q_k(z) = v; z_i(x) > M\}$$

and minimise

$$L(k, I_{L_{k,v}}, \hat{\theta}_{k,v}) + L(k, I_{R_{k,v}}, \hat{\theta}_{k,v}).$$

Hyperparameter selection

The model contains **hyperparameters** (K, γ, tree depth, ...).

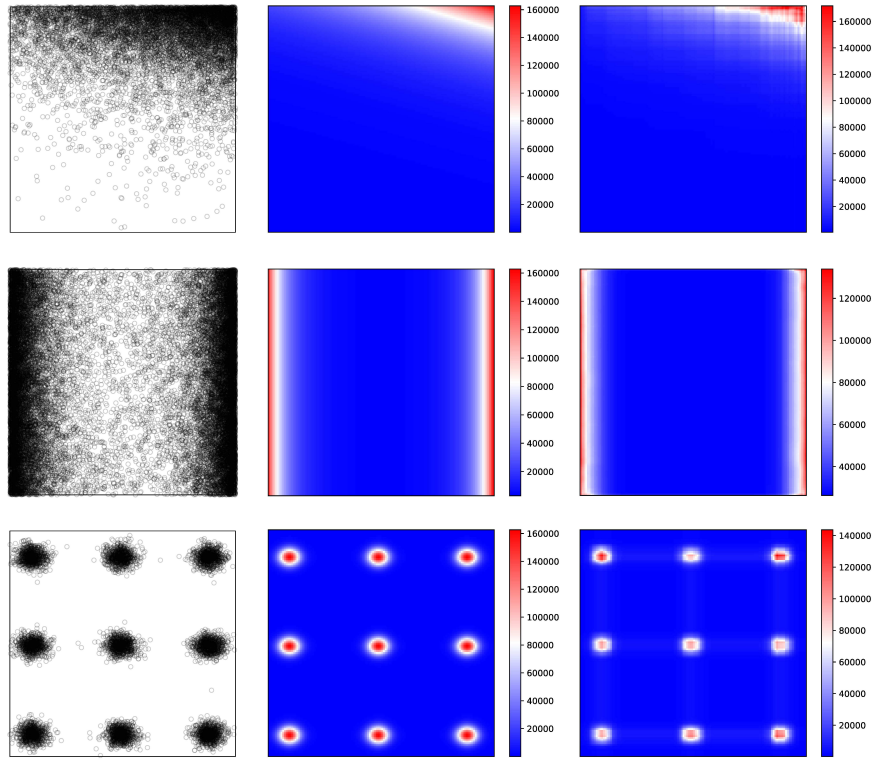
We select them by **cross-validation**. For each combination of parameters:

- randomly assign the data points to (say) four subsets $\mathbf{x_i}$, $i = 1, \ldots, 4$;
- for $i = 1, \ldots, 4$, calculate $\hat{\lambda}_i$ based on the $\mathbf{x_j}$, $j \neq i$, (so $\hat{\lambda}_i$ estimates 3 times the intensity function of $\mathbf{x_i}$) and
- maximise the cross-validation Poisson log-likelihood

$$\sum_{i=1}^{4} \left\{ \sum_{x \in \mathbf{x_i}} \log \left[\frac{1}{3} \hat{\lambda}_i(x) \right] - \frac{1}{3} \int_W \hat{\lambda}_i(x) dx \right\}$$

over K, γ etc.

Simulated examples

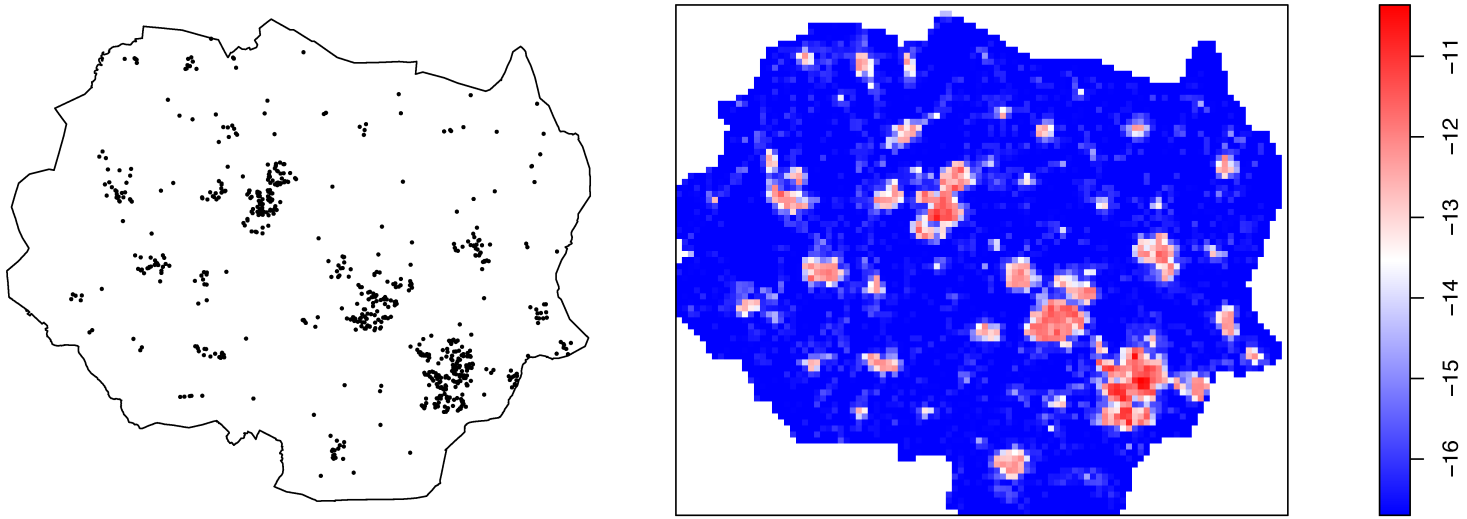


Left: data pattern (Poisson sample).

Middle: true intensity function.

Right: estimated intensity function using the two coordinates as covariates.

Kitchen fires in Twente

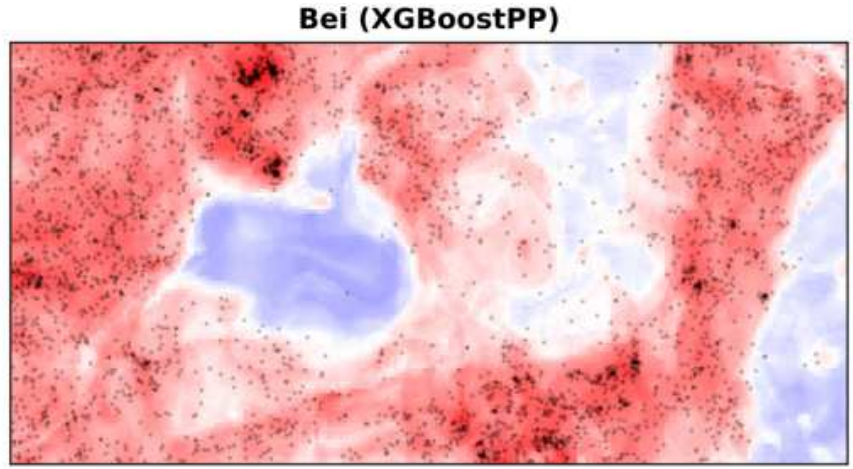


Left: pattern of 699 kitchen fire incidents in Twente (2004–2020).
Right: estimated log intensity function using 29 covariates including building density, type and use, composition of the population and energy consumption.

Conclusions

Extensive simulations show that XGBoostPP

- outperforms kernel estimators in integrated absolute error;
- outperforms neural network based competitors when there are more than a few covariates, but
- is more computationally demanding than kernel estimation.



Thank you for your attention!