Does latency prevent the Ogata recursion trick?

Marcos Costa Santos Carreira

Jul-2020

1 Setup

1.1 Two Kernels

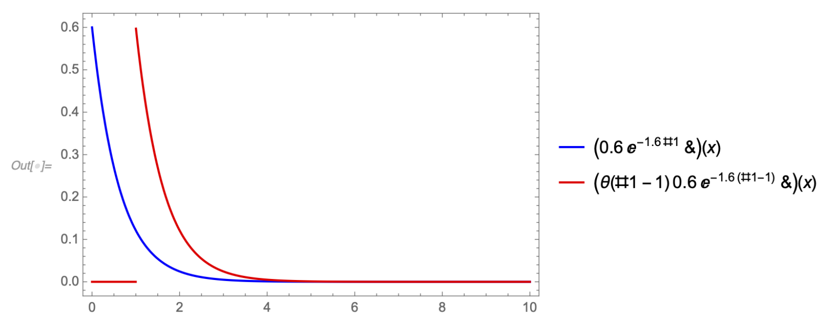

Figure 1: Kernels

1.2 Ogata recursion trick

The double summation:

$$\sum_{i=1}^{n} \log\left(\lambda_0 + \alpha \cdot \sum_{j=1}^{i-1} \exp\left(-\beta\left(t_i - t_j\right)\right)\right) \tag{1}$$

Can be converted into (Ogata 1978):

$$\sum_{i=1}^{n} \log\left(\lambda_0 + \alpha \cdot A_i\right) \tag{2}$$

With:

$$A_i = \exp\left(-\beta\left(t_i - t_{i-1}\right)\right) \cdot A_{i-1} \tag{3}$$

$$A_1 = 0 \tag{4}$$

But it seems to me that we can only do that for the exponential kernel without latency; if we had 4 timestamps in sequence with a constant difference of 0.3*latency, the last timestamp could only be influenced by the first, and I don't see how to do it with the recursion.

2 Take the long way home

I programmed the cluster simulation algorithm and the Log-Likelihood estimation (without the Ogata trick and considering latency) in Mathematica.

I checked the average number of points with a large latency against Tick and the average with zero latency against the expected theoretical value and against Tick.

For $\{\lambda_0, \alpha, \beta\} = \{1.2, 0.6, 0.8\}$, $T = 200$ and the exponential kernel with latency 1: $\alpha \cdot \exp(-\beta \cdot (t-1)) \cdot \mathbb{1}_{t \geq 1}$ I have two results for the estimation (without latency and with latency):

```
Timing[NMaximize[{loglk[λ0, α, β, teste1, 0], 0.5 < λ0 < 2.0 && 0.2 < α < β < 1.2}, {λ0, α, β}]]
{364.383, {555.102, {λ0 → 1.20848, α → 0.27753, β → 0.366348}}}

Timing[NMaximize[{loglk[λ0, α, β, teste1, 1], 0.5 < λ0 < 2.0 && 0.2 < α < β < 1.2}, {λ0, α, β}]]
{355.78, {563.68, {λ0 → 1.33498, α → 0.532636, β → 0.730975}}}
```

Figure 2: Mathematica estimation

3 Question

I'm going to program the Log-Likelihood estimation (without the Ogata trick and considering latency) in Python now. Is it needed or can I do something else? Is the Ogata trick impossible with latency?