Lasso for Hawkes process

January 29, 2019

1 Notation and definition (may be imprecise)

- M, total number of neurons
- K, number of bins
- δ, size
- $Ntot$, total number of spikes
- $\mathbb{I}_k =](k-1)\,\delta,\ k\,\delta] = ((k-1)\,\delta,\ k\,\delta], \quad \forall\, k\, \in [\![1, K]\!]$
- for any θ, $k \in [\![1, K]\!]$, $\theta + \mathbb{I}_k$ is an interval such that

$$\theta + \mathbb{I}_k = (\theta + \mathbb{I}_k) =]\theta + (k-1)\delta, \theta + k\,\delta] = (\theta + (k-1)\delta, \theta + k\,\delta]$$

- for any θ, $k \in [\![1, K]\!]$, $\theta - \mathbb{I}_k$ is an interval such that

$$\theta - \mathbb{I}_k = (\theta - \mathbb{I}_k) = [\theta - k\,\delta, \theta - (k-1)\,\delta[= [\theta - k\,\delta, \theta - (k-1)\,\delta)$$

- A is the scope, such that $A = K\,\delta$ and a maximum length. If the distance between two spikes is strictly greater than A, they don't influence each other
- $T_{\min}$ is the beginning of the study time interval
- $T^{\max}$ is the end of the study time interval
- $Int(x) = \lfloor x \rfloor$ represents the integer part of x
- $\lambda^{(r)}$ is the functional intensity of neuron r, function depending on time
- $\mu^{(r)}$ is the spontaneous part of $\lambda^{(r)}$
- $\mathcal{N}^{(l)}$ is the set of spike values of neuron l
- $N^{(l)}_{t-\mathbb{I}_k} = N^{(l)}_{[t-k\delta, t-(k-1)\delta[}$ represents the number of spikes θ of $\mathcal{N}^{(l)}$ such that

$$\theta \in (t - \mathbb{I}_k) = [t - k\delta, t - (k-1)\delta[$$

$$N^{(l)}_{t-\mathbb{I}_k} = N^{(l)}_{[t-k\delta, t-(k-1)\delta[} = \sum_{\substack{\theta < t, \\ \theta \in \mathcal{N}^{(l)}}} \mathbb{1}_{\mathbb{I}_k}(t - \theta)\ \forall\, t$$

2 Problem

We want to solve several minimization problems

$$\min_{a_r \in \mathbb{R}^{(1+MK)}} \frac{1}{2} a_r^t \, \mathbf{G} \, a_r - b_r^t \, a_r + d_r^t \, |a_r| \qquad (\mathcal{P}_{min}^{(r)})$$

where $|v|$ is a vector such that $|v|_i = |v_i| \; \forall i$.
b and d are matrices of size $(1 + MK)$-by-M, so, for any r, $b^{(r)}$ and $d^{(r)}$ are vectors of $\mathbb{R}^{(1+MK)}$ and are the rth column of b and d, respectively.
In the following, We use the class *DataSpike*. A *DataSpike* instance has two public attributes:

- $\_T$: an array of double values containing all spike values independently of the neuron numbers. Values are sorted. $\_T$ has only one row.
- *_neur*: an array of integer values of the same size of $\_T$ containing the neuron numbers.

In the algorithms, an instance of the class *DataSpike* is replaced by a matrix of double values with two rows.

3 Auxiliary functions

We define the following variables

- α is the first index of T such that $T[\alpha] > T_{\min}$
- β is the last index of T such that $T[\beta] \leqslant T^{\max}$
- *depart* is the first index of T such that $T[depart] > T_{\min} - A$
- *departbis* is the last index of T such that $T[departbis] \leqslant T^{\max} - A$

get_low_index and *get_up_index* are functions which define α, β, ...

- *get_low_index* function works as follows:

  get_low_index$(x, T) = \gamma$ where γ is the first index of T such that $T[\gamma] > x$

- *get_up_index* function works as follows:

  get_up_index$(x, T) = \gamma$ where γ is the last index of T such that $T[\gamma] \leqslant x$

get_k function works as follows:

$$get\_k(x, \delta) = \begin{cases} k & = \mathrm{Int}(\frac{x}{\delta}) + 1 = \lfloor \frac{x}{\delta} \rfloor + 1 \quad \text{if } x > k\delta, \\ k & = \mathrm{Int}(\frac{x}{\delta}) = \lfloor \frac{x}{\delta} \rfloor \qquad\qquad \text{if } x \equiv k\delta \end{cases}$$

or $get\_k(x, \delta) = k \geqslant 1$ such that $x \in \mathbb{I}_k =](k-1)\,\delta, \; k\,\delta]$
For example $get\_k(1.5\,\delta, \delta) = 2$ and $get\_k(\delta, \delta) = 1$.

Algorithm 1 *get_low_index* function

Usage: get_low_index(x, T)

Description: Computation of ind, the first index in T such that $T[ind] > x$

Input: x; scalar value

Output: ind - integer

```
ind ← 1
n ← length(T)
while (ind<(n+1)) do
    if (T[ind]⩽x) then
        ind ← ind+1
    else
        break
return ind
```

Algorithm 2 *get_up_index* function

Usage: get_up_index(x, T)

Description: Computation of ind, the last index in T such that $T[ind] \leqslant x$

Input: x; scalar value

Output: ind - integer

```
ind ← length(T)
while (ind⩾1) do
    if (T[ind]>x) then
        ind ← ind-1
    else
        break
return ind
```

Algorithm 3 *get_k* function

Usage: get_k(x, delta, eps)

Description: Computation of k such that $x \in \mathbb{I}_k =](k-1)\,\delta,\ k\,\delta]$

Input: x; scalar value

Input: delta; δ

Input: eps; for numerical precision, default value 1e-12 (optional argument)

Output: k; integer

```
k ← floor(x/delta) + 1
if |k * delta − x| < eps then
    return k
if |(k − 1) * delta − x| < eps then
    return k-1
return k
```

4 Computation of b

$$b^{(r)} = \begin{pmatrix} \#^{(r)} = \#\left\{T : T_{\min} < T \leqslant T^{\max}, T \in \mathcal{N}^{(r)}\right\} \\ \dots \\ \displaystyle\int_{T_{\min}}^{T^{\max}} N^{(l)}_{[t-k\delta, t-(k-1)\delta[}\, dN^{(r)}_t = \int_{T_{\min}}^{T^{\max}} N^{(l)}_{t-\mathbb{I}_k}\, dN^{(r)}_t = \sum_{\substack{T_{\min} < T \leqslant T^{\max}, \\ T \in \mathcal{N}^{(r)}}} \sum_{\substack{\theta < T, \\ \theta \in N^{(l)}}} \mathbb{1}_{\mathbb{I}_k}(T - \theta) \\ \dots \end{pmatrix}$$

$\boxed{\text{WARNING}}$

- In what follows, numbering starts at 1 (think Matlab/Octave and R) and you should be careful when it differs.
- In what follows
  $for(i = \alpha : \beta)$ is empty if $\alpha > \beta$ (unlike what happens in R)

Algorithm 4 Computation of b

Usage: compute_b(M, K, Tmin, Tmax, DS, delta)
Description: Computation of b - numbering of neurons and k starts at 1
Input: M; integer - number of neurons
Input: K; integer - number of bins
Input: Tmin - minimum time
Input: Tmax - maximum time
Input: DS - DataSpike: matrix with two rows, the first one contains spike values (every spikes in a single row), the second one contains the neuron numbers to identify them in the first one
Input: delta - δ
Output: b - (1+M*K)-by-M matrix
Output: low - array of size Ntot(=length(T)), taking into account the scope
Output: cnt - array of size M, containing $\#^{(r)}$

```
T ← DS[1,]                                         # first row of DS: all spike values
neur ← DS[2,]                                      # second row of DS: neuron numbers
Ntot ← length(T)                                   # total number of spikes
A ← K ∗ δ                                          # scope
b ← matrix(0, nrow=(1+M*K), ncol= M)               # init of b
low ← vector(mode='integer', length=Ntot)          # init of low - for the State
cnt ← vector(mode='integer', length=M)             # init of cnt - for the algo
ilow ← 1
eps ← 1.e-12                                       # for numerical precision
for (i=1:Ntot) do                                  # loop over spikes
    t ← T[i]; r ← neur[i]
    while (|T[ilow] − t| > A) do
        ilow ← ilow+1
    low[i] ← ilow
    if (T_min < t ⩽ T^max) then
        cnt[r] ← cnt[r]+1
    for (j=ilow:(i-1)) do
        if (|t − T[j]| < eps) then
            break
        k ← get_k((t-T[j])/delta)+1                # more complex than integer part
        l ← neur[j]
        b[(l-1)*K+k+1, r] += 1
b[1,] ← cnt                                        # 1st line of b
return b, low, cnt
```

5 Computation of G

G is a matrix of size $(1+MK)$-by-$(1+MK)$

- First row or first column of G

$$
\begin{aligned}
G_{0,l,k} &= \int_{T_{\min}}^{T^{\max}} N^{(l)}_{[t-k\delta,t-(k-1)\delta[}\, dt \\
&= \int_{T_{\min}}^{T^{\max}} \sum_{\substack{\theta<t,\\ \theta\in\mathcal{N}^{(l)}}} \mathbb{1}_{\mathbb{I}_k}(t-\theta)\, dt \simeq \int_{T_{\min}}^{T^{\max}} \sum_{\substack{(t-A)\leqslant\theta<t,\\ \theta\in\mathcal{N}^{(l)}}} \mathbb{1}_{\mathbb{I}_k}(t-\theta)\, dt \\
&\simeq \int_{T_{\min}}^{T^{\max}} \sum_{\substack{(t-A)\leqslant\theta<t,\\ \theta\in\mathcal{N}^{(l)}}} \mathbb{1}_{\mathbb{I}_k+\theta}(t)\, dt \\
G_{0,l,k} &= \int_{T_{\min}}^{T^{\max}} N^{(l)}_{[t-k\delta,t-(k-1)\delta[}\, dt \simeq \sum_{\substack{\theta\in\mathcal{N}^{(l)},\\ (T_{\min}-A)\leqslant\theta<T^{\max}}} \left(\min\left(T^{\max}, k\,\delta+\theta\right) - \max\left(T_{\min}, (k-1)\,\delta+\theta\right)\right)
\end{aligned}
$$

- Inner part of G

$$
\begin{aligned}
G_{l_1,l_2,k_1,k_2} &= \int_{T_{\min}}^{T^{\max}} N^{(l_1)}_{[t-k_1\delta,t-(k_1-1)\delta[}\, N^{(l_2)}_{[t-k_2\delta,t-(k_2-1)\delta[}\, dt \\
G_{(l_1,k_1),(l_2,k_2)} &= \int_{T_{\min}}^{T^{\max}} \left(\sum_{\substack{\tau\in\mathcal{N}^{(l_1)},\\ \tau<t}} \mathbb{1}_{\mathbb{I}_{k_1}+\tau}(t) \right) \left(\sum_{\substack{\theta\in\mathcal{N}^{(l_2)},\\ \theta<t}} \mathbb{1}_{\mathbb{I}_{k_2}+\theta}(t) \right) dt \\
&\simeq \int_{T_{\min}}^{T^{\max}} \left(\sum_{\substack{\tau\in\mathcal{N}^{(l_1)},\\ t-A\leqslant\tau<t}} \mathbb{1}_{\mathbb{I}_{k_1}+\tau}(t) \right) \left(\sum_{\substack{\theta\in\mathcal{N}^{(l_2)},\\ t-A\leqslant\theta<t}} \mathbb{1}_{\mathbb{I}_{k_2}+\theta}(t) \right) dt \\
&= \sum_{\substack{\tau\in\mathcal{N}^{(l_1)},\theta\in\mathcal{N}^{(l_2)},\\ (T_{\min}-A)\leqslant\tau<T^{\max},\\ (T_{\min}-A)\leqslant\theta<T^{\max}}} \int_{T_{\min}}^{T^{\max}} \mathbb{1}_{\mathbb{I}_{k_1}+\tau}(t)\, \mathbb{1}_{\mathbb{I}_{k_2}+\theta}(t)\, dt \\
&= \sum_{\substack{\tau\in\mathcal{N}^{(l_1)},\theta\in\mathcal{N}^{(l_2)},\\ (T_{\min}-A)\leqslant\tau<T^{\max},\\ (T_{\min}-A)\leqslant\theta<T^{\max}}} \int_{T_{\min}}^{T^{\max}} \mathbb{1}_{(\mathbb{I}_{k_1}+\tau)\cap(\mathbb{I}_{k_2}+\theta)\cap]T_{\min},T^{\max}]}(t)\, dt
\end{aligned}
$$

Algorithm 5 Computation of G

Usage: compute_G(M, K, Tmin, Tmax, T, neur, delta, A)
Description: Computation of b - numbering of neurons and k starts at 1
Input: δ; delta
Input: M; integer - number of neurons
Input: K; integer - number of bins
Input: Tmin - minimum time
Input: Tmax - maximum time
Input: T - flattened array of spike values (every spikes in a single row)
Input: neur - array of neuron numbers to identify them in T
Input: delta - size
Input: A - scope, maximum distance to be taken into account
Output: b - array of size (1+M*K)*M

```
G ← matrix(0, nrow=(1+M*K), ncol= (1+M*K))                 # init of G
G[1,1] ← T^max − T_min
A ← K ∗ δ                                                  # scope
depart ← get_low_index((T_min − A), T)
β ← get_up_index(T^max, T)
for (i in depart:β) do
    ti ← T[i]; l_1 ← neur[i]
    for (k in (1:K)) do                                    # 1st row and 1st column of G
        x_1 ← min(T^max, ti + k δ)
        x_2 ← max(T_min, ti + (k − 1) δ)
        dx = x_1 − x_2
        if (dx > 0) then
            G[1,(l_1-1)*K+k+1] += dx
            G[(l_1-1)*K+k+1,1] += dx
    for (j in (low[i]:(i-1))) do                           # inner part of G, l_1 ≠ l_2
        tj ← T[j]; l_2 ← neur[j]
        for (k_1 in (1:K)) do
            for (k_2 in (1:K)) do
                x_1 ← min(T^max, ti + k_1 δ, tj + k_2 δ)
                x_2 ← max(T_min, ti + (k_1 − 1) δ), (tj + (k_2 − 1) δ)
                dx = x_1 − x_2
                if (dx > 0) then
                    G[(l_1-1)*K+k_1+1,(l_2-1)*K+k_2+1] += dx
                    G[(l_2-1)*K+k_2+1,(l_1-1)*K+k_1+1] += dx
    for (k_1 in 1:K) do                                    # l_1 == l_2
        x_1 ← min(T^max, ti + k_1 δ)
        x_2 ← max(T_min, ti + (k_1 − 1) δ)
        dx = x_1 − x_2
        if (dx > 0) then                                   # diagonal part
            G[(l_1-1)*K+k_1+1,(l_1-1)*K+k_1+1] += dx
```

for (k_2 in (k_1+1):K) **do** # extradiagonal part
  $x_2 \leftarrow \max\left(T_{\min}, ti + (k_2 - 1)\,\delta\right)$
  $dx = x_1 - x_2$
  if $(dx > 0)$ **then**
    G[(l_1-1)*K+k_1+1,(l_1-1)*K+k_2+1] += dx
    G[(l_1-1)*K+k_2+1,(l_1-1)*K+k_1+1] += dx
return G

6 Computation of d

For d we need the following quantity

$$\mu_2^{(r)} = \begin{pmatrix} \dots \\ \int_{T_{\min}}^{T^{\max}} \left(N^{(l)}_{[t-k\delta, t-(k-1)\delta[}\right)^2 dN_t^{(r)} = \int_{T_{\min}}^{T^{\max}} (N^{(l)}_{t-\mathbb{I}_k})^2 \, dN_t^{(r)} \\ \dots \end{pmatrix}$$