Reconstruction de la connectivité fonctionnelle en Neurosciences: une amélioration des algorithmes actuels

Gilles Scarella [1] & Cyrille Mascart [2] & Alexandre Muzy [3] & Tien Cuong Phi [4] & Patricia Reynaud-Bouret [5]

[1] *Université Côte d'Azur, CNRS, LJAD/I3S - gilles.scarella@univ-cotedazur.fr*
[2] *Université Côte d'Azur, CNRS, I3S - mascart@i3s.unice.fr*
[3] *Université Côte d'Azur, CNRS, I3S - alexandre.muzy@cnrs.fr*
[4] *Université Côte d'Azur, CNRS, LJAD - Tien_cuong.phi@univ-cotedazur.fr*
[5] *Université Côte d'Azur, CNRS, LJAD - reynaudb@univ-cotedazur.fr*

Résumé. Afin d'identifier la connectivité fonctionnelle entre neurones, des travaux précédents (dans [2] notamment) ont utilisé des processus de Hawkes pour modéliser les intensités conditionnelles des trains de spikes et ont reconstruit la connectivité fonctionnelle par moindres carrés pénalisés. On propose ici une nouvelle méthode de construction des matrices du même problème Lasso obtenu et l'utilisation d'un autre solveur, qui semble plus efficace, pour une amélioration du temps de calcul.

Mots-clés. Neurosciences, processus de Hawkes, Lasso, connectivité fonctionnelle

Abstract. To identify the functional connectivity between neurons, previous works (see [2] in particular) have used Hawkes processes to model conditional intensities of spike trains and have reconstructed functional connectivity by penalized least square method. We propose here a new method to construct the matrices in the same resulting Lasso problem and the choice of another solver, which seems to be more efficient, to improve computational times.

Keywords. Neuroscience, Hawkes processes, Lasso, functional connectivity

1 Présentation du problème

Le but est d'étudier la connectivité fonctionnelle entre neurones, qui est un enjeu important en Neurosciences. La présente étude est basée sur l'enregistrement simultané des temps d'émission des potentiels d'action, ou spikes, de M neurones. Comme cela a été présenté dans [2], on considère M trains de spikes simultanés, modélisés comme des processus de Hawkes multivariés. L'intensité du i-ième train de spikes N^i a la forme suivante

$$\lambda_i(t) = \left(\nu_i + \sum_{j=1}^{M} \sum_{T \in N^j, T<t} h_{j \to i}(t - T) \right)_+, \quad \forall\, t, \quad \forall i \in [\![1, M]\!].$$

Le coefficient ν_i est le taux de décharge spontané du i-ième train de spikes, donnant la fréquence moyenne d'apparition d'un nouveau spike en l'absence d'excitation ou d'inhibition, et la fonction $h_{j\to i}$, qui dépend du temps, modélise l'interaction excitatrice ou inhibitrice du j-ième train sur le i-ième. On suppose que les fonctions $h_{j\to i}$ sont constantes par morceaux sur une partition de K intervalles de taille δ, telles que:

$$h_{j\to i} = \sum_{k=1}^{K} a_{j\to i}^k \, \varphi_k \quad \text{où } \varphi_k = \mathbb{1}_{((k-1)\delta, k\delta]}.$$

Les coefficients (ν_i) et $(a_{j\to i}^k)$ doivent être estimés, pour tous $(i,\, j) \in [\![1, M]\!]^2$ et $k \in [\![1, K]\!]$.

Le code *neuro-stat*, introduit dans [2], permet de retrouver une estimation *sparse* des coefficients (ν_i) et $(a_{j\to i}^k)$ et donc du graphe de connectivité fonctionnelle (où l'existence d'une connexion entre j et i correspond à la non nullité de $h_{j\to i}$). De plus, le code permet aussi, en fonction de différentes phases de l'enregistrement, d'estimer différents jeux de paramètres et différents graphes (typiquement on peut alors associer un graphe à un comportement animal).

Ici nous nous intéressons à l'amélioration de l'algorithme sur une plage de temps fixée $(T_{\min}, T^{\max}]$.
De manière générale, on se focalise sur le critère des moindres carrés suivant, que nous pénaliserons ensuite:

$$\int_{T_{\min}}^{T^{\max}} \overline{\lambda_i}^2(t)\, dt - 2 \int_{T_{\min}}^{T^{\max}} \overline{\lambda_i}(t)\, dN_t^i$$

où $\overline{\lambda}_i$ est un candidat intensité de la forme

$$\begin{aligned} \overline{\lambda_i}(t) &= \overline{\nu_i} + \sum_{j=1}^{M} \sum_{T\in N^j, T<t} \overline{h}_{j\to i}(t-T), \quad \forall\, t, \quad \forall i \in [\![1, M]\!], \\ \text{avec } \overline{h}_{j\to i} &= \sum_{k=1}^{K} \overline{a}_{j\to i}^k \, \varphi_k \end{aligned}$$

Soit $\psi_t^l(\varphi_k)$ la fonction prévisible définie par:

$$\psi_t^l(\varphi_k) = \int_{-\infty}^{t^-} \varphi_k(t-u)\, dN_u^l = \sum_{T<t, T\in N^l} \mathbb{1}_{((k-1)\delta, k\delta]}(t-T), \tag{1}$$

on en déduit de (1) que $\overline{\lambda}_i$ s'écrit donc sur le dictionnaire des fonctions prévisibles sous la forme suivante

$$\overline{\lambda}_i(t) = \overline{\nu}_i + \sum_{j=1}^{M} \sum_{k=1}^{K} \overline{a}_{j\to i}^k \, \psi_t^j(\varphi_k)$$

En introduisant comme dans [2] les matrices b et G, de dimension $(1+MK)$-by-M (resp. $(1+MK)$-by-$(1+MK)$), et en notant $\alpha(l,k) = 1+(l-1)\,K+k$, l'indice dans $[\![2, 1+M\,K]\!]$, pour $k \in [\![1, K]\!]$ et $l \in [\![1, M]\!]$, on voit que le critère des moindres carrés devient

$$-2\,{}^T\mathbf{b}^i\,\beta + {}^T\beta\,\mathbf{G}\,\beta \quad \forall i \text{ avec}$$

$$\begin{aligned}
\mathbf{b}^i_{\alpha(l,k)} &= \int_{T_{\min}}^{T^{\max}} \psi^l_t(\varphi_k)\, dN^i_t \quad \text{et} \qquad \mathbf{b}^i_1 = \#\left\{T \in N^i,\ \ T \in (T_{\min}, T^{\max}]\right\},\\
\mathbf{G}_{\alpha(l_1,k_1),\alpha(l_2,k_2)} &= \int_{T_{\min}}^{T^{\max}} \psi^{l_1}_t(\varphi_{k_1})\,\psi^{l_2}_t(\varphi_{k_2})dt, \quad \mathbf{G}_{\alpha(l,k),1} = \int_{T_{\min}}^{T^{\max}} \psi^l_t(\varphi_k)\,dt \text{ et } \mathbf{G}_{1,1} = T^{\max} - T_{\min}
\end{aligned}$$

En suivant [1], on pénalise par une norme l_1 à poids tels que $\mathbf{d}$ est une matrice de dimension $(1+MK)$-by-M définie à partir de $\mu_{\mathbf{2}}$ (de même dimension) et $\mu_{\mathbf{A}}$ vecteur de taille $(1+MK)$.

On utilise la même définition que [2] pour $\mathbf{d}$, dans laquelle on prend $\gamma = 3$.

$$\begin{aligned}
\mathbf{d}^i &= \sqrt{2\gamma\, c_{log}\, \mu_{\mathbf{2}i}} + \frac{\gamma}{3} c_{log}\, \mu_{\mathbf{A}},\ \forall i \in [\![1, 1+MK]\!],\quad c_{log} = \log((1+MK)M)\\
(\mu_{\mathbf{A}})_{\alpha(l,k)} &= \sup_{t \in (T_{\min}, T^{\max}]} |\psi^l_t(\varphi_k)|,\quad (\mu_{\mathbf{A}})_1 = 1,\\
(\mu_{\mathbf{2}})^i_{\alpha(l,k)} &= \int_{T_{\min}}^{T^{\max}} \left(\psi^l_t(\varphi_k)\right)^2\, dN^i_t,\quad (\mu_{\mathbf{2}})^i_1 = \#\left\{T \in N^i,\ \ T \in (T_{\min}, T^{\max}]\right\}.
\end{aligned}$$

On suit la même procédure que dans [2] et l'on doit résoudre un ensemble de problèmes Lasso de dimension $(1+M\,K)$, pour tout $i \in [\![1, M]\!]$ pour obtenir les coefficients $a^k_{j\to i}$

$$\begin{aligned}
\mathbf{a}^i_{BL} &= \arg \min_{\beta \in \mathbb{R}^{(1+MK)}} -2\,{}^T\mathbf{b}^i\,\beta + {}^T\beta\,\mathbf{G}\,\beta + 2\,{}^T\mathbf{d}^i\,|\beta|\\
&\quad \text{avec } \mathbf{a}^i_{BL} = (\nu_i, a^1_{1\to i}, a^2_{1\to i}, \cdots, a^K_{1\to i}, a^1_{2\to i}, \cdots, a^K_{M\to i})
\end{aligned}$$

2 Nouvelle méthode et résultats

La méthode mise en œuvre dans [2], pour le package *neuro-stat*, est coûteuse en temps calcul et utilisable uniquement dans le logiciel R, ce qui conduit à certaines limitations. En effet, cette méthode utilisait des calculs d'intégrales de fonctions constantes par morceaux pour définir les matrices intervenant dans le problème d'estimation. Si $Ntot$ désigne le nombre total de spikes, la complexité était alors en $O(M\,Ntot\,K^2)$ pour $\mathbf{G}$ et en $O(M\,Ntot\,K)$ pour $(\mathbf{b}, \mu_{\mathbf{A}}, \mu_{\mathbf{2}})$.

L'objectif est de considérer ici entre 1 000 et 10 000 neurones, soit un nombre supérieur à celui considéré dans [2]. La référence est le *BlueBrain project*[1] qui simule des colonnes corticales d'environ 10 000 neurones.

[1] `https://www.epfl.ch/research/domains/bluebrain/`

La nouvelle méthode présentée ici est moins coûteuse, on utilise la portée $A = K\delta$ de telle sorte que l'influence du spike τ_1 est nulle sur le spike τ_2 si $|\tau_1 - \tau_2| > A$. Par exemple, on obtient pour la partie intérieure de $\mathbf{G}$,

$$\mathbf{G}_{\alpha(l_1,k_1),\alpha(l_2,k_2)} \simeq \sum_{\substack{\tau \in N^{l_1},\, \theta \in N^{l_2}, \\ (T_{\min}-A) \leqslant \tau, \theta < T^{\max}}} \int_{T_{\min}}^{T^{\max}} \mathbb{1}_{(\tau+(k_1-1)\delta, \tau+k_1\delta] \cap (\theta+(k_2-1)\delta, \theta+k_2\delta] \cap (T_{\min}, T^{\max}]}(t)\, dt$$

Ce terme peut être calculé explicitement et son calcul est décrit dans l'Algorithme 1, où l'on passe en revue chacun des spikes et l'on affecte sa contribution au bon coefficient de $\mathbf{G}$. Dans l'Algorithme 1, T est un tableau de taille $Ntot$, contenant les valeurs des spikes indépendamment des neurones dans l'ordre croissant, *neur* est un tableau de même taille que T contenant le numéro de neurone du spike correspondant.

Algorithm 1 Calcul de G

Usage: compute_G(M, K, $T_{\min}$, $T^{\max}$, T, neur, δ)

```
 1: G ← matrix(0, nrow=(1+M*K), ncol= (1+M*K))          # initialization of G
 2: G[1,1] ← T^max − T_min
 3: A ← K ∗ δ                                            # scope
 4: α ← get_low_index((T_min − A), T)                    # lowest index in T such that T[α] > T_min − A
 5: β ← get_low_index(T^max, T) - 1                      # highest index in T such that T[β] ⩽ T^max
 6: for (i in α:β) do
 7:     ti ← T[i]; l_1 ← neur[i]
 8:     for (j in (low[i]:(i-1))) do                     # low[i] is the lowest index s.t T[i] − T[low[i]] ⩽ A
 9:         tj ← T[j]; l_2 ← neur[j]
10:         for (k_1 in (1:K)) do
11:             for (k_2 in (1:K)) do
12:                 x_1 ← min(T^max, ti + k_1 δ, tj + k_2 δ)
13:                 x_2 ← max(T_min, ti + (k_1 − 1) δ), (tj + (k_2 − 1) δ)
14:                 dx = x_1 − x_2
15:                 if (dx > 0) then
16:                     G[(l_1-1)*K+k_1+1,(l_2-1)*K+k_2+1] += dx
17:                     G[(l_2-1)*K+k_2+1,(l_1-1)*K+k_1+1] += dx
18:     ...
```

Les Figures 1 et 2 montrent une comparaison des temps de construction de $\mathbf{G}$ entre *neuro-stat* et la nouvelle méthode, sur deux exemples, le premier à $Ntot$ fixé ($Ntot \simeq 9.0\,10^6$), simulant des processus de Poisson; le second tel que $Ntot = M\,\nu(T_{\min} - T^{\max})$ où ν est une fréquence donnée ($\nu = 20$ Hz), simulant des processus de Hawkes sur l'intervalle de temps $(0, 100]$ avec le code SPIKES (voir [3]). La nouvelle méthode est plus efficace.

D'autre part, le solveur Lasso a été modifié par rapport à [2], c'est désormais la méthode LARS (voir [4]) qui est utilisée et semble plus efficace que la précédente. Les

Figures 3 et 4 donnent une comparaison des temps de résolution du Lasso pour deux tests, la Figure 3 correspond au test 2 pour $K = 2$ et la Figure 4 correspond à un test simulant des processus de Hawkes sur $(0, 20]$ pour $K = 5$ (utilisant la fonction *Hawkesmulti* de *neuro-stat*).

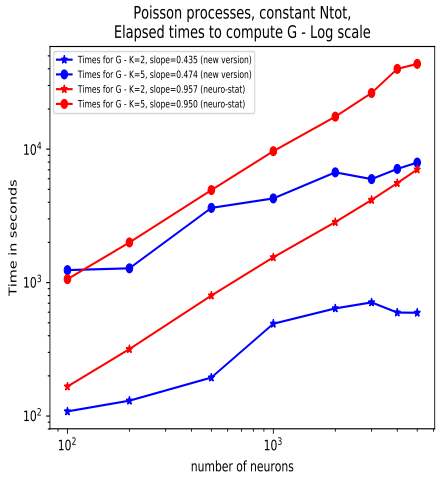

Figure 1: Comparaison des temps calcul de **G** - $K = 2$ et 5 - Test 1

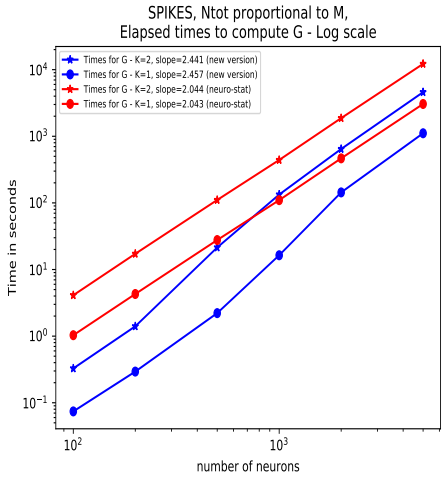

Figure 2: Comparaison des temps calcul de **G** - $K = 1$ et 2 - Test 2

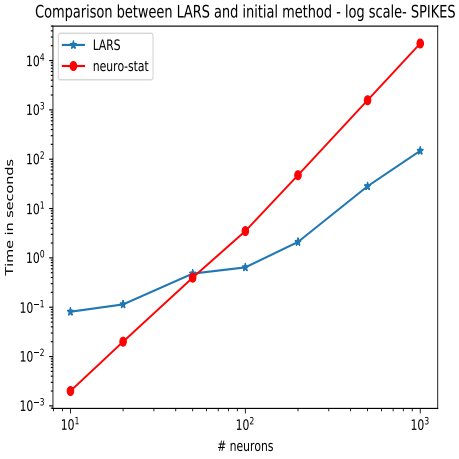

Figure 3: Temps calcul pour Lasso Test 2, K=2

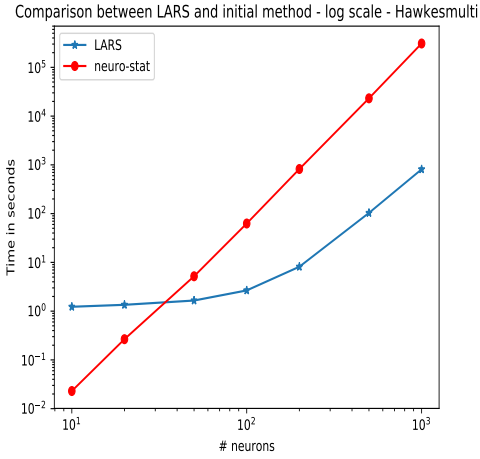

Figure 4: Temps calcul pour Lasso Test 3, K=5

3 Conclusion

On a expliqué comment améliorer le temps de calcul pour la construction de la matrice $\mathbf{G}$ par rapport à la méthode précédente. On procède de manière semblable pour $\mathbf{b}$, $\mu_{\mathbf{2}}$ et $\mu_{\mathbf{A}}$. Cependant, la nouvelle version du calcul de $\mu_{\mathbf{2}}$ reste coûteuse, il devrait être possible de l'améliorer, ce qui fait l'objet d'un travail en cours de finalisation.

Bibliographie

[1] Hansen, N-R, Reynaud-Bouret, P. and Rivoirard, V. (2015), *Lasso and probabilistic inequalities for multivariate point processes*, Bernoulli, 21(1), 83–143.

[2] Lambert, R. et al. (2018), *Reconstructing the functional connectivity of multiple spike trains using Hawkes models*, Journal of Neuroscience Methods, 297, 9–21.

[3] Mascart, C., Muzy, A. and Reynaud-Bouret, P. (2020), *Discrete event simulation of point processes: Computational complexity analysis on sparse graphs*, submitted to ACM Trans. Algor.

[4] Efron, B. and Hastie, T. and Johnstone, I. and Tibshirani, R. (2004), *Least angle regression*, The Annals of Statistics, 2-32; 407–499.