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

Gilles Scarella, Cyrille Mascart, Alexandre Muzy, Tien Cuong Phi, Patricia Reynaud-Bouret

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 [5] 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, méthode d'active set

Abstract. To identify the functional connectivity between neurons, previous works (see [5] 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, active set method

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 [5], 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:

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

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

Le code *neuro-stat*[1], inclus dans le package R *UnitEvents*[2], introduit dans [5], permet de retrouver une estimation *sparse* des coefficients (ν_i) et $(a^k_{j\to i})$ 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:

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

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

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

Soit $\psi^l_t(\varphi_k)$ la fonction prévisible définie par:

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

[1] `https://github.com/ybouret/neuro-stat`
[2] `https://sourcesup.renater.fr/frs/?group_id=3267`

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

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

En introduisant comme dans [5] 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

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

$$\begin{array}{lll}
\mathbf{b}^i_{\alpha(l,k)} & = \displaystyle\int_{T_{\min}}^{T^{\max}} \psi^l_t(\varphi_k)\,dN^i_t \quad \text{et} \qquad \mathbf{b}^i_1 = \#\left\{T\in N^i,\;\; T\in(T_{\min},T^{\max}]\right\},\\
\mathbf{G}_{\alpha(l_1,k_1),\alpha(l_2,k_2)} & = \displaystyle\int_{T_{\min}}^{T^{\max}} \psi^{l_1}_t(\varphi_{k_1})\,\psi^{l_2}_t(\varphi_{k_2})dt, \quad \mathbf{G}_{\alpha(l,k),1} = \int_{T_{\min}}^{T^{\max}} \psi^l_t(\varphi_k)\,dt \text{ et } \mathbf{G}_{1,1}=T^{\max}-T_{\min}
\end{array} \tag{2}$$

En suivant [3], 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 [5] pour $\mathbf{d}$, dans laquelle on prend $\gamma = 3$.

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

On suit la même procédure que dans [5] 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 $a^k_{j\to i}$

$$\begin{array}{lll}
\mathbf{a}^i_{BL} = \arg & \displaystyle\min_{\beta\in\mathbb{R}^{(1+MK)}} & -2\,{}^T\mathbf{b}^i\,\beta + {}^T\beta\,\mathbf{G}\,\beta + 2\,{}^T\mathbf{d}^i\,|\beta|\\
\text{avec} & \mathbf{a}^i_{BL} & = (\nu_i, a^1_{1\to i}, a^2_{1\to i}, \cdots, a^K_{1\to i}, a^1_{2\to i}, \cdots, a^K_{M\to i})\\
\text{et} & |\beta| & = (|\beta_1|, |\beta_2|, \cdots, |\beta_{MK}|, |\beta_{1+MK}|)
\end{array} \tag{4}$$

2 Nouvelle méthode et résultats

La méthode mise en œuvre dans [5], dans le code *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 [5]. La référence est le *BlueBrain project*[3] qui simule des colonnes corticales d'environ 10 000 neurones.

La nouvelle méthode présentée ici utilise, comme la précédente, des fonctions à support borné mais 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 $|\tau_1 - \tau_2| > A$. Dans la nouvelle méthode de calcul, T est un tableau de taille $Ntot$, contenant les valeurs des temps de spikes dans l'ordre croissant et indépendamment des neurones; $neur$ est un tableau de même taille que T contenant le numéro de neurone du spike correspondant. La Figure 1 illustre la nouvelle méthode utilisée pour calculer $\mathbf{G}$, $\mathbf{b}$, $\mu_{\mathbf{2}}$ et $\mu_{\mathbf{A}}$, où l'on passe en revue chacun des spikes et l'on affecte sa contribution à l'indice approprié de la quantité à calculer.

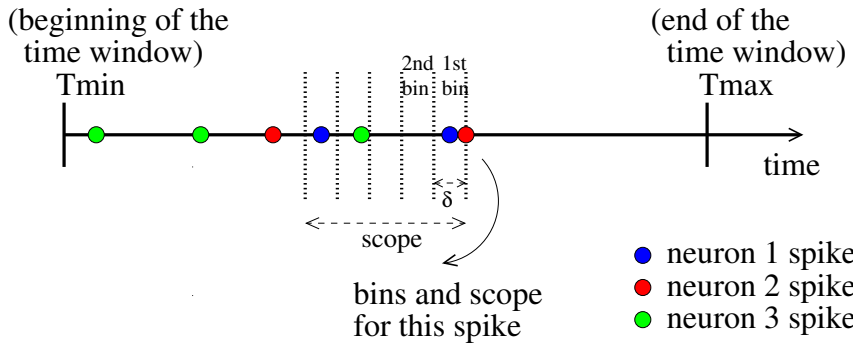

Figure 1: Exemple pour M=3 neurones et K=5 bins

Par exemple, on obtient pour les différents termes de $\mathbf{G}$ définie dans (2):
Pour tous $l \in [\![1, M]\!]$, $k \in [\![1, K]\!]$, on a

$$\mathbf{G}_{1,\alpha(l,k)} = \sum_{\substack{\theta \in N^l, \\ (T_{\min} - A) \leqslant \theta < T^{\max}}} \left(\min\left(T^{\max}, k\,\delta + \theta\right) - \max\left(T_{\min}, (k-1)\,\delta + \theta\right)\right),$$

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

On obtient pour tous $(k_1, k_2) \in [\![1, K]\!]^2$, $(l_1, l_2) \in [\![1, M]\!]$

$$\mathbf{G}_{\alpha(l_1,k_1),\alpha(l_2,k_2)} = \sum_{\substack{(\theta,\tau)\in(N^{l_1}\times N^{l_2}),\\ (T_{\min}-A\leqslant\theta<\tau<T^{\max})}} (\min(\theta+k_1\delta,\, \tau+k_2\delta,\, T^{\max}) - \max(\theta+(k_1-1)\delta,\, \tau+(k_2-1)\delta,\, T_{\min}))$$

Le calcul de $\mathbf{G}$ est décrit en Figure 2 (avec une numérotation des indices commençant à 0).

Algorithm 1 Computation of matrix G

```
function COMPUTE_G(DataSpike, T_min, T_max)
    i_begin = INDEX(T_min − Kδ, DataSpike)
    i_end = INDEX(T_max, DataSpike)-1 ⟶ [-1 for getting the largest spike strictly smaller than T_max]
    g[0, 0] = T_max − T_min
    for θ ∈i_begin:i_end do
        for k ∈ 1 : K do
            dx = min(T_max,T[θ]+kδ)-max(T_min, T[θ]+(k-1)δ)
            if dx > 0 then
                g[0, neur[θ]K + k] += dx
                g[neur[θ]K + k, 0] += dx
        for k_1 ∈ 1 : K do
            x_1=min(T_max, T[θ]+k_1δ)
            dx = x_1-max(T_min, T[θ]+(k_1-1)δ)
            if dx > 0 then
                g[neur[θ]K + k_1, neur[θ]K + k_1] += dx
            for k_2 ∈ (k_1 + 1) : K do
                dx = x_1-max(T_min, T[θ]+(k_2-1)δ)
                if dx > 0 then
                    g[neur[θ]K + k_1, neur[θ]K + k_2] += dx
                    g[neur[θ]K + k_2, neur[θ]K + k_1] += dx
        for τ ∈ (θ + 1):end  and T[τ]<T[θ]+Kδ do
            for k_1 ∈ 1 : K do
                for k_2 ∈ 1 : k_1 do
                    dx = min(T_max, T[θ]+k_1δ, T[τ]+k_2δ) - max(T_min, T[θ]+(k_1 − 1)δ, T[τ]+(k_2 − 1)δ)
                    if dx > 0 then
                        g[neur[θ]K + k_1, neur[τ]K + k_2] += dx
                        g[neur[τ]K + k_2, neur[θ]K + k_1] += dx
```

Figure 2: Nouvel algorithme pour G

On calcule les autres quantités définies dans (2) et (3) en suivant le même procédé.

Les Figures 3 et 4 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 $Ntot{=}M\,\nu(T^{\max} - T_{\min})$ 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 [6]). La nouvelle méthode est plus efficace.

Pour le calcul des estimateurs définis dans (4), on utilise désormais la méthode d'Active Set semblable à celle expliquée dans [1] afin de réduire les temps calculs. Pour chaque problème Lasso intermédiaire, on utilise un solveur LARS (cf [2]).

Sur la Figure 5, le test considéré ici est un processus de Hawkes utilisant le code SPIKES [6] sur (0,100] avec un taux de décharge spontanée de 10 Hz pour chaque neurone et un nombre moyen de connexions pour chaque neurone égal à 25. Pour la reconstruction, on choisit $\delta = 0.02\,s$. Après comparaison à *neuro-stat*, les valeurs numériques obtenues sont identiques entre les deux méthodes (à la précision numérique près).

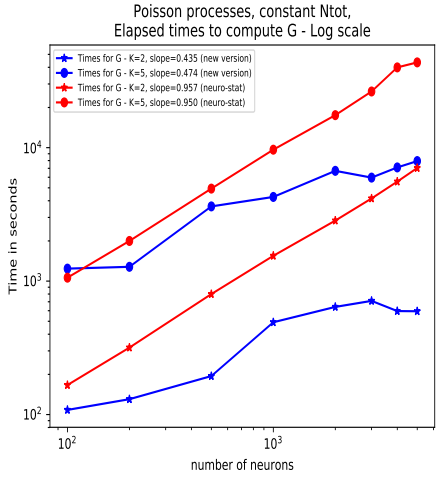

Figure 3: Comparaison des temps calcul de $\mathbf{G}$ - $K = 2$ et 5 - Test 1

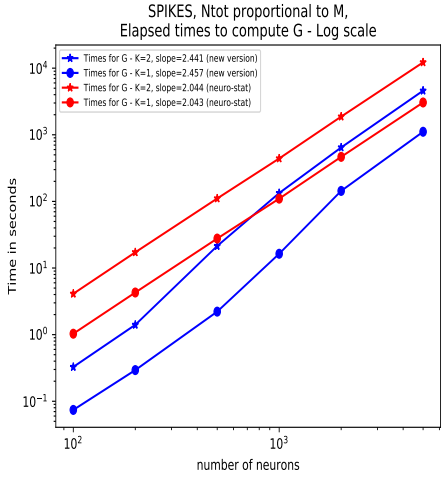

Figure 4: Comparaison des temps calcul de $\mathbf{G}$ - $K = 1$ et 2 - Test 2

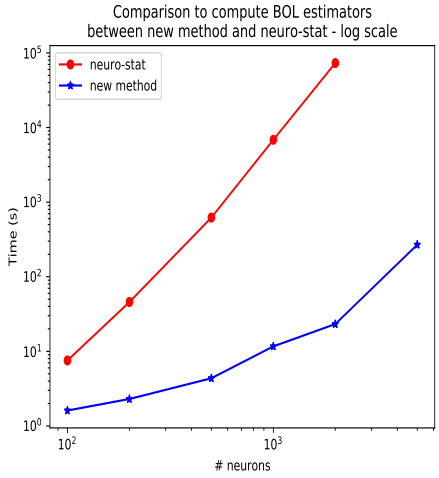

Figure 5: Comparaison des temps calcul des estimateurs - Test 3

En utilisant des trains de spikes simulés à l'aide du code SPIKES décrit dans [6], une série de tests a été menée sur des réseaux de grande taille, entre 5 000 et 20 000 neurones. Le Tableau 1 présente les caractéristiques de chaque réseau.

M	nombre de spikes	Tmin (s)	Tmax (s)	taux de décharge (Hz)	coefficients d'interaction
5 000	20 410 540	0	100	20	25
10 000	85 572 676	0	100	20	25
10 000	107 032 361	0	100	75% à 20 Hz, 12.5% à 10 Hz, 12.5% à 40 Hz	25
10 000	128 228 049	0	100	50% à 10 Hz, 50% à 50 Hz	25
15 000	121 800 711	0	100	20	25
20 000	163 450 693	0	100	20	25

Table 1: Description des réseaux

Le Tableau 2 montre les temps calcul obtenus pour les trains de spikes du Tableau 1.

test	M	temps calcul pour **G** (s)	temps pour b (s)	temps pour muA (s)	temps pour mu_2 (s)	temps pour estimateur BOL (s)	mémoire max (Go)
1	5 000	5 405.72	2 190.49	0.34	1 736.79	691.82	3.06
2	10 000	34 395.4	14 432	1.37	23 605.3	2 181.3	11
3	10 000	53 100.8	21 491.7	1.71	42 457.8	2 083.58	11
4	10 000	69 998.9	30 334.2	2.05	64 523.9	2 254.43	11
5	15 000	74 096.1	30 063.3	1.95	48 759.5	6 087.92	26
6	20 000	134 760	56 184.6	2.64	90 188.4	11 492	47

Table 2: Temps calcul pour les exemples du Tableau 1

3 Conclusion et perspectives

On a expliqué comment améliorer le temps de calcul de la matrice $\mathbf{G}$ par rapport à la méthode précédente et on a montré des résultats de comparaison. On procède de manière semblable pour améliorer les temps calcul pour $\mathbf{b}$, $\mu_{\mathbf{2}}$ et $\mu_{\mathbf{A}}$. Les temps calcul des estimateurs du problème Lasso ont aussi été réduits en utilisant la méthode d'Active Set.

Des premiers tests pour des nombres de neurones élevés ont été menés. L'article [4] est une référence récente présentant des résultats de reconstruction de la connectivité pour des réseaux allant jusqu'à 10 000 neurones, à l'aide des corrélations croisées.

Remerciements

Ces travaux ont bénéficié d'un accès aux moyens de calcul de l'IDRIS au travers de l'allocation de ressources 2020-A0080311481 attribuée par GENCI.

Bibliographie

[1] Dragoni, L., Flamary, R., Lounici, K., and Reynaud-Bouret, P. (2019) *Large scale Lasso with windowed active set for convolutional spike sorting*, arXiv:1906.12077.

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

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

[4] Kobayashi, R., Kurita, S., Kurth, A., Kitano, K., Mizuseki, K., Diesmann, M., Richmond, B. J. and Shinomoto, S. (2019) *Reconstructing neuronal circuitry from parallel spike trains*, Nature Communications, 10 (1).

[5] Lambert, R., Tuleau-Malot, C., Bessaih, T., Rivoirard, V., Bouret, Y., Leresche, N. and Reynaud-Bouret, P., (2018), *Reconstructing the functional connectivity of multiple spike trains using Hawkes models*, Journal of Neuroscience Methods, 297, 9–21.

[6] Mascart, C., Muzy, A. and Reynaud-Bouret, P. (2020), *Efficient Simulation of Sparse Graphs of Point Processes*, submitted to ACM.