Fast Constrained Non-negative Matrix Factorization for Whole-Brain Calcium Imaging Data

Johannes Friedrich[1,2], **Daniel Soudry**[1], **Yu Mu**[2], **Jeremy Freeman**[2], **Misha Ahrens**[2], **Liam Paninski**[1]
[1]Department of Statistics, Columbia University, New York, NY 10027
[2]Janelia Research Campus, Ashburn, VA 20147
{j.friedrich, ds3293}@columbia.edu
{muy, freemanj11, ahrensm}@janelia.hhmi.org, liam@stat.columbia.edu

Abstract

Advances in optical recording technologies allow whole-brain recordings with single cell resolution of small animals, such as larval zebrafish. A crucial step for further neural analysis is the move from voxel-space to neural-space, i.e. the detection of all neurons and extraction/demixing of how each neurons activity evolves in time. The sheer amount of data has led experimenters towards simple but fast methods such as averaging voxels in a small neighborhood around identified neuron centers. More statistically-principled methods based on non-negative matrix factorization (NMF) better conserve information, yielding higher signal-to-noise ratio and more recovered neurons, and allow the demixing of overlapping neurons. However, previous implementations of these methods have been too slow for the large data sizes obtained with calcium imaging and have not yet been employed routinely in practice. Here we exploit advances in NMF paired with downsampling schemes for the specific data at hand to obtain a fast NMF method. Given access to a small cluster, processing of the data takes the same time as the experiment. Though we focus on batch processing performed after the experiment, our method opens the way to online real-time processing.

1 Introduction

Calcium imaging has become a standard tool for monitoring large populations of neurons. Recent advances in imaging technologies have enabled whole-brain imaging of small organisms with single cell resolution [1]. In a typical experiment, lasting up to one hour, about 1TB of imaging data is collected. The first step in the neural data analysis pipeline is to identify the location of each neuron, the second to infer the time course of its fluorescence activity. Nuclear localized calcium indicators facilitate simple solutions to the spatial location identification problem. (For better temporal resolution a cytosolic calcium indicator emitting at different wavelength could be co-expressed.) A common method to extract the calcium concentration over time of the detected neurons is to average the voxels in a small neighborhood around the identified neural centers. Though particularly fast, the method suffers from contaminating contributions from other nearby neurons within the region over which averaging takes place, or high noise levels if this region is kept small, which can compromise further neural analysis. Another approach relies on matrix factorization that expresses the spatio-temporal fluorescence activity as the product of a spatial matrix that encodes the spatial footprint of each neuron and a temporal matrix that characterizes the calcium concentration of each neuron over time. In the case of significant spatial overlap, even the more sophisticated, but inherently linear, Independent Component Analysis (ICA) [5] fails, because there is no linear demixing matrix that can lead to independent outputs [7]. In contrast, non-negative matrix factorization (NMF) enables source separation for the case of multiple neurons contributing to the activity of one voxel. Further, the obtained spatial footprints allow to judge whether a suspected neuron is indeed one or a false alarm. In spite of these benefits, NMF has not yet been employed routinely in practice, because it is too slow for the large data sizes obtained with calcium imaging. We address this problem here and make NMF times comparable with simple methods.

2 Results

Motivated by the benefits of NMF over ICA [4], we applied the implementation of scikit-learn [6] to some representative patches of size 100×100 pixels, that have been extracted from a medial z-layer of a whole-brain light-sheet imaging recording of 3000 frames using nuclear localized GCaMP6f in zebrafish. However, we consistently found that the extracted components didn't correspond to individual neurons, but were distributed over each patch.

To ensure that the the spatial components are localized, we constrained them to lie within spatial patches (which are not large compared to the size of the cell body; Fig. 1) around the neuron centers, thus imposing sparsity by construction. The neural centers were detected by spatial deconvolution and detection of local maxima (to be published elsewhere [7]). This leads to the optimization problem:

$$\min_{\substack{\boldsymbol{A}\in\mathbb{R}_+^{n\times T},\boldsymbol{f}\in\mathbb{R}_+^{T},\boldsymbol{b}\in\mathbb{R}_+^{d},\\ \boldsymbol{S}\in\mathbb{R}_+^{n\times d}\mid s_{ik}=0\ \forall k\notin \boldsymbol{p}_i}} \|\boldsymbol{D} - \boldsymbol{A}^T\boldsymbol{S} - \boldsymbol{f}\boldsymbol{b}^T\|_F^2 \tag{1}$$

with fluorescence data $\boldsymbol{D} \in \mathbb{R}_+^{T\times d}$, neural activities $\boldsymbol{A} \in \mathbb{R}_+^{n\times T}$, neural shapes $\boldsymbol{S} \in \mathbb{R}_+^{n\times d}$, global background intensity $\boldsymbol{f} \in \mathbb{R}_+^{T}$, background spatial structure $\boldsymbol{b} \in \mathbb{R}_+^{d}$ and patches $\boldsymbol{P} \in \{0,1\}^{n\times d}$. For convenience, instead of singling out the background, in the following we add its intensity as row to $\boldsymbol{A}$ and its shape to $\boldsymbol{S}$. The objective in Eq. 1 is bi-convex and can be solved by first obtaining the residual based on some initial guess, followed by iterations over neurons consisting of adding a neuron, performing alternating non-negative least squares (NNLS), subtracting it again and moving to the next neuron. Instead of solving the NNLS problems for each neuron exactly, it was faster to perform only one block-coordinate decent step each time and move on to the next neuron. We found that initializing $\boldsymbol{S}$ by placing Gaussians at the neural centers and taking the 20% percentile of the data over time for the background worked well in practice. We further initialized $\boldsymbol{A}$ as fluorescence at the center pixels and vector of ones for neurons and background respectively. This procedure converged to a good solution as indicated by a low mean-squared error (MSE) and also by visual inspection of the residual, however it required to be run for about one minute on a laptop with 2.7 GHz Core i5 processor to approximately achieve this value (Fig. 2A, black line).

We found that great speedups can be obtained by changing the order in which the variables in this block-coordinate decent scheme are updated. Instead of updating activity and shape of one neuron at a time, it turned out beneficial to update the activities of all neurons while keeping their shapes fixed, and then updating all shapes while keeping their activities fixed (Fig. 2A, vermilion line). The ensuing method consists of hierarchical alternating least squares (HALS)[2] iterations that operate on smaller matrices obtained as dot-products of data and activities or shapes respectively and avoids

[†]We use the convention that all vectors are assumed to be column vectors, thus the ith row of $\boldsymbol{P}$ is $\boldsymbol{p}_i^T$.

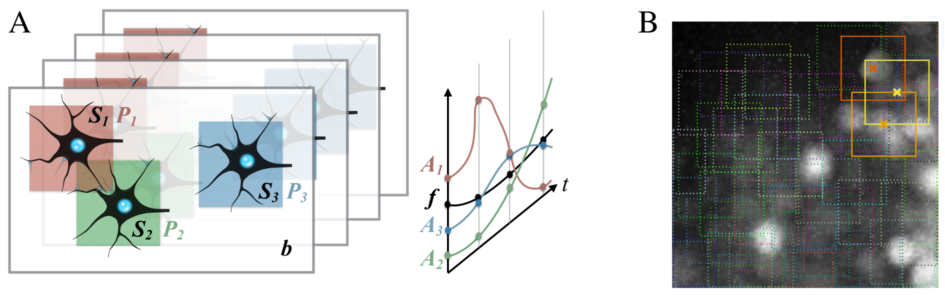

Figure 1: Spatially constrained NMF. **(A)** The spatial footprint of neuron i lies within patch $\boldsymbol{p}_i$[†], thus $s_{ik} = 0\ \forall k \notin \boldsymbol{p}_i$. Its contribution to the data is $\boldsymbol{a}_i\boldsymbol{s}_i^T$ and the background is modeled as $\boldsymbol{f}\boldsymbol{b}^T$. **(B)** Real data summary image obtained as max-projection along time-axis. Rectangles indicated patches centered at suspected neurons. The three highlighted neurons are considered in Figure 3.

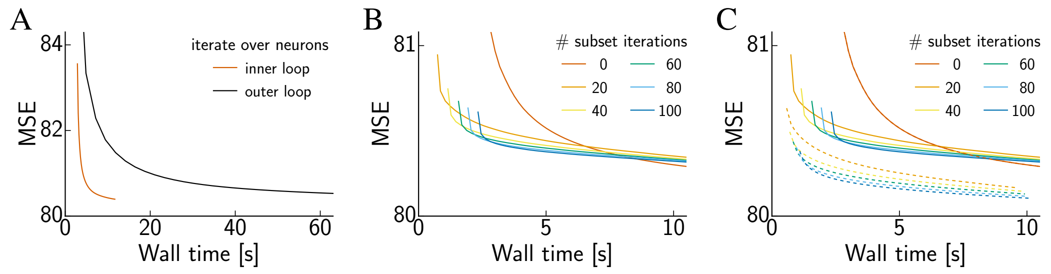

Figure 2: Towards faster NMF. Mean-squared-error as a function of wall time. **(A)** The order in which coordinates in the block-coordinate decent method are updated affects convergence speed. **(B)** Temporal downsampling speeds up NMF. Curves show the MSE on the whole data after a number of initial iterations (color coded, c.f. legend) are performed on a smaller dataset obtained by successively taking the mean of 30 images. The curve for zero subset iterations is identical to the one in (A). **(C)** Spatial downsampling yields further improvements, in particular a better (local) minimum of the MSE. Dashed curves show the results for additional spatial downsampling by taking the mean of 2x2 pixels. Solid curves are replotted from (B) to ease comparison. There is only one curve for the degenerate case of zero subset iterations where both methods are identical.

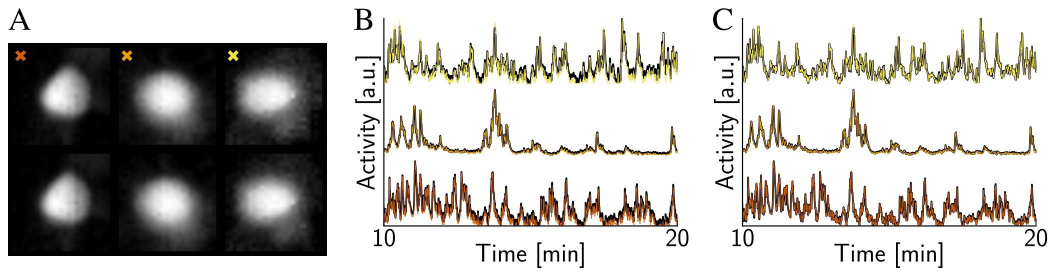

Figure 3: NMF with and without downsampling. **(A)** Extracted shapes with (top row, algorithm run for merely 1 s) and without (bottom row, algorithm run until convergence) are nearly identical. The three neurons have been highlighted in Fig. 1 using the same color code. **(B)** Extracted time traces without (thick black) and with downsampling (color as in A) overlap well after merely 1 s. **(C)** They overlap almost perfectly if the algorithm using downsampling is run not only for 1 but 10 s.

keeping track of the residual. The two functions for updating the activities and shapes respectively (each taking 90 ms), that together constitute one complete HALS iteration, are shown at the bottom lines of Algorithm 1. We computed the residual only for sake of performance comparison, but excluded the substantial time spent on its computation from the reported wall time values.

To gain further benefits in speed, we reasoned that a lower number of frames can be enough to infer the shapes, which don't change during the experiment. Once the shapes have been found the activities can be quickly obtained by solving the remaining convex NNLS problem. We subsampled the data by taking only every kth frame and inferred the shapes. Keeping the obtained shapes fixed, we solved the convex problem for the activities over all frames using some iterations of HALS for the activities (line 11 in Alg. 1). Finally, we performed some HALS iterations on the whole data. However, plain subsampling, which is close to a stochastic update rule, did hardly improve over using the whole data right from the beginning (not shown). Next, instead of taking every kth frame, we partitioned the data into chunks of k frames and aggregated the activity over time to produce a summary statistic for each chunk, thus preserving more information compared to subsampling. We found mean or median to work well, however, we chose mean because it is faster to evaluate (80 vs 412 ms). Figure 2B shows the results obtained for a varying number of iterations on the small aggregated data.

The promising results gained from temporal downsampling motivated us to study the effects of spatial downsampling as well. We further reduced the downsampled data by a factor of four via averaging over 2x2 pixels, which yielded our final Algorithm 1. Figure 2C compares the obtained performance curves to the version without spatial downsampling. Strikingly, by spatial downsampling the final iterations on the whole data are initialized in a way that enables the finding of a better local minimum of the objective. Even without further HALS iterations on the whole data a good solution with MSE of 80.403 is obtained in less than one second (959 ms) for 80 iterations on the downsampled data. Running the algorithm for 5 and 10 s using 100 iterations on the downsampled data we obtained a MSE of 80.162 and 80.107 respectively. For sake of comparison we also determined the neural activities using the simple method, that subtracts some percentile as background and merely averages the region around the centers. In order to calculate the corresponding MSE we optimized neural shapes (keeping traces fixes) and background until convergence. Using the 20% percentile as background estimate yielded the lowest MSE of 85.591 within 0.16 s.

We obtained remarkable speedups using downsampling without compromising the obtained solution in terms of MSE, but how does it affect the extracted shapes and traces? We ran the algorithm without downsampling until convergence and with downsampling for 1 and 10 s respectively. Figure 3 shows the results for three neurons with overlapping patches (highlighted in Fig. 1). Both, shapes and series, agree well even if the downsampled algorithm is run for merely 1 s (Fig. 3A,B) and are nearly identical if run longer (Fig. 3C), hence downsampling does not impair their final performance.

Algorithm 1 NMF algorithm using localization constraints and spatio-temporal downsampling

```
Require: data $\boldsymbol{D} \in \mathbb{R}_+^{T \times d}$, $n$ neuron centers, number of iterations $I'$ and $I$
 1: successively take mean of e.g. 30 frames and downscale using local mean: $\boldsymbol{D} \to \boldsymbol{D}' \in \mathbb{R}_+^{T' \times d'}$
 2: generate patches $\boldsymbol{P}'$ around centers
 3: initialize spatial background as 20% percentile, neural shapes as Gaussians within patch: $\boldsymbol{S}' \in \mathbb{R}_+^{(n+1) \times d'}$
 4: initialize temporal background as $\mathbf{1}_{T'}$, neural activities as data at centers: $\boldsymbol{A}' \in \mathbb{R}_+^{(n+1) \times T'}$
 5: for $i = 1, \dots, I'$ do
 6:     $\boldsymbol{A}' \leftarrow$ HALSACTIVITY($\boldsymbol{D}', \boldsymbol{A}', \boldsymbol{S}'$)
 7:     $\boldsymbol{S}' \leftarrow$ HALSSHAPE($\boldsymbol{D}', \boldsymbol{A}', \boldsymbol{S}', \boldsymbol{P}'$)
 8: initialize activities $\boldsymbol{A} \in \mathbb{R}_+^{(n+1) \times T}$ as mean of $\boldsymbol{A}'$ for each neuron
 9: initialize shapes $\boldsymbol{S} \in \mathbb{R}_+^{(n+1) \times d}$ by zero-order-hold upsampling of $\boldsymbol{S}'$
10: generate patches $\boldsymbol{P}$
11: $\boldsymbol{A} \leftarrow$ HALSACTIVITY($\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{S}, 5$)
12: $\boldsymbol{S} \leftarrow$ HALSSHAPE($\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{P}, 5$)
13: for $i = 1, \dots, I$ do
14:     $\boldsymbol{A} \leftarrow$ HALSACTIVITY($\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{S}$)
15:     $\boldsymbol{S} \leftarrow$ HALSSHAPE($\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{P}$)
16: return $\boldsymbol{S}, \boldsymbol{A}$
```

```
1: function HALSACTIVITY($\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{S}, I = 1$)
2:     $\boldsymbol{U} = \boldsymbol{S}\boldsymbol{D}^T$
3:     $\boldsymbol{V} = \boldsymbol{S}\boldsymbol{S}^T$
4:     for $i = 1, \dots, I$ do
5:         for $k = 1, \dots, n+1$ do
6:             $\boldsymbol{a}_k \leftarrow \max(0, \boldsymbol{a}_k + \frac{\boldsymbol{u}_k - \boldsymbol{A}^T \boldsymbol{v}_k}{v_{kk}})$
7:     return $\boldsymbol{A}$
```

```
1: function HALSSHAPE($\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{S}, \boldsymbol{P}, I = 1$)
2:     $\boldsymbol{U} = \boldsymbol{A}\boldsymbol{D}$
3:     $\boldsymbol{V} = \boldsymbol{A}\boldsymbol{A}^T$
4:     for $i = 1, \dots, I$ do
5:         for $k = 1, \dots, n+1$ do
6:             $\boldsymbol{s}_k \leftarrow \boldsymbol{p}_k \odot \max(0, \boldsymbol{s}_k + \frac{\boldsymbol{u}_k - \boldsymbol{v}_k^T \boldsymbol{S}}{v_{kk}})$
7:     return $\boldsymbol{S}$
```

3 Discussion

We have presented a fast NMF method for the extraction of neural time series and spatial footprints. While we presented results for one patch, all analysis were performed on patches from different brain areas in two zebrafish and found to hold generally. Our method outperforms simple methods based on local averaging, which are further only applicable to recordings that use nuclear localized indicators, whereas our NMF method extends to cytosolic indicators as well. Good results are obtained after merely one second per patch on a dual-core laptop, however whole-brain imaging of zebrafish requires processing of $\sim$15000 patches, which would take ten times longer than the 1500 s recording. Our method can be deployed to a cluster to process patches in parallel. On a small cluster of 5 nodes with 16 cores each the analysis would just take one quarter of the recording time, not including the time spent for loading the data and initialization. To facilitate the use of a cluster we are currently adding our method to the Thunder library for large-scale neural data [3]. Real-time processing seems feasible and future work involves online formulations of the algorithm as well as fast (online) initialization methods.

References

[1] M. B. Ahrens, M. B. Orger, D. N. Robson, J. M. Li, and P. J. Keller. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nature methods*, 10(5):413–420, 2013.

[2] A. Cichocki, R. Zdunek, and S.-i. Amari. Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. In *Independent Component Analysis and Signal Separation*, pages 169–176. Springer, 2007.

[3] J. Freeman, N. Vladimirov, T. Kawashima, Y. Mu, N. J. Sofroniew, D. V. Bennett, J. Rosen, C.-T. Yang, L. L. Looger, and M. B. Ahrens. Mapping brain activity at scale with cluster computing. *Nature methods*, 11(9):941–950, 2014.

[4] R. Maruyama, K. Maeda, H. Moroda, I. Kato, M. Inoue, H. Miyakawa, and T. Aonishi. Detecting cells using non-negative matrix factorization on calcium imaging data. *Neural Networks*, 55:11–19, 2014.

[5] E. A. Mukamel, A. Nimmerjahn, and M. J. Schnitzer. Automated analysis of cellular signals from large-scale calcium imaging data. *Neuron*, 63(6):747–760, 2009.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] E. A. Pnevmatikakis, D. Soudry, Y. Gao, T. Machado, D. Pfau, T. Reardon, Y. Mu, C. Lacefield, K. E. Poskanzer, M. Ahrens, D. S. Peterka, R. Bruno, T. Jessell, R. Yuste, and L. Paninski. Simultaneous denoising, deconvolution, and demixing of calcium imaging data. under review.