Fast spike train inference from calcium imaging

Joshua T. Vogelstein, other fuckers, Liam Paninski

February 13, 2009

Abstract

1 Introduction

2 Methods

2.1 Model

We assume a simple discrete-time state-space model relating spikes, n_t, baseline subtracted intracellular calcium concentration, denoted by C_t, and fluorescence measurements, F_t. First, we assume a linear relationship between F_t and C_t, with gaussian noise. Second, we assume that C_t jumps after each spike, and then decays according to some time constant. Finally, we assume that spikes are distributed according to a Poisson process. The above assumptions lead to the follwoing model:

$$F_t = \alpha C_t + \beta + \varepsilon_t, \qquad \varepsilon_t \sim \mathcal{N}(0, \sigma) \tag{1}$$

$$C_t = \gamma C_{t-1} + n_t, \qquad n_t \sim \text{Poisson}(n_t; \lambda\Delta) \tag{2}$$

where α and β set the fluorescence baseline and offset respectively, σ indicates the variance of the noise, and γ sets the decay rate.

2.2 Inference

Given such a model, our goal is to find the maximum *a posteriori* (MAP) spike train, i.e., the most likely spike train, $\boldsymbol{n}$ (where we use the shorthand notation $\boldsymbol{X} = [X_1, \ldots, X_T]$, and T is the final time step), given the fluorescence measurements, $\boldsymbol{F}$. Thus, the objective function that we'd like to solve may be written as:

$$\begin{aligned}
\widehat{\boldsymbol{n}} &= \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmin}} P(\boldsymbol{n}|\boldsymbol{F}) = \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmin}} P(\boldsymbol{F}|\boldsymbol{n}) P(\boldsymbol{n}) \\
&= \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmax}} \prod_{t=1}^{T} P(F_t|C_t) P(n_t) = \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmax}} \sum_{t=1}^{T} \big(\log P(F_t|C_t) + \log P(n_t) \big) \\
&= \underset{n_t \in \mathbb{N}_0 \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2} (F_t - \alpha C_t - \beta)^2 - n_t \log \lambda\Delta + \log n_t! \right),
\end{aligned} \tag{3}$$

where $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$, and C_t is implicitly a function of n_t. Unfortunately, the computational complexity of solving Eq. (3) scales exponentially with T, i.e., is $O(\mathrm{e}^T)$, making Eq. (3) an NP-hard problem. Thus, instead of an exact solution, we propose to make an approximation that reduces the complexity to be *polynomial* in T. In particular, we relax the assumption that we must have an integer number of spikes at any time step, by approximating the Poisson distribution with an exponential. Note that this is a common approximation technique in the machine learning literature

[Hastie et al., 2001], as it is the closest convex relaxation to its non-convex counterpart. The constraint on n_t in Eq. (3) is therefore relaxed from $n_t \in \mathbb{N}_0$ to $n_t \geq 0$:

$$\widehat{\boldsymbol{n}} \approx \underset{n_t \geq 0 \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - \alpha C_t - \beta)^2 + n_t \lambda \Delta \right). \tag{4}$$

While this convex relaxation makes the problem tractable, the "sharp" threshold imposed by the nonnegativity constraint prohibits the use of standard gradient ascent techniques [Boyd and Vandenberghe, 2004]. We therefore take an "interior-point" (or "barrier") approach, in which we drop the sharp threshold, and add a barrier term, which must approach $-\infty$ as n_t approaches zero (e.g., $-\log n_t$) [Boyd and Vandenberghe, 2004]. By iteratively reducing the weight of the barrier term, $z > 0$, we are guaranteed to converge to the correct solution [Boyd and Vandenberghe, 2004], $\widehat{\boldsymbol{n}}$. Thus, our goal is to efficiently solve:

$$\widehat{\boldsymbol{n}}_z = \underset{n_t \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - \alpha C_t - \beta)^2 + n_t \lambda \Delta - z \log(n_t) \right), \tag{5}$$

which is concave and may therefore be solved exactly and efficiently using a gradient ascent technique. To see how, we first note that spikes and calcium are related to one another via a simple linear transformation, namely, $n_t = f(C_t, C_{t-1}) = C_t - \gamma C_{t-1}$, or, in matrix notation:

$$\boldsymbol{M}\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots \\ 1 & -\gamma & 0 & \cdots & \cdots \\ 0 & 1 & -\gamma & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 1 & -\gamma \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ \vdots \\ C_T \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ \vdots \\ n_T \end{bmatrix} = \boldsymbol{n} \tag{6}$$

where $\boldsymbol{M} \in \mathbb{R}^{T \times T}$ is a bidiagonal matrix, and $\boldsymbol{b}$, $\boldsymbol{C} = [C_1, \ldots, C_T]$, and $\boldsymbol{n}$ are T dimensional column vectors. Given Eq. (6), we may rewrite Eq. (5) in terms of $\boldsymbol{C}$:

$$\begin{aligned} \widehat{\boldsymbol{C}}_z &= \underset{C_t - \gamma C_{t-1} \geq 0 \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - \alpha C_t - \beta)^2 + (C_t - \gamma C_{t-1})\lambda\Delta - z \log(C_t - \gamma C_{-1} - \nu) \right) \\ &= \underset{\boldsymbol{M}\boldsymbol{C} \geq \boldsymbol{0}}{\operatorname{argmin}} \frac{1}{2\sigma^2} \left\| \boldsymbol{F} - \alpha \boldsymbol{C} - \beta \boldsymbol{1} \right\|^2 + \lambda\Delta(\boldsymbol{M}\boldsymbol{C})'\boldsymbol{1} - z \log(\boldsymbol{M}\boldsymbol{C})'\boldsymbol{1}, \end{aligned} \tag{7}$$

where $\boldsymbol{M}\boldsymbol{C} \geq \boldsymbol{0}$ indicates that every element of $\boldsymbol{M}\boldsymbol{C}$ is greater than or equal to zero, $'$ indicates transpose, and $\boldsymbol{1}$ is a T dimensional column vector, and $\log(\cdot)$ indicates an element-wise logarithm. As (7) is concave (it is identical to Eq. (5) with different notation), we may use any descent technique to find $\widehat{\boldsymbol{C}}_z$. We elect to use the Newton-Raphson approach:

$$\widehat{\boldsymbol{C}}_z \leftarrow \widehat{\boldsymbol{C}}_z + s\boldsymbol{d} \tag{8a}$$

$$\boldsymbol{H}\boldsymbol{d} = \boldsymbol{g} \tag{8b}$$

$$\boldsymbol{g} = -\frac{\alpha}{\sigma^2}(\boldsymbol{F} - \alpha\widehat{\boldsymbol{C}}_z - \beta) + \lambda\Delta\boldsymbol{M}'\boldsymbol{1} - z\boldsymbol{M}'(\boldsymbol{M}\widehat{\boldsymbol{C}}_z)^{-1} \tag{8c}$$

$$\boldsymbol{H} = \frac{\alpha^2}{\sigma^2}\boldsymbol{I} + z\boldsymbol{M}'(\boldsymbol{M}\widehat{\boldsymbol{C}}_z)^{-2}\boldsymbol{M} \tag{8d}$$

where s is the step size, $\boldsymbol{d}$ is the step direction, and $\boldsymbol{g}$ and $\boldsymbol{H}$ are the gradient (first derivative) and Hessian (second derivative) of the likelihood, respectively (the likelihood is the argument to be minimized in Eq. (7)), and the exponents

indicate element-wise operations. Note that we use "backtracking linesearches", meaning that for each iteration, we find the maximal s that is between 0 and 1 and decreases the likelihood.

Typically, implementing Newton-Raphson requires inverting the Hessian, i.e., $\boldsymbol{d} = \boldsymbol{H}^{-1}\boldsymbol{g}$, a computation consuming $O(T^3)$ time. Instead, because $\boldsymbol{M}$ is bidiagonal, the Hessian is tridiagonal, so the solution may be found in $O(T)$ time via standard banded Gaussian elimination techniques (which can be implemented efficiently in Matlab using $\boldsymbol{H}\backslash\boldsymbol{g}$). The resulting fast algorithm for solving the optimization problem in (4) is the main result of this paper, i.e. we may approximately infer the optimal solution for models characterized by equations such as (1) and (2) in linear time, whereas an exact solution would require exponential time, and a naïve approximate solution would require polynomial time.

2.3 Learning

In the above, we assumed that the parameters governing our model, $\boldsymbol{\theta} = \{\alpha, \beta, \sigma, \gamma, \nu, \rho.\lambda\} \in \boldsymbol{\Theta}$, were known. In general, however, these parameters may be estimated from the data. To find the maximum likelihood estimator for the parameters, $\widehat{\boldsymbol{\theta}}$, we must integrate over the unknown variable, $\boldsymbol{n}$. However, integrating over all possible spike trains is typically approximated by Monte Carlo approaches, which is relatively slow. Thus, one often approximates:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} \iint P(\boldsymbol{F}|\boldsymbol{C}, \boldsymbol{n}, \boldsymbol{\theta}) P(\boldsymbol{C}, \boldsymbol{n}|\boldsymbol{\theta}) d\boldsymbol{n} d\boldsymbol{C} \approx \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} P(\boldsymbol{F}|\widehat{\boldsymbol{C}}, \widehat{\boldsymbol{n}}, \boldsymbol{\theta}) P(\widehat{\boldsymbol{C}}, \widehat{\boldsymbol{n}}|\boldsymbol{\theta}) \tag{9}$$

where $\widehat{\boldsymbol{n}} = [\widehat{n}_1, \ldots, \widehat{n}_T]$ is estimate of $\widehat{\boldsymbol{n}}$ from the inference algorithm described above ($\widehat{\boldsymbol{C}}$ is defined similarly). The approximation in (14) is good whenever the likelihood is very peaky, meaning that most of the mass is around the MAP sequence.[1] In particular, for state-space models, one often approximates the integral in (14) with the Viterbi path, i.e., the MAP path of the hidden states [Rabiner, 1989]. Finding the Viterbi path is often far easier than solving the integral in (14), as in the case here. Due to the state space nature of the above model (Eqs (1) and (2)), the optimization in Eq (9) simplifies significantly. More specifically, we can write the argument from the left-hand-side of Eq (9) as a product of terms that we have defined in our model:

$$P(\boldsymbol{F}|\widehat{\boldsymbol{C}}, \widehat{\boldsymbol{n}}, \boldsymbol{\theta}) P(\widehat{\boldsymbol{C}}, \widehat{\boldsymbol{n}}|\boldsymbol{\theta}) = \prod_{t=1}^{T} P(F_t|\widehat{C}_t, \alpha, \beta, \sigma) P(\widehat{C}_t|\widehat{C}_{t-1}, \widehat{n}_t, \gamma) P(\widehat{n}_t|\lambda) \tag{10}$$

Thus, this likelihood is jointly concave in all the parameters. To solve for $\{\alpha, \beta, \sigma\}$, we solve:

$$\begin{aligned}
\{\widehat{\alpha}, \widehat{\beta}, \widehat{\sigma}\} &= \underset{\alpha,\beta,\sigma>0}{\operatorname{argmax}} \sum_{t=1}^{T} \log P(F_t|\widehat{C}_t, \alpha, \beta, \sigma) \\
&= \underset{\alpha,\beta,\sigma>0}{\operatorname{argmax}} -\frac{T}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^{T}(F_t - \alpha\widehat{C}_t - \beta)^2 \\
&= \underset{\alpha,\beta,\sigma>0}{\operatorname{argmax}} -\frac{T}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left\|\boldsymbol{F} - \widehat{\boldsymbol{X}}\boldsymbol{\eta}\right\|^2
\end{aligned} \tag{11}$$

where $\boldsymbol{\eta} = [\alpha, \beta]$ and $\widehat{\boldsymbol{X}}_t = [\widehat{C}_t, 1]$. Thus, solving for $\{\widehat{\alpha}, \widehat{\beta}\}$, is a constrained quadratic optimization problem, which can be solved efficiently using standard tools, such as Matlab's `quadprog`. The variance may then be solved for analytically:

$$\widehat{\sigma}^2 = \frac{1}{T}\left\|\boldsymbol{F} - \widehat{\boldsymbol{X}}\widehat{\boldsymbol{\eta}}\right\|_2^2 \tag{12}$$

Similarly, taking the log of the prior term, $P(\widehat{\boldsymbol{n}}|\boldsymbol{\theta})$ yields:

[1] The approximation in (14) may be considered a first-order Laplace approximation

$$\log P(\widehat{\boldsymbol{n}}|\boldsymbol{\theta}) = \sum_{t=1}^{T} \log(\lambda\Delta) - \lambda\Delta\widehat{n}_t = T(\lambda\Delta) - \lambda\Delta\widehat{\boldsymbol{n}}'\mathbf{1} \tag{13}$$

Plugging Eqs (11) and (13) back into Eq (9), and noting that log is a monotonic function, we have:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}}{\operatorname{argmax}} -\frac{T}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left\|\boldsymbol{F} - \widehat{\boldsymbol{X}}\boldsymbol{\eta}\right\|_2^2 + T\log(\lambda\Delta) - \lambda\Delta\widehat{\boldsymbol{n}}'\mathbf{1} \tag{14}$$

Estimating the parameters then separates into three log-concave problems. In particular, solving for $\widehat{\boldsymbol{\eta}}$ is simply:

$$\widehat{\boldsymbol{\eta}} = \underset{\boldsymbol{\eta},\, 0<\eta_1<1}{\operatorname{argmin}} \frac{1}{2}\left\|\boldsymbol{F} - \widehat{\boldsymbol{X}}\boldsymbol{\eta}\right\|_2^2 = \underset{\boldsymbol{\eta},\, 0<\eta_1<1}{\operatorname{argmin}} \frac{1}{2}\boldsymbol{\eta}'\boldsymbol{Q}\boldsymbol{\eta} - \boldsymbol{L}'\boldsymbol{\eta} \tag{15}$$

where $\boldsymbol{Q} = \widehat{\boldsymbol{X}}'\widehat{\boldsymbol{X}}$, and $\boldsymbol{L} = \widehat{\boldsymbol{X}}'\boldsymbol{F}$, and the constraint, $0 < \eta_1 < 1$, ensures that the time constant is both non-negative and finite. Eq (15) may be solved using standard quadratic optimization tools, such as Matlab's `quadprog`.

Unfortunately, the recursive nature of our algorithm makes solving for $\widehat{\boldsymbol{\eta}}$ in this manner unidentifiable. In particular, because both $\widehat{\boldsymbol{X}}$ and $\boldsymbol{\eta}$ can scale and shift $\boldsymbol{C}$, alternating between inferring $\widehat{\boldsymbol{X}}$ and $\widehat{\boldsymbol{\eta}}$ does not converge (data not shown). However, this is not particularly problematic, because the fluorescence measurements are in arbitrary units. This unit arbitrariness suggests that $\boldsymbol{F}$ may be scaled and shift without loss of generality. The prior term and non-negativity constraint on $\boldsymbol{n}$, however, imposes small costs associated with scaling and shifting $\boldsymbol{n}$ and $\boldsymbol{C}$. We therefore estimate $\boldsymbol{\eta}$ from the raw fluorescence trace. More specifically, to estimate γ, we manually find a fluorescence sequence that seem to be decaying, estimate the time constant τ, and let $\gamma = 1 - \Delta/\tau$. For ν, we manually find a fluorescence sequence that seems to be near baseline, label it $[F_s, \ldots, F_t]$, and let $\nu = \frac{1}{t-s}\sum_{u=s}^{t} F_u$. Finally, we estimate ρ by letting it be equal to what we manually determine it be, by eye. In practice, we have found that by first scaling and shifting $\boldsymbol{F}$ to be between 0 and 1, subsequent traces from the same or different cells have very similar values for $\boldsymbol{\eta}$, and therefore, these parameters need not be tweaked much. Furthermore, the effective signal-to-noise ratio (SNR) of the inferred spike train does not depend strongly on these parameters, in agreement with previous results [Yaksi and Friedrich, 2006]. The other parameters, σ and λ, can be solved for analytically:

$$\widehat{\sigma}^2 = \frac{1}{T}\left\|\boldsymbol{F} - \widehat{\boldsymbol{X}}\widehat{\boldsymbol{\eta}}\right\|_2^2 \tag{16}$$

$$\widehat{\lambda} = \frac{1}{T\Delta}\widehat{\boldsymbol{n}}'\mathbf{1} \tag{17}$$

3 Results

4 Discussion

5 other stuff

6 Old Results

6.1 Main Result

To evaluate the efficacy of our fast filter, we compare it with the optimal linear (i.e., Wiener) filter [Wiener, 1949]. The top three panels of Fig. 1 depict a typical dataset simulated according to our model, Eqs. (1) and (2). Beneath the simulation, we show both the Wiener filter output (fourth panel) and our fast filter output (bottom panel). For both filters, we provided only the fluorescence observations depicted in the top panel, and $\boldsymbol{\eta}$. From this data, we estimate the remaining parameters, and infer the hidden spike train. Several differences between these two approaches should be apparent. First, the effective signal-to-noise ratio (SNR) of our fast filter improves upon the optimal linear filter.

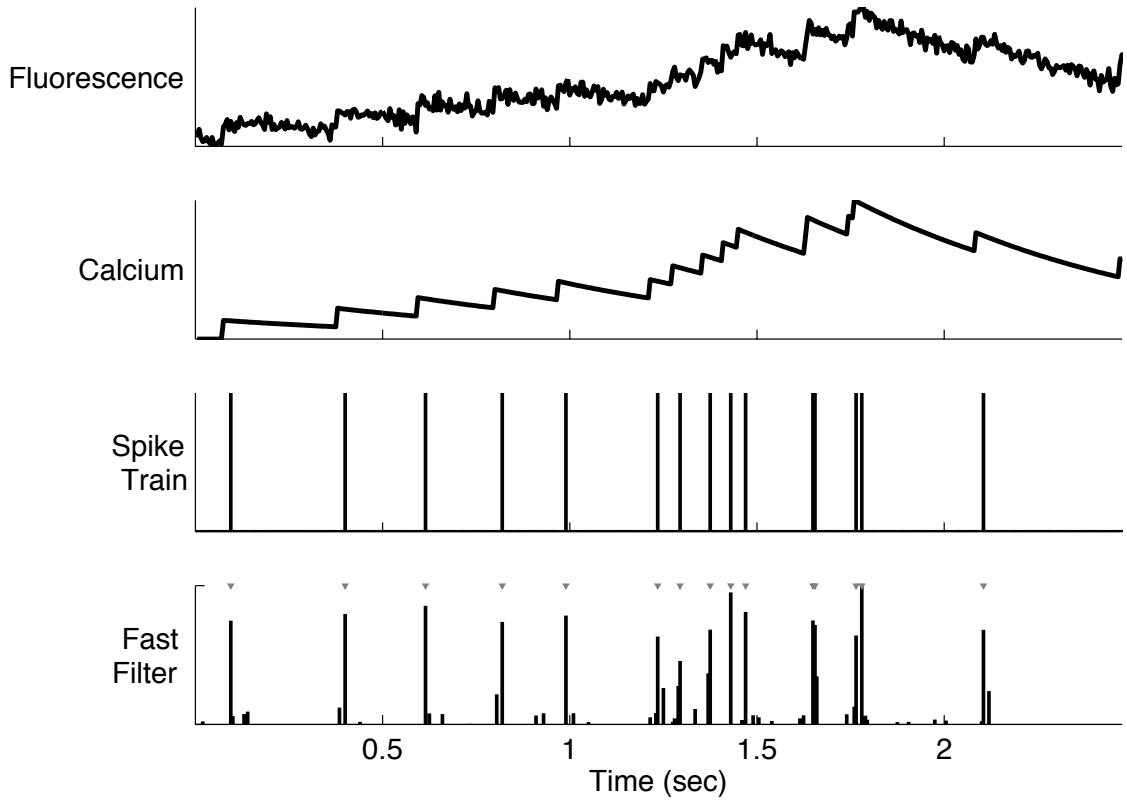

Figure 1: A simulation demonstrating that given a fluorescence time series, and the parameters of our model, we can accurately infer a close approximation to the most likely spike train. Top panel: simulated fluorescence time series. Second panel: simulated intracellular calcium concentration. Third panel: simulated spike train. Fourth panel: the fast filter's inferred spike train (gray triangles indicate true spike times here, and elsewhere, unless indicated otherwise). Parameters: $\Delta = 5$ msec, $\alpha = 1$ a.u., $\beta = 0$ a.u., $\sigma = 0.25$ a.u., $\tau = 700$ msec, $\lambda = 10$ Hz.

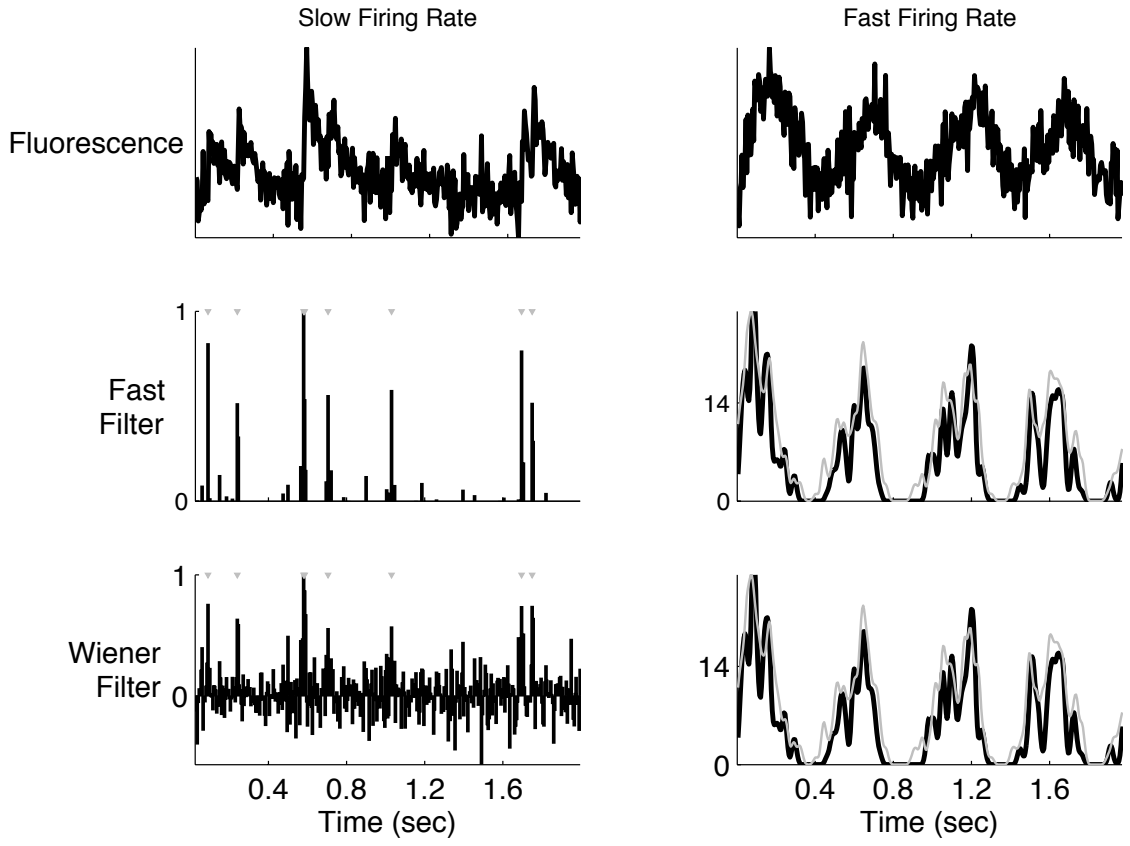

Figure 2: A simulation demonstrating that the fast filter performs at least as well as the optimal linear filter. The left panels show that in the slow firing regime, the fast filter — which approximates the Poisson distribution governing spiking with an exponential distribution — outperforms the Wiener filter — which approximates the Poisson distribution with a Gaussian. The right panels show that both approximations are sufficient in the fast firing regime. Top left panel: fluorescence time series for a neuron with a slow firing rate. Middle left panel: the fast filter's inferred spike train. Bottom left panel: Wiener filter's inferred spike train. Note that (i) the Wiener filter does not impose a non-negativity constraint, and (ii) the effective SNR of this filter in this example is higher than the fast filter's. Top right panel: same as top left panel, for a neuron with a high firing rate. Middle right panel: the fast filter's inferred spike train smoothed with a Gaussian kernel for visualization purposes (black line), and the true spike train smoothed with the same Gaussian kernel (gray line). Bottom right panel: same as middle right panel, but with the Wiener filter. Parameters for left panels: same as above. Parameters for right panels: same as above, except: $\sigma = 8$ a.u., $\lambda = 500$ Hz.

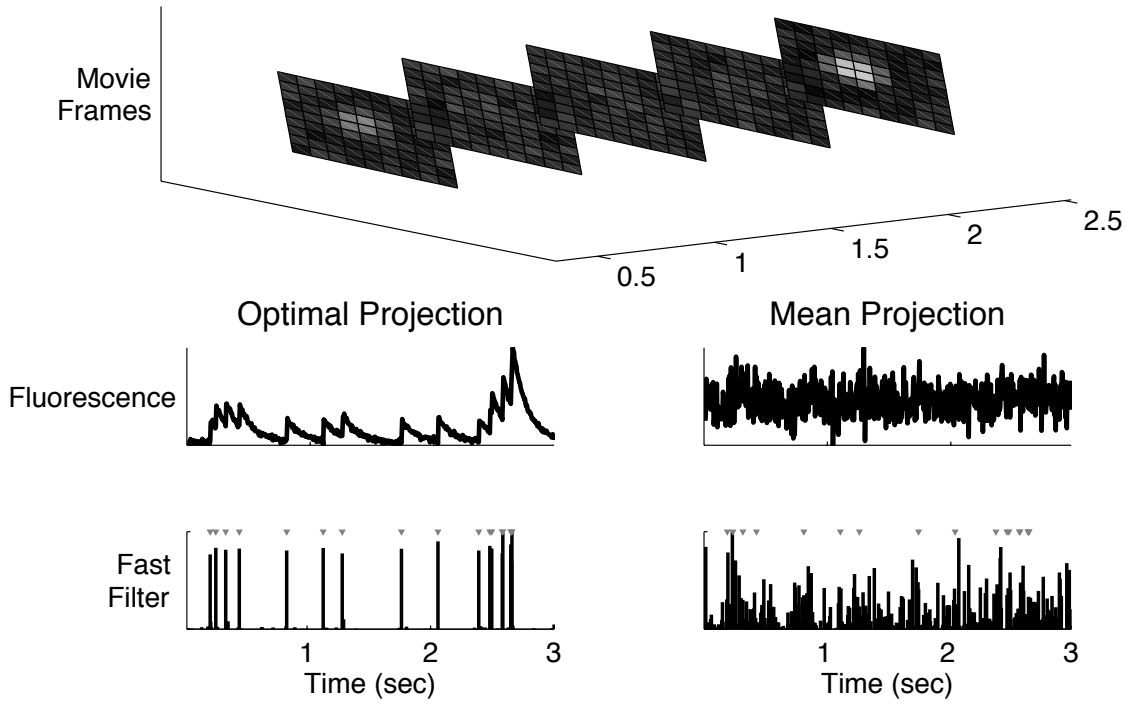

Figure 3: A simulation demonstratig that using a better spatial filter can significantly enhance the effective SNR (see Supplementary Movie 1 for the full movie associated with this simulation). Top: five movie frames, equally spaced from the entire move. Middle left: projection of the entire movie onto the optimal spatial filter, yielding a 1D fluorescence time series, with a small SNR. Bottom left: fast filter's inferred spike train using the optimal spatial filter. Middle right: projection of the entire movie onto the "mean" spatial filter, yielding a 1D fluorescence time series with a large SNR. Bottom right: fast filter's inferred spike train using the mean spatial filter. Parameters different from Fig 1: $\boldsymbol{\alpha}$ is a mixture of two Gaussian kernels, each with the same mean, but different variances and component coefficients (see main text for details). $\boldsymbol{\beta} = 1$.

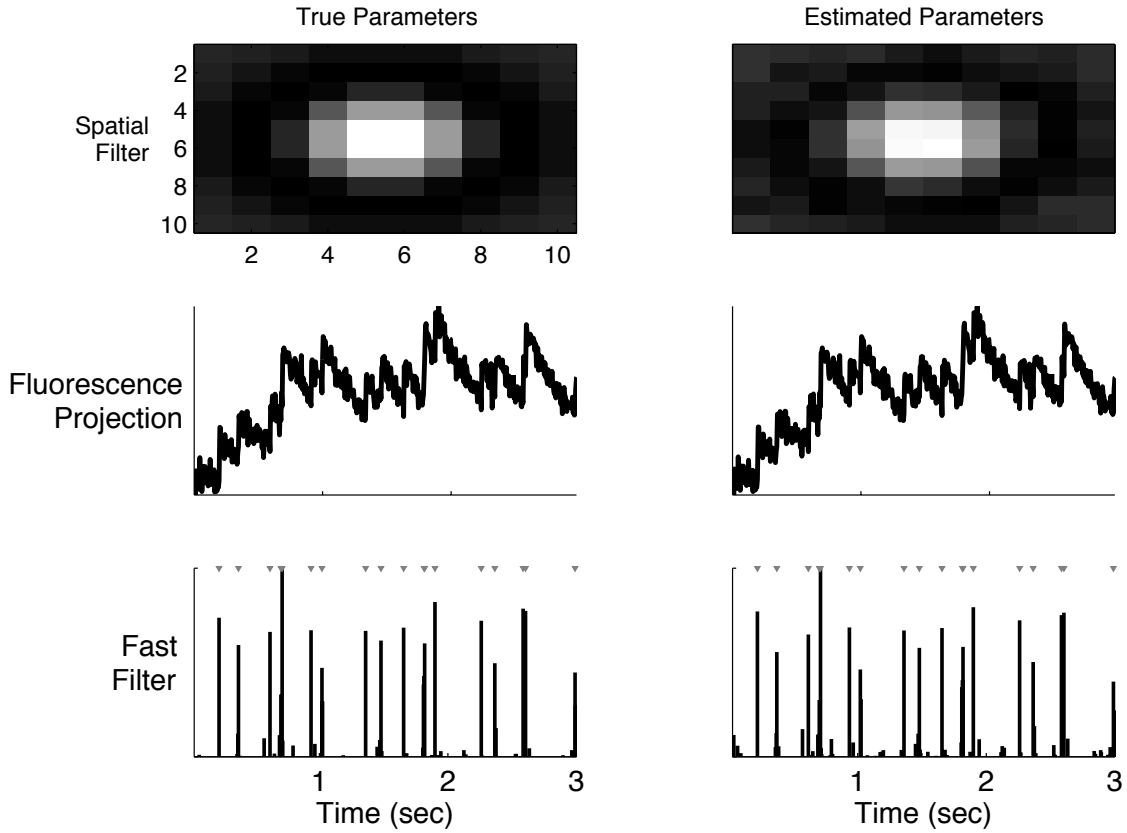

Figure 4: A simulation demonstrating that given only the fluorescence movie, the parameters may be estimated, and the spike train inferred (c.f. Supplementary Movie 2). Top left panel: true spatial filter. Middle left panel: projection of movie onto true spatial filter. Bottom left panel: inferred spike train using true parameters. Right panels: same as left except estimating parameters. $\boldsymbol{\alpha}$ initialized with SVD. $\boldsymbol{\beta}$ initialized with $\mathbf{0}$. λ and σ were initialized at double their true value. τ was assumed known. Parameters same as above.

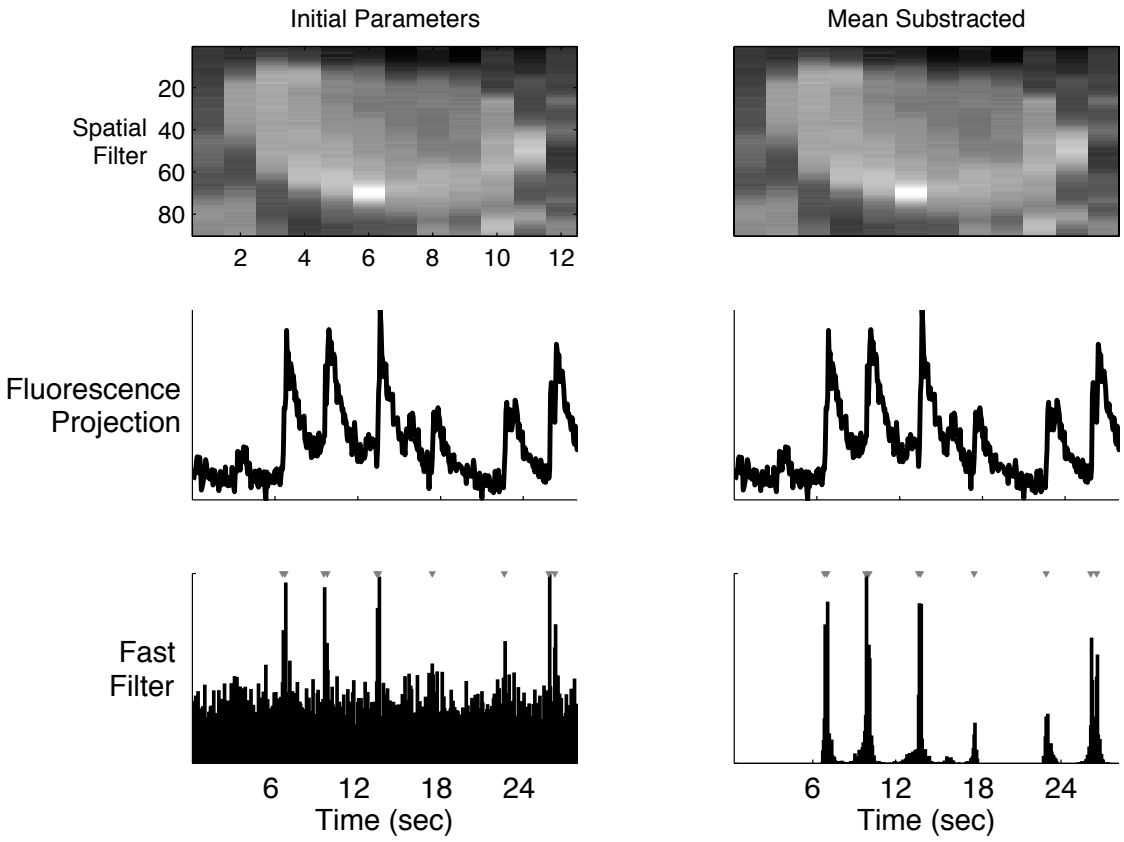

Figure 5: Given only a fluorescence movie, recorded in vivo, we can learn the parameters necessary to correctly infer the spike trains. Top left: mean frame. Middle left: projection of movie onto mean frame. Bottom left: the fast filter's inference using the mean frame as the filter (λ and σ estimated as described by equations (**??**) and (**??**), respectively). Right panels: same as left, but estimating α and β as described in text.

Figure 6: distribution of errors in spike inference from real data

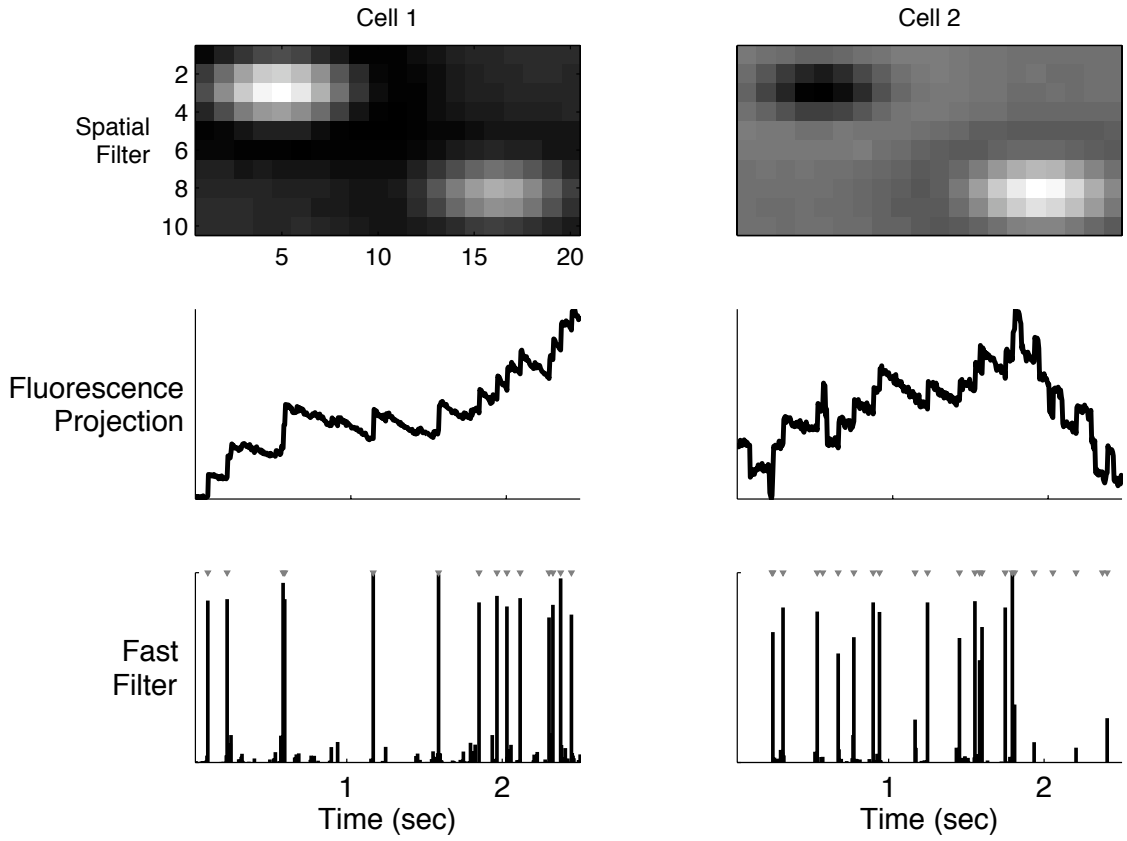

Figure 7: same as spatial EM, but with 2 cells in ROI

Figure 8: full movie, fully automated

Figure 9: full movie, fully automated, real data

Figure 10: same as spatial EM, but now have slow rise time on F

Figure 11: same as spatial EM, but now have nonlinear observations

Figure 12: same as spatial EM, but now have poisson observations

Figure 13: drift

Figure 14: thresholding

Figure 15: thresholding

Figure 16: Simulation demonstrating our neuron model and inference. Note that the effective signal-to-noise ratio (SNR) of our fast filter improves on the Wiener filter, largely by eliminating the ringing effect present in the Wiener filter output. For both filters, only the fluorescence was provided to the algorithm, so both the parameters and the inference were performed using this minimal amount of data. Top panel: Simulated fluorescence. Second panel: Simulated intracellular calcium concentration. Third panel: Simulated spike train. Fourth panel: Wiener filter (blue for positive inference, red for negative), superimposed on simulated spike train (black triangles). Bottom panel: Fast filter (same conventions as fourth panel). Parameters: $\gamma = 0.94$, $\nu = 0$, $\rho = 1$, $\sigma = 0.3$, $\lambda = 8$, $\Delta = 0.005$ msec.

Figure 17: Comparing the Wiener filter and our fast filter for a fast spiking neuron. In this scenario, the two filters perform approximately equally well. Note that both recover fast fluctuations in the firing rate that are smoothed out by the calcium dynamics. Conventions as in Fig. 1. Actual and inferred spike trains were normalized similarly, to ease comparisons. Parameters: $\gamma = 0.94$, $\nu = 0$, $\rho = 1$, $\sigma = 0.3$, $\lambda = 100$ (modulated by a sinusoid), $\Delta = 0.05$ msec.

Second, while the Wiener filter induces a "ringing" effect (where the inferred signal oscillates above and below zero), our fast filter completely eliminates this effect. These two improvements are common observations upon imposing a non-negative constraint when appropriate [Shumway and Stoffer, 2006]. Importantly, both our implementation of the Wiener filter and our fast filter require only $O(T)$ time, whereas the naïve implementation of the Wiener filter requires $O(T \log(T))$, and the naïve implementation of a non-negative filter requires $O(T^3)$. Thus, when our approximation in Eq (4) is good, our filter outperforms the Wiener filter, and imposes approximately the same computational burden.

6.2 Fast Spiking Neuron

When the observed neuron is spiking quickly, the Poisson distribution may be well approximated by a Gaussian distribution, suggesting that the Wiener filter may be optimal in this regime. But the exponential approximation of a Poisson is also very accurate in the fast spiking regime. To compare these two strategies, we simulated a fast spiking neuron, where the expected number of spikes per bin exceeds 10 (Fig. 17). In this scenario, both the Wiener filter and our fast filter perform approximately equally well. Both filters infer peaks in the firing rate that are obscured by the low-pass filter properties of the calcium dynamics. Thus, it seems from this analysis that regardless of the firing rate of the observable neuron, (1) filtering the signal may provide valuable information, and (2) our fast filter performs at least as well as the Wiener filter, without requiring more computational time.

6.3 Spatial Filtering

In the above, we assumed a one-dimensional (1-D) observation, $\boldsymbol{F}$. The raw data, however, is actually a series of multidimensional images. To get the 1-D time-series, two pre-processing steps are required. First, for each neuron, one must define a region-of-interest (ROI), essentially assigning sets of pixels to neurons. This yields a vector time-series for each neuron, $\vec{\boldsymbol{F}}_t \in \mathbb{R}^m$. Second, one must project the $m-$D observations into a 1-D time-series. Typically, this step is performed by averaging all the pixels within the ROI. In theory, one can improve on this uniform averaging by estimating the optimal spatial filter. Here, we provide details on how to incorporate this second step (projecting the $m-$D observations into 1-D) into our fast filter. First, we replace Eq (3) with:

$$\vec{\boldsymbol{F}}_t = \boldsymbol{\alpha} C_t + \boldsymbol{\beta} + \boldsymbol{\Sigma} \boldsymbol{\varepsilon}_t, \qquad \varepsilon_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \tag{18}$$

where $\vec{\boldsymbol{F}}_t$, $\boldsymbol{\alpha}$, and β are all m dimensional column vectors, $\boldsymbol{\Sigma}$ is the covariance matrix, and $\boldsymbol{\varepsilon}_t$ is an $m-$dimensional standard normal random vector. $\boldsymbol{\alpha}$ now represents the optimal spatial filter, and $\boldsymbol{\beta}$ represents the baseline for each pixel. Our goal then is to estimate $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, to provide maximal information about the underlying spike train. We would expect this optimal spatial filtering improve the effective SNR after filtering in many situations. For example, if certain pixels are anti-correlated with others, a uniform spatial filter would average out those differences, whereas the optimal filter would take advantage of that information.

To estimate $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, we first plug Eqs (18) and (2) into (9), to obtain:

$$\vec{\boldsymbol{\theta}} = \underset{\vec{\boldsymbol{\theta}} \in \vec{\boldsymbol{\Theta}}}{\operatorname{argmax}} -\frac{T}{2}\log(2\pi|\boldsymbol{\Sigma}|) + T\log(\lambda\Delta) - \lambda\Delta \boldsymbol{n}'\mathbf{1}$$
$$-\frac{1}{2}\sum_{t=1}^{T}(\vec{\boldsymbol{F}}_t - \boldsymbol{\alpha}(\gamma\widehat{C}_{t-1} - \nu - \rho\widehat{n}_t) - \boldsymbol{\beta})'\boldsymbol{\Sigma}^{-1}(\vec{\boldsymbol{F}}_t - \boldsymbol{\alpha}(\gamma\widehat{C}_{t-1} - \nu - \rho\widehat{n}_t) - \boldsymbol{\beta}) \tag{19}$$

where $|\cdot|$ indicates the determinant. By pre-whitening the observation matrix, $\vec{\boldsymbol{F}} = [\vec{\boldsymbol{F}}_1, \vec{\boldsymbol{F}}_2, \ldots, \vec{\boldsymbol{F}}_T]$, we can estimate the covariance matrix by the identity, i.e., $\boldsymbol{\Sigma} = \boldsymbol{I}$. Without loss of generality, we may assume that $\nu = 0$ and $\rho = 1$, similar to our assumption of $\alpha = 1$ and $\beta = 0$ above. This leaves γ, $\boldsymbol{\alpha}$, and $\boldsymbol{\beta}$. Assuming that γ is known (or estimated from the raw fluorescence signal, as we did above), we have:

$$\widehat{\boldsymbol{\eta}}_v = \underset{\|\boldsymbol{\eta}_v\|<1}{\operatorname{argmax}} \left\| \vec{\boldsymbol{F}} - \boldsymbol{\eta}_v \boldsymbol{X}_v \right\|^2 \tag{20}$$

where $\boldsymbol{\eta}_v = [\boldsymbol{\alpha}, \boldsymbol{\beta}]$, and $\boldsymbol{X}_v = [\gamma \boldsymbol{C} + \boldsymbol{n}, \mathbf{1}]'$. Note that we have constrained the norm of $\boldsymbol{\eta}_v$, which is necessary because otherwise there is a free scale between $\boldsymbol{X}_v$ and $\boldsymbol{\eta}_v$. Problems of the form as in Eq (20) are known as "trust region subproblems". Fortunately, many algorithms for solving such a problem are available [Fortin, 2000].

6.4 Karel's data

6.5 Incorporating a nonlinear observation model

6.6 Slow dynamics

6.7 Two spike trains (overlapping SPF's)

6.8 Optimal thresholding

7 Discussion

Acknowledgments Support for JTV was provided by NIDCD DC00109. LP is supported by an NSF CAREER award, by an Alfred P. Sloan Research Fellowship, and the McKnight Scholar Award. BOW was supported by NDS grant F30 NS051964. The authors would like to thank A. Packer for helpful discussions.

References

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Oxford University Press.

[Fortin, 2000] Fortin, C. (2000). *A Survey of the Trust Region Subproblem within a Semidefinite Framework*. PhD thesis, University of Waterloo.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[Ikegaya et al., 2004] Ikegaya, Y., Aaron, G., Cossart, R., Aronov, D., Lampl, I., Ferster, D., and Yuste, R. (2004). Synfire chains and cortical songs: temporal modules of cortical activity. *Science*, 304(5670):559–564.

[MacLean et al., 2005] MacLean, J. N., Watson, B. O., Aaron, G. B., and Yuste, R. (2005). Internal dynamics determine the cortical response to thalamic stimulation. *Neuron*, 48(5):811–823.

[Niell and Smith, 2005] Niell, C. M. and Smith, S. J. (2005). Functional imaging reveals rapid development of visual response properties in the zebrafish tectum. *Neuron*, 45(6):941–951.

[Ohki et al., 2005] Ohki, K., Chung, S., Ch'ng, Y. H., Kara, P., and Reid, R. C. (2005). Functional imaging with cellular resolution reveals precise micro-architecture in visual cortex. *Nature*, 433(7026):597–603.

[Rabiner, 1989] Rabiner, L. R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 72(2):257–286.

[Sato et al., 2007] Sato, T. R., Gray, N. W., Mainen, Z. F., and Svoboda, K. (2007). The functional microarchitecture of the mouse barrel cortex. *PLoS Biol*, 5(7):e189.

[Shumway and Stoffer, 2006] Shumway, R. and Stoffer, D. (2006). *Time Series Analysis and Its Applications*. Springer, 2nd edition.

[Vogelstein et al., 2008] Vogelstein, J., Watson, B., AM, P., Yuste, R., B, J., and Paninski, L. (2008). Spike inference from calcium imaging using sequential monte carlo methods. *Biophysical Journal*.

[Wiener, 1949] Wiener, N. (1949). *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. MIT Press, Cambridge, Mass.

[Yaksi and Friedrich, 2006] Yaksi, E. and Friedrich, R. W. (2006). Reconstruction of firing rate changes across neuronal populations by temporally deconvolved Ca^{2+} imaging. *Nat Methods*, 3(5):377–383.

A Wiener Filter

Sections 2.2 and 2.3 outline one approach to solving Eq. (3), by approximating the Poisson distribution with an exponential distribution, and imposing a non-negative constraint on the inferred $\widehat{\boldsymbol{n}}$. Perhaps a more straightforward approach would be to approximate the Poission distribution with a Gaussian distribution. In fact, as rate increases above about 10 spikes/sec, a Poisson distribution with rate $\lambda\Delta$ is well approximated by a Gaussian with mean and variance $\lambda\Delta$. Given such an approximation, instead of Eq. (4), we would obtain:

$$\widehat{\boldsymbol{n}}_w \approx \underset{n_t \in \mathbb{R}, \forall t}{\operatorname{argmin}} \sum_{t=1}^{T} \left(\frac{1}{2\sigma^2}(F_t - C_t)^2 + \frac{1}{2\lambda\Delta}(n_t - \lambda\Delta)^2 \right) \tag{21}$$

As above, we can rewrite Eq. (21) in matrix notation in terms of $\boldsymbol{C}$:

$$\widehat{\boldsymbol{C}}_w = \underset{C_t \in \mathbb{R}, \forall t}{\operatorname{argmin}} \frac{1}{2\sigma^2} \left\| \boldsymbol{F} - \boldsymbol{C} \right\|^2 + \frac{1}{2\lambda\Delta} \left\| \boldsymbol{M}\boldsymbol{C} - \lambda\Delta\boldsymbol{1} \right\|^2 \tag{22}$$

which is quadratic in $\boldsymbol{C}$, and may therefore be solved analytically using quadratic programming, $\widehat{\boldsymbol{C}}_w = \widehat{\boldsymbol{C}}_0 + \boldsymbol{d}_w$, where $\widehat{\boldsymbol{C}}_0$ is the initial guess and $\boldsymbol{d}_w = \boldsymbol{H}_w \backslash \boldsymbol{g}_w$, where

$$\boldsymbol{g}_w = \frac{1}{\sigma^2}(\boldsymbol{C}_0' - \boldsymbol{F}) + \frac{1}{\lambda\Delta}((\boldsymbol{M}\widehat{\boldsymbol{C}}_0)'\boldsymbol{M} - \lambda\Delta\boldsymbol{M}'\boldsymbol{1}) \tag{23}$$

$$\boldsymbol{H}_w = \frac{1}{\sigma^2}\boldsymbol{I} + \frac{1}{\lambda\Delta}\boldsymbol{M}'\boldsymbol{M} \tag{24}$$

Note that this solution is the optimal linear solution, under the assumption that spikes follow a Gaussian distribution, and if often referred to as the Wiener filter, regression with a smoothing prior, or ridge regression. To estimate the parameters for the Wiener filter, we take the same approach as above:

$$\widehat{\boldsymbol{\theta}}_w \approx \underset{\boldsymbol{\theta}_w}{\operatorname{argmax}} P(\boldsymbol{F}|\widehat{\boldsymbol{n}}_w, \boldsymbol{\theta}_w) P(\widehat{\boldsymbol{n}}_w|\boldsymbol{\theta}_w) \tag{25a}$$

$$= \underset{\boldsymbol{\theta}_w}{\operatorname{argmax}} -\frac{T}{2}\log(4\pi^2\sigma^2\lambda\Delta) - \frac{1}{2\sigma^2}\left\| \boldsymbol{Y}_w + \boldsymbol{\eta}_w\boldsymbol{X}_w \right\|^2 - \frac{1}{2\lambda\Delta}\left\| \widehat{\boldsymbol{n}}_w - \lambda\Delta\boldsymbol{1} \right\|^2 \tag{25b}$$

where $\boldsymbol{Y}_w$, $\boldsymbol{\eta}_w$, and $\boldsymbol{X}_w$ are defined as their subscriptless counterparts in Eq. (9).

Figure 18: Simulation demonstrating our neuron model and inference. Note that our filter accurately infers the unobserved spike train, given only the noisy fluorescence measurements. Top panel: Simulated fluorescence. Second panel: Simulated intracellular calcium concentration. Third panel: Simulated spike train. Fourth panel: Output of optimal nonnegative filter (green) superimposed on simulated spike train (black). Parameters: $\lambda = 10$, $\Delta = 0.025$ msec, $\tau = 0.5$ sec, $\sigma = 1$.

B old

Generalization of the optimal nonnegative filter The above filter design assumed a very simple model relating spikes to $[Ca^{2+}]$, and $[Ca^{2+}]$ to fluorescence. Both (1) and (2) may be generalized in a straightforward manner. In fact, our model may be thought of as a special case of the more general state-space framework:

$$\boldsymbol{C}_t = \boldsymbol{D}\boldsymbol{C}_{t-1} + \mathbf{1}n_t \tag{26a}$$

$$\boldsymbol{F}_t = \boldsymbol{A}\boldsymbol{C}_t + \boldsymbol{b} + \boldsymbol{\varepsilon}_t \tag{26b}$$

which we can solve in linear time for any log-concave distribution of n_t and $\boldsymbol{\varepsilon}_t$. For our particular scenario of interest, i.e., that of inferring spike trains from calcium sensitive fluorescence observations, these generalizations facilitate considering a model with much richer dynamics. For instance, a more accurate model relating spikes to $[Ca^{2+}]$ would consider myriad calcium extrusion mechanisms, buffering, etc. In such a situation, we would represent $[Ca^{2+}]$ as a vector, $\boldsymbol{C}_t \in \mathbb{R}^{N\times 1}$, where each element of $\boldsymbol{C}_t$ would correspond to a different spike dependent calcium process. The differential operator, $\boldsymbol{D} \in \mathbb{R}^{N\times N}$, accounts for the relative strength of each of these mechanisms, and their time constants. The spikes, n_t, are multiplied by a column vector of ones, $\mathbf{1} \in \mathbb{R}^{N\times 1}$, because the magnitude of the effect of a spike on each element in $\boldsymbol{C}_t$ is scaled by $\boldsymbol{\rho}$.[2] It should be clear that as in (2), spikes are related to calcium by a simple linear transformation. In the multidimensional scenario, however, $\boldsymbol{D}$ — a matrix — would replace a — a scalar — so $\boldsymbol{M}$ becomes block-bidiagonal, making the Hessian block tridiagonal. Nonetheless, Gaussian elimination may still solve $\boldsymbol{H}\boldsymbol{C} = \boldsymbol{g}$ in $O(T)$, so our filter may be applied to this scenario as well.

Another natural extension of this work would be to let F_t also be multidimensional. In (1), although we assumed that we had access to a monodimensional fluorescent magnitude at each time, the raw data is actually a multidimensional movie. These images, however, are necessarily blurred by the point-spread-function of the camera. Thus, we could replace (1) with (26b), where $\boldsymbol{A}$ is the point-spread function of the camera, and $\boldsymbol{b}$ sets the relative baseline intensity per pixel. Furthermore, the noise, $\boldsymbol{\varepsilon}_t$, could have any log-concave distribution, and these results would still hold. Given these generalizations, this filter may be applied to a large class of problems.

Projection Pursuit Regression Projection pursuit regression (PPR) is another technique one could use to infer a spike train from fluorescence measurements. PPR is different from the optimal nonnegative filter in a few ways. First, it solves a related, but slightly different problem: instead of finding the MAP estimate of the spike train, PPR finds the maximum likelihood estimate, i.e., the spike train that makes the fluorescence measurements most likely. Second, PPR constrains the solution to have only nonnegative integers. Therefore, $\boldsymbol{n}_{PPR} = \operatorname{argmax}_{n_t\in\mathbb{N}_0} p(\boldsymbol{F}|\boldsymbol{n})$, where $\mathbb{N}_0$ is the set of nonnegative integers.. Third, because of this constraint, PPR requires an additional parameter, w, that sets the size of the calcium transient caused by a single an action potential. This forces us to substitute (2) with $C_t = aC_{t-1} + wn_t$. Given these differences, to find $\boldsymbol{n}_{PPR}$, PPR proceeds iteratively, adding a spike with each iteration, as long as doing so reduces the residual square error (i.e., the sum of the squared difference between the inferred $\boldsymbol{C}$ and $\boldsymbol{F}$). One obtains the time of the next inferred spike by finding the maximum of the convolution of the current residual square error with the calcium kernel, $we^{-t\Delta/\tau}$. In general, this procedure takes $O(T\log T)$ per iteration, as the convolution is computed in the Fourier domain. However, the recursive state-space representation in (26) implies that this convolution requires just $O(T)$ time here. Henceforth, we therefore refer to this approach as fast PPR (or fPPR). This approach may generalized to the multidimensional cases, as in the previous filter. However, unlike the previous filter, the likelihood function is not concave (recall that PPR is a greedy optimization method), so we are no longer guaranteed to converge to the optimal solution.

[2] For the same reasons that there is no parameter scaling the spikes in the monodimensional case

Figure 19: Comparison of various linear filters for a simulated Poisson neuron spiking with a rate of 1 Hz (left panels) and 200 Hz (right panels). The main result is that the two filters proposed outperform the optimal linear (i.e., Wiener) filter. Top panels: Fluorescence measurements. Second panels: Number of spikes per frame. Third panels: Optimal linear (i.e., Wiener) filter output given the above fluorescence signal. The Wiener filter does not impose a nonnegativity constraint, thus the inferred spike train can either be positive (green) or negative (red). Fourth panels: Same as third panels, but for the optimal nonnegative filter. Note the absence of negative spikes. Fifth panels: Same as third panels, but using the fast Projection Pursuit Regression (fPPR) filter. Parameters: $\Delta = 0.025$ sec, $\tau = 0.5$ sec, $\lambda = 1/(\text{firing rate})$, left $\sigma = 0.4$, right $\sigma = 1$.

C Results

To evaluate the efficacy of our filters, we compare their results to that of the optimal linear (i.e., Wiener) filter [Wiener, 1949]. The Wiener filter differs in construction from our filters in a few ways. First, it imposes no constraint on on the spike train. Second, it is optimal upon assuming that the prior distribution of n_t is Gaussian. In our model, spiking was Poisson, (2), and the nonnegativity of n_t in the sparse-spiking regime makes the Gaussian model assumed by the Wiener filter inaccurate (Fig. 2, left). However, when spike rates are fast — e.g., on average, several spikes per image frame — a Poisson distribution is better approximated by a Gaussian distribution. Furthermore, at high firing rates, the mean of the Gaussian would be relatively far from zero, and the variance proportional, so the probability of sampling a negative number would be relatively small, obviating the need for the nonnegativity constraint. Thus, one might expect the Wiener filter to perform as well as our nonnegative filter (and fPPR) when the observed neuron has a high firing rate (and relatively low imaging rate). Furthermore, given that the calcium kernel is exponential, the Wiener filter, like the filters we developed here, only requires $O(T)$ time, as opposed to the typical $O(T \log T)$.

Fig. 19 depicts a comparison between the filters developed above and the Wiener filter for two different scenarios: a slow firing rate simulation (left panels) and a fast firing rate simulation (right panels). When action potentials are sparse, the two filters we propose above outperform the Wiener filter. Moreover, at high firing rates, all three filters perform approximately equally well.

Figure 20: Inferring a spike train given fluorescence measurements from a live *in vitro* neuron. Simultaneous recording of a saturating fluorescence signal (black line in top panel), and its associated spike train (black impluses in all panels). Note how the three filters handle saturation differently. The optimal nonnegative filter's signal-to-noise ratio degrades as saturation increases, but does not suffer a catastrophic failure (green in second panel). fPPR suffers more than the optimal nonlinear particle filter when the signal saturates, due to the hard threshold (green in third panel). The optimal nonlinear particle filter [Vogelstein et al., 2008], which explicitly incorporates saturation, correctly identifies each spike time (green in bottom panel).

While in simulations, all the above algorithms perform reasonably well, the true test is how well they perform given data from live cells. We simultaneously recorded both electrophysiologically and imaged with an epifluorescent microscope, from a pyramidal neuron in a somatosensory cortical slice, as described in [MacLean et al., 2005]. Fig. 20 shows an example fluorescence time-series in which the neuron spiked with a relatively low rate, but the calcium accumulated nonetheless, leading to fluorescence saturation (top panel). In practice, this kind of strong saturation is rarely observed (personal communication, anonymous), so this example is designed to test the limits of performance of our filters. In fact, the optimal nonnegative filter accurately infers every spike, even when the fluorescence is strongly saturating, and the signal-to-noise ratio is very poor (second panel). On the other hand, fPPR, which has a sharp threshold for including an additional spike, performs relatively poorly (third panel), demonstrating the dependency of this method on a good model fit. For comparison purposes, we also show the performance of an optimal nonlinear particle filter [Vogelstein et al., 2008], which specifically incorporates a nonlinear saturation function. While the optimal nonlinear particle filter performs better in scenarios such as this one, the computational burden is increased by approximately 100-fold relative to the fast nonnegative filter. Thus these two filters serve complementary purposes: the fast nonnegative filter is better geared to rapid online analysis of large-scale multi-neuronal data, whereas the nonlinear particle filter is better suited for offline refinement of the results from the fast nonnegative filter.

D Conclusions

We show here that for certain nonnegative deconvolution problems, we can derive an algorithm that is both optimal and efficient. More specifically, our algorithm may be applied to any model with a nonnegative signal that is linearly filtered by a matrix linear ordinary differential equation. We apply this approach to the problem of inferring the most likely spike train given noisy calcium sensitive fluorescence observations (c.f. Fig. 18), and demonstrate, in simulations, that the optimal nonnegative filter outperforms the optimal linear (i.e., Wiener) filter in both slow and fast firing rate regimes (c.f. Fig. 19). Furthermore, when applied to data from a live cell, the optimal nonnegative filter outperforms a fast projection pursuit regression filter, which constrains the inferred spike train to be nonnegative integers (c.f. Fig. 20). On the other hand, the nonnegative filter is based on a linear observation model, and therefore suffers a loss of precision in the presence of strong saturation effects, in contrast to the optimal nonlinear particle filter (c.f. Fig. 20).

The implications of these results are severalfold. First, it seems as if there is no reason to use the Wiener filter for scenarios in which our algorithm may apply. Second, as our filter is so efficient, it may be used for many real-time processing applications. Specifically, upon simultaneously imaging a population of neurons [Ikegaya et al., 2004, Niell and Smith, 2005, Ohki et al., 2005, Yaksi and Friedrich, 2006, Sato et al., 2007], our filter may be applied essentially online. This could greatly expedite the tuning of important experimental parameters — such as laser intensity — to optimize signal-to-noise ratio for inferring spikes. Third, the parameters estimated from this filter may be used to initialize the parameters of the optimal nonlinear particle filter, which may then be used offline, to further refine the spike train inference. Future work will consider multidimensional models for this application, incorporating both more sophisticated calcium models, and spatial filtering for extracting the fluorescence signal, obviating the need for additional algorithms for image segmentation.

E other

We have found empirically that despite the approximation in (4), upon initializing the parameters with a reasonable guess, the algorithm tends to converge to parameters that are reasonably close to the true parameters, resulting in an accurate spike reconstruction. Importantly, estimating these parameters typically requires only a very short sequence of observations, e.g., several seconds of data including about 5–10 spikes (not shown).