Real-Time Nonparametric Bayesian Inference for Spike Sorting

Anonymous Author(s)
Affiliation
Address
email

Abstract

SMUG

1 Introduction

The recently announced BRAIN initiative calls for the development of technologies that will advance our understanding of neuroscience [**?**]. Crucial to the success of this endeavor will be the advancement of our ability to understand the *dynamics* of the brain, via the measurement of large populations of neural activity at the single neuron level. Such reverse engineering efforts will benefit from real-time decoding of neural activity, to facilitate adapting the stimuli to probe the functional connectivity more effectively. Regardless of the experimental apparati used to measure such data (e.g., electrodes or calcium imaging), real-time decoding of individual neuron responses requires identifying and labelling individual spikes from recordings from large populations of neurons. In other words, real-time decoding requires real-time spike sorting.

Automatic spike sorting methods are continually evolving to deal with more sophisticated experiments. Most recently, several methods have been proposed to (i) learn the number of separable neurons on each electrode or "multi-trode" [**?**], or (ii) operate online to resolve overlapping spikes from multiple neurons [**?**]. To our knowledge, no method to date is able to simultaneously address both of these challenges.

We develop a nonparametric Bayesian continuous-time generative model of population activity. Our model explains the continuous output of each neuron by a latent marked Poisson process, with the "marks" characterizing the shape of each spike. Previous efforts to address overlapping spiking often assume a fixed kernel for each waveform, but joint intracellular and extracellular recording clearly indicate that this assumption is false (see Figure **??**). To we assume that the statistics of the marks are time-varying. Moreover, rather than assuming a priori a fixed number of separable neurons per channel, we take a nonparametric Bayesian approach. We use the framework of completely random measures to jointly characterize spike times and waveforms from a potentially infinite number of neurons [**?**].

We characterize the above infinite-dimensional continuous-time stochastic process as the limiting process of an intuitive discrete-time model. We then develop an online variational Bayesian inference algorithm for this model [**?**]. Via numerical simulations, we demonstrate that our inference procedure improves over the previous state-of-the-art, even though we allow the other methods to use the entire dataset for training, whereas we learn online. Moreover, we demonstrate that we can effective track the time-varying changes in waveform, and detect overlapping spikes. Indeed, it seems that the false positives detections from our approach have indistinguishable first order statistics from the true positives, suggesting that that second-order methods may be required to reduce false positive rate (i.e., template methods may be inadequate). Our work therefore suggests that further improvements in real-time decoding of activity may be most effective if directed at simultaneous real-time

spike sorting and decoding. To facilitate such developments and support reproducible research, all code and data associated with this work is provided in the Supplementary materials.

2 Model

2.1 Notation

Unless otherwise specified, we let lower-case English alphabet characters indicate scalars $x \in \mathbb{R}$, bold indicates column vectors $\boldsymbol{x} \in \mathbb{R}^p$, and upper-case bold indicates matrices, $\mathbf{X} \in \mathbb{R}^{p \times q}$. Parameters and constants ~~will be~~ Greek characters. Time ~~will be~~ $t \in (0, T)$, $i \in [N]$ ~~will~~ index the N neurons, where $[N] = \{1, 2, \ldots, N\}$. Script ~~will~~ denote sets and pipes will denote the cardinality of the set, e.g. $|\mathcal{T}|$.

2.2 Input

Our data is a time-series of multielectrode recordings $\mathbf{X} \equiv (\mathbf{x}_1, \cdots, \mathbf{x}_T)$, and consists of T recordings from M channels. The set of recording times lie on regular grid with interval length Δ, while $\mathbf{x}_t \in \mathbb{R}^M$ for all t. This time-series of electrical activity is driven by an unknown number of neurons, and we let the number be unbounded. The neurons themselves generate continuous-time voltage traces, with the outputs of all neurons superimposed and discretely sampled to produce the recordings $\mathbf{X}$. At a high level, we model the output of each neuron as a series of idealized spikes which are smoothed with appropriate kernels, the latter determining the shape of each spike. We describe this in detail, starting first with a model for a single channel recording $\mathbf{x}^\mathsf{T} \equiv (x_1, \cdots, x_T)$.

2.3 Modeling a single electrode recording

There is a rich literature characterizing the spiking activity of a single neuron [**?**], accounting in detail for factors like non-stationarity, refractoriness and spike waveform. We however make a number of simplifying assumptions (some of which we later relax, others we leave for future work). Figure **??** provides a schematic depiction of our generative process. First, we model the spiking activity of each neuron as stationary and memoryless, so that its set of spike times are distributed as a homogeneous Poisson process (PP). justify? We model the neurons themselves are heterogeneous, with the i^{th} neuron having an (unknown) firing rate λ_i. Call the ordered set of spike times of the i^{th} neuron $\mathcal{T}_i$; then the time between successive elements of $\mathcal{T}_i$ is exponentially distributed with mean $1/\lambda_i$. We write this as

$$\mathcal{T}_i \sim \text{PP}(\lambda_i) \tag{1}$$

The actual electrical output of a neuron is not binary; instead each spiking event is a smooth perturbation in voltage about a resting state. This perturbation forms the shape of the spike, and without any loss of generality, we set the resting state to zero. (figure? better biological description? comment on how we preprocess the data to get zero mean?). While the spike shapes vary across neurons as well as across different spikes of the same neuron, each neuron has its own characteristic distribution over shapes. We let $\boldsymbol{\theta}_i^* \in \Theta$ parametrize this distribution for neuron i. Whenever this neuron emits a spike, a new shape is drawn independently from the corresponding distribution. This waveform is then offset to the time of the spike, and contributes to the voltage trace associated with that spike. The complete detected output of the neuron is the superposition of all these spike waveforms plus noise. We assume the noise at each observation time is i.i.d. Gaussian.

We model the random spike shapes themselves as weighted superpositions of a dictionary of K basis functions $\mathbf{d}(t) \equiv (d_1(t), \cdots, d_K(t))^\mathsf{T}$. The dictionary elements are shared across all neurons, and each is a real-valued function of time, e.g., $d_k \in L_2$. For the i^{th} neuron, the j^{th} spike $\tau_{ij} \in \mathcal{T}_i$, is associated with a random K-dimensional weight vector $\mathbf{y}_{ij}^* \equiv (y_{ij1}^*, \ldots y_{ijK}^*)^\mathsf{T}$, and the shape of this spike at time t is given by the weighted sum $\sum_{k=1}^K y_{ijk}^* d_k(t)$. We assume $\mathbf{y}_{ij}^* \sim \mathcal{N}_K(\boldsymbol{\mu}_i^*, \Sigma_i^*)$, indicating a K-dimensional Gaussian distribution with mean and covariance given by $\theta_i^* \equiv (\boldsymbol{\mu}_i^*, \Sigma_i^*)$. Then, at any time t, the output of neuron i is

$$x_i(t) = \sum_{j=1}^{|\mathcal{T}_i|} \sum_{k=1}^{K} y_{ijk}^* d_k(t - \tau_{ij}). \tag{2}$$

The total signal recorded from any electrode is the superposition of the outputs of all neurons. Assume for the moment there are N neurons, and define $\mathcal{T} = \cup_{i \in [N]} \mathcal{T}_i$ as the (ordered) union of the

spike times of all neurons. Let $\tau_l \in \mathcal{T}$ indicate the time of l^{th} overall spike, whereas $\tau_{ij} \in \mathcal{T}_i$ is the j^{th} spike of neuron i. This defines a pair of mappings: $\nu : [|\mathcal{T}|] \to [N]$, and $p : [|\mathcal{T}|] \to \mathcal{T}_{\nu_i}$, with $\tau_l = \tau_{\nu_l p_l}$. In words, $\nu_l \in N$ is the neuron to which the l^{th} element of $\mathcal{T}$ belongs, while p_l indexes this spike in the spike train $\mathcal{T}_{\nu_l}$. Let $\boldsymbol{\theta}_l \equiv (\boldsymbol{\mu}_l, \Sigma_l)$ be the neuron parameter associated with spike l, so that $\boldsymbol{\theta}_l = \boldsymbol{\theta}^*_{\nu_l}$. Finally, define $\mathbf{y}_l = (y_{l1}, \ldots, y_{lK})^\mathsf{T} \equiv \mathbf{y}^*_{\nu_j p_j}$ as the weight vector of spike τ_l. Then, we have that

$$x(t) = \sum_{i \in [N]} x_i(t) = \sum_{l \in |\mathcal{T}|} \sum_{k \in [K]} y_{lk} d_k(t - \tau_l), \qquad \text{where } \mathbf{y}_l \sim \mathcal{N}_K(\boldsymbol{\mu}_l, \Sigma_l). \tag{3}$$

From the superposition property of the Poisson process [1], the overall spiking activity $\mathcal{T}$ is Poisson with rate $\Lambda = \sum_{i \in [N]} \lambda_i$. Each event $\tau_l \in \mathcal{T}$ is associated with a pair of labels, the parameter of the neuron to which it is assigned ($\boldsymbol{\theta}_l = (\boldsymbol{\mu}_l, \Sigma_l)$), and the weight-vector characterizing the spike shape ($\mathbf{y}_l$). We view these as the "marks" of a marked Poisson process $\mathcal{T}$. From the properties of the Poisson process, we have that the marks $\boldsymbol{\theta}_l$ are drawn i.i.d. from a probability measure

$$G(\mathrm{d}\boldsymbol{\theta}) = \frac{1}{\Lambda} \sum_{i \in [N]} \lambda_i \delta_{\boldsymbol{\theta}^*_i} \tag{4}$$

Note that with probability one, the neurons have distinct parameters, so that the mark $\boldsymbol{\theta}_l$ associated with spike l identifies the neuron which produced it: $G(\boldsymbol{\theta}_l = \boldsymbol{\theta}^*_i) = P(\nu_l = i) = \lambda_i / \Lambda$. Given $\boldsymbol{\theta}_l$, $\mathbf{y}_l$ is distributed as in Eq. (3). The output waveform $x(t)$ is then a linear functional of this marked Poisson process.

2.4 Completely random measures (CRMs)

The previous section assumed a known number of neurons N. In practice however, our recordings are a superposition of the outputs of an unknown number of neurons. We deal with this uncertainty by taking a nonparametric Bayesian approach, and letting N be infinite. While the number of neurons observed over any finite observation interval is finite, this number increases with the observation interval. Such an approach leads to an elegant and flexible modeling framework, and has already proved successful in neuroscience applications [**?**]. Since only a finite number of spikes are observed in any finite interval, the total rate Λ must also be finite. Moreover, we want this to be dominated by a few λ_i: the corresponding neurons contribute the majority of the spiking activity in the observation interval. A natural framework that captures these modeling requirements is that of *completely random measures* (CRMs) [3]. Completely random measures are stochastic processes that form flexible and convenient priors over infinite dimensional objects like probability distributions [4], hazard functions [5], latent features [6] etc. These have been well studied in the Bayesian nonparametrics and machine learning communities, and there exists a wealth of literature on their theoretical properties, as well as on computational approaches to posterior inference.

Recall that each neuron is characterized by a pair of parameters $(\lambda_i, \boldsymbol{\theta}^*_i)$; the former characterizes the distribution over spike times, and the latter, over spike shapes. With Eq. (4) in mind, we map the infinite collection of pairs $\{(\lambda_i, \boldsymbol{\theta}^*_i)\}$ to an atomic measure on Θ:

$$\Lambda(\mathrm{d}\boldsymbol{\theta}) = \sum_{i=1}^{\infty} \lambda_i \delta_{\boldsymbol{\theta}^*_i}. \tag{5}$$

For any subset $\mathit{\Theta}$ of Θ, the measure $\Lambda(\mathit{\Theta})$ equals $\sum_{\{i : \boldsymbol{\theta}^*_i \in \mathit{\Theta}\}} \lambda_i$. We allow $\Lambda(\cdot)$ to be random, modeling it as a realization of a completely random measure. Such a random measure has the property that for any two disjoint subsets $\mathit{\Theta}_1$, $\mathit{\Theta}_2 \subseteq \Theta$, the measures $\Lambda(\mathit{\Theta}_1)$ and $\Lambda(\mathit{\Theta}_2)$ are independent. This distribution over measures is induced by a distribution over the infinite sequence of weights (the λ_i's), and a distribution over the sequence of their locations (the $\boldsymbol{\theta}^*_i$'s). For a CRM, the weights λ_i are the jumps of a Lévy process [7], and their distribution is characterized by a Lévy measure $\rho(\lambda)$. The locations $\boldsymbol{\theta}^*_i$ are drawn i.i.d. from a base probability measure $H(\boldsymbol{\theta}^*)$. As is typical, we assume these to be independent (though this is not necessary).

The particular class of CRM is determined by the Lévy measure $\rho(\lambda)$. For our application, we set $\rho(\lambda) = \alpha \lambda^{-1} \exp(-\lambda)$; this results in a CRM called the Gamma process (ΓP) [**?**]. The Gamma process has the convenient property that the total rate $\Lambda \equiv \Lambda(\Theta) = \sum_{i=1}^{\infty} \lambda_i$ is Gamma distributed (and thus conjugate to the Poisson process prior on $\mathcal{T}$). The Gamma process is also closely connected with the Dirichlet process [8], which will prove useful later on. Other choices of

the Lévy intensity can be used to capture greater uncertainty in the number of neurons active in any finite interval, or to model power-law behavior in the number of spikes emitted by different neurons.

To complete the specification on the Gamma process, we choose a base-measure $H(\boldsymbol{\theta}^*)$. Recalling that $\boldsymbol{\theta}^* \equiv (\mu^*, \Sigma^*)$ gives the mean and variance of the weight-vector $\mathbf{y}^*$ of a neuron, we set $H(\boldsymbol{\theta}^*)$ to be the conjugate normal-Wishart distribution with parameters ϕ. Our overall model is then:

$$\mathcal{T}_i \sim \text{PP}(\lambda_i) \quad i \in \mathbb{N}, \qquad \text{where } \Lambda(\cdot) = \sum_{i=1}^{\infty} \lambda_i \delta_{\theta_i^*} \sim \Gamma\text{P}(\alpha, H(\cdot|\phi)), \tag{6a}$$

$$x_i(t) = \sum_{j=1}^{|\mathcal{T}_i|} \sum_{k=1}^{K} y_{ijk}^* d_k(t - \tau_{ij}), \qquad \text{where } \mathbf{y}_{ij}^* \sim \mathcal{N}_K(\boldsymbol{\mu}_i^*, \Sigma_i^*) \quad i, j \in \mathbb{N}, \tag{6b}$$

$$x(t) = \sum_{i=1}^{\infty} x_i(t) + \varepsilon_t, \qquad \text{where } \varepsilon_t \sim \mathcal{N}(0, \Sigma_x) \tag{6c}$$

It will be more convenient to work with the marked Poisson process representation of Eq. (3). The superposition process $\mathcal{T}$ is a rate Λ Poisson process, and under a Gamma process prior, Λ has a Gamma distribution with shape and scale parameters α and 1 respectively [8]. As we saw (Eq. (4)), the labels $\boldsymbol{\theta}_i$ assigning events to neurons are drawn i.i.d. from a normalized Gamma process $G(\mathrm{d}\boldsymbol{\theta})$:

$$G(\mathrm{d}\boldsymbol{\theta}) = \frac{1}{\Lambda} \sum_{l=1}^{\infty} \lambda_l. \tag{7}$$

$G(\mathrm{d}\boldsymbol{\theta})$ is a random probability measure called a *normalized random measure* [4]. Crucially, a normalized Gamma process is the Dirichlet process (DP) [8], so that G is a draw from a Dirichlet process. For the l^{th} spike in $\mathcal{T}$, given its parameter $\boldsymbol{\theta}_l$, its shape vector is drawn from a normal distribution with parameters $(\boldsymbol{\mu}_l, \Sigma_l)$. Thus the spike parameters $\boldsymbol{\theta}$ are i.i.d. draws from a Dirichlet process, while the weight vectors are draws from a DP mixture model of Gaussians (DPMM) [9].

The connection with the DP allows us to simplify the representation of our model. In particular, we exploit a remarkable property of the DP that allows us to integrate out the infinite-dimensional variable $G(\cdot)$. The resulting marginal distribution over observations follows the so-called Chinese restaurant process (CRP) [10]. Under this scheme, the l^{th} spike is assigned the same parameter as an earlier spike with probability proportional to the number of earlier spikes having that parameter. It is assigned a new parameter (and thus, a new neuron is observed) with probability proportional to α. Letting C_t be the number of neurons observed until time t, and $|\mathcal{T}_i^t| = |\mathcal{T}_i \cap [0, t)|$ be the number of spikes produced by neuron i before time t, we then have for spike l at time $t = \tau_l$:

$$P(\nu_l = i) \propto \begin{cases} |\mathcal{T}_i^t| & i \in \{1, \cdots, C_t\}, \\ \alpha & i = C_t + 1, \end{cases} \tag{8}$$

and $\boldsymbol{\theta}_l = \boldsymbol{\theta}_{\nu_l}^*$. This marginalization property of the DP allows us to integrate out the infinite-dimensional rate vector $\Lambda(\cdot)$, and sequentially assign spikes to neurons based on the assignments of earlier spikes. Doing so requires one final property: for the Gamma process, the random probability measure $G(\cdot)$ is independent of the total mass Λ. Consequently, the clustering of spikes (determined by $G(\cdot)$) is independent of the rate Λ at which they are produced. We then have the following model equivalent to the one above:

$$\mathcal{T} \sim \text{PP}(\Lambda), \qquad \text{where } \Lambda \sim \text{Gamma}(\alpha, 1), \tag{9a}$$

$$(\boldsymbol{\mu}_l, \Sigma_l) \equiv \boldsymbol{\theta}_l, \qquad \text{where } \boldsymbol{\theta}_l \sim \text{CRP}(\alpha, H_\phi(\cdot)), \quad l \in [|\mathcal{T}|], \tag{9b}$$

$$x(t) = \sum_{l=1}^{|\mathcal{T}|} \sum_{k=1}^{K} y_{lk} d_k(t - \tau_l) + \varepsilon_t \quad \text{where } \mathbf{y}_l \sim \mathcal{N}_K(\boldsymbol{\mu}_l, \Sigma_l),\ l \in [|\mathcal{T}|],\ \varepsilon_t \sim \mathcal{N}(0, \sigma^2). \tag{9c}$$

Unlike most applications which observe the outputs of a CRP, our observation at any time t is a convolution-like function of the CRP outputs of all earlier times. Consequently, we cannot directly apply standard techniques for posterior inference. In §3, we develop a novel online algorithm for posterior inference; first, we provide a discrete-time approximation to our model.

2.5 A discrete-time approximation

In the previous subsections, we modeled the continuous-time voltage waveform output of a neuron. Our data on the other hand consists of recordings at a discrete set of times. While it is possible

to make inferences about the continuous-time process that underlies these discrete recordings, in this paper, for simplicity, we restrict ourselves to the discrete case. We thus provide a discrete-time approximation to the model above. Our approximation follows easily from the marked Poisson process characterization of the model.

Recall first the Bernoulli approximation to the Poisson process: a sample from a Poisson process with rate Λ can be approximated by discretizing time at a granularity Δ, and assigning each bin an event independently with probability $\Lambda\Delta$. The accuracy of this approximation increases as Δ tends to 0. In our case, we have to approximate the marked Poisson process $\mathcal{T}$. All this requires additionally is to assign marks $\boldsymbol{\theta}_i$ and $\mathbf{y}_i$ to each event in the Bernoulli approximation. Following Eqs. (9b) and (9c), the $\boldsymbol{\theta}_l$'s are distributed according to a Chinese restaurant process, while each $\mathbf{y}_l$ is drawn from a normal distribution parametrized by the corresponding $\boldsymbol{\theta}_l$. We discretize the elements of dictionary $d_k \equiv \{d_k(t)\}_{t\in(0,T)}$ as well, defining a mapping from L_2 to $\mathbb{R}^L$ yielding discrete dictionary elements $\widetilde{d}_k = (\widetilde{d}_k[1], \ldots, \widetilde{d}_k[L])^\mathsf{T}$, so $\mathbf{D} \in \mathbb{R}^{K\times L}$. The shape of the j^{th} spike is now a vector of length L, and for a weight vector $\mathbf{y}$, is given by $\mathbf{Dy}$.

We can simplify notation a bit for the discrete-time model. Let t index time-bins (so that for an observation interval of length T, $t \in [T/\Delta]$). Let the binary variable z_t indicate whether or not a spike in present in time bin t (recall that $z_t \sim$ Bernoulli$(\Lambda\Delta)$). Let ν_t and θ_t be the neuron and neuron parameter associate with the spike in time bin t (these remain undefined if $z_t = 0$). Then the output at time t, x_t is given by

$$x_t = \sum_{\tau=1}^{L} \mathbf{D}_\tau^\mathsf{T} \mathbf{y}_{t-\tau-1} + \epsilon_t \quad \text{where } \mathbf{D}_\tau \text{ is column } \tau \text{ of } \mathbf{D}, \text{ and } \epsilon_t \sim \mathcal{N}(0, \sigma^2). \tag{10}$$

2.6 Correlations in time and across electrodes

We end by describing two extensions to the model outlined so far. The first relaxes the requirement that the parameters θ^* of each neuron remain constant, instead allowing the mean of the weight-vector ($\boldsymbol{\mu}^*$) evolve with time. Justify, eg cells dying. In order that the means remain marginally Gaussian distributed, we model their time-evolution as realizations of a Gaussian process (GP) [**?**] In continuous-time, we allow these parameters to evolve according to a Gaussian process (GP) [], so that our original model now involves a DP mixture of Gaussian processes. We choose a Markov kernel, this results in a first-order autoregressive process for the discrete case. Letting $\boldsymbol{B}$ be the transition matrix of this process, and r be an independent and normally distributed innovation term, we have

$$\boldsymbol{\mu}^*_{t+1} = \mathbf{B}\boldsymbol{\mu}^*_t + \mathbf{r}_t. \tag{11}$$

Our second extension is to generalize out model from a single electrode to the case of multielectrode recordings. We do this allowing each channel to have its own latent weight vector: for channel m at time t, call this $\mathbf{y}_t^m$. Of course, these vectors need to be correlated (since they correspond to the same spike). We do this by modeling the set of vectors as a matrix-variate normal. Need to check how we set details of this correlation matrix. Algorithm 1 outlines generative mechanism of the data for the discrete-time model.

3 Inference

We now address the problem of posterior inference over the latent variables given the matrix $\mathbf{X}$ of multielectrode recordings. Unsurprisingly, exact inference is intractable, and we have to resort to approximating the posterior distribution. There exists a vast literature on approximate inference for Bayesian nonparametric models, especially so for models based on the Dirichlet process. Traditional approaches are sampling-based, typically involving Markov chain Monte Carlo techniques (see eg. [11, 12]), and recently there has also been work on constructing deterministic approximations to the intractable posterior (eg. [13, 14]). Posterior inference in our problem is further confounded by two additional factors. The first is the convolutional nature of our observation process, where at each time, we observe a function of the previous observations drawn from the DP mixture model. This is in contrast to the usual situation where one directly observes the DPMM outputs themselves. The second complication is a computational requirement: typical inference schemes are batch methods

Pseudocode 1 Generative mechanism for the multi-electrode, non-stationary, discrete-time process

Input: a) the number of bins T, and the bin-width Δ
b) the K-by-L dictionary $\mathbf{D}$ of K basis functions
c) the DP hyperparameters α and ϕ.
d) the transition matrix $\mathbf{B}$ of the neuron AR process

Output: An M-by-T matrix $\mathbf{X}$ of multielectrode recordings.

Initialize the number of clusters C to 0.
Draw the overall spiking rate $\Lambda \sim \text{Gamma}(1, \alpha)$.
for t in $[T]$ **do**
  Sample $z_t \sim \text{Bernoulli}(\Lambda\Delta)$, with $z_t = 1$ indicating a spike in bin t.
  if $z_t = 1$ **then**
    Sample ν_t, assigning the spike to a neuron, with $P(\nu_t = i) \propto \begin{cases} |\mathcal{T}_i| & i \in [C] \\ \alpha & i = C+1 \end{cases}$
    if $\nu_t = C + 1$ **then**
      $C \leftarrow C + 1$.
      Set $\theta^*_C \sim H_\phi(\cdot)$, and n_C=1.
    else
      $n_{\nu_t} \leftarrow n_{\nu_t} + 1$.
    end if
    Set $\theta_t = \theta^*_{\nu_t}$, and recall that $\theta_t \equiv (\boldsymbol{\mu}_t, \Sigma_t)$.
    Sample $\mathbf{Y}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \Sigma \otimes K)$, determining the spike shape at all electrodes. *define* $\otimes$.
  end if
  Update the cluster parameters: $\boldsymbol{\mu}^*_i = \mathbf{B}\boldsymbol{\mu}^*_i + r_i \quad i \in \{1, \cdots, C\}$
  $x_t = \sum_{\tau=1}^{L} \mathbf{D}_\tau^\intercal \mathbf{y}_{t-\tau} + \epsilon_t$
end for
seems like C should be indexed by t?

that are slow and computationally expensive. Our ultimate goal, on the other hand, is to perform inference in real time, making these approaches are unsuitable for our purposes.

With the latter objective in mind, we develop an online algorithm for posterior inference. Our algorithm is based on [15], though that work was concerned with the usual case of i.i.d. observations from a DPMM. We generalize this algorithm to our observation process, and also to account for the time-evolution of the cluster parameters.

At a high level, at time t, our algorithm maintains the set of times of the spikes it has inferred from the observations until time t. It also maintains the identities of the neurons that it assigned each of these spikes to, as well as the weight vectors determining the shapes of the associated spike waveforms. In addition to these point estimates, the algorithm also keeps a set of posterior distributions $q_{it}(\theta^*_i)$ where i spans over the set of neurons associated with the spikes seen so far. Let the number of unique neurons observed at time t be C_t, so that i varies from 1 to C_t. For each i, $q_{it}(\theta^*_i)$ approximates the distribution over the parameters $\theta^*_i \equiv (\mu^*_i, \Sigma^*_i)$ of neuron i given the observations until time t.

Having identified the location and shape of spikes from earlier times, we can calculate their contribution to the recording x_{t+1} at time $t+1$. We subtract this term from x_{t+1}, and treat the residual $\widetilde{\mathbf{x}}_{t+1}$ as an observation from a DP mixture model. Given this residual, our algorithm then makes a hard decision about whether or not this was produced by an underlying spike, what neuron that spike belongs to (one of the earlier neurons or a new neuron), and what the shape of the associated spike waveform is. The latter is used to calculate $q_{i,t+1}(\theta^*_i)$, the new distribution over neuron parameters at time $t+1$. Our algorithm proceeds recursively in this manner.

Below, we describe each of these steps for the case a single electrode; generalizing to the multielectrode case is straightforward. We start by recalling that the basis functions $\mathbf{D}$, and thus all spike waveforms, span L time bins. Let z_t indicate whether or not a spike is present at time t, with $\mathbf{y}_t$ giving its shape, and ν_t its associated neuron. The residual at time t is then given by

$$\widetilde{\mathbf{x}}_t = \mathbf{x}_t - \sum_{i=1}^{L} \mathbf{D}\mathbf{y}_{t-i} \tag{12}$$

Given this residual, the first step is to decide whether or not there is a spike underlying this residual. By Bayes' rule,

$$P(z_t = 1|\widetilde{\mathbf{x}}_t) \propto P(z_t = 1, \widetilde{\mathbf{x}}_t) = \sum_{\nu_t=1}^{C_{t-1}+1} P(\widetilde{\mathbf{x}}_t, \nu_t|z_t = 1)P(z_t = 1) \tag{13}$$

Recall that C_{t-1} is the number of unique neurons underlying the observations before time t. Letting n_i be the number of spikes from neuron i, $P(\nu_t = i|z_t = 1)$ follows from the CRP update rule (equation (8)). On the other hand,

$$P(\widetilde{\mathbf{x}}_t|\nu_t = i, z_t = 1) = \int_\Theta P(\widetilde{\mathbf{x}}_t|\theta_t)q_{it}(\theta_t)\mathrm{d}\theta_t \tag{14}$$

Here, $P(\widetilde{\mathbf{x}}_t|\theta_t)$ is just the normal distribution with mean μ_t and variance θ_t, while we restrict $q_{it}(\cdot)$ be the normal-Wishart distribution. We can then evaluate integral (14), and then summation (13) to calculate $P(z_t = 1|\widetilde{\mathbf{x}}_t)$. If this exceeds a threshold of 0.5 we decide that there is a spike present at time t, otherwise, we set $z_t = 0$. Observe that making this decision involves marginalizing over all possible cluster assignments ν_t, and all values of the weight vector $\mathbf{y}_t$. On the other hand, having decided that a spike is present , we simplify matters collapsing these posterior distributions to point estimates. In particular, we assign the spike to the neuron with highest posterior probability, and similarly set the weight-vector to its MAP value.

Given these point estimates, we can update the posterior distribution over parameters of cluster ν_t to obtain $q_{i,t+1}$ from $q_{i,t}$; this is straightforward because of conjugacy. We need to follow this up with an additional update step for the parameter distributions of *all* clusters because of the AR process on the parameters. This too is straightforward Get exact update rule.

4 Experiments

4.1 Performance on partial ground truth data

need to discuss preprocessing

fig 1a: include all data in main, in supplement, include current 1a and with errorbars

table of run times

compare with DPMM

check matlab initializes with hierarchical clustering for GMM & k-means

add waveforms over time in PC space

scatterplot of waveforms of TP, FP, and FN in PC space (either learned from all data or only these waveforms). let's see whether the distributions look different at all in this space, which should inform us with regard to how well we are doing. perhaps plot ellipsoids (corresponding to level-sets) with means and covariances fit from the data

show waveforms in 3b and 3c in PCA of channel 3 to show that they are overlapping. done

rotate 3a and 4a panels to be N rows and 1 column, rather than 1 row and N columns

show that we capture waveform adaptation, via showing the μ moving around in PC space

show 10 waveforms of AR and non-AR means & errorbars over time

for overlapping, show in black dashed lines, the "expected" trace (sum of overlapping), and beside that, show the three time-varying residuals, reporting the sum-squared residuals

then let's plot the two distributions of sum-squared-residuals (one for assuming overlapping, one for only the ground-truth neuron). depending on how different those two distributions look and what they look like, we can test to demonstrate the differences.

4.2 Data

Experiments were performed on a subset of the publicly available[1] dataset described in [16]. We used the dataset d533101 that consisted of an extracellular tetrode and a single intracellular electrode. The recording was made simultaneously on all electrodes and was set up such cell with the intracellular electrode was also recorded on the extracellular array implanted in the hippocampus of an anesthetized rat. *We pre-processed the data via .... (did we "cheat" in this pre-processing, ie, include future info?)*

The intracellular recording is relatively noiseless and gives nearly certain firing times of the intracellular neuron. The extracellular recording contains the spike waveforms from the intracellular neuron as well as an unknown number of additional neurons. The data is a 4-minute recording at a 10 kHz sampling rate and preprocessed with a high-pass filter at 800 Hz. *@dec: always a space between # and unit. i fixed throughout.*

Each algorithm gives a clustering of the detected spikes. In this dataset, we only have a partial ground truth, so we are only able to analyze whether a spike comes from the intracellular (IC) recording or not. In these experiments, we define a detected spike to be an IC spike if the IC recording has a spike within 0.5 ms *@dec: always "0." prior to fraction, because "." is easy to miss.* of the detected spike in the extracellular recording. We define the cluster with the greatest number of intracellular spikes as a the "IC cluster" *@dec: double quote, latex style please*. We refer to these data as "partial ground truth data", because we know the ground truth spike times for one of the neurons, but not all the others.

4.3 Bake-off Recipes

We compare a number of variants of SMUG, as well as several previously proposed methods, as described below. SMUG can operation on either multichannel data, SMUG-M, or single channel data, SMUG-1. The waveforms can either be assumed to follow an auto-regressive process, SMUG-MA, or be i.i.d., SMUG-MW (W for white). We also compare to Gaussian mixture models with N components (GMM-N) *or does N refer to number of PCs? either way, we must specify both*, and k-means-N, where N indicates the number of principle components on which we operate. Finally, we include a Focus Mixture Model [**?**], a recently proposed Bayesian generative model with state-of-the-art performance. Only SMUG methods were online as we could not find any previous code available for online spike sorting, despite a small literature on the topic [**?**].

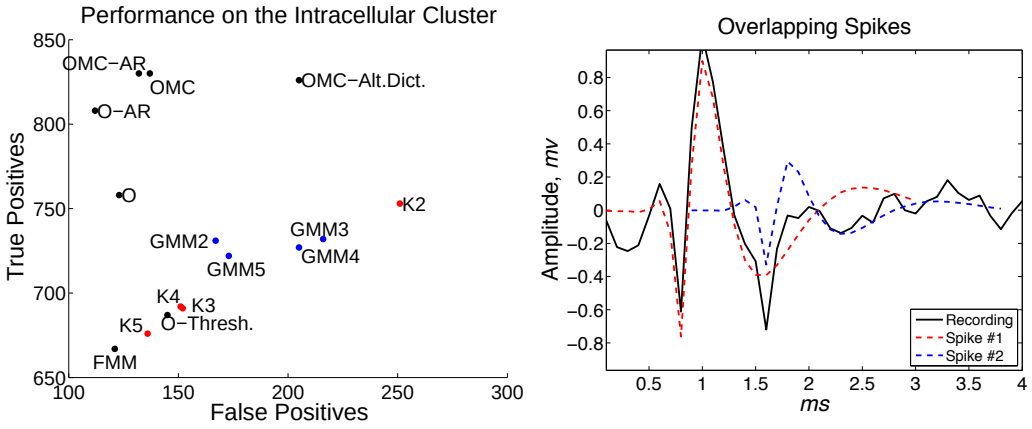
The single-channel experiments were all run on channel 2 (the results were nearly identical for all channels). The spike detections for the offline methods used a threshold of 3 times the noise standard deviation [17]. (*Need to do more comparisons and descriptions*) and windowed at a size $L = 30$ (*check if L matches with Vinayak notation*) *vinayak uses L, i believe*. The action potential window was set to 30, and PCA was used to reduce the space to K=3 for the experiments.

4.4 Results

The main empirical result of our contribution is that all variants of SMUG detect more true positives with fewer false positives than any of the other algorithms on the partial ground truth data (see Figure 1a). Our improved sensitivity and specificity is *despite* that SMUG is fully online, whereas all the algorithms that we compare to are batch algorithms using all data for all spikes. Of the SMUG variants, both multi-channel (MC) variants outperformed variants utilizing only a single channel (see §4.7 for more details). Moreover, SMUG with time-varying distribution of waveforms yielded fewer false positives than SMUG with i.i.d. waveforms, and both yielded the same number of true positives (see §4.5 for more details). Perhaps most importantly, SMUG found more spikes than all the other algorithms, at least partially because it could detect overlapping spikes (see §**??** for more details).

The results were very similar for K=2, K=3, and K=4. On the full 4-minute dataset, there were 3742 spikes detected with the threshold, and 753 of those corresponded to the intracellular spikes. For comparison, the algorithms we implemented here detected 3593 spikes of which 840 were intracellular spikes; this means we detected fewer total "spikes" while capturing more of the intracellular spikes. (Is it worthwhile to give a plot showing what we don't detect that a threshold does and

[1] http://crcns.org/data-sets/hc/hc-1/



(a) (b)

Figure 1: Results on the d533101 dataset. (a) This shows the average number of true positives versus the average number of false positives in the intracellular cluster for 2 minute segments of the 4 minutes of day. The methods that combine the clustering and the detection have similar numbers of false positives but greater numbers of true positives. I have error bars as well, but they look ugly (b) An example of an inferred overlapping spike.

vice versa?) To get uncertainty estimates, we split the data into 10 random 2 minute segments and process them with the algorithms. The results are qualitatively similar (see Figure **??**).

The online algorithms were all run with weakly informative parameters (*add parameters once I get vinayak's notation*). The parameters were insensitive to minor changes. Running time in unoptimized MATLAB code for 4 minutes of data was 31s was a single channel and 3 minutes for all 4 channels on a 3.2 GHz Intel Core i5 machine with 6GB of memory.

@dec: i changed this to highlight that we used other data to learn the dictionary. let's replace the figs with those results. Because our method is fully online, we cannot use the data to estimate the dictionary weights. We therefore learn a dictionary via PCA applied to data from a the d566102 dataset in the HC-1 database. This dataset is recorded from the same type of tetrode device implanted by the same lab in a hippocampus of a different anesthetized rat. Our approach is robust to such variations in the dictionary. We also compare our results using the dictionary learned from the data, and obtain quantitatively quite similar results (see Figure **??** in the Supplementary materials).

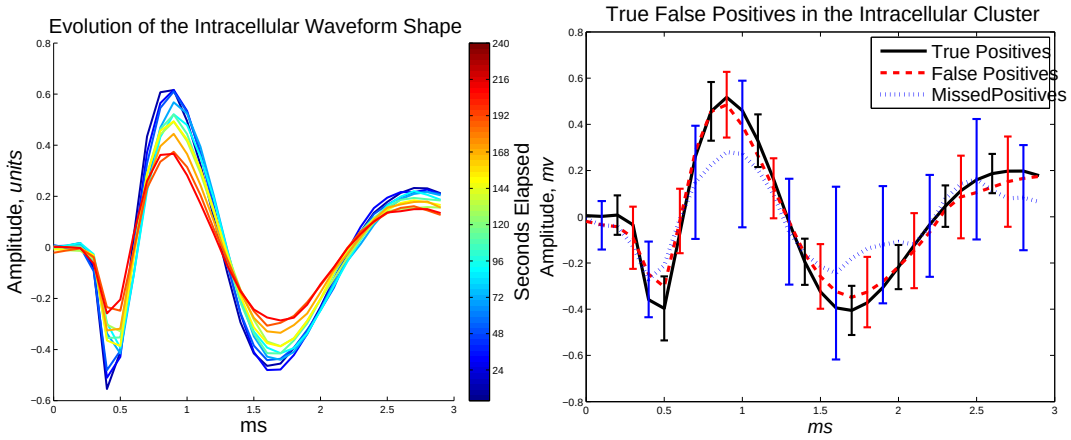
For the IC cluster, it is of interest to investigate the properties of the true positives, the false positives, as well as the IC spikes that are missed by the algorithm. Errorbar plots for these groups are shown in Figure 2b; this figure demonstrates the great similarity of the true positives and the false positives, while the missed positives (spike waveforms not detected or not in the IC cluster) have high variability and a different mean shape.

4.5 Time-Varying Waveform Adaptation

As as been demonstrated previously [18], the waveform shape of a neuron may change over time. The mean waveform over time for the intracellular neuron is shown in Figure 2a. Allowing the mean of each component to evolve over time improved single-channel performance over the 4-minute dataset, and gave similar performance in the multi-channel algorithms.

4.6 Overlapping Spike Detection

It is possible for action potentials to fire close to simultaneously so that a given window would have 2 or move action potent ions. It is possible for the algorithm to detect and fit overlapping spikes as they come in. Out of the 3593 spikes detected by the algorithm, there are 124 pairs of overlapping



(a) (b)

Figure 2: (a) Mean IC waveforms over time. Each colored line represents the mean of the waveform averaged over 24s and the color gives the elapsed time. This neuron decreases in amplitude over the period of the recording. (b) Errorbar plots of the true positives and the false positives in the IC cluster. While the false positives have slightly more variability, the mean shape for the false positives and the true positives is nearly identical. The true misses have a significantly lower amplitude as well as high variability

spikes within 1ms of one another (3.45%). An example of an overlapping waveform is shown in Figure 1b.

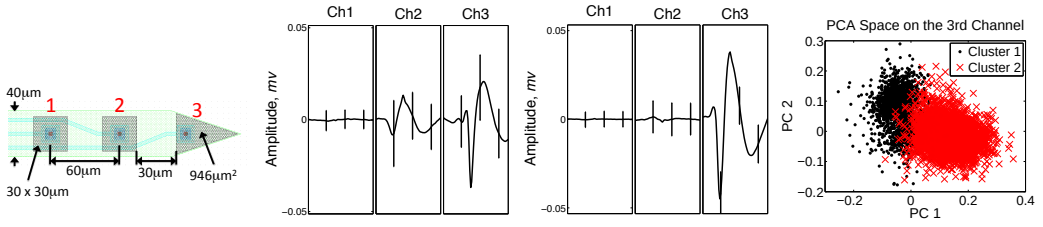
4.7 Multi-Electrode Array

In the tetrode case the waveform undoubtedly appears on all channels at once; it is possible to concatenate the channels to jointly process the data [19]. When the action potential will only appear on a subset of channels it is nice to allow the action potential to vary in a low-dimensional subset in each of the channels instead of a low-dimensional subset over all the channels. [20]

We use processed data from novel NeuroNexus devices implanted in the rat motor cortex. The data was recorded at 32.5kHz in freely-moving rats, and the data was first processed by high-pass filtering at 800Hz. The first device we consider is a set of 3 channels of data shown in Figure 4a. The neighboring electrode sites in these devices have 30 μm between electrode edges and 60 μm between electrode centers. These devices are close enough that a locally-firing neuron could appear on multiple electrode sites (cite that paper on action potential overlap), and neighboring channels warrant joint processing. The second, 8-channel device is shown in Figure **??**, and has similar properties to the first device.

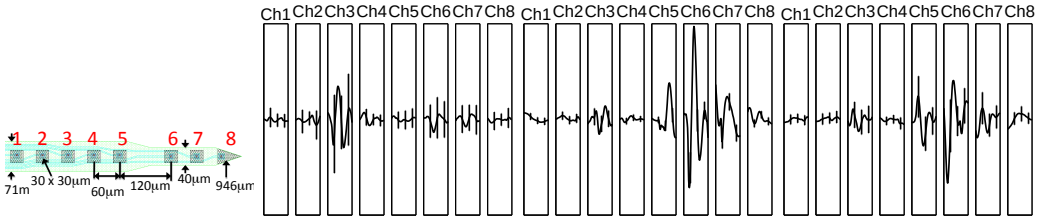
The top 2 clusters found in the first 10 minutes of data on the 3-channel device are shown in Figures 3b, 3c. The waveform in channel 3 is very similar for the waveforms in Figure 3b and 3c, and would be difficult to separate if we were analyzing each channel individually; the representation of those two clusters in PCA space on channel 3 is shown in Figure **??**. We gain the ability to distinguish the waveforms by looking at all the channels simultaneously. The top 3 clusters found in the 8-channel device can be found in Figures 4b, 4c, and 4d.

perhaps add comments about time-evolution and the false positives we avoid by using multi-channel analysis



(a) (b) (c) (d)

Figure 3: (a)3 electrode device showing local proximity of electrodes with channel indexes in large, red numbers. (b,c) Top 2 most prevalent waveforms. Each waveform shape is 2ms long. Note in (a) we have a waveform that appears on both channel 2 and channel 3, wheras in (b) the waveform only appears in channel 3. If only channel 3 was used, it would be difficult to separate the waveform in (a) and (b), as is demonstrated in Figure (d) by the representation of detected spikes on the 3rd channel in PCA space.



(a) (b) (c) (d)

Figure 4: (a)8 electrode device showing local proximity of electrodes with channel indexes in large, red numbers. (b,c,d) Top 3 most prevalent waveforms. Each waveform shape is 2ms long.

5 Discussion

in discussion:

embarrassingly parallel per 4

ignore time steps that aren't useful

let λ_i vary as a function of: (i) spike histories, (ii) stimulus, (iii) possibly baseline drift?

1 paragraph on:

1 summary

2 relate work - including Franke and Pillow

3 future work - including adaptive stimulus design & decoding

References

[1] J. F. C. Kingman. *Poisson processes*, volume 3 of *Oxford Studies in Probability*. The Clarendon Press Oxford University Press, New York, 1993. Oxford Science Publications.

[2] F. Wood, S. Goldwater, and M. J. Black. A non-parametric Bayesian approach to spike sorting. In *Proceedings of the IEEE Conference on Engineering in Medicine and Biologicial Systems*, volume 28, 2006.

[3] J.F.C. Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

[4] L.F. James, A. Lijoi, and I. Pruenster. Posterior analysis for normalized random measures with independent increments. *Scand. J. Stat.*, 36:76–97, 2009.

[5] N. L. Hjort. Nonparametric Bayes estimators based on beta processes in models for life history data. *Annals of Statistics*, 18(3):1259–1294, 1990.

[6] R. Thibaux and M. I. Jordan. Hierarchical beta processes and the Indian buffet process. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, volume 11, 2007.

[7] Ken ti Sato. *Lévy Processes and Infinitely Divisible Distributions*. Cambridge University Press, 1990.

[8] Thomas S Ferguson. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[9] A.Y. Lo. On a class of bayesian nonparametric estimates: I. density estimates. *Annals of Statistics*, 12(1):351–357, 1984.

[10] J. Pitman. Combinatorial stochastic processes. Technical Report 621, Department of Statistics, University of California at Berkeley, 2002. Lecture notes for St. Flour Summer School.

[11] R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9:249–265, 2000.

[12] H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

[13] D. M. Blei and M. I. Jordan. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144, 2006.

[14] T. P. Minka and Z. Ghahramani. Expectation propagation for infinite mixtures. Presented at NIPS2003 Workshop on Nonparametric Bayesian Methods and Infinite Models, 2003.

[15] L. Wang and D.B. Dunson. Fast bayesian inference in dirichlet process mixture models. *Journal of Computational & Graphical Statistics*, 2009.

[16] D A Henze, Z Borhegyi, J Csicsvari, A Mamiya, K D Harris, and G Buzsaki. Intracellular feautures predicted by extracellular recordings in the hippocampus in Vivo. *J. Neurophysiology*, 2000.

[17] M. S. Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 1998.

[18] Ana Calabrese and Liam Paninski. Kalman filter mixture model for spike sorting of non-stationary data. *Journal of neuroscience methods*, 196(1):159–169, 2011.

[19] Jan Gasthaus, Frank Wood, Dilan Gorur, and Yee Whye Teh. Dependent dirichlet process spike sorting. *Advances in neural information processing systems*, 21:497–504, 2009.

[20] Jason S Prentice, Jan Homann, Kristina D Simmons, Gašper Tkačik, Vijay Balasubramanian, and Philip C Nelson. Fast, scalable, Bayesian spike identification for multi-electrode arrays. *PloS one*, 6(7):e19884, January 2011.