Online Bayesian Inference for a Latent Marked Poisson Process: Applications in Spike Sorting

Anonymous Author(s)
Affiliation
Address
email

Abstract

1 Introduction

1 paragraph on the following:

1. opening
2. challenge
3. approach
4. resolution

2 Model

3 The model/introduction

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 C channels. The set of recording times lie on regular grid with interval length Δ, while $\mathbf{x}_t \in \mathbb{R}^C$ for all t. This time-series of electrical activity is driven by an unknown number of neurons and we want to... recap scientific goals. We let the number of neurons be unbounded, though only a few of the infinite neurons dominate. These neurons contribute the majority of the activity in any finite interval of time; however, as time passes, the total number of observed neurons increases (Justify?). 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 $X \equiv (x_1, \cdots, x_T)$.

3.1 Modelling a single electrode recording

There is a rich literature characterizing the spiking activity of a single neuron [], accounting for factors like nonstationarity and refractoriness. However we shall make the simplifying assumption that each neuron as stationary and memoryless, so that its set of spike times are distributed as a homogeneous Poisson process. justify? We model the neurons themselves are heterogeneous, with the ith neuron having an (unknown) firing rate r_i. Call the ordered set of spike times of the ith neuron E_i; then the time between successive elements of E_i is exponentially distributed with mean $1/r_i$. We write this as

$$E_i \sim \text{PoissProc}(r_i) \tag{1}$$

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 (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. Figure? We let $\theta^* \in \Theta$ parametrize this distribution, with neuron i having parameter θ_i^*. Whenever this neuron emits a spike, a new shape is drawn independently from the corresponding distribution. This is then offset to the time of the spike, and forms the voltage trace associate with that spike. The complete output of the neuron is the superposition of all these spike waveforms. (Figure?) We model the random spike shapes themselves as weighted superpositions of a dictionary of K basis functions $A \equiv (A_1(t), \cdots, A_K(t))$. The dictionary elements are shared across all neurons, and each is a real-valued function of time. For the ith neuron, the jth spike $e_{ij} \in E_i$, is associated with a random K-dimensional weight vector $\tilde{\mathbf{y}}_{ij}$, and the shape of this spike is given by the weighted sum $\sum_{k=1}^{K} \tilde{y}_{ijk} A_k(t)$. We let $\tilde{\mathbf{y}}_{ij}$ be Gaussian distributed, with $\theta_i^* \equiv (\mu_i^*, \Sigma_i^*)$ determining its mean and variance. Then, at any time t, the output of neuron i is

$$x_i(t) = \sum_{j=1}^{|E_i|} \sum_{k=1}^{K} \tilde{y}_{ijk} A_k(t - e_{ij}) \tag{2}$$

The total signal recorded $x(t)$ at any electrode is the superposition of the outputs of all neurons. Assume for the moment there are N neurons, and define $E = \cup_{i=1}^{N} E_i$ as the (ordered) union of the spike times of all neurons. To map elements $e_j \in E$ to those in the individual spike trains E_i, we need to introduce some more notation. Let ν_j be the neuron to which the jth element of E belongs, and let $\theta_j \equiv (\mu_j, \Sigma_j)$ be the neuron parameter associated with that spike. Thus $\theta_j = \theta_{\nu_j}^*$. Furthermore, let p_j index the position of the spike j of E in E_{ν_j}, the spike train of neuron ν_j. Thus $e_j = e_{\nu_j p_j}$. Finally, define $\mathbf{y}_j = \tilde{\mathbf{y}}_{\nu_j p_j}$. Then, we have that

$$x(t) = \sum_{i=1}^{N} x_i(t) = \sum_{j=1}^{|E|} \sum_{k=1}^{K} y_{jk} A_k(t - e_j) \tag{3}$$

where

$$\mathbf{y}_j \sim N(\mu_j, \Sigma_j) \tag{4}$$

From the superposition property of the Poisson process [1], the overall spiking activity E is a Poisson process with rate $R = \sum_{i=1}^{N} r_i$. Each event $e_j \in E$ is associated with a pair of labels, the parameter of the neuron to which it is assigned ($\theta_j = (\mu_j, \Sigma_j)$), and the weight-vector characterizing the spike shape ($\mathbf{y}_j$). We can view these as the 'marks' of a marked Poisson process E. From the properties of the Poisson process, we have that the marks θ_j are drawn i.i.d. from a measure

$$G(\mathrm{d}\theta) = \frac{1}{R} \sum_{i=1}^{N} r_i \delta_{\theta_i^*} \tag{5}$$

Note that with probability one, the neurons have distinct parameters, so that the mark θ_j associated with spike j identifies the neuron which produced it: $G(\theta_j = \theta_i^*) = P(\nu_j = i) = \frac{r_i}{R}$. Given θ_j, y_j is distributed as in equation 4. The output waveform $x(t)$ is then a linear functional of this marked Poisson process (equation (3)).

3.2 Completely random measures (CRMs)

While the previous section assumed N neurons, our recordings are actually a superposition of the outputs of an unknown number of neurons. We deal with this by taking a nonparametric Bayesian approach, and letting the number of neurons N tend to infinity. Such an approach leads to an elegant and flexible modelling framework, and has already proved successful in neuroscience applications [2]. Since only a finite number of spikes are observed in any finite interval, the total rate R must also be finite; moreover, as we described earlier, we want this to be dominated by a few r_i. A natural framework that captures these modelling requirements is that of completely random measures [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 (r_i, θ_i^*); the former characterizes the distribution over spike times, and the latter over spike shapes. Recalling equation (5), map the infinite collection of pairs $\{(r_i, \theta_i^*)\}$ to an atomic measure on Θ:

$$R(\mathrm{d}\theta) = \sum_{i=1}^{\infty} r_i \delta_{\theta_i^*} \tag{6}$$

For any subset $\mathit{\Theta}$ of Θ, the measure $R(\mathit{\Theta})$ equals $\sum_{\{i:\theta_i^* \in \mathit{\Theta}\}} r_i$. We allow $R(\cdot)$ to be random, modelling it as a realization of a completely random measure. Such a random measure has the property that for any two disjoint subsets $\mathit{\Theta}_1$ and $\mathit{\Theta}_2 \in \Theta$, the measures $R(\mathit{\Theta}_1)$ and $R(\mathit{\Theta}_2)$ are independent. This distribution over measures is induced by a distribution over the infinite sequence of weights (the r_i's), and a distribution over the sequence of their locations (the θ_i^*'s). For a CRM, the weights r_i are the jumps of a Lévy process [7], and their distribution is characterized by a Lévy measure $\rho(r)$. The locations θ_i^* are drawn i.i.d. from a base probability measure $H(\theta^*)$. As is typical, we assume these to be independent (though this is not necessary). if there's space, I can elaborate on the construction of the CRM from its Levy measure, though this is not necessary

The CRM we choose for our application is the Gamma process (ΓP); this has Lévy intensity $\rho(r) = r^{-1}\exp(-r\alpha)$. The Gamma process has the convenient property that the total rate $R \equiv R(\Theta) = \sum_{i=1}^{\infty} r_i$ is Gamma distributed (and thus conjugate to the Poisson process prior on E). 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, power-law behaviour etc.

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

$$R(\cdot) \sim \Gamma\mathrm{P}(\alpha, H(\cdot)) \tag{7}$$

$$E_i \sim \mathrm{PoissProc}(r_i) \quad i \text{ in } 1, 2, \cdots \tag{8}$$

$$\tilde{\mathbf{y}}_{ij} \sim N(\mu_i^*, \Sigma_i^*) \quad i, j \text{ in } 1, 2, \cdots \tag{9}$$

$$x_i(t) = \sum_{j=1}^{|E_i|} \sum_{k=1}^{K} \tilde{y}_{ijk} A_k(t - e_{ij}) \tag{10}$$

$$X = \sum_{i=1}^{\infty} x_i \tag{11}$$

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

$$G(\mathrm{d}\theta) = \frac{1}{R} \sum_{j=1}^{\infty} r_j \tag{12}$$

$G(\mathrm{d}\theta)$ is a random probability measure that belongs to a class called normalized random measures [4]. Importantly, a normalized Gamma process is the Dirichlet process [8], so that G is a draw from a Dirichlet process. For the j spike, given its parameter θ_j), its shape vector is drawn from a normal distribution with parameters (μ_j, Σ_j). Thus the spike parameters θ are i.i.d. draws from a Dirichlet process, while the weight vectors are draws from a DP mixture model of Gaussians [9].

With this insight, we can now marginalize out the infinite-dimensional rate vector $R(\cdot)$, and assign spikes to neurons via the Chinese restaurant process (CRP) [10]. Under the CRP, the jth 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. A final point is that for the Gamma process, the random probability measure $G(\cdot)$ is independent of the total mass $R(\Theta)$. Thus, the clustering of spikes is independent of the rate R at which they are produced. We thus have the following model equivalent to the one above:

$$R \sim \text{Gamma}(1, \alpha) \tag{13}$$
$$E \sim \text{PoissProc}(R) \tag{14}$$
$$\nu_j \sim \text{CRP}(\alpha, H(\cdot)), \quad j = 1, \cdots, |E| \tag{15}$$
$$y_e \sim N(\mu_{\nu(e)}, \Sigma_{\nu(e)}), \quad j = 1, \cdots, |E| \tag{16}$$
$$x(t) = \sum_{j=1}^{|E|} \sum_{k=1}^{K} y_{jk} A_k(t - e_j) \tag{17}$$

Unlike most applications which observe the outputs of a CRP, we observe a convolution-like functional of these outputs. This means that standard inference techniques cannot be directly applied. In section 4, we develop a novel online algorithm for posterior inference.

3.3 A discrete-time approximation

In the previous subsections, we modelled a continuous-time voltage waveform output by 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, this 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 R can be approximated by discretizing time at a granularity Δ, and assigning each interval an event independently with probability $R\Delta$. The accuracy of this approximation increases as Δ tends to 0. In our case, we have to approximate the marked Poisson process E. All we require additionally is to assign the marks θ_i and $\mathbf{y}_i$ to each event in the Bernoulli approximation. Following equations (15) and (16), the θ_i's are distributed according to a Chinese restaurant process, while each $\mathbf{y}_i$ is drawn from a normal distribution parametrized by the corresponding θ_i. The elements of dictionary $\mathbf{A}$ are discretized as well, and each element is now a finite-dimensional vector instead of a function. We set the length of each dictionary element to L, so that $\mathbf{A}$ is an K-by-L matrix. The shape of the jth spike is now a vector of length L, and is given by $\mathbf{Ay}$.

3.4 Noise and nonstationarity

We have so far not considered measurement noise: in practice the signal recorded by an electrode is the neuron output corrupted by noise. Biology. Let ϵ_t be the noise at time t, we model this as independent of the signal, additive and Gaussian. However, rather than modelling the noise as independent across time, we model it as a first-order autoregressive process. This can capture effects like the movement of electrodes during the experiment. Furthermore, rather than keeping the cluster parameters fixed, we model these as AR processes as well, capturing the evolution of the neuron shape distribution with time. Justify, eg cells dying

3.5 Modelling multielectrode recordings

Not too much to add, but I need to talk to David. I will also given a mechanistic summary of the overall discrete time generative process

4 Inference

We now address the problem of posterior inference over the latent variables gives 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 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]), while recently there has also been work on constructing deterministic approximations to the intractable posterior ([13, 14]). Our problem is complicated by two factors. The first is the convolutional nature of our observation process, where we observe a functional of the sequence of 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 that are slow and computationally expensive. Our goal, on the other hand, is ultimately to perform inference in real time, so that these batch approaches 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 this was developed for the usual situation where one has i.i.d. observations drawn from a DPMM. We generalize this algorithm to our observation process, and to account for the time-evolution of the cluster parameters and the Markov nature of the observation noise.

At a high level, at time t, our inference algorithm maintains a set of the times of spikes underlying the observations until time t. It also maintains the identities of the neurons those spikes were assigned to, as well as the weight vectors determining the shapes of the associated spike waveforms. Besides these point estimates, we also maintains a set of posterior distributions $q_{it}(\theta_i^*)$ where i spans over the set of neurons associated with the spikes seen so far. 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 observation at time $t+1$. We subtract this from the actual observation at time $t+1$, and treat the residual $\tilde{x}_{t+1}$ as an observation from a DP mixture model. Given this, our algorithm then makes a hard decision about whether a spike is present, whether that spike belongs to one of the earlier neurons or a new neuron, and what the shape of the associated spike waveform is. We use this to recursively update the distribution over neuron parameters. We describe this recursive algorithm below.

$$\tilde{x}_t = x_t - \sum_{i=1}^{L} \mathbf{A}\mathbf{y}_{t-i} \tag{18}$$

Given the spike statistics from earlier times, as well as the distribution over parameters, our algorithm decides whether there is a spike underlying the observations at time t, which cluster it is assigned to, and what the shape of that spike is. We simultaneously update the distribution over parameters of clusters.

Recall that each spike waveform spans L time bins. Lett z_t indicate whether or not a spike is present at time t, and with $\mathbf{y}_t$ giving its shape. Equation (18) gives the residual at time t, we must first decide whether or not there is a spike underlying this residual. By Bayes' rule,

$$P(z_t = 1|\tilde{x}_t) \propto P(z_t = 1, \tilde{x}_t) = \sum_{\gamma_t=1}^{C_t} P(\tilde{x}_t, \gamma_t | z_t = 1) P(z_t = 1) \tag{19}$$

$$P(\tilde{x}_t|\gamma_t, z_t = 1) = \int_{\Theta} P(\tilde{x}_t|\theta_t) q_{\gamma_t}(\theta_t|\gamma_t) \mathrm{d}\theta_t \tag{20}$$

Having calculated $P(z_t = 1|\tilde{x}_t)$, we decide that there is a spike present if this exceeds a threshold of 0.5, otherwise, we set $z_t = 0$. In the event that we decide that a spike is present (after marginalizing over all possible cluster assignments), we simplify matters by assigning the spike to the cluster with highest posterior probability. Similarly, we set the weight vector $\mathbf{y}_t$ to its MAP value. Given these two, we update the posterior distribution over parameters of the cluster recursively calculating q_{t+1}.

5 Results

5.1 Performance on Partial Ground Truth Data

5.2 Time-Varying Waveform Adaptation

5.3 Overlapping Spike Detection

5.4 Michigan Data

5.5 dec draft stuff

5.6 Publicly Available Data

Experiments were performed on a subset of the publicly available[1] dataset described in [**?**]. 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.

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 10kHz sampling rate and preprocessed with a high-pass filter at 800 Hz.

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 .5ms of the detected spike. We define the cluster with the greatest number of intracellular spikes as a the ”IC cluster.” Figure 1a shows the performance on the intracellular cluster versus competing methods. The single-channel experiments were all run on channel 2 (the results were nearly identitcal for all channels). The spike detections for the offline methods were given by using a threshold at 3 times the noise standard deviation [**?**]. (Need to do more comparisons and descriptions) and windowed at a size $P = 30$ (check if P matches with Vinayak notation). The action potential window was set at 30, and PCA was used to reduce the space to K=3 for the experiments. The results were very similar for K=2, K=3, and K=4. There were 3742 spikes detected with the threshold, and 753 of those corresponded to the intracellular spikes.

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 3.2GHzIntel Core i5 machine with 6GB of memory. We show the inferred y_k fordetected spikes in the 2-PCA space in Figure 1b.

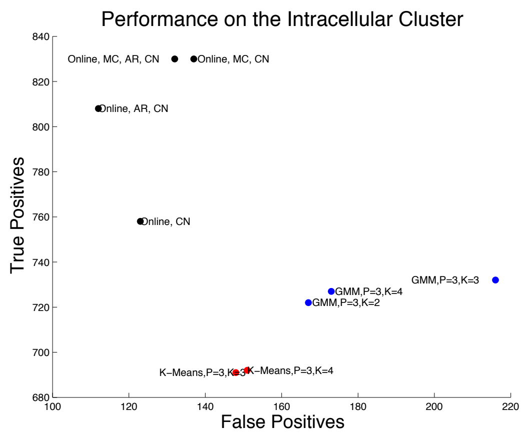
As as been demonstrated previously (cite paninski), the waveform shape of a neuron may change over time. The mean waveform over time for the intracellular neuron is shown in Figure 2a. For the IC cluster, it is of interest to investigate the properties of the true positives, the false positives, as well 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.

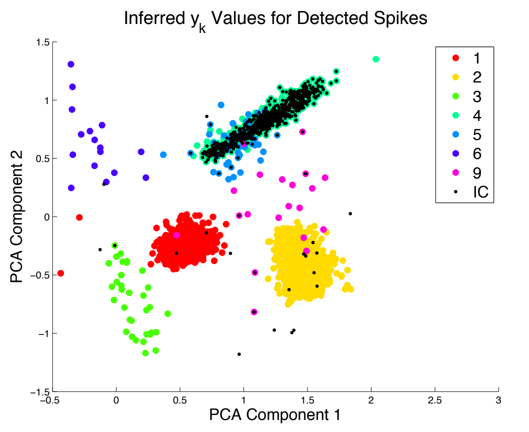
To get an idea of the robustness of the algorithm, we split the data into random segments of 2minutes, and ran the algorithms on that dataset. Add plot and information.

6 Multi-Electrode Sensors

In the tetrode case the spike undoubtedly appears on all channels at once. Often, people will just concatenate the channels to process the data (eg. cites). 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

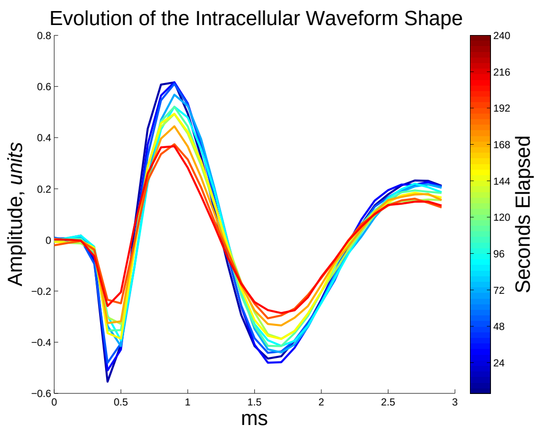
[1]http://crcns.org/data-sets/hc/hc-1/

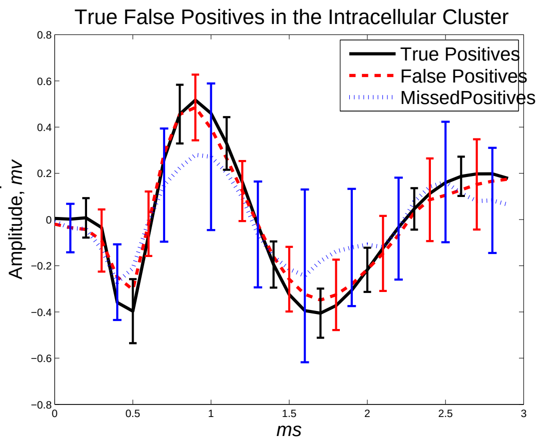




(a) (b)

Figure 1: Results on the d533101 dataset. (a) This shows the number of true positives versus the number of false positives. Our methods do better, yay! (b) Results on the d533101 dataset from the online algorithm with an AR mean





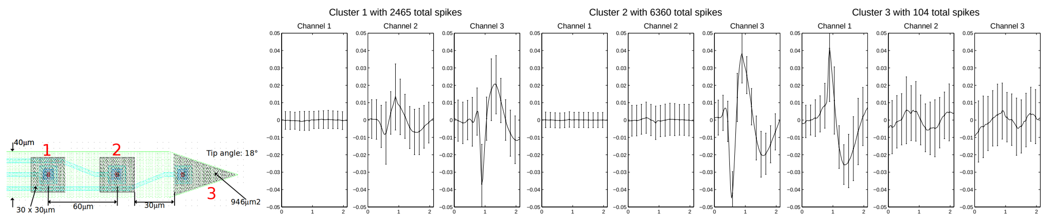
(a) (b)

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

in each of the channels instead of a low-dimensional subset over all the channels. This is especially important when the appropriate dictionary is not known beforehand.

We use processed data from novel NeuroNexus devices. (Need to give a description of this). The first device we consider is a set of 3 channels of data shown in Figure 3a. 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 top 3 clusters found in the first 10 minutes of data are shown in Figure (add in a bit).



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

Figure 3: Device used. Channels in large, red numbers.

7 Discussion

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.