Online Neural Sequence Detection with Hierarchical Dirichlet Point Process

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural sequence detection plays an important role in neuroscience researches. Recent impressive works utilize convolutive nonnegative matrix factorization and Neyman-Scott process to solve this problems. However, they still face with two limitations. Firstly, they accommodate the entire dataset into memory and perform iterative updates of multiple passes, which can be inefficient when dataset is large or grows frequently. Secondly, they rely on the prior knowledge of the number of sequence types, which can be impractical with real data when the future situation is unknown. To tackle these limitations, we propose a hierarchical Dirichlet point process model for efficient neural sequence detection. Instead of computing with the entire data, our model can sequentially detect sequences in an online unsupervised manner with Particle filter. Besides, the Dirichlet prior enables our model to automatically introduce new sequence types on the fly as needed, thus avoiding specifying the number of types in advance. We manifest these advantages on synthetic data and real world recordings from songbird higher vocal center and rodent hippocampus.

1 Introduction

Neural sequences are a set of sequential firing neurons occurring repeatedly over time. Examples of neural sequences appear in working memory [6, 10], motor production [9], and memory replay [4]. Usually, methods that using linear dimensionality reduction of large neural populations may be inappropriate to handle such sequences since they form a high-dimensional structure and cannot be efficiently summarized by a low-dimensional representation [20, 15]. Besides, the relationships between neural sequences and behavior are not always so regular and predictable [27], thus a unsupervised method to identify neural sequences is required.

While a great process in neural sequence detection has been made by recent works [19, 15, 25], they have two drawbacks when it comes to the contexts of online learning. Firstly, they are ill-suited to accommodate new data or handle large dataset. Many real world spike train data involve plenty of hours' or days' signal recording that sometimes may be hard to accommodate entirely in the memory. The inference algorithms of these works (e.g., coordinate descent, Gibbs sampling, etc.) maintain the entire data and perform iterative updates of multiple passes, which may be a waste of both computational and storage resources. Secondly, these works require the number of sequence types to be specified in advance, which is inappropriate when future situation is unknown.

To address these problems, we propose a hierarchical Dirichlet point process (HDPP) with an efficient online inference algorithm. Briefly, HDPP considers the observed spikes as a mixture of neural sequences, and those spikes belonging to different neural sequences are modeled via different nonhomogeneous Poisson processes, for which the intensity of each process varies across neurons and times. Likewise, the detected neural sequences are also a mixture of certain types of sequences. The

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

temporal dynamics of these sequences are captured by several Hawkes processes [11], where each process's distinctive temporal relation will help HDPP to disambiguate different types of sequences with highly overlapping neurons. Besides, the prior of each sequence type is given by a Dirichlet distribution, so that either the number of sequences or the number of sequence types could vary during inference.

We derive a Particle filter [2] to learn the HDPP model in an online manner: (1) learn the model with a single pass of the data; (2) make the decisions on a regular basis without requiring future data [1]. In particular, we use a collection of particles to sequentially approximate posterior, taking in one spike at a time to make the update. We also develop specific mechanisms to merge similar sequences and prune inaccurate ones during inference, thus further improving the performance. We evaluate the proposed model on synthetic data as well as real world data from songbird higher vocal center [15] and rodent hippocampus [3, 7, 8]. Results show empirically the proposed model can properly identify neural sequence and be robust to noise in a single pass over data.

The contributions of our work include: (1) We propose a novel hierarchical Dirichlet point process for neural sequence detection, and different to prior works [14, 15, 25], our model does not specify the number of sequence type in advance, while determines adaptively during inference. (2) We propose an efficient online inference algorithm with the ability to scale up to large dataset with near constant processing time per spike. (3) We use a group of Hawkes processes to capture the temporal dynamics of different types of sequences, which enables our model to distinguish sequences with highly overlapping neurons.

2 Related works

Previous studies about neural sequences detection can be divided into three classes: supervised, bottom-up unsupervised, and top-down unsupervised [28].

Supervised methods Averaging spikes times over multiple trials is the simplest and most used supervised method to identify neural sequences. This approach requires simultaneous sensory cues or behavioral actions recording, and uses a template matching procedure to discover exact timestamps of each sequence [18, 12]. However, as previous mentioned, the relationship between neural activity and behavior are not always reliable [27], thus this approach may be useless under certain circumstances.

Bottom-up unsupervised methods Bottom-up unsupervised methods [21, 22, 23, 20] usually identify neural sequences via a variety of statistics between pairs of neurons (e.g., cross-correlations). These methods may be beneficial when involving small numbers of neurons [28], but they have high computational complexity and low statistical power when dealing with large number of neurons.

Top-down unsupervised methods Top-down unsupervised methods can be defined as a kind of methods which discover a sequence from the perspective of the full population and is robust to noise at the level of individual neurons [28]. Maboudi et al. [14] identifies neural sequences by visualizing the transition matrix of an Hidden Markov Models. Peter et al. [19] and Mackevicius et al. [15] use convolutive nonnegative matrix factorization (ConvNMF) with different regularization to factorize the neural data into a couple of neural factors and temporal factors, where each neural factor encodes a type of neural sequence and each temporal factor indicates the times of sequences. Williams et al. [25] models neural sequences as a Neyman-Scott process (PP-Seq), which is a spatial-temporal point process originally designed to discover clusters in the galaxy. Each sequence in this model corresponds to a latent cause which generates spikes as the offspring events.

3 Background

A basic probabilistic model of continuous-time event stream is temporal point process whose realization can be represented as an event sequences $\{x_i\}_1^N = \{t_1, t_2, ...\}$ with times of occurence $t_i \in [0, T]$. A way to characterize temporal point processes is via intensity function $\lambda(t)$, for which $\lambda(t)$ equals the expected instantaneous rate of happening next event given the history. The functional form of $\lambda(t)$ varies among different phenomena of interests. In a *homogeneous* Poisson process (PP), $\lambda(t)$ is assumed to be a constant over time, while in a *nonhomogeneous* Poisson process, $\lambda(t)$ is time-varying and independent of the history.

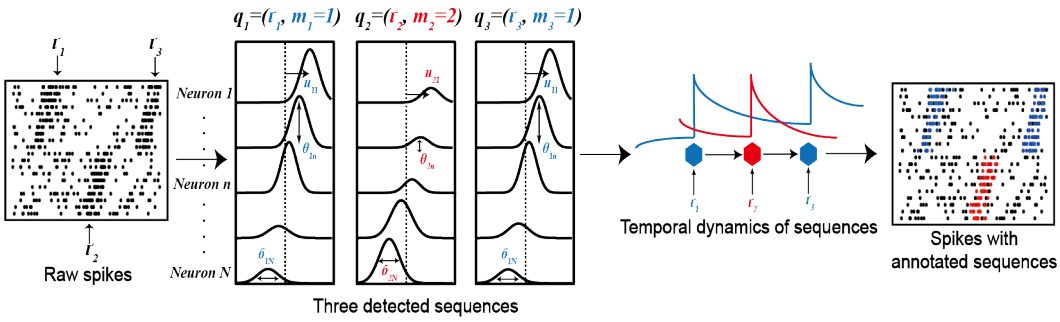

Figure 1: A case with $K = 3$ sequences and $M = 2$ types of sequences is shown. The proposed model considers observed spikes as a mixture of neural sequences, where spikes within a sequence can be modeled via a nonhomogeneous Poisson process that catches their firing rates. Likewise, the detected sequences are also a mixture of certain types of sequences, where their temporal dynamics are captured by several Hawkes processes.

Hawkes process A Hawkes process (HP) [11] is a self-exciting nonhomogeneous Poisson process in which past events from the history have influences on the intensity of current and future events. Such influences are positive but decay exponentially with time:

$$\lambda(t) = \lambda_0 + \alpha \sum_{t_i < t} \exp(-\delta(t - t_i)), \tag{1}$$

where λ_0 is the base intensity independent of the history, α is the kernel parameter, and δ is the decay rate of that influence. An important extension of hawkes process is to incorporate additional information of each event, for example, an associated event "label" k_i with each event $x_i = (t_i, k_i)$. As a consequence, we can define a Hawkes process as a sum of independent Hawkes processes from different types based on the superposition principle: $\lambda(t) = \sum_{k=1}^{K} \lambda_k(t)$, where $\lambda_k(t)$ is the intensity function from k-th category.

Dirichlet Hawkes process Dirichlet Hawkes process (DHP) [5, 16] is a method to naturally handle infinite mixture clusters with temporal dependency. As opposed to traditional parametric models, DHP allows the number of clusters to vary over time. It uses a Hawkes process to model the rate intensity of events (e.g., the times of neural sequences), while Dirichlet process captures the diversity of event types (e.g., the clusters of neural sequences). Similar to Dirichlet process, a DHP, typically denoted by DHP(λ_0, θ) is characterized by a concentration parameter λ_0 and a base distribution θ. It uses the samples $\theta_{1:K}$ from base distribution θ and a collection of triggering kernel functions $\phi_{\theta_i}(t, t_i)$ as the parameters for modeling K different event types. Then, the events (t_i, θ_i) from DHP(λ_0, θ) can be simulated as follows:

1. Sample t_1 from PP(λ_0) and θ_1 from θ.
2. For $n > 1$, sample t_n from HP$(\lambda_0 + \sum_{k=1}^{K} \lambda_k(t_n))$
   - Sample a new θ_{K+1} from θ with probability: $\lambda_0/(\lambda_0 + \sum_{k=1}^{K} \lambda_k(t_n))$
   - Reuse previous θ_k with probability: $\lambda_k(t_n)/(\lambda_0 + \sum_{k=1}^{K} \lambda_k(t_n))$

where $\lambda_k(t_n) = \sum_{i=1}^{n-1} \phi_{\theta_i}(t_n, t_i)\mathbb{I}(\theta_i = \theta_k)$ is the intensity function of a Hawkes process for past events with type k.

4 Hierarchical Dirichlet Point Process of Neural Sequences

We model the neural sequences as a hierarchical Dirichlet point process (HDPP). We use a Dirichlet nonhomogeneous Poisson process as the prior for observed spikes, while use a Dirichlet Hawkes process as the prior for neural sequences. As in Fig. 1, the goal of HDPP is to sequentially identify sequences from observed spikes, then partition these sequences into different types based on both temporal dynamics and included neurons.

Consider a continuous-time spike train data with N neurons, its observed spikes can be denoted by a set of S labeled events $x_s = (t_s, n_s, k_s)$ referring to the spike time, $t_s \in [0, T]$, neuron, $n_s \in \{1, ..., N\}$ and indicator for neural sequence, $k_s \in \{1, ..., K\}$. Similarly, neural sequences in this dataset are also a collection of tuples $q_k = (\tau_k, m_k)$ referring to the time $\tau_k \in [0, T]$ and indicator for sequence type $m_k \in \{1, .., M\}$.

According to the definition of Dirichlet Hawkes process, for a spike x_s occurring at time t_s its probability of belonging to the k-th neural sequence is:

$$P(k_s = k|t_s) = \frac{g_k(t_s)}{g_0 + g_{\text{noise}} + \sum_{k=1}^{K} g_k(t_s)}, \tag{2}$$

where g_0 is the concentration parameter, g_{noise} is the firing rate for background spikes, and $g_k(t)$ is the intensity function of sequence q_k, controlling the temporal dynamics of related spikes. Likewise, the probability of assigning spike x_s to either background noise or a new sequence is given by:

$$P(k_s = k|t_s) = \begin{cases} \frac{g_{\text{noise}}}{g_0 + g_{\text{noise}} + \sum_{k=1}^{K} g_k(t_s)} & \text{for } k = 0, \\ \frac{g_0}{g_0 + g_{\text{noise}} + \sum_{k=1}^{K} g_k(t_s)} & \text{for } k = K + 1. \end{cases} \tag{3}$$

After a neural sequence $q_{k>0}$ is chosen, we assign it to an exist type m_k or create a new type on the fly as needed with probability:

$$P(m_k = m|\tau_k) = \begin{cases} \frac{\lambda_m(\tau_k)}{\lambda_0 + \sum_{m=1}^{M} \lambda_m(\tau_k)} & \text{for } m \in \{1, ..., M\}, \\ \frac{\lambda_0}{\lambda_0 + \sum_{m=1}^{M} \lambda_m(\tau_k)} & m = M + 1. \end{cases} \tag{4}$$

where $\lambda_m(\tau)$ is the intensity function that captures temporal dynamics of sequences in type m. Note that we assume $g_k(t)$ and $\lambda_m(\tau)$ take different functional forms, and $g_{kn}(t)$ can be viewed as the firing rate of neuron n caused by sequence q_k. Following the work in [25], we represent $g_{kn}(t)$ via a Gaussian form as:

$$g_{kn}(t) = \theta_{m_k n} \cdot \beta_k \cdot \mathcal{N}(t|\tau_k + \mu_{m_k n}, \sigma_{m_k n}). \tag{5}$$

Here $\theta_{m_k n}$, $\mu_{m_k n}$ and $\sigma_{m_k n}$ are parameters of each sequence type m_k, indicating the weight, offset, and width of neuron n's firing rate. β_k is the amplitude of sequence q_k, denoting the expected number of spikes induced by q_k [25]. For $\lambda_m(\tau)$, we opt the similar form as (1), with parameter $\alpha_m > 0$ controls the self-excitation of previous sequences, hyperparameter $\delta > 0$ controls the decay and hyperparameter η denotes the expected time interval of sequences:

$$\lambda_m(\tau) = \alpha_m \sum_{\tau_k} \exp(-\delta(\tau - \tau_k - \eta))\mathbb{I}(\theta_k = \theta_m). \tag{6}$$

Based on the definition of $g_k(t)$ and $\lambda_m(\tau)$, we regard that the temporal dynamics of spikes and neural sequences are captured by a nonhomogeneous Poisson process and a self-exciting Hawkes process, respectively. For sequence type parameters α_m, $\{\theta_{mn}\}_{n=1}^{N}$ and $\{(\mu_{mn}, \sigma_{mn})\}_{n=1}^{N}$, we assume they follow a Gamma distribution, a Dirichlet distribution, and a normal-inverse-chi-squared distribution [17, 25] for every neuron and type. For amplitude β_k and background spike rate g_{noise}, we assume they obey a Gamma distribution. Then, we can describe the proposed hierarchical Dirichlet point process in a generative way as:

1. For neural sequence q_k, $\tau_k \sim \text{HP}(\lambda_0 + \sum_{m=1}^{M} \lambda_m(\tau_{k-1}))$, $\beta_k \sim \text{Gamma}(a_1, b_1)$.
2. Sample sequence type m_k using (4).
   - If $m_k = M + 1$: $\theta_{M+1} \sim \text{Dir}(\theta_0)$, $(\mu_{M+1}, \sigma_{M+1}) \sim \text{NI}\chi^2(\mu_0, \kappa_0, \upsilon_0, \sigma_0)$, $\alpha_{M+1} \sim \text{Gamma}(a_2, b_2)$ and $M = M + 1$.
3. Sample spikes induced by sequence q_k: $s_k = \text{Poisson}(\beta_k)$, $\{n_s\}_{s=1}^{s_k} \sim \text{Categorical}(\theta_{m_k})$ and $\{t_s\}_{s=1}^{s_k} \sim \mathcal{N}(\tau_k + \mu_{m_k n_s}, \sigma_{m_k n_s})$
4. Sample background spikes during time interval $\Delta t = \tau_k - \tau_{k-1}$: $g_{\text{noise}} \sim \text{Gamma}(a_0, b_0)$, $s_0 = \text{Poisson}(g_{\text{noise}} \cdot \Delta t)$, $\{n\}_{s=1}^{s_0} \sim \text{Categorical}(\theta_{\text{noise}})$ and $\{t_s\}_{s=1}^{s_0} \sim \text{PP}(g_{\text{noise}})$.

where θ_0 is a hyperparameter parameter for Dirichlet prior, $a_0, b_0, a_1, b_1, a_2, b_2$ are hyperparameter parameters for Gamma prior, and $\mu_0, \kappa_0, \upsilon_0, \sigma_0$ are hyperparameter parameters for normal-inverse-chi-squared prior.

5 Inference

Given spikes $\{x_s = (t_s, n_s)\}_{s=1}^{S}$, as previous discussed, the model aims to sample neural sequences indicator k_s for each spike. Using this insight, we can use a set of particles to sequentially approximate posterior $P(k_{1:s}|t_{1:s}, n_{1:s})$, while sampling indicator m_k for each sequence.

We derive a Particle filter approach for our proposed model. This inference algorithm exploits the temporal dependencies in the observed spikes to sequentially sample the latent variables. To efficiently sample from the posterior distribution, a Particle filter keeps track of an approximation of $P(k_{1:s-1}|t_{1:s-1}, n_{1:s-1})$, and updates it to have an approximation for $P(k_{1:s}|t_{1:s}, n_{1:s})$ when receiving a new observed spike. The approximation is evaluated by a set of weighted particles $\{p\}^P$, where the weight of the each particle reflects how well the observed spikes are explained.

For each particle p, its weight is defined as the ratio between true posterior and importance distribution: $w_s^p = \frac{P(k_{1:s}|t_{1:s},n_{1:s})}{Q(k_{1:s}|t_{1:s},n_{1:s})}$. As the work in [1, 5] did, we choose $P(k_s|k_{1:s-1}, t_{1:s}, n_{1:s})$ to be the importance distribution. Next, the unnormalized weight w_s^p can be updated as:

$$w_s^p \propto w_{s-1}^p \cdot P(n_s|k_{1:s}, n_{1:s-1}). \tag{7}$$

Here, let $C^{m_{k_s}\backslash x_s}$, $C_{n_s}^{m_{k_s}\backslash x_s}$ denote the spike counts of sequence type m_{k_s} and spike counts of neuron n_s in sequence type m_{k_s}, both excluding x_s. The likelihood $P(n_s|k_{1:s}, n_{1:s-1})$ is given by a Dirichlet-Categorical conjugate relation [16]:

$$\frac{C_{n_s}^{m_{k_s}\backslash x_s} + \theta_0}{C^{m_{k_s}\backslash x_s} + N\theta_0}. \tag{8}$$

After sampling indicator k_s from posterior $P(k_s|t_s, n_s, \text{rest})$ for each spike, we then sample the type indicators m_k from posterior $P(m_k|\tau_k, X_k, \text{rest})$. The posteriors of both indicators are determined by the likelihood of the spike/sequence assigned to a given sequence/sequence type and by the temporal prior under Dirichlet nonhomogeneous Poisson process (2)(3) or Dirichlet Hawkes process (4):

$$P(k_s|t_s, n_s, \text{rest}) \propto P(n_s|k_s, \text{rest}) \cdot P(k_s|t_s, \text{rest}), \tag{9}$$

$$P(m_k|\tau_k, X_k, \text{rest}) \propto P(X_k|m_k, \text{rest}) \cdot P(m_k|\tau_k, \text{rest}), \tag{10}$$

where *rest* represents all other latent variables with $x_{1:s-1}$, $X_k = \{n_s : k_s = k\}$ is the neurons in k-th sequence. The exact form of two posteriors are given in Supplement A.

Updating parameters There are three groups of parameters need to be updated after attributing each spike to a neural sequence: sequence type parameters $\Theta_m = (\theta_m, \alpha_m, \{\mu_{mn}\}_{n=1}^N, \{\sigma_{mn}\}_{n=1}^N)$, sequence parameters $\Theta_k = \{\tau_k, \beta_k\}$ and background parameters $\Theta_\emptyset = g_{\text{noise}}$. Following the literature in Particle filter devoted to the estimation of a online parameter [2], for most parameters, we perform a single Gibbs sampling step to sample from their closed-form posterior under conjugate relation. Details about derivations are in Supplement B.

Merge and Prune Generally speaking, neural sequences created during this sequential inferring procedure are often truly needed, as the decisions of adding new sequences are based on the accumulated knowledge from past. However, there is still chance that some sequences generated at early samples would become incorrect and that multiple sequences may be close in time. Thus, we introduce a mechanism to merge similar sequences and prune incorrect ones.

The similarity of two sequences q_k and $q_{k'}$ can be defined in terms of the time difference between τ_k and $\tau_{k'}$, as $d(k, k') = |\tau_k - \tau_{k'}|$. We will merge sequences q_k, $q_{k'}$ and their associated sufficient statistics when $d(k, k') < \Delta_d$. We also check the variance of firing times for each neuron in every sequence. That is, we reserve the spikes which fire consecutively in the same neuron n: $\text{Var}(\{t_s\}_{s=1}^{s_n}) < \epsilon$, while remove those appear to be sparse distributed. Besides, we will remove an entire sequence if it has too few spikes after pruning. Note that there is no need to perform such merge or prune at every iteration. Since the operation of merge sequences takes $O(K)$ and computing variance for each sequence takes $O(s_n \cdot N)$, we propose to examine similarity and variance at a fix interval $O(i \cdot T)$ and only for the current sampled sequence.

The overall pseudocode for inference is presented in Algorithm 1.

Algorithm 1: Particle filter for HDPP

Input: Spikes $\{x_s = (t_s, n_s)\}_{s=1}^{S}$, hyperparameters and threshold ϵ.
Initialize particle weight w^p to $\frac{1}{P}$ for all $p \in \{1, .., P\}$;
for *each spike* $(t_s, n_s), s = 1, ..., S$ **do**
  for $p \in \{1, ..., P\}$ **do**
    Sample sequence indicator k_s from (9);
    Sample sequence type indicator m_k from (10);
    Update the parameters Θ_m, Θ_k, and $\Theta_\emptyset$;
    Update the particle weight by (7);
  end
  Normalize particle weight;
  if $\|w^p\|_2^{-2} < \epsilon$ **then**
    Resample particles
  end
end

6 Experiments

6.1 Synthetic Data

We create three kinds of synthetic data to investigate the robustness to noise, the effectiveness of temporal dynamics, and the ability to adaptively learn the number of sequence types, respectively, of HDPP: (1) Two types of sequences with background noise. (2) Two types of sequences occur side-by-side with highly overlapping neurons. (3) The number of sequence types varies with time.

Robustness to background noise. Figure 2(a) presents the first case of synthetic data, which demonstrates the robustness of HDPP. This data have $N = 60$ neurons and $T = 30$ seconds. The raw spikes (Fig. 2(a) left panel) shows no visible neural sequences. However, sequences are revealed by sorting the neurons with the learned weight, type, and offset of this sequence. Besides, background spikes can also be correctly identified as noise.

Distinguishing overlapping sequences. Figure 2(b) shows a much more complex case where two sequences with overlap populations of neurons occur side-by-side. This data have $N = 60$ neurons and $T = 20$ seconds. Usually, it's hard to distinguish these sequences from the perspective of neurons. But with the help of temporal dynamics, HDPP could be able to capture sequences' distinctive intensity functions (Fig. 2(b) top panel), and disambiguate them to different types. Note that the intensities at the very beginning are missing since HDPP resamples a motif indicator for the first sequence whenever it receives a spike and those empty motifs are dropped as well as their recorded intensities.

Learning the number of sequence types automatically. Finally, Fig. 2(c) shows HDPP can adaptively determine the number of sequence types which grows with time. This data have $N = 90$ neurons and $T = 90$ seconds. From both sorted results (Fig. 2(c) right panel) and learned intensities (Fig. 2(c) top panel), we can see that HDPP introduces these types (blue, red and green) sequentially.

Further, we also test the robustness of HDPP quantitatively with area under reciever operating characteriztic (ROC) curves as a metric. We simulate a dataset with $M = 1$ sequence type and $N = 100$ neurons, where we vary the background spikes g_{noise} to quantitatively measure the robustness. From the results of area under ROC in Fig. 2(d), HDPP achievess promising results as we increase the noise levels.

6.2 Performance with Neural Recordings

6.2.1 Songbird Higher Vocal Center Recording

For real world data, we first apply our model to the deconvolved spike train from functional imaging data recorded in songbird HVC during singing [15][1]. One of the core features of this data is the

[1] `http://github.com/FeeLab/seqNMF`

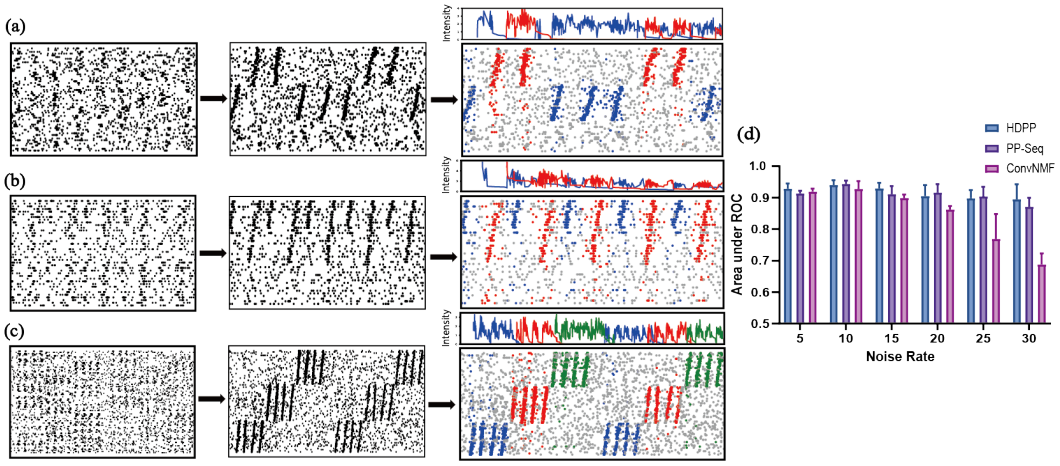

Figure 2: Synthetic data. (a) Two types of sequences with background noise. Raw spikes (left), neurons re-sorted by HDPP (middle), sequences annotated by HDPP (right), and corresponding intensities captured by Hawkes processes (top). (b) Two types of highly overlapping sequences occur side-by-side with background noise. (c) Three types of sequences appear sequentially over time. (d) HDPP is robust to background noise. Comparison of HDPP, PP-Seq, and convNMF to identify sequences under different levels of background noise.

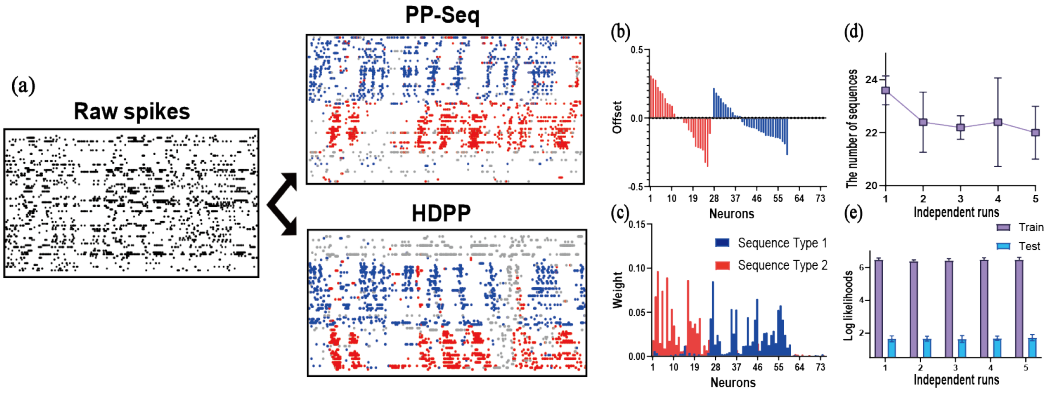

Figure 3: Songbird Higher Vocal Center data. (a) Raw spike train (left) and comparison of detected sequences by HDPP (bottom) and PP-Seq (top). (b)(c) Visualization of learned offsets and weights for sequence type 1 (blue) and sequence type 2 (red). (d) The number of detected sequences over five independent runs. (e) The log-likelihoods of training and testing sets over five independent runs.

high variability of song, thus making it difficult to identify neural sequences by a template matching procedure.

Comparison of HDPP and PP-Seq. Figure 3(a) visualizes the results from HDPP and PP-Seq[2], where neurons are sorted based on each sequence's type m_k, offset $\{\mu_{mn}\}_{n=1}^N$, weight $\{\theta_{mn}\}_{n=1}^N$, and HDPP is able to identify similar sequences as PP-Seq. Figure 3(b)(c) visualize the learned offset and weight from HDPP. For offset $\{\mu_{mn}\}_{n=1}^N$, there is a clear time shift among different neurons ($\pm$ 0.25s for type 1 and $\pm$ 0.3s for type 2), and for weight $\{\theta_{mn}\}_{n=1}^N$, the two sequence types have their own preference for neurons (neuron 27~62 for type 1 and neuron 1~26 for type 2).

Converging to similar parameter ranges. Figure 3(d) presents the total number of detected sequences over five independent runs (different random seed for initiation), and we can see that these numbers are close over five runs. Similarly, Fig 3(e) shows the log-likelihoods of training and testing sets (partitioned by method in [25]), which also remain stable over these runs. Therefore, Fig. 3(d)(e) indicate HDPP can converge to similar estimates.

[2] `https://github.com/lindermanlab/PPSeq.jl`

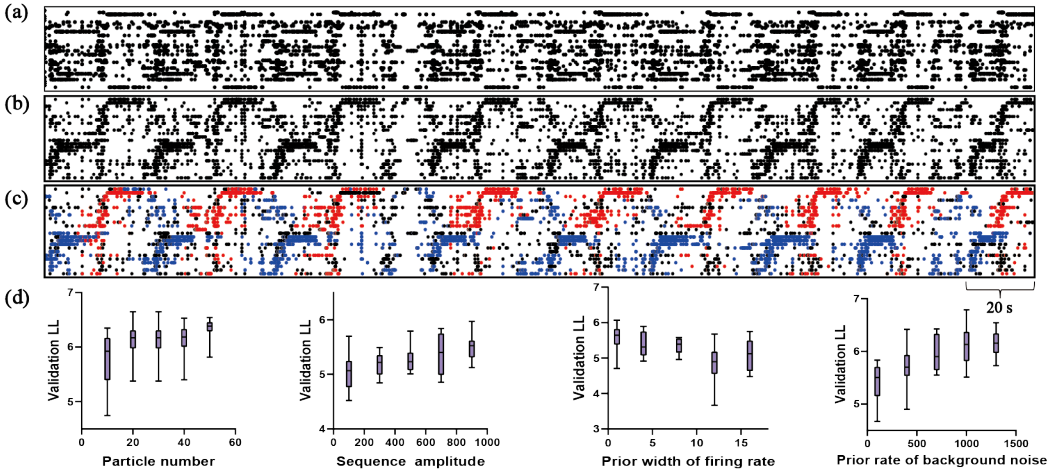

Figure 4: Rodent hippocampal recording. (a) Raw spike train ($T \approx 4.6$ minutes is shown). (b) Neurons sorted by HDPP. (c) Two types of sequences identified by HDPP. (d) The validation log-likelihoods on four key hyperparameters. Each boxplot involves five different values of a hyperparameter, while other hyperparameters are set to fix values.

6.2.2 Rodent Hippocampal Recording

We then test our model on a more complex spike train data from a rat's hippocampal recording when it performs repeating runs down a wooden 1.6m linear track [3, 8, 7][3]. The session we use is "Achilles 10252013", whose duration is $T = 2068$ seconds. As in [8], we expect to observe two sequences encoding the two opposite running directions on this liner track. In order to have a good knowledge of hyperparameter settings, we perform cross-validation via a "speckled holdout pattern" strategy, which was previously used in PCA [29], and has recently been applied to neural data [25, 26]. We then evaluate the performance of HDPP by calculating marginal likelihood on the held-out set.

Visualizing detected sequences and validation log-likelihoods. Figure 4(a)-(c) shows the raw spikes, sorted results and two type of sequence annotated by HDPP, where we can clearly observe two groups of place coding neurons revealed by the sequences. Figure 4(d) presents the validation log-likelihoods of four key hyperparameters in HDPP, while other parameters are either less important to the performance or easy to locate the optimal range. Not surprisingly, without taking into account of efficiency, a larger number of particles has a positive affect on the performance of the model, but the improvement is not obvious during certain range (e.g. 20~40). What's more, HDPP on this data prefers larger sequence amplitude β_k, smaller prior width of firing rate σ_m, and higher prior rate b_0 in gamma distribution, which instead means smaller noise firing rate g_{noise}.

6.3 Scalability to Large Dataset

In order to scale with large datasets, the online inference algorithm is supposed to process each spike in an expected constant time. In other word, the time cost of either calculating intensity functions or updating parameters should not grow with the number of sequences detected so far. In HDPP, we can note that the influences of past sequences decay exponentially, using this insight, we can reduce the computational load by ignoring those past sequences far away from current time. More specifically, given a time window Δt, we maintain an active list of sequences $\{q_k\}_{k=1}^{K}$ where $t - \tau_k \leq \Delta t$. We set the particle number to 20 and empirically verify the expected time cost per 1000 spikes in Fig. 5 based on the dataset from a rat's hippocampal recording, which has hundreds of thousands of spikes in a couple of minutes.

[3] `http://crcns.org/data-sets/hc/hc-11`

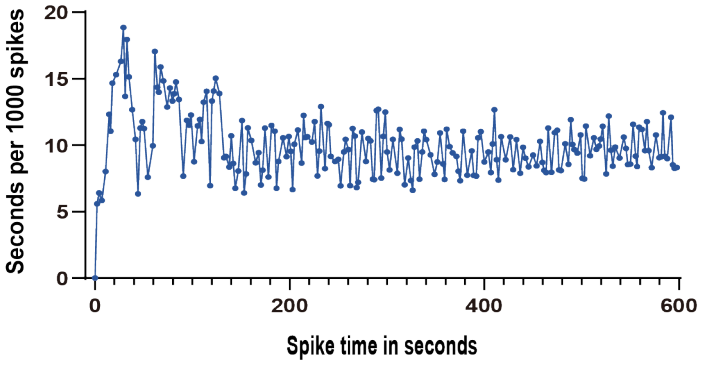

Figure 5: Time cost per 1000 spikes of HDPP, confirming the ability of proposed online inference algorithm to scale up to a large dataset with $S \approx 210000$ spikes in 600 seconds.

7 Conclusion

We present a hierarchical Dirichlet point process (HDPP) for neural sequences detection with an efficient online inference algorithm. The proposed model follows the framework of Dirichlet point process, where spike firng rates are modeled as a nonhomogeneous Poisson process and temporal dynamic of different sequences is captured via a Hawkes process. Unlike previous works, HDPP learns the number of sequence types adaptively during inference, and takes in data in a streaming fashion, thus can be easily adapted to new data. In the experiments, favorable results from both synthetic data and real world spike recordings have demonstrated HDPP can reliably and efficiently identify neural sequences in a single pass. Furthermore, it is an interesting future direction to investigate how we can learn HDPP via online variational inference without truncated approximation of the stick breaking construction [24, 13].

References

[1] Kevin Canini, Lei Shi, and Thomas Griffiths. Online inference of topics with latent dirichlet allocation. In *Artificial Intelligence and Statistics*, pages 65–72. PMLR, 2009.

[2] Olivier Cappé, Simon J Godsill, and Eric Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

[3] Zhe Chen, Andres D Grosmark, Hector Penagos, and Matthew A Wilson. Uncovering representations of sleep-associated hippocampal ensemble spike activity. *Scientific reports*, 6(1):1–13, 2016.

[4] Thomas J Davidson, Fabian Kloosterman, and Matthew A Wilson. Hippocampal replay of extended experience. *Neuron*, 63(4):497–507, 2009.

[5] Nan Du, Mehrdad Farajtabar, Amr Ahmed, Alexander J Smola, and Le Song. Dirichlet-hawkes processes with applications to clustering continuous-time document streams. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 219–228, 2015.

[6] Mark S Goldman. Memory without feedback in a neural network. *Neuron*, 61(4):621–634, 2009.

[7] AD Grosmark, J Long, and G Buzsáki. Recordings from hippocampal area ca1, pre, during and post novel spatial learning. *CRCNS. org*, 10:K0862DC5, 2016.

[8] Andres D Grosmark and György Buzsáki. Diversity in neural firing dynamics supports both rigid and learned hippocampal sequences. *Science*, 351(6280):1440–1443, 2016.

[9] Richard HR Hahnloser, Alexay A Kozhevnikov, and Michale S Fee. An ultra-sparse code underliesthe generation of neural sequences in a songbird. *Nature*, 419(6902):65–70, 2002.

[10] Christopher D Harvey, Philip Coen, and David W Tank. Choice-specific sequences in parietal cortex during a virtual-navigation decision task. *Nature*, 484(7392):62–68, 2012.

[11] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[12] Daoyun Ji and Matthew A Wilson. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature neuroscience*, 10(1):100–107, 2007.

[13] Dahua Lin. Online learning of nonparametric mixture models via sequential variational approximation. *Advances in Neural Information Processing Systems*, 26, 2013.

[14] Kourosh Maboudi, Etienne Ackermann, Laurel Watkins de Jong, Brad E Pfeiffer, David Foster, Kamran Diba, and Caleb Kemere. Uncovering temporal structure in hippocampal output patterns. *Elife*, 7:e34467, 2018.

[15] Emily L Mackevicius, Andrew H Bahle, Alex H Williams, Shijie Gu, Natalia I Denisenko, Mark S Goldman, and Michale S Fee. Unsupervised discovery of temporal sequences in high-dimensional datasets, with applications to neuroscience. *Elife*, 8:e38471, 2019.

[16] Charalampos Mavroforakis, Isabel Valera, and Manuel Gomez-Rodriguez. Modeling the dynamics of learning activity on the web. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1421–1430, 2017.

[17] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[18] Eva Pastalkova, Vladimir Itskov, Asohan Amarasingham, and Gyorgy Buzsaki. Internally generated cell assembly sequences in the rat hippocampus. *Science*, 321(5894):1322–1327, 2008.

[19] Sven Peter, Elke Kirschbaum, Martin Both, Lee Campbell, Brandon Harvey, Conor Heins, Daniel Durstewitz, Ferran Diego, and Fred A Hamprecht. Sparse convolutional coding for neuronal assembly detection. *Advances in Neural Information Processing Systems*, 30, 2017.

[20] Pietro Quaglio, Vahid Rostami, Emiliano Torre, and Sonja Grün. Methods for identification of spike patterns in massively parallel spike trains. *Biological cybernetics*, 112(1):57–80, 2018.

[21] Eleonora Russo and Daniel Durstewitz. Cell assemblies at multiple time scales with arbitrary lag constellations. *Elife*, 6:e19428, 2017.

[22] Emiliano Torre, Carlos Canova, Michael Denker, George Gerstein, Moritz Helias, and Sonja Grün. Asset: analysis of sequences of synchronous events in massively parallel spike trains. *PLoS computational biology*, 12(7):e1004939, 2016.

[23] Roemer van der Meij and Bradley Voytek. Uncovering neuronal networks defined by consistent between-neuron spike timing from neuronal spike recordings. *Eneuro*, 5(3), 2018.

[24] Chong Wang and David Blei. Truncation-free online variational inference for bayesian nonparametric models. *Advances in neural information processing systems*, 25, 2012.

[25] Alex Williams, Anthony Degleris, Yixin Wang, and Scott Linderman. Point process models for sequence detection in high-dimensional neural spike trains. *Advances in neural information processing systems*, 33:14350–14361, 2020.

[26] Alex H Williams, Tony Hyun Kim, Forea Wang, Saurabh Vyas, Stephen I Ryu, Krishna V Shenoy, Mark Schnitzer, Tamara G Kolda, and Surya Ganguli. Unsupervised discovery of demixed, low-dimensional neural dynamics across multiple timescales through tensor component analysis. *Neuron*, 98(6):1099–1115, 2018.

[27] Alex H Williams and Scott W Linderman. Statistical neuroscience in the single trial limit. *Current opinion in neurobiology*, 70:193–205, 2021.

[28] Alexander Henry Williams. *Unsupervised Methods for Neural Data Analysis with Single Trial Resolution*. Stanford University, 2019.

[29] Svante Wold. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405, 1978.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly swage paid to participants and the total amount spent on participant compensation? [N/A]