Unsupervised Clusterless Decoding using a Switching Poisson Hidden Markov Model

Etienne Ackermann etienne.ackermann@rice.edu
Caleb T. Kemere caleb.kemere@rice.edu
Department of Electrical and Computer Engineering
Rice University
Houston, TX 77005-1892, USA

John P. Cunningham jpc2181@columbia.edu
Department of Statistics
Columbia University
New York, NY 10027, USA

Abstract

Spike sorting is a standard preprocessing step to obtain ensembles of single unit data from multiunit, multichannel recordings in neuroscience. However, more recently, some researchers have started doing analyses directly on the unsorted data. Here we present a new computational model that is an extension of the standard (unsupervised) switching Poisson hidden Markov model (where observations are time-binned spike counts from each of N neurons), to a clusterless approximation in which we observe only a d-dimensional mark for each spike. Such an unsupervised yet clusterless approach has the potential to incorporate more information than is typically available from spike-sorted approaches, and to uncover temporal structure in neural data without access to behavioral correlates. We show that our approach can recover model parameters from simulated data, and that it can uncover task-relevant structure from real neural data.

Keywords: Hidden Markov Models, Clusterless Decoding

1. Introduction

Decoding neural activity is fundamental to much of neural data analysis. Most of the approaches to decoding are *supervised*, meaning that we are given labeled data (e.g., a set of an animal's positions along a track), as well as the corresponding neural data, and our task becomes to learn a mapping from the neural activity to position, or whichever other extrinsic correlate we are interested in. In this context, neural activity typically refers to spike events from an ensemble (tens, hundreds or maybe even thousands) of neurons. However, neural data are most commonly recorded as continuous voltage traces on multiple electrode channels, so that prior to decoding, spikes first have to be detected from these voltage traces, and each of those spikes has to be assigned to one of N neurons in a process called *spike sorting* (Gerstein and Clark, 1964; Lewicki, 1998). Figure 1.A shows an example of the information that may be available for supervised decoding with spike sorted data, including spike timing information, the neuron identity for each spike, spike waveform feature information, and the extrinsic correlates (position in this figure).

Spike sorting workflows are often characterized by a high degree of subjectivity and variability (Febinger et al., 2018), and even with fully automated spike sorting approaches like

that developed by Chung et al. (2017), results may still be sensitive to a host of parameter choices. Spike sorting also throws away potentially valuable information: most spikes are discarded in the spike sorting process for being too small or too "difficult" to sort, and so they get lumped into background multiunit activity or "hash". For those spikes that are sorted, we typically throw away the uncertainty information about each spike assignment. With the rapid rise in popularity of data mining and machine learning, it is not surprising that researchers have tried to find ways to interpret neural data directly, without having to perform spike sorting first. Indeed, the desire to circumvent the subjectivity and tedium of the spike sorting process, along with the observation that spike sorting throws away valuable information, has led to the recent development of many so-called "clusterless" approaches (Ventura, 2008; Kloosterman et al., 2013; Deng et al., 2015). Clusterless approaches operate directly on spike waveform features (such as the peak amplitudes on each of the four channels of a tetrode), without requiring or assuming knowledge of the underlying neuron identity that generated each individual spike. Figure 1.B illustrates typical information that may be available during supervised clusterless decoding, namely spike timing information, spike waveform features, and the behavioral correlates.

A complimentary paradigm to the supervised decoding approaches mentioned above, is that of *unsupervised* approaches to understand and interpret neural data (see e.g., Low et al., 2018; Williams et al., 2018; Maboudi et al., 2018; Williams et al., 2019; Mackevicius et al., 2019). These unsupervised approaches provide powerful ways of understanding the latent dynamics of neural activity, but to date they all require sorted spikes from ensembles of neurons. Figure 1.C shows an example of the information available during unsupervised decoding of sorted data, namely the spike timing information and the neuron identities responsible for generating those spikes. Note that there are no (observable) extrinsic correlates, and that fine spike timing information is lost during the binning process.

In this paper, we extend one particular unsupervised approach, namely the (switching Poisson) hidden Markov model (HMM; see e.g., Linderman et al., 2016; Maboudi et al., 2018, as well as Figure 1.E) to the clusterless setting, in a new model that we call the *clusterless* hidden Markov model. This clusterless HMM builds on the switching Poisson framework used by Gat and Tishby (1993); Kemere et al. (2008), and others, by treating the neuron identities responsible for the spikes as latent variables in the model (see Figure 2.F), and by incorporating our uncertainty about those identities in the form of parametric distributions of the waveform features associated with each hidden neuron, the *mark distributions*. Such a clusterless approach allows us (i) to incorporate more information than typically available with a standard spike sorting approach, and (ii) to eliminate the subjectivity and variability arising from the spike sorting process. The incorporation of additional information is particularly appealing for short time-scale event analysis (e.g., replay or theta sequences), where there are often not sufficiently many spikes remaining after spike sorting to decode with much accuracy.

We present the clusterless HMM and one possible way of doing inference on the model in Section 2, and we show that we can accurately recover the underlying model parameters in a simulation study in Section 3. We also show that the model can uncover temporal structure in real neural data from the rat hippocampus in Section 4, and finally we provide some directions for future improvement and investigation in Section 5.

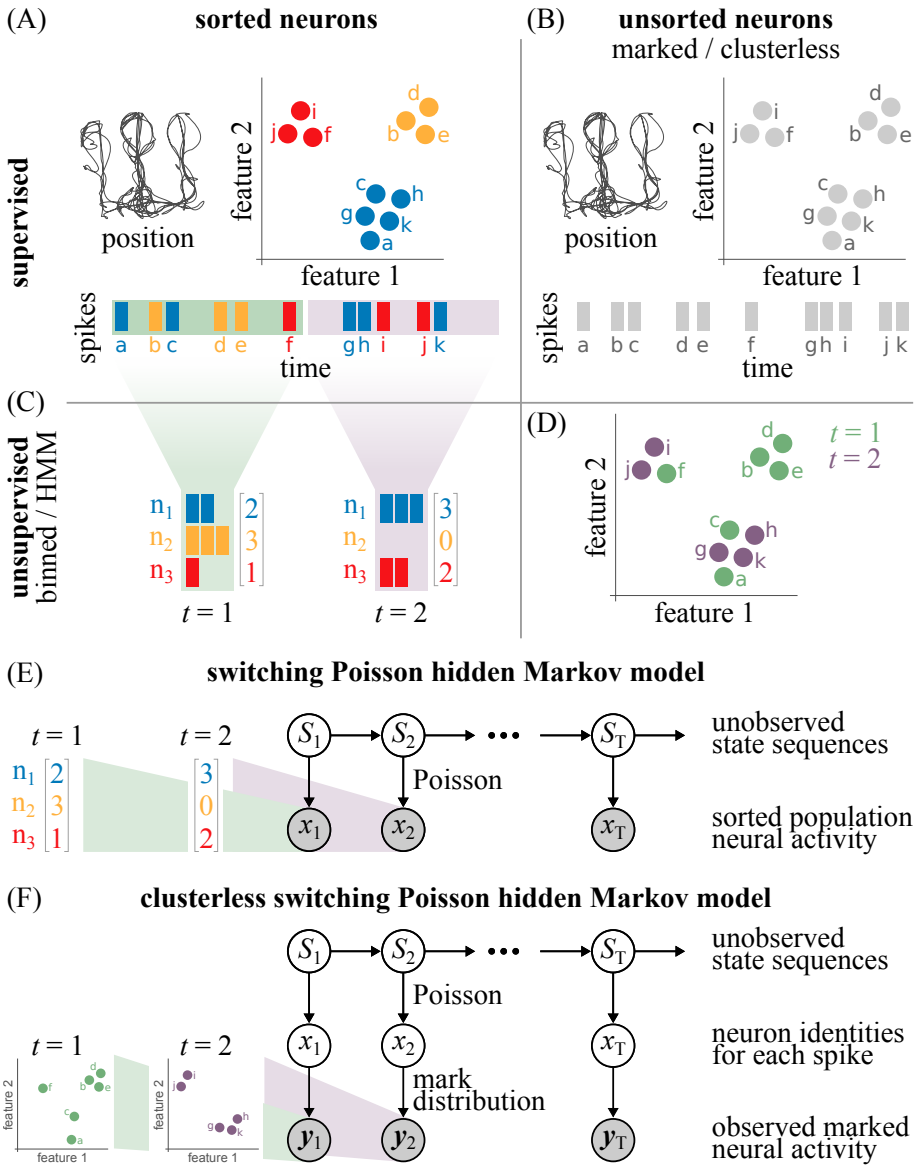

Figure 1: Switching Poisson HMM versus clusterless HMM.
(A)–(D) Illustration of information available within different decoding paradigms. (A) Supervised + sorted decoding. (B) Supervised + clusterless decoding. (C) Unsupervised + sorted decoding (e.g., switching Poisson HMM). (D) Unsupervised + clusterless decoding (this paper). (E) Graphical model illustrating the switching Poisson HMM, where the sequence of states are unobserved, and each state generates an observable collection of spike counts within the associated time window. Examples of observations during two time windows are shown on the left, namely spike counts with neuron identities. (F) Graphical model for our clusterless HMM, where the neuron identities generating the spikes are unobservable. The unobservable neuron identities become latent variables in our model, and these generate the observable marks. Examples of observations are shown on the left, namely a collection of unsorted spike waveform features during each time window.

2. Methods

We present a novel clusterless HMM for analyzing and decoding multiunit sequential neural activity in an unsupervised manner. The Baum–Welch algorithm is an expectation maximization (EM)-based algorithm for estimating the maximum-likelihood parameters of an HMM (Bilmes et al., 1998), and we will use it in this paper to do inference on our model. The Baum–Welch algorithm critically relies on our ability to evaluate $P(\mathbf{y}_t \mid S_t)$, from which we can efficiently compute several other quantities of interest to enable inference with our models (see e.g., Rabiner (1989) and Appendix B). That is, we need to be able to evaluate the probability of observing a particular outcome $\mathbf{y}_t$ for each (hidden) state S_t. In the following sections, we will present our full clusterless HMM, and develop a sampling-based approach to approximate $P(\mathbf{y}_t \mid S_t)$.

The idea here is very simple. In particular, *if we did know the neuron identities for each spike*, then we would not have to use a "clusterless" approach, and we could instead fit a standard switching Poisson HMM. So in our clusterless model, we can similarly compute probabilities *if we assume that we know the neuron identities*, and we simply treat these identities as hidden or latent variables in our model. Moreover, we do not need to know exactly how many neurons there are, since a specification of the number of neurons simply determines the partitioning of our waveform feature space, so that an over-specification (or perhaps even a small under-specification) of the number of neurons should not have a significant effect on our model's behavior. These ideas will be made precise below.

2.1 Notation and preliminaries

Suppose that we record from a population of N neurons, with each neuron identified by $n \in \{1, 2, \dots N\}$. Further, suppose that we observe a d-dimensional mark $\left(m \in \mathbb{R}^d\right)$ for each spike from a neuron. The marks could be, for example, the peak amplitudes from the four channels of a tetrode, or principal components of the observed waveform on a collection of electrodes, and so on.

If we assume that the neurons have *state*-dependent firing rates $r : \mathbb{Z} \to \mathbb{R}^N$ with $S_t \mapsto r(S_t) \equiv \mathbf{r}_t$, then we consider each neuron as generating a train of events from an inhomogeneous (state-dependent) Poisson process, with an associated (state-dependent) rate, $r^n(S_t) \in \mathbb{R}^+$. To be more precise, we assume that the neurons are *independent*, and that they have N state-dependent rates, $\mathbf{r}_t = \left(r^1(S_t), r^2(S_t), \dots, r^N(S_t)\right)$ at time $t \in \mathbb{R}$, and for some state $S_t \in \{1, 2, \dots, Z\}$. We may then collect all of these rates as the matrix $\mathbf{\Lambda} \in \mathbb{R}_{\geq 0}^{Z \times N}$, and we will estimate these rates as part of the HMM parameter estimation process.

For simplicity, we will assume that we have only one probe (or tetrode). In general, we assume that the tetrodes are independent, so that tetrodes record from disjoint subsets of neurons, and therefore generalization to the multiprobe case is straightforward.

Now, consider a single observation window $(t-1, t]$ simply identified with t, during which we observe $K(t) = K$ spike events (equivalently, we observe K marks: $\mathbf{y}_t = \{m_1, \dots, m_K\}, m_i \in \mathbb{R}^d, i = 1, \dots, K$), and for which we assume that each neuron has a constant firing rate for the duration of the time window. Since observation windows are conditionally independent (given the underlying states), we only need to concern ourselves

with evaluating $P(\mathbf{y}_t \,|\, S_t)$ for a single observation window (the process remains unchanged for each observation window).

Finally, let us assume that the marks (the spike waveform features) can be modeled as coming from unit-specific distributions parameterized by $\mathbf{\Phi}$. The choice of this distribution will depend on which features are ultimately chosen to represent each mark. In this paper, we simply use the peak amplitudes on each of the tetrode channels, which can be adequately modeled as coming from unit-specific multivariate normal distributions in practice.

2.2 Model Specification

As suggested before, let $\mathbf{y}_t$ denote the *observation* at time t, where $\mathbf{y}_t \in \mathbb{R}^{d \times K(t)}$, and where $K(t)$ is the number of marks observed during time window t. We assume that the observations are sampled at discrete, equally-spaced time intervals, so that t can be an integer-valued index.

The hidden state space is assumed to be *discrete*, with $S_t \in \{1, \ldots, Z\}$. To define a probability distribution over sequences of observations, we need to specify a probability distribution over the initial state, $\boldsymbol{\pi} \equiv P(S_1)$, with $\boldsymbol{\pi}_i = P(S_1 = 1)$, the $Z \times Z$ state transition probability matrix, $\mathbf{A}$, with $a_{ij} = P(S_t = j \,|\, S_{t-1} = i)$, and the emission model defining $P(\mathbf{y}_t \,|\, S_t)$.

In the usual multiple-spike-train switching Poisson model (see e.g., Jones et al., 2007; Kemere et al., 2008), we simultaneously observe N spike trains (corresponding to N neurons). Each spike train is modeled as an independent Poisson process with rate $\lambda_n(t)$, and the vector of firing rates, $\boldsymbol{\lambda}(t)$, switches randomly between one of Z states. We will use the same idea here, but since the number of neurons and more importantly the identities of those neurons are *hidden*, we will have to be slightly more careful in our treatment. Nevertheless, assuming that our data were generated by N hidden neurons, we associate the N-dimensional rate vector $\mathbf{r}_t$ with the expected number of marks from each of the neurons during time window t. There are Z possible rate vectors (one for each state S_t), so that we can collect all the possible rates captured by the model in the rate (or observation) matrix $\mathbf{\Lambda} \in \mathbb{R}_{\geq 0}^{Z \times N}$.

We further assume that our model is time-invariant in that the state transition probability matrix and the emission model do not change over time, and that the states form a first-order Markov chain, namely that $P(S_t \,|\, S_{t-1}, \ldots, S_1) = P(S_t \,|\, S_{t-1})$. The HMM is then fully specified by the set of parameters $\mathbf{\Theta} = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{\Lambda}\}$. There are therefore a total of $Z(1 + Z + N)$ parameters to estimate for the model[1].

1. Technically we may also need to estimate the number of hidden states, the number of hidden neurons, as well as the neuron cluster parameters, but those are typically estimated *a priori* and / or outside of the Baum-Welch algorithm, so that we do not consider them as part of the HMM parameter estimation process.

2.3 Sampling approach to evaluate $P(\mathbf{y}_t \,|\, S_t)$

In order to use the Baum-Welch algorithm to estimate the set of parameters $\mathbf{\Theta} = \{\boldsymbol{\pi}, \mathbf{A}, \mathbf{\Lambda}\}$, we need to be able to evaluate

$$\begin{aligned} P(\mathbf{y}_t \,|\, S_t = j) &= P\big(\mathbf{y}_t \,|\, r(S_t = j)\big) \\ &= P\big((m_k)_{k=1}^{K(t)}, K \,\big|\, \mathbf{r}_t^{(j)}; \{\mathbf{\Phi}_n\}_{n=1}^{N}\big) \\ &= \mathcal{L}\big(\mathbf{r}_t^{(j)} \,\big|\, (m_k)_{k=1}^{K(t)}, K; \{\mathbf{\Phi}_n\}_{n=1}^{N}\big) \end{aligned} \tag{1}$$

where the dependence of the sequence of observed marks $\mathbf{y}_t \equiv (m_k)_{k=1}^{K(t)}$ on the hidden state S_t is realized through the state-dependent firing rates $r(S_t = j) \equiv \mathbf{r}_t^{(j)}$, as well as the unit-specific mark distribution parameters $\{\mathbf{\Phi}_n\}_{n=1}^{N}$. That is, conditioning on S_t is equivalent to conditioning on $\mathbf{r}_t^{(j)}$. Note also the $P\big((m_k)_{k=1}^{K}\big) \equiv P\big((m_k)_{k=1}^{K}, K\big)$, and we will simply write $P\big((m_k)_{k=1}^{K}\big)$ for convenience.

A graphical representation during one time window of our clusterless HMM is shown in Figure 2. The collection of K marks depend on the unobserved neuron identities (q_n^k), as well as on the possibly state dependent mark distribution parameters, $\mathbf{\Phi}$. The neuron identities depend on the firing rates, ρ and r, and on how many marks were observed (K). The number of marks, K, is assumed to follow a Poisson distribution with mean rate r, the total number of expected spikes from all the neurons in state S.

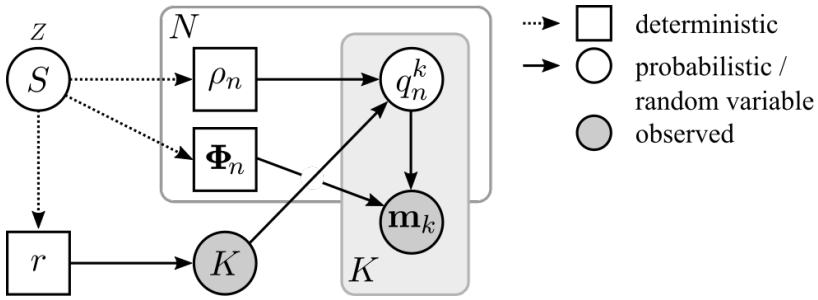

Figure 2: Probabilistic graphical model during single time window for the clusterless HMM. S: discrete state, $|S| = Z$; r: aggregate firing rate, ρ_n: relative firing rate of neuron n; $K \sim \text{Pois}(r)$: number of marks observed; $q_k \sim \text{Multinom}(K, \boldsymbol{\rho})$: neuron identity indicator; $\mathbf{\Phi}_n$: mark distribution parameters; $\mathbf{m}_k$: d-dimensional mark.

Dropping the dependence of K on t (purely for notational simplicity), and dropping the explicit parameterization of $P(\cdot)$ by $\mathbf{\Phi}$, we note that we desire to evaluate

$$P_{\mathbf{y}_t, |\mathbf{r}_t^{(j)}}\big((m_k)_{k=1}^{K} \,\big|\, \mathbf{r}_t^{(j)}\big), \tag{2}$$

which is hard to evaluate explicitly for two key reasons, namely (i) we do not know which neurons / units gave rise to each of the marks, and (ii) we do not know *a priori* how many marks we will observe in a particular observation window, so that we cannot easily specify a simple (possibly multivariate) probability distribution over the sequence of observed marks. Indeed, the dimensionality of this distribution will have to depend on how many marks we observe in each window.

To address challenge (i) above, we introduce the auxiliary hidden random variable $\mathbb{I}^K \in \mathbb{Z}^{N\times K}$, that encodes which of the latent units gave rise to each of the observed marks. Each column of $\mathbb{I}^K$ is a standard unit vector $\mathbf{e}_n$ whose elements are all zeros, except for the nth element, which is equal to one. That is, the kth column encodes the neuron identity n that generated the kth mark. We denote this as $\mathbb{I}^K \equiv \big(\mathbf{e}_{u(k)}\big)_{k=1}^K$, such that $u(k) \in \mathbb{Z}$ is the neuron identity of the kth mark.

In particular, we note that

$$P_{\mathbf{y}_t|\mathbf{r}_t^{(j)}}\big((m_k)_{k=1}^K \,\big|\, \mathbf{r}_t^{(j)}\big) = \int_{\mathbb{I}^K} P_{\mathbf{y}_t,\mathbb{I}^K|\mathbf{r}_t^{(j)}}\big((m_k)_{k=1}^K, \mathbb{I}^K \,\big|\, \mathbf{r}_t^{(j)}\big)\,\mathrm{d}\mathbb{I}^K. \tag{3}$$

Making the dependence on time t and state j implicit (i.e., letting $\mathbf{r} \equiv r(S_t = j)$ as before), we note that

$$P_{\mathbf{y},\mathbb{I}^K|\mathbf{r}}\big((m_k)_{k=1}^K, \mathbb{I}^K \,\big|\, \mathbf{r}\big) = P_{\mathbf{y}|\mathbb{I}^K,\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbb{I}^K, \mathbf{r}\big) \cdot P_{\mathbb{I}^K|\mathbf{r}}(\mathbb{I}^K \,|\, \mathbf{r}) \tag{4}$$

so that

$$\begin{aligned}\int_{\mathbb{I}^K} P_{\mathbf{y},\mathbb{I}^K|\mathbf{r}}\big((m_k)_{k=1}^K, \mathbb{I}^K \,\big|\, \mathbf{r}\big)\,\mathrm{d}\mathbb{I}^K &= \int_{\mathbb{I}^K} P_{\mathbf{y}|\mathbb{I}^K,\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbb{I}^K, \mathbf{r}\big) \cdot P_{\mathbb{I}^K|\mathbf{r}}(\mathbb{I}^K \,|\, \mathbf{r})\,\mathrm{d}\mathbb{I}^K \\ &= \mathbb{E}_{\mathbb{I}^K|\mathbf{r}}\big[P_{\mathbf{y}|\mathbb{I}^K,\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbb{I}^K, \mathbf{r}\big)\big]. \end{aligned} \tag{5}$$

Note that it is generally difficult to compute the integral directly, whereas it is somewhat simpler to estimate the expected value, assuming of course, that we can sample from $\mathbb{I}^K \,|\, \mathbf{r}$ according to it's distribution. Indeed, if we are able to sample $\mathbf{Q}^{(i)} \sim P_{\mathbb{I}^K|\mathbf{r}}(\mathbb{I}^K \,|\, \mathbf{r})$, where $\mathbf{Q}^{(i)} \in \mathbb{Z}^{N\times K}$ for $i = 1, \ldots, M$, then

$$\begin{aligned} P_{\mathbf{y}|\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbf{r}\big) &= \mathbb{E}_{\mathbb{I}^K|\mathbf{r}}\big[P_{\mathbf{y}|\mathbb{I}^K,\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbb{I}^K, \mathbf{r}\big)\big] \\ &\approx \frac{1}{M}\sum_{i=1}^M P_{\mathbf{y}|\mathbb{I}^K,\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbb{I}^K = \mathbf{Q}^{(i)}, \mathbf{r}\big) \\ &= \frac{1}{M}\sum_{i=1}^M P_{\mathbf{y}|\mathbb{I}^K,\mathbf{r}}\big((m_k)_{k=1}^K, K \,\big|\, \mathbb{I}^K = \mathbf{Q}^{(i)}, \mathbf{r}\big) \\ &= \frac{1}{M}\sum_{i=1}^M P_{\mathbf{y}|\mathbb{I}^K}\big((m_k)_{k=1}^K \,\big|\, \mathbb{I}^K = \mathbf{Q}^{(i)}\big) \cdot P_{\mathbf{y}|\mathbb{I}^K,\mathbf{r}}\big(K \,\big|\, \mathbb{I}^K = \mathbf{Q}^{(i)}, \mathbf{r}\big) \\ &= \frac{1}{M}\sum_{i=1}^M \prod_{k=1}^K P\big(m_k;\, \mathbf{\Phi}(\mathbf{q}^{k(i)})\big) \cdot \prod_{n=1}^N \mathrm{Pois}\big(V_n^{(i)}; r_n\big) \end{aligned} \tag{6}$$

where we have explicitly brought K back into $P\big((m_k)_{k=1}^K, K \,|\, \mathbb{I}^K, \mathbf{r}\big)$, and where we have used the fact that the marks are assumed independent to express $P\big((m_k)_{k=1}^K \,|\, \mathbb{I}^K\big)$ as a product. We have also made use of the conditional independence (see Figure 2) of the marks on the neuron identities ($\mathbf{Q}$) and the number of observed marks (K) to factorize

$P\big((m_k), K \,|\, \mathbb{I}^K, \mathbf{r}\big)$. Then we can finally approximate the data log likelihood simply as:

$$\log\Big[P_{\mathbf{y}\,|\,\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbf{r}\big)\Big] \approx$$
$$\approx \log\left[\sum_{i=1}^{M} \exp\left(\sum_{k=1}^{K} \log\Big[P\Big(m_k; \mathbf{\Phi}\big(\mathbf{q}^{k(i)}\big)\Big)\Big] + \right.\right.$$
$$\left.\left. + \sum_{n=1}^{N} \log\Big[\mathrm{Pois}\big(V_n^{(i)}; r_n\big)\Big]\right)\right] - \log(M) \quad (7)$$

where, for each mark, $\mathbf{\Phi}$ depends on the neuron identity encoded by $\mathbf{q}^{k(i)}$, the kth column of the ith sample $\mathbf{Q}^{(i)}$, and where $V_n^{(i)}$ is the total number of spikes from neuron n, for the ith sample. That is,

$$V_n = \sum_{k=1}^{K} \big(u(k) = n\big) \equiv \sum_{k=1}^{K} \big[\mathbb{I}^K\big]_{nk}, \quad n = 1, 2, \ldots, N, \text{ so that } V_n^{(i)} = \sum_{k=1}^{K} q_n^{k(i)} \quad (8)$$

and where q_n^k is the nth element of the kth column of $\mathbf{Q}$, which equals one if we assume that the kth mark was generated by the nth neuron, and zero otherwise.

In order to sample $\mathbf{Q} \sim P_{\mathbb{I}^K|\mathbf{r}}(\mathbb{I}^K \,|\, \mathbf{r})$, it is convenient to factorize the rate $(\mathbf{r})$ into the aggregate rate $r \equiv \sum_n^N \mathbf{r}_n$, and the relative rates $(\rho_1, \rho_2, \ldots, \rho_N) \equiv \boldsymbol{\rho} \in \mathbb{R}^N$ s.t. $\sum_n^N \rho_n = 1$ (see Figure 2). In this way, the (true) rate associated with neuron n is simply $\mathbf{r}_n = r\rho_n$, and we can simply sample from the multinomial distribution of $\mathbf{V}$, one trial at a time, for each mark identity in $\mathbf{Q}^{(i)}$. That is, for $k = 1, \ldots, K$, sample $\big[\mathbf{Q}^{(i)}\big]_k \sim \mathrm{Multinomial}\big(1, \frac{\mathbf{r}}{\|\mathbf{r}\|_1}\big)$, where $[\cdot]_k$ is used to denote the kth column:

$$P_{\mathbb{I}^K|\mathbf{r}}\big(\mathbb{I}^K \,|\, \mathbf{r}_t\big) \propto P_{\mathbf{V}^K|\mathbf{r}}\big(\mathbf{V}^K \,|\, \mathbf{r}\big) \sim \mathrm{Multinom}\left(K, \frac{\mathbf{r}}{\|\mathbf{r}\|_1}\right) = \mathrm{Multinom}\big(K, \boldsymbol{\rho}\big). \quad (9)$$

Note that this sampling strategy yields the correct number of neuron identities (proportional to their rates), but it does not affect the order of the columns of $\mathbf{Q}$.

2.4 Updating the state-dependent firing rates, $\mathbf{r}^{(j)}$

Ordinarily, we may consider updating the rates according to

$$\hat{r}^n\big|_{S_t=j} = \frac{\sum_{t=1}^{T} \gamma_j(t) V_n^{(j)}(t)}{\sum_{t=1}^{T} \gamma_j(t)} \quad (10)$$

where $\sum_{t=1}^{T} \gamma_j(t)$ is the expected number of times that we are in state j (or equivalently, the expected number of transitions away from state j; even more explicitly, $\gamma_j(t) = P(S_t = j \,|\, \mathbf{Y})$), and $V_n^{(j)}(t)$ is the number of marks in time window t that were generated / emitted by neuron n (and given that the system is currently in state j). However, we do not know which marks were generated by which neurons, so we have to consider

$$\hat{r}^n\big|_{S_t=j} = \frac{\sum_{t=1}^{T} \gamma_j(t) \mathbb{E}\big[V_n^{(j)}(t)\big]}{\sum_{t=1}^{T} \gamma_j(t)} \quad (11)$$

instead.

It turns out that we can efficiently compute this expectation, for time window $(t-1, t]$, as follows:

$$\mathbb{E}_{\mathbb{I}^K|\mathbf{r},(m_k)_{k=1}^K}\left[V_n^{(j)}(t)\right] = \sum_{k=1}^{K(t)} \mathfrak{q}_n^k(t,j) \tag{12}$$

where

$$\mathfrak{q}_i^k(t,j) = \frac{P\big(m_k^{(t)}; \mathbf{\Phi}_i\big) \cdot \rho_i^{(j)}}{\sum_{n=1}^N P\big(m_k^{(t)}; \mathbf{\Phi}_n\big) \cdot \rho_n^{(j)}}, \quad i = 1, 2, \ldots, N, \tag{13}$$

and where the dependence on t should be clear, namely that the sequence of marks (m_k) are those observed during time window t. Note that (13) is computed *independent* of any sampling.

The prior distribution over states, $\boldsymbol{\pi}_i \equiv P(S_1 = i)$, and the transition probability matrix, $\mathbf{A}$, can be estimated in the standard way, namely

$$\hat{\boldsymbol{\pi}}_i = \gamma_i(1), \quad i = 1, 2, \ldots, Z \tag{14}$$

and

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{15}$$

where $\xi_{ij}(t) = p(S_t = i, S_{t+1} = j \,|\, \mathbf{Y})$, so that $\sum_{t=1}^{T-1} \xi_{ij}(t)$ is the expected number of transitions from state i to state j.

The calculations of $P(\mathbf{y}_t \,|\, S_t)$ and $\mathbb{E}[V_n^{(j)}$ as well as the parameter estimation process are summarized in Algorithms 3–1 in Appendix B.

3. Simulation Case Study

We tested our clusterless HMM on simulated data so that we could determine exactly how well model parameters were recovered. In particular, we were interested in recovering the state transition probability matrix $\mathbf{A} \in \mathbb{R}^{Z \times Z}$, the observation—or rate—matrix $\mathbf{\Lambda} \in \mathbb{R}^{Z \times N}$, and a sequence of latent states $(S_1, S_2, \ldots S_T)$.

For the remainder of this paper, we will assume that the mark distributions are independent of the hidden states, and that the marks can be modeled as multivariate normal distributions with parameters $\mathbf{\Phi} = \{\boldsymbol{\mu}_n, \mathbf{\Sigma}_n\}_{n=1}^N$ (see Figure 3). In this case, we can estimate the mark distributions before doing parameter estimation for our clusterless HMM, and we chode to use a simple Gaussian mixture model (GMM) to estimate the mark distribution parameters. Moreover, substituting the multivariate normal mark distributions into (6), (7), and (13) lead to

$$P_{\mathbf{y}|\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbf{r}\big) \approx \frac{1}{M} \sum_{i=1}^{M} \prod_{k=1}^{K} \mathcal{N}\big(m_k; \boldsymbol{\mu}(\mathbf{q}^{k(i)}), \mathbf{\Sigma}(\mathbf{q}^{k(i)})\big) \cdot \prod_{n=1}^{N} \mathrm{Pois}\big(V_n^{(i)}; r_n\big),$$

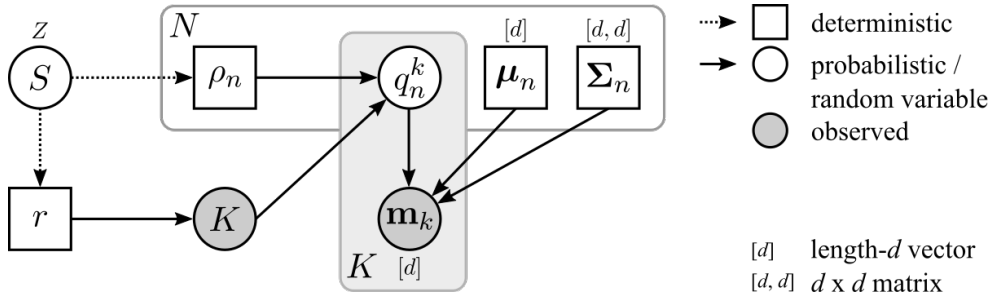

Figure 3: Probabilistic graphical model for the clusterless HMM with state-independent multivariate normal mark distributions.
S: discrete state, $|S| = Z$; r: aggregate firing rate, ρ_n: relative firing rate of neuron n; $K \sim \text{Pois}(r)$: number of marks observed; $q_k \sim \text{Multinom}(K, \boldsymbol{\rho})$: neuron identity indicator; $\boldsymbol{\mu}_n$: neuron centroid; $\boldsymbol{\Sigma}_n$: neuron covariance; $\mathbf{m}_k \sim \mathcal{N}\big(\sum_n q_n^k \boldsymbol{\mu}_n, \sum_n q_n^k \boldsymbol{\Sigma}_n\big)$.

$$\log\Big[P_{\mathbf{y}\,|\,\mathbf{r}}\big((m_k)_{k=1}^K \,\big|\, \mathbf{r}\big)\Big] \approx$$
$$\approx \log\left[\sum_{i=1}^{M} \exp\left(\sum_{k=1}^{K} \log\Big[\mathcal{N}\Big(m_k; \boldsymbol{\mu}\big(\mathbf{q}^{k(i)}\big), \boldsymbol{\Sigma}\big(\mathbf{q}^{k(i)}\big)\Big)\Big] + \right.\right.$$
$$\left.\left. + \sum_{n=1}^{N} \log\Big[\text{Pois}\big(V_n^{(i)}; r_n\big)\Big]\right)\right] - \log(M)$$

and

$$\mathsf{q}_i^k(t,j) = \frac{\mathcal{N}\Big(m_k^{(t)}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i\Big) \cdot \rho_i^{(j)}}{\sum_{n=1}^{N} \mathcal{N}\Big(m_k^{(t)}; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\Big) \cdot \rho_n^{(j)}}, \quad i = 1, 2, \ldots, N.$$

3.1 Simple two-state, three-neuron example

As a simple yet representative example, we present parameter estimation results from a system with state transition probability matrix $\mathbf{A} = \begin{bmatrix} 0.8 & 0.2 \\ 0.5 & 0.5 \end{bmatrix}$, and a rate matrix $\boldsymbol{\Lambda} \approx \begin{bmatrix} 4.72 & 0.07 & 3.21 \\ 4.75 & 2.37 & 0.88 \end{bmatrix}$.

More specifically, we randomly sampled the centroids and covariances of our $N = 3$ latent units, which we modeled as ($d = 2$ dimensional) multivariate normal distributions. This choice of mark distribution works reasonably well in practice when the marks are waveform peak amplitudes on channels of a tetrode, and we can use a Gaussian mixture model as a preprocessing step to estimate the parameters $\boldsymbol{\Phi}_n = (\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$ for each neuron $n = 1, \ldots, N$. In the rest of this paper we will adopt this strategy. Consequently, we now have that $P\big(m_k; \boldsymbol{\Phi}(q^{k(i)})\big) = \mathcal{N}\big(m_k; \boldsymbol{\mu}(\mathbf{q}^{k(i)}), \boldsymbol{\Sigma}(\mathbf{q}^{k(i)})\big)$ in (6) and (7), and that $P\big(m_k^{(t)}; \boldsymbol{\Phi}_n\big) = \mathcal{N}\big(m_k^{(t)}; \boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\big)$ in (13).

We then sampled a sequence of states $(S_1, S_2, \ldots, S_T)$ from the state transition probability matrix, $\mathbf{A}$, and for each of those states we generated state-dependent spike events by sampling from a multivariate Poisson distribution with rates given by the corresponding row of $\mathbf{\Lambda}$. For each of those spike events, we then generated a $d = 2$ dimensional mark by sampling from the unit-specific multivariate normal distributions, leading to a sequence of observations $(\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T)$.

Using the sequence of observations, $(\mathbf{y}_t)_{t=1}^T$, we then fit the clusterless HMM by evaluating $P(\mathbf{y}_t \,|\, S_t)$ using (6) for each possible state S_t and time window t, and by using (15) to update the transition probability matrix, $\mathbf{A}$, and (11) to update the observation / rate matrix, $\mathbf{\Lambda}$, during each iteration of the Baum-Welch algorithm. We used $M = 5000$ samples to approximate (6).

We sampled a length $T = 200$ state sequence, and used the first 100 samples to fit the model, and the next 100 samples to evaluate the Viterbi (state-) decoding accuracy of the model. The results from the parameter estimation process, as well as from the state decoding process, are shown in Figure 4.

Notice that the parameters seem to have been estimated reasonably well, at least qualitatively. We define the relative error between matrices $\hat{\mathbf{B}}$ and $\mathbf{B}$ as

$$\varepsilon_{\text{rel}}(\hat{\mathbf{B}}, \mathbf{B}) = \frac{\|\hat{\mathbf{B}} - \mathbf{B}\|_F}{\|\mathbf{B}\|_F} \tag{16}$$

where $\|\cdot\|_F$ is the Frobenius norm: $\|\mathbf{B}\|_F = (\sum_{ij} |a_{ij}|^2)^{1/2}$.

The relative parameter estimation error for the state transition probability matrix was $\varepsilon_{\text{rel}}(\hat{\mathbf{A}}, \mathbf{A}) = 0.05$, for the rate map, $\varepsilon_{\text{rel}}(\hat{\mathbf{\Lambda}}, \mathbf{\Lambda}) = 0.12$, and the state decoding accuracy was 97.5%.

3.2 Misspecification of Hyperparameters

In the previous section we had assumed that both the number of latent states as well as the number of latent neurons were known, and the parameter estimation was performed with the exact numbers of each. In reality, we may neither know the "true" number of states, nor the number of neurons that participated in generating our data.

3.2.1 Overspecification of Number of States

For the unknown number of states, a more advanced Bayesian treatment may be considered, where the number of states is itself learned directly from the data such as in the hierarchical Dirichlet process hidden Markov model (see e.g., Teh et al., 2006), or a likelihood-based approach may be followed, as described by Celeux and Durand (2008). However, as far as decoding accuracy is concerned, it may be argued that the HMM approach is largely insensitive to the precise number of states (see e.g., Maboudi et al., 2018, as well as the example in this section). In effect, choosing a larger number of hidden states partitions our latent space into a finer partition, and conversely, a lower number of states specifies a coarser partition. In an alternate view, if we consider a mapping from the state space (our domain) to some external space of interest (our codomain, e.g., an animal's position in a maze), then we may expect to decode to the codomain reasonably well as long as we can

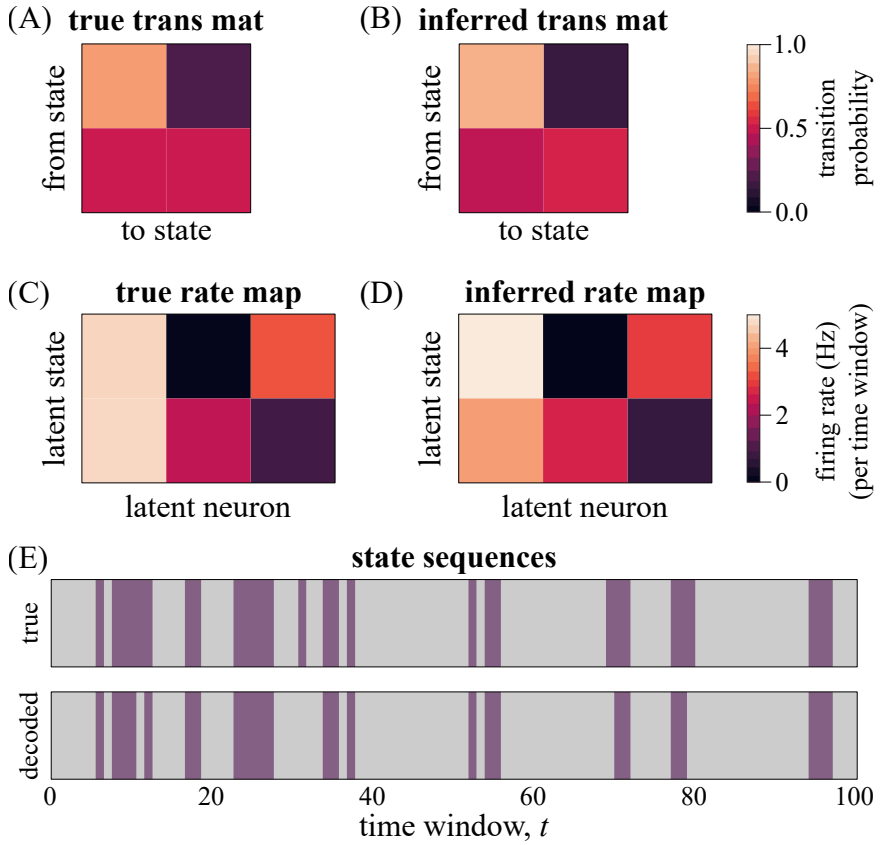

Figure 4: Parameter estimation and decoding example from synthetic dataset. (A) The true, and (B) inferred state transition probability matrices. (C) The true rate map that was used to generate the marks, and (D) the inferred rate map obtained by fitting the model. (E) Top: true state sequence generated from the transition matrix in (A); bottom: decoded state sequence. Colors indicate different states: $S_t \in \{z_1, z_2\}$.

find a surjective mapping; adding additional states should not affect our ability to find such a mapping, even though distinctness may become lost.

Indeed, we find that the decoding accuracy remains unchanged at 97.5% if we artificially use $Z = 4$ hidden states instead of the $Z = 2$ that were used to generate the data, and if we associate both dark and light purple states with "true" state z_1, and both dark and light gray states with "true" state z_2 (see Figure 5). This clearly illustrates the loss of distinctness, without affecting our ability to decode to the desired space (the original, data-generating state space in this instance).

3.2.2 Overspecification of Number of Neurons

For the unknown number of neurons, things are a bit simpler. We may get some idea by simple visual inspection of our data, or by performing some reasonable clustering in our

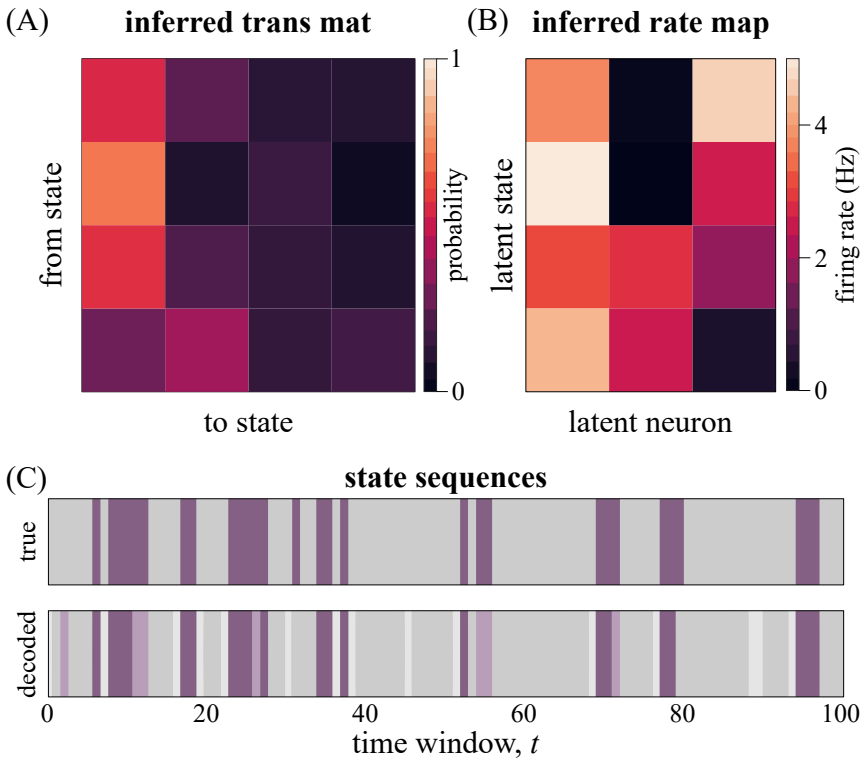

Figure 5: Overspecification of number of states. (A) The inferred transition matrix, and (B) the inferred rate map. (C) Decoding results. Top: true state sequence; bottom: decoded state sequence. Colors indicate different states: $S_t \in \{z_1, \ldots, z_4\}$.

mark space. But as was the case for the number of hidden states, an overspecification of the number of neurons should have a relatively small effect on the model's ability to decode (or to do most inference tasks, for that matter).

Indeed, if we over-specify $N = 5$ instead of the $N = 3$ neurons that were used to generate the data, we again find that our decoding accuracy is unchanged at 97.5% (see Figure 6). The relative estimation error in the transition matrix is also unchanged from when we had used the exact number of neurons.

This insensitivity to the number of neurons should not be very surprising, since neural decoding is often robust to subjective differences in the spike sorting process, where one researcher may combine two putative units together, while another may keep them as separate clusters. However, when we have limited data—such as during replay detection and analysis—the decoding robustness quickly fades, and small differences in spike sorting quality can have an outsized effect on analysis results. Thankfully, the clusterless approach presented in this paper (and in general) allows us to incorporate much more data than is typically available in a spike-sorting pipeline, thereby restoring some of the typical decoding robustness.

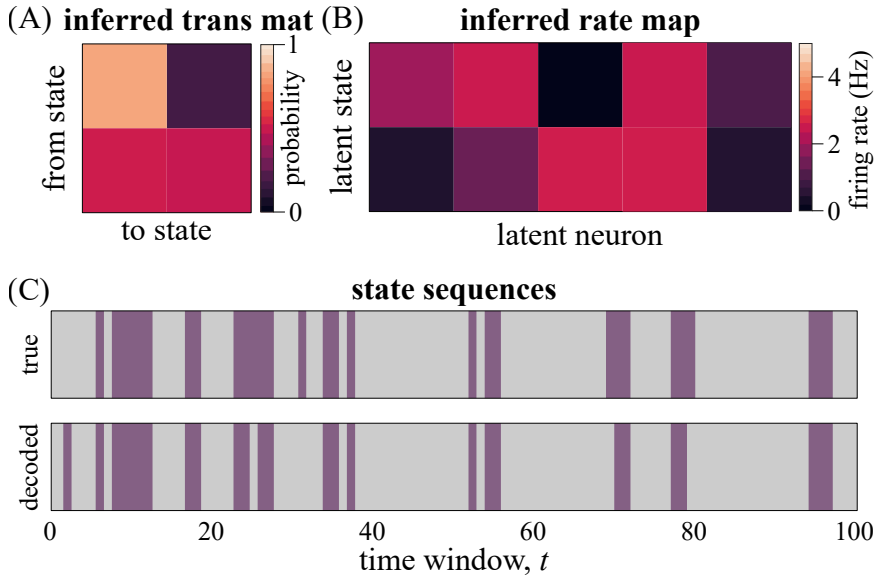

Figure 6: Overspecification of number of neurons. (A) The inferred transition matrix, and (B) the inferred rate map. (C) Decoding results. Top: true state sequence; bottom: decoded state sequence. Colors indicate different states: $S_t \in \{z_1, z_2\}$.

4. Application to Hippocampal Place Cell Data

We also fit our clusterless HMM to real neural data that were recorded from area CA1 in the rat hippocampus. In particular, a rat was trained to run back-and-forth on a 100 cm long linear track for liquid reward, and we subsequently recorded neural activity while the animal was performing this task. We expected that the clusterless HMM would capture the characteristic ensemble population "place cell" activity during periods in which the animal was running, and we expected to see that latent states correspond roughly to locations along the linear track (see e.g., Linderman et al., 2016; Maboudi et al., 2018).

4.1 Decoding Place Cell Activity with the Clusterless HMM

The animal was on the linear track for approximately 15 minutes. Periods of running activity were identified by applying a minimum speed threshold of 8 cm/s, and with this criterion, the animal was found to be running for a total duration of just over 4 minutes. Contiguous bouts of running (longer than 400 ms) were then partitioned into 400 ms observation windows. The distribution of these contiguous segments of running activity is shown in Figure 7.B, from where we can see that there are many short sequences (each consisting of only a few observation windows), and relatively few "long" sequences of 10 observation windows or more. With overwhelmingly short sequences, we can not expect the estimation of the transition probability matrix to be as reliable as if we had longer sequences, but we still find that the estimated transition probability matrix (Figure 7.C) is sparse, and strongly clustered around the diagonal and super-diagonal, with only a few other transitions scattered

throughout. This sparse structure is suggestive of sequential hippocampal dynamics, as we would expect during running behavior (Maboudi et al., 2018).

We included all (pyramidal and interneuron) spikes > 70 μV for our clusterless HMM, which resulted in about twice as many spikes during run as compared to the manual-spike-sorted data (7,154 sorted vs. 14,453 unsorted spikes), and about 30% of the total number of spikes (using 0 μV as baseline; see Figure 7.A).

By augmenting our dataset with some external correlate (the animal's position in this case), we can learn a mapping between the state space and the external correlate. We refer to this mapping as the latent state place field (lsPF). In particular, we use our clusterless HMM to decode some neural activity to the the state space, and we use the associated position data to then learn a mapping between the state space and position. We notice that the states localize relatively well in space (Figure 7.E), and we can use these lsPFs to decode neural activity to position (Figure 7.D).

In Figure 7.D we see the posterior probability distribution over position for each 400 ms observation window, and more importantly, we see that the regions of highest probability typically coincide with the animal's true position (black trace).

4.2 Comparison to Switching Poisson HMM

Unlike when we used simulated data, it is impossible to determine the "true" relative errors in estimating any of the parameters for the real neural data, especially since the latent states are merely mathematical abstractions (albeit with biological support or justification), so no true values for these parameters even exist. An alternative approach of evaluating our model's performance is by analyzing its position-decoding ability instead. Here we have chosen to compare our clusterless HMM's (position) decoding performance to the decoding performance of the switching Poisson HMM, where manually sorted data are used.

For both the clusterless and the switching Poisson HMMs, we kept as many of the hyperparameters the same as we could, including the observation window sizes, the sequences used for training and for testing, the number of (assumed) hidden states, etc. Using the manually-sorted spikes (where we have fewer spikes, but arguably higher quality spikes without interneurons or other "noise"), we obtain a median decoding error of only 6.3 cm, compared to a median decoding error of 8.2 cm for the clusterless approach (see Figure 7.F). However, the area under the curve (AUC) for the cumulative error distribution of the switching Poisson HMM is 87.3%, whereas the clusterless HMM achieved an AUC of 88.1%, although these differences did not appear to be statistically significant[2].

Consequently, we may conclude that the clusterless HMM appears to have similar decoding accuracy compared to the switching Poisson HMM, even though no manual spike sorting was performed for the clusterless HMM.

2. Statistical significance was evaluated with a bootstrap approach where we aggregated the AUCs of 5000 sampled CDFs (5000 for the switching Poisson group, and 5000 for the clusterless group), where each sampled CDF was obtained by randomly sampling, with replacement, from the pooled switching Poisson and clusterless HMM CDFs. Testing for a difference in the means of the two groups with Welch's t-test resulted in a two-sided p-value of 0.37.

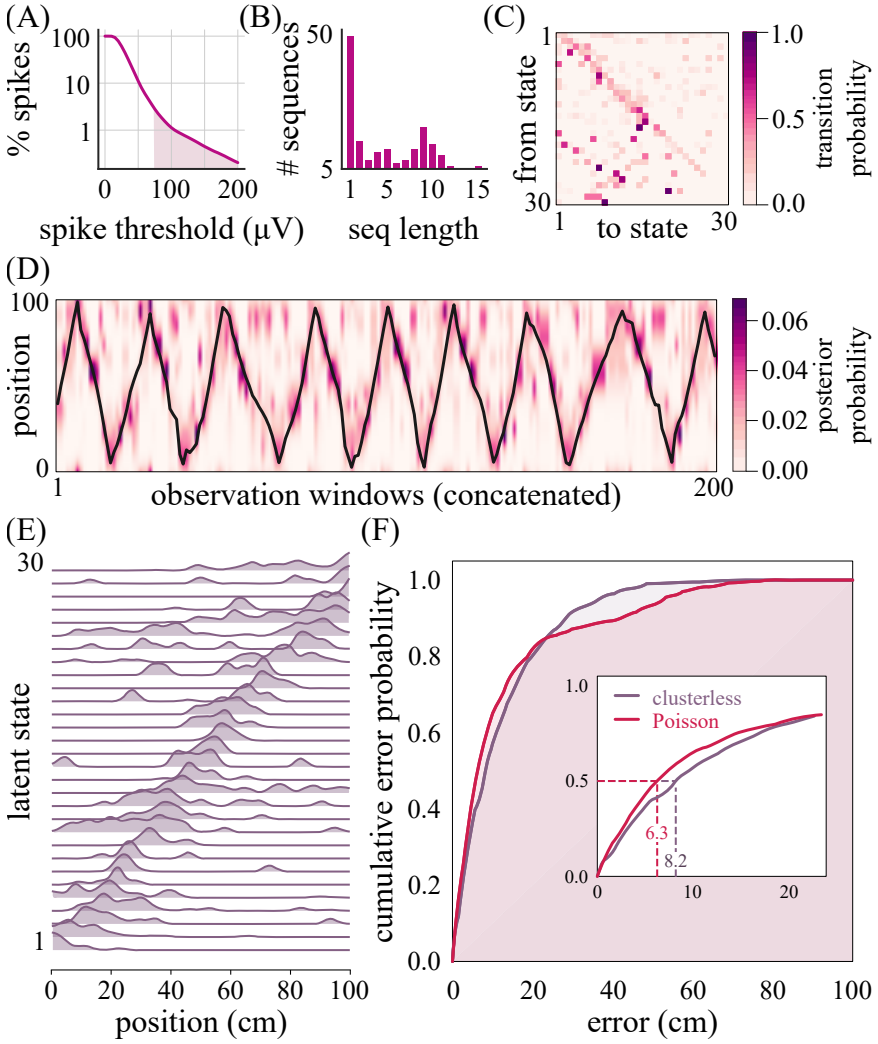

Figure 7: Clusterless hidden Markov model of hippocampal data during RUN. (A) Percentage of spikes retained as a function of detection threshold. We kept all spikes $> 70\ \mu$V (or $\approx 30\%$) (B) Histogram of sequence lengths (in number of observation windows). (C) Transition probability matrix. (D) Decoded posterior distribution over position for several run segments (200×400 ms windows). Black trace indicates animal's actual position. (E) Latent state place fields (lsPFs) relating state space to animal position. (F) Cumulative distribution function of position decoding errors.

5. Discussion and Future Directions

We have presented a new model, the clusterless HMM, which we have shown able to decode position just as well as the standard switching Poisson HMM, without access to the manually-sorted neuron identities. Even though the difference between the two approaches

was not statistically significant here, others have shown that with the right dataset, the clusterless approaches can indeed improve decoding accuracy (Kloosterman et al., 2013; Matano and Ventura, 2018), and this potential improvement may be especially important for the study of severely data-limited analyses such as the study of replay (see e.g., Ackermann et al., 2017; Maboudi et al., 2018, for examples of applying the HMM framework to the identification and characterization of replay events).

In Section 2 we have only presented the sampling approach for spikes from a single (potentially multi-channel) probe. However, the extension to multiple probes is straightforward, and such an approach was applied to the real data example in Section 4. The key assumption for the multiprobe case is that each probe records from an independent subset of neurons, so that sampling can be restricted to neurons associated with each individual probe. Since probes are assumed independent, likelihoods can be computed by taking the product of the likelihoods for individual probes. The multiprobe extension therefore demands a sightly more sophisticated implementation, rather than any real algorithmic changes.

It is also possible to derive update and estimation equations for the mark distribution parameters directly in the Baum–Welch context (instead of using a separate estimation algorithm like the GMM that we have used in Sections 3 and 4), but such an approach would only really be beneficial if we assume that the mark distributions are state dependent.

One interesting consequence of our sampling approach to approximate $P(\mathbf{y}_t \mid S_t)$ is that the data likelihood is no longer guaranteed to increase during each iteration of the EM algorithm (unless approximations are reused as suggested in Algorithm 1 in Appendix A). When we have found that the likelihood decreased during an iteration, we did not update the model parameters. Instead, we would repeat the iteration with a new set of samples, and if we again did not find an improvement in the likelihood, we would terminate the EM process. In practice, the early iterations made the biggest differences to the final likelihood, and those early iterations never seemed to result in a decrease in likelihood, so that whether we were to terminate immediately upon seeing a decrease in likelihood or whether we were to do some smarter iterative approach, it didn't really seem to significantly affect the final results or model parameters.

Even though our clusterless HMM was able to recover the approximate parameters for the simulated dataset, and even though it clearly recovered temporal and task-relevant structure in the real dataset, there are a number of remaining challenges, the most important of which may be the high computational cost of the sampling approach that we have taken here. The model took about 6 hours to fit to the real dataset used here, which is already a relatively small dataset by today's standards. However, several improvements could be made to speed up the computation time. For example, the samples in (6) are assumed to be independent, and so their calculation can be trivially parallelized. Some of the samples can also potentially be re-used in the forward backward algorithm (for example, the calculation of $\alpha_j(t+1)$ and $\beta_j(t)$ both make use of $P(\mathbf{y}_{t+1} \mid S_t = j)$; see the Appendix). It is however fairly challenging to determine how many samples we need for sufficiently accurate estimates of $P(\mathbf{y}_t \mid S_t)$, since it depends on many factors, including the degree of separability of the neuron clusters, the dimensionality of the feature space, the underlying firing rates of the neurons, and so on. Deriving an alternative way to approximate $P(\mathbf{y}_t \mid S_t)$, or establishing some bounds or guidelines to determine the number of samples needed, is

certainly a worthwhile future endeavor. Nevertheless, even with the relatively slow computation time, the clusterless HMM already provides a unique approach to the clusterless analysis of replay or other events where behavioral correlates are not available (e.g., during sleep), necessitating the use of unsupervised approaches.

Acknowledgments

EA would like to acknowledge the helpful discussions with E. Buchanan, K. Kay, and X. Deng during the conception of this project.

Animal Use

One male Long Evans rat (Charles River Laboratories) was implanted with a micro-drive array with independently adjustable tetrodes (implant coordinates3.66 mm AP and 2.4 mm ML relative to bregma). Tetrodes were slowly lowered into hippocampal CA1 over a period of about one week. Tetrode locations were verified by characteristic LFP waveforms attributed to the target area in conjunction with estimated tetrode depth. All experiments were approved by the Rice University Institutional Animal Care and Use Committee's guidelines and adhered to National Institute of Health guidelines.

Code Availability

A modified version of the Python package `hmmlearn` (`https://github.com/hmmlearn/hmmlearn`) has been made available at `https://github.com/nelpy/hmmlearn`. This modified package implements the multiprobe parameter estimation and decoding of our clusterless HMM with multivariate normal mark distributions.

Appendix A: Parameter Estimation Algorithm

The iterative Baum–Welch parameter estimation procedure for our clusterless HMM is given in Algorithm 1. Algorithm 1 makes use of Algorithm 3 to approximate $P(\mathbf{Y}_t \,|\, S_t)$, and Algorithm 2 to calculate $\mathbb{E}[V_n^{(j)}]$.

Algorithm 1 Clusterless HMM parameter estimation.

inputs: sequence of observations: $(\mathbf{y}_t)_{t=1}^T$, current or initial model parameters: $\boldsymbol{\Theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\Lambda}\}$, and mark distribution parameters (e.g., $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^N$)
outputs: updated model parameters $\boldsymbol{\Theta} = \{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\Lambda}\}$
for EM algorithm iteration $l = 1$ to L (or until convergence criterion reached) **do**
  compute $p(\mathbf{y}_t \,|\, S_t = j)$ according to Algorithm 3, for $t = 1, \ldots, T$ and $j = 1, \ldots, Z$
  compute $\mathbb{E}[V_n^{(j)}(t)]$ according to Algorithm 2, for $t = 1, \ldots, T$; $j = 1, \ldots, Z$; and $n = 1, \ldots, N$
  for state $j = 1$ to Z **do**
    update prior state distribution $\boldsymbol{\pi}$ according to (14)
    for neuron $n = 1$ to N **do**
      update rate $\hat{r}^n\big|_{S_t=j} \equiv \boldsymbol{\Lambda}_{jn}$ according to (11)
    end for
    for state $i = 1$ to Z **do**
      update transition probability $\mathbf{A}_{ij}$ according to (15)
    end for
  end for
end for

Algorithm 2 Calculation of $\mathbb{E}[V_n^{(j)}]$.

inputs: sequence of observations: $(\mathbf{y}_t)_{t=1}^T$, current or initial rate matrix: $\boldsymbol{\Lambda}$, and mark distribution parameters (e.g., $\{\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n\}_{n=1}^N$)
outputs: $\mathbb{E}[V_n^{(j)}(t)]$ for $t = 1, \ldots, T$; $j = 1, \ldots, Z$; and $n = 1, \ldots, N$
for time window $t = 1$ to T **do**
  observe $\mathbf{y}_t = \big(m_k\big)_{k=1}^{K(t)}$ in time window $(t-1, t]$
  for state $j = 1$ to Z **do**
    $\mathbf{r} \leftarrow j$th row of $\boldsymbol{\Lambda}$; $\rho \leftarrow \mathbf{r}/\|\mathbf{r}\|_1$
    for neuron $n = 1$ to N **do**
      compute $\mathbb{E}[V_n^{(j)}(t)]$ according to (12)
    end for
  end for
end for

Algorithm 3 Sampling approach to approximate $p(\mathbf{y}_t \,|\, S_t)$.

inputs: sequence of observations: $(\mathbf{y}_t)_{t=1}^{T}$, current or initial rate matrix: $\mathbf{\Lambda}$, and mark distribution parameters (e.g., $\{\boldsymbol{\mu}_n, \mathbf{\Sigma}_n\}_{n=1}^{N}$)
outputs: $p(\mathbf{y_t} \,|\, S_t)$ for $t = 1, \ldots, T$ and $S_t = 1, \ldots, Z$
for time window $t = 1$ to T **do**
  observe $\mathbf{y}_t = \big(m_k\big)_{k=1}^{K(t)}$ in time window $(t-1, t]$
  for state $j = 1$ to Z **do**
    $\mathbf{r} \leftarrow j$th row of $\mathbf{\Lambda}$; $\rho \leftarrow \mathbf{r}/\|\mathbf{r}\|_1$
    for sample $i = 1$ to M **do**
      sample $\mathbf{Q}^{(i)} \sim P_{\mathbb{I}^K|\mathbf{r}}(\mathbb{I}^K \,|\, \mathbf{r})$:
      for mark $k = 1$ to $K(t)$ **do**
        sample $[\mathbf{Q}^{(i)}]_k \sim \text{Multinom}(1, \rho)$
      end for
      for neuron $n = 1$ to N **do**
        calculate $V_n^{(i)}$ according to (8)
      end for
    end for
    approximate $b_j(\mathbf{y}_t) \equiv p(\mathbf{y}_t \,|\, S_t = j) = p(\mathbf{y}_t \,|\, \mathbf{r})$ according to (6)
  end for
end for

Appendix B: Baum–Welch Algorithm

The Baum–Welch algorithm makes use of the forward-backward algorithm to compute the posterior marginals of all hidden state variables given a sequence of observations (Bilmes et al., 1998). That is, it computes $P(S_t \,|\, \mathbf{y}_{1:T})$.

Following standard notation (and for notational simplicity), we define $b_j(\mathbf{y}_t) \equiv p(\mathbf{y}_t \,|\, S_t = j)$, which we evaluate by sampling as described in Section 2.3.

The forward algorithm gives us a way to compute the probability of seeing the partial sequence $\mathbf{y}_1, \ldots, \mathbf{y}_t$ and ending up in state i at time t. That is,

$$\begin{aligned}
\alpha_i(t) &= p\big(\mathbf{y}_1, \ldots \mathbf{y}_t, S_t = i \,\big|\, \mathbf{r}_{1:t}; \{\boldsymbol{\mu}\}, \{\mathbf{\Sigma}\}\big) \\
&= p\big(\mathbf{Y}_{1:t}, S_t = i \,\big|\, \mathbf{r}_{1:t}\big) \\
&= p\big(\big\{m_1^{[1]}, \ldots m_{k(1)}^{[1]}\big\}, \ldots \big\{m_1^{[t]}, \ldots m_{k(t)}^{[t]}\big\}, S_t = i \,\big|\, \mathbf{r}_{1:t}\big) \qquad (17)
\end{aligned}$$

where $\mathbf{y}_t$ denotes the tth observation window, and where $m_k^{[t]}$ denotes the kth mark in time window t. In particular, $\alpha_i(t)$ may be recursively computed as follows:

1. $\alpha_i(1) = \boldsymbol{\pi}_i b_i(\mathbf{y}_1)$
2. $\alpha_j(t+1) = \big[\sum_{i=1}^{Z} \alpha_i(t) a_{ij}\big] b_j(\mathbf{y}_{t+1})$
3. $p(\mathbf{Y} \,|\, \mathbf{r}) = \sum_{i=1}^{Z} \alpha_i(T)$

The backward procedure is similarly used to compute

$$\begin{aligned}\beta_i(t) &= p\big(\mathbf{y}_{t+1}, \dots \mathbf{y}_T \,\big|\, S_t = i, \mathbf{r}_{t+1:T}; \{\boldsymbol{\mu}\}, \{\boldsymbol{\Sigma}\}\big) \\ &= p\big(\mathbf{Y}_{t+1:T} \,\big|\, S_t = i, \mathbf{r}_{t+1:T}\big) \\ &= p\big(\{m_1^{[t+1]}, \dots m_{k(t+1)}^{[t+1]}\}, \dots \{m_1^{[T]}, \dots m_{k(T)}^{[T]}\} \,\big|\, S_t = i, \mathbf{r}_{t+1:T}\big)\end{aligned} \tag{18}$$

which is the probability of observing the partial sequence $\mathbf{y}_{t+1} \dots \mathbf{y}_T$ given that we were in state i at time t.

Similar to before, $\beta_i(t)$ may be recursively computed as follows:

1. $\beta_i(T) = 1$
2. $\beta_i(t) = \sum_{j=1}^{Z} a_{ij} b_j(\mathbf{y}_{t+1}) \beta_j(t+1)$
3. $p(\mathbf{Y} \,|\, \mathbf{r}) = \sum_{i=1}^{Z} \beta_i(1) \boldsymbol{\pi}_i b_i(\mathbf{y}_1)$

Having defined the forward and backward passes, we may now turn our attention to the parameter estimation problem more directly.

In particular, we define

$$\gamma_i(t) = p(S_t = i \,|\, \mathbf{Y}, \mathbf{r}) \tag{19}$$

which can be shown (by Markovian conditional independence) to be equal to

$$\gamma_i(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{Z} \alpha_j(t)\beta_j(t)}.$$

Let

$$\xi_{ij}(t) = p(S_t = i, S_{t+1} = j \,|\, \mathbf{Y}, \mathbf{r}) \tag{20}$$

which we can show to be equal to

$$\xi_{ij}(t) = \frac{\gamma_i(t) a_{ij} b_j(\mathbf{y}_{t+1}) \beta_j(t+1)}{\beta_i(t)}.$$

The parameter estimation updates then follow naturally in terms of γ_i and ξ_{ij} as described in Section 2.4. More specifically, the expected relative frequency spent in state i at time $t = 1$ gives us an estimate for $\boldsymbol{\pi}$:

$$\hat{\boldsymbol{\pi}}_i = \gamma_i(1), \tag{21}$$

while the rate estimates are updated according to

$$\hat{r}_t^n\big|_{S_t = j} = \frac{\sum_{t=1}^{T} \gamma_j(t) \mathbb{E}\big[V_n(t)\big]}{\sum_{t=1}^{T} \gamma_j(t)} \tag{22}$$

and the transition probabilities are updated according to

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}. \tag{23}$$

The update equations for multiple sequences of observations are easy to derive (see e.g., Rabiner, 1989, for details).

References

Etienne Ackermann, Caleb Kemere, Kourosh Maboudi, and Kamran Diba. Latent variable models for hippocampal sequence analysis. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 724–728. IEEE, 2017.

Jeff A Bilmes et al. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126, 1998.

Gilles Celeux and Jean-Baptiste Durand. Selecting hidden markov model state number with cross-validated likelihood. *Computational Statistics*, 23(4):541–564, 2008.

Jason E Chung, Jeremy F Magland, Alex H Barnett, Vanessa M Tolosa, Angela C Tooker, Kye Y Lee, Kedar G Shah, Sarah H Felix, Loren M Frank, and Leslie F Greengard. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394, 2017.

Xinyi Deng, Daniel F Liu, Kenneth Kay, Loren M Frank, and Uri T Eden. Clusterless decoding of position from multiunit activity using a marked point process filter. *Neural Computation*, 27(7):1438–1460, 2015. ISSN 1530888X. doi: 10.1162/NECO_a_00744.

Heidi Y Febinger, Alan D Dorval, and John D Rolston. A sordid affair: Spike sorting and data reproducibility. *Neurosurgery*, 82(3):N19–N20, 2018.

Itay Gat and Naftali Tishby. Statistical modeling of cell assemblies activities in associative cortex of behaving monkeys. In *Advances in neural information processing systems*, pages 945–952, 1993.

GL Gerstein and WA Clark. Simultaneous studies of firing patterns in several neurons. *Science*, 143(3612):1325–1327, 1964.

Lauren M Jones, Alfredo Fontanini, Brian F Sadacca, Paul Miller, and Donald B Katz. Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proceedings of the National Academy of Sciences*, 104(47):18772–18777, 2007.

Caleb Kemere, Gopal Santhanam, M Yu Byron, Afsheen Afshar, Stephen I Ryu, Teresa H Meng, and Krishna V Shenoy. Detecting neural-state transitions using hidden markov models for motor cortical prostheses. *Journal of neurophysiology*, 2008.

Fabian Kloosterman, Stuart P Layton, Zhe Chen, and Matthew A. Wilson. Bayesian decoding using unsorted spikes in the rat hippocampus. *Journal of Neurophysiology*, 111(1):217–227, 2013. ISSN 0022-3077. doi: 10.1152/jn.01046.2012. URL `http://www.physiology.org/doi/10.1152/jn.01046.2012`.

Michael S Lewicki. A review of methods for spike sorting: the detection and classification of neural action potentials. *Network: Computation in Neural Systems*, 9(4):R53–R78, 1998.

Scott W Linderman, Matthew J Johnson, Matthew A Wilson, and Zhe Chen. A bayesian nonparametric approach for uncovering rat hippocampal population codes during spatial navigation. *Journal of neuroscience methods*, 263:36–47, 2016.

Ryan J Low, Sam Lewallen, Dmitriy Aronov, Rhino Nevers, and David W Tank. Probing variability in a cognitive map using manifold inference from neural dynamics. *bioRxiv*, page 418939, 2018.

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.

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.

Francesca Matano and Valérie Ventura. Computationally efficient model selection for joint spikes and waveforms decoding. pages 1–16, 2018. ISSN 20869614. doi: 10.14716/ijtech.v6i2.905. URL `http://arxiv.org/abs/1808.01693`.

Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 1989.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. doi: 10.1198/016214506000000302. URL `https://doi.org/10.1198/016214506000000302`.

Valérie Ventura. Spike Train Decoding Without Spike Sorting. *Neural Computation*, 20: 923–963, 2008.

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.

Alex H Williams, Ben Poole, Niru Maheswaranathan, Ashesh K Dhawale, Tucker Fisher, Christopher D Wilson, David H Brann, Eric Trautmann, Stephen Ryu, Roman Shusterman, et al. Discovering precise temporal patterns in large-scale neural recordings through robust and interpretable time warping. *BioRxiv*, page 661165, 2019.