Uncovering spatial topology represented by rat hippocampal population neuronal codes

J Comput Neurosci manuscript No.
(will be inserted by the editor)

Uncovering spatial topology represented by rat hippocampal population neuronal codes

Zhe Chen · Fabian Kloosterman · Emery N. Brown · Matthew A. Wilson

Received: October 27, 2011 / Revised: January 16, 2011 / Accepted: January 23, 2012

Abstract Hippocampal population codes play an important role in representation of spatial environment and spatial navigation. Uncovering the internal representation of hippocampal population codes will help understand neural mechanisms of the hippocampus. For instance, uncovering the patterns represented by rat hippocampus (CA1) pyramidal cells during periods of either navigation or sleep has been an active research topic over the past decades. However, previous approaches to analyze or decode firing patterns of population neurons all assume the knowledge of the place fields, which are estimated from training data a priori. The question still remains unclear how can we extract information from population neuronal responses either without a priori knowledge or in the presence of finite sampling constraint. Finding the answer to this question would leverage our ability to examine the population neuronal codes under different experimental conditions. Using rat hippocampus as a model system, we attempt to uncover the hidden "spatial topology" represented by the hippocampal population codes. We develop a hidden Markov model (HMM) and a variational Bayesian (VB) inference algorithm to achieve this computational goal, and we apply the analysis to extensive simulation and experimental data. Our empirical results show promising direction for discovering structural patterns of ensemble spike activity during periods of active navigation. This study would also provide useful insights for future exploratory data analysis of population neuronal codes during periods of sleep.

Z. Chen
Neuroscience Statistics Research Lab
Massachusetts General Hospital, Harvard Medical School, Boston, MA 02114, USA
Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
E-mail: zhechen@mit.edu

F. Kloosterman · M. A. Wilson
Department of Brain and Cognitive Sciences and Picower Institute for Learning and Memory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

E. N. Brown
Massachusetts General Hospital, Harvard Medical School, Boston, MA 02114, USA
Department of Brain and Cognitive Sciences and Harvard-MIT Division of Health and Science Technology, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Keywords Hidden Markov model · Expectation-Maximization · Variational Bayesian inference · Place cells · Population codes · Spatial topology · Force-based algorithm

1 Introduction

1.1 Motivation

Hippocampal population codes play an important role in representation of spatial environment and spatial navigation [O'Keefe and Nadel 1978, Buzsaki 2006]. It is known that the receptive fields of hippocampal pyramidal cells encode information of the position of space, hence those cells are referred to as "place cells" [O'Keefe and Nadel 1978]. Using the multielectrode technique, spiking activity of ensemble hippocampal place cells can be simultaneously recorded from rodents, which enable us to examine the internal representation of the population codes at different behavioral stages [Wilson and McNaughton 1993, Wilson and McNaughton 1994]. One of the goal in exploratory data analysis is to discover the hidden structures or firing patterns of spiking activity from simultaneously recorded hippocampal population neurons, either during periods of active behavior [Wilson and McNaughton 1993, Harris et al. 2003, Foster and Wilson 2006] or during periods of sleep [Wilson and McNaughton 1994, Louie and Wilson 2001, Lee and Wilson 2002, Ji and Wilson 2007]. For instance, finding rodent hippocampus neuronal "replay" [Foster and Wilson 2006, Davidson et al. 2009] or "preplay" patterns [Dragoi and Tonegawa 2011] in cell assemblies during either quiet awakefulness or slow-wake sleep (SWS), as compared to the firing patterns during periods of active navigation, has been an important research topic in recent years [Skaggs and McNaughton 1996, Diba and Buzsaki 2007, Karlsson and Frank 2009]. Two types of neuronal codes were used in previous studies. One is based on temporal code, which assumes that the individual cells of neuronal assembly fire in a specific order when the animal navigates in the spatial environment [Lee and Wilson 2002, Ji and Wilson 2007]. The other is based on rate code, which assumes that the spiking activity of population cells follows a probabilistic rule [Brown et al. 1998, Zemel et al. 1998, Zhang et al. 1998, Davidson et al. 2009]. However, these approaches have some drawbacks. First, all previous approaches rely on the assumption that the receptive fields of population neurons (i.e., place fields of hippocampal pyramidal neurons) are known, which are commonly constructed from empirical training data. This assumption could be problematic since the receptive fields are plastic, thus the empirical internal representation of the stimulus space could change at different stages (e.g., navigation vs. sleep) or at different learning phases (first day vs. second day), or when the shape of the stimulus space changes [Lever et al. 2002, Frank et al. 2004, Wills et al. 2005]. The change in hippocampal place-cell representation is known as *remapping*. Second, if the goal of the analysis is to examine the internal representation of the population codes, we shall assume no or little knowledge about the environment (i.e., either linear track, or T-maze, or open field). This is critically important especially when the firing patterns are examined during SWS or REM sleep periods, or the animal has been exposed to multiple distinct spatial environments before the experimental recording. Meanwhile, noticing the fact that knowing the receptive fields of a real environment is not completely necessary for the replay or preplay analysis,

since the place fields are only a proxy to examine the relative proximity of spatial position in the environment. Therefore, one could imagine the possibility that population neurons encode an internal representation of the "virtual environment" which could be an abstract representation of the real environment. To our best knowledge, very few study has been done in this area in the literature, except for the work by Curto and Itskov [Curto and Itskov 2008]. Specifically, with the same motivation (but completely different methodology) and with no assumption of the hippocampal place fields, Curto and Itskov showed that simply knowing which groups of cells fire together would reveal structure in the stimulus space, which then enables the brain to construct its own internal representations. Put in their words, "*a rather unexplored question is how the output of hippocampal place cells (without access to corresponding place fields) might be used by downstream structures in order to reconstruct position and the underlying space*". In their method, the authors made certain assumptions of the place fields in an open field environment, and identified the cell groups (a group of place cells that collectively fire within a two theta-cyle, or 250 ms time window), and further computed the homology groups and extracted the topological features of the spatial environment, and finally constructed an internal representation of the environment using a graph (that contains a vertex for every cell group and an edge between neighboring cell groups) and a distance metric (that contains distances between any two cell groups).

Motivated by these above-mentioned open questions, we develop a probabilistic generative model and a statistical inference approach to solve the above-mentioned problems. Our approach is different from the method of [Curto and Itskov 2008] in terms of the assumptions of place fields and the use of mathematical tools. Finding the internal representation of hippocampal population codes is viewed as an *unsupervised* learning problem. More precisely, we propose a solution based on a hidden Markov model (HMM) and an associated efficient Bayesian inference procedure. Our computational goal is to infer or uncover the spatial topology represented by the hippocampal population neuronal codes in rodent. It shall be pointed out that the term "spatial topology" used here has a narrower meaning than its conventional sense, it is simply referred to the structure of the stimulus space or behavior sequences underlying the hippocampal population neuronal codes. As a byproduct of our estimation procedure, we also recover the receptive fields of hippocampal population neurons with respect to the virtual environment, which are referred to as the "virtual place fields".

1.2 Overview of methods

Inferring the spatial topology represented by the hippocampal population codes is considered as an inverse problem with missing data [Dabaghian et al. 2008, Dabaghian et al. 2011]. To our best knowledge, very few study has been found in the literature. In this study, we examine this problem from a computational perspective. From a statistical data analysis viewpoint, the observed data are the spiking activity of hippocampal ensemble place cells, whereas the missing data are the hidden trajectory (in the virtual environment) associated with the firing patterns exhibited by the place cells, as well as the neuronal tuning curves with respect to the spatial environment. The unobserved trajectory is treated

as a hidden state, which is assumed to follow a Markovian structure. For simplicity, we also assume that the number of hidden states is finite. To model the dynamical system, HMM is a powerful tool for inferring hidden variables given partially observed data. In the computational neuroscience field, to name a few, HMM [Cappé et al. 2005, Rabiner 1989] has been widely used either for decoding natural stimuli [Jones et al. 2007], or for inferring states of population neurons during periods of SWS [Chen et al. 2009], or for detecting neural-state transition for motor cortical prostheses [Kemere et al. 2008], or for sorting neuronal spikes [Herbst et al. 2008], or for spatial-temporal clustering of neural data [Darmanjian and Principe 2009].

Once the statistical model is determined, two kinds of inference approaches can be considered: one is the maximum likelihood approach [Pawitan 2001], the other is the Bayesian approach [Robert 2001, MacKay 2003, Gelman et al. 2004]. Maximum likelihood estimate is asymptotically optimal and invariant, but it is prone to overfitting in the presence of small sample size. In contrast, Bayesian inference imposes priors (e.g., sparsity, invariance) onto the model, and its estimate is more meaningful and efficient; in addition, the uncertainty of the estimate can be represented by the posterior in place of the point estimate. There are various Bayesian inference methods available in the literature [Scott 2002], such as the Markov chain Monte Carlo (MCMC) [Gilks et al. 1995, Rydén 2008], Laplace approximation [MacKay 2003], and variational methods [MacKay 2003, Bishop 2006]. Specifically, in contrast to the MCMC methods, variational Bayesian (VB) methods are more computationally appealing, and they have been proposed for learning a number of statistical models [Beal 2003, Bishop 2006, Katahira et al. 2010, Chen et al. 2011, Wu et al. 2011].

Spatial topology is typically visualized by graphs. Force-based algorithms are a class of algorithms for drawing graphs in a way that the nodes of a graph are positioned in two dimensional or three dimensional space so that all the edges are of more or less equal length and there are as few crossing edges as possible [Tollis et al. 1999]. The force-based algorithms achieve this by assigning forces amongst the set of edges and the set of nodes; the most straightforward method is to assign forces as if the edges were springs (*Hooke's law*) and the nodes were electrically charged particles (*Coulomb's law*). The entire graph is then simulated in the same fashion as a physical system. The forces are applied to the nodes, pulling them closer together or pushing them further apart. This process is repeated iteratively until the system reaches an equilibrium state (i.e., their relative positions no longer change or change very little from one iteration to the next). The physical interpretation of this equilibrium state is that all the forces are in mechanical equilibrium.

Our computational approach consists of two steps: first, infer the unknown parameters of the HMM using VB inference; second, infer the spatial topology of the animal behavior within the environment based on the parameters of the HMM using a force-based algorithm. The rest of the paper is organized as follows. Section 2 presents the background of the finite-state HMM. Section 3 presents the VB inference algorithm for HMM. Section 4 introduces the force-based algorithm for visualizing the spatial topology. Section 5 presents results from a number of computer simulations and experimental data. Interpretations and implications of these results are discussed in detail. Finally, in Section 6 we present some discussions on important issues and conclude the paper in Section 7.

2 Finite-state hidden Markov model

Let us consider a discrete-time homogenous Markov chain. By discrete time, we assume that the time is evenly discretized into fixed-length intervals, which have time indices $t = 1, \ldots, T$. The standard HMM is characterized by three elements: *transition probability, emission probability*, and *initial state probability.*

- The initial probability of state is denoted by a vector $\boldsymbol{\pi} = \{\pi_i\}$, where $\pi_i = \Pr(S_0 = i)$ $(i = 1, \ldots, m)$. Without loss of generality, we assume that the discrete variable $S_t \in \{1, \ldots, m\}$, and size of the discrete state is $dim\{S\} = m$.
- The m-by-m transition probability matrix is written as

$$\boldsymbol{P} = \begin{pmatrix} P_{11} & P_{12} & \ldots & P_{1m} \\ P_{21} & P_{22} & \ldots & P_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ P_{m1} & P_{m2} & \ldots & P_{mm} \end{pmatrix} \tag{1}$$

  with P_{ij} corresponding to the transition (conditional) probabilities from state i to state j.
- For c-th cell, the Poisson spike counts $y_{c,t}$ observed at the t-th time bin follows products of exponentiated Poisson distributions (denoted by `Poi`)

$$\begin{aligned} p(y_{c,t}|S_t) &= \prod_{i=1}^{m} p(y_{c,t}|S_t = i)^{S_{t,i}} \\ &= \prod_{i=1}^{m} \texttt{Poi}(y_{c,t}|\lambda_{ic}, S_t = i)^{S_{t,i}} \\ &= \prod_{i=1}^{m} \left(\frac{\exp(-\lambda_{ic})\lambda_{ic}^{y_{c,t}}}{y_{c,t}!} \right)^{S_{t,i}} \end{aligned} \tag{2}$$

  where the exponent $S_{t,i}$ denotes a Kronecker delta, i.e., $S_{t,i} = 1$ if and only if $S_t = i$. The $\lambda_{ic} \geq 0$ denotes the rate parameter for cell c at the i-th hidden state. Given all $c = 1, \ldots, C$ cells, the emission probability for the i-state is given by $\prod_{c=1}^{C} p(y_{c,t}|S_t = i)^{S_{t,i}}$.

Let $\boldsymbol{\Lambda} = \{\lambda_{ic}\}$ be an m-by-C matrix, and let $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})$ denote all the unknown parameters. Under the assumption of Poisson distribution for spike counts, the observations $\mathbf{y}_t$ at different time indices t are mutually independent, the observed data likelihood is given by

$$\begin{aligned} p(\mathbf{y}_{1:T}|S_{1:T}, \boldsymbol{\theta}) &= \Pr(\mathbf{y}_{1:T}|S_{1:T}, \boldsymbol{\theta}) \\ &= \prod_{t=1}^{T} \prod_{c=1}^{C} \prod_{i=1}^{m} \left(\frac{\exp(-\lambda_{ic})\lambda_{ic}^{y_{c,t}}}{y_{c,t}!} \right)^{S_{t,i}}. \end{aligned} \tag{3}$$

The hidden variables $S_{1:T}$ are treated as the missing data, $\mathbf{y}_{1:T}$ as the observed (incomplete) data, and their combination $\{S_{1:T}, \mathbf{y}_{1:T}\}$ as the complete data, we write the complete data likelihood as

$$\begin{aligned} p(S_{1:T}, \mathbf{y}_{1:T}|\boldsymbol{\theta}) &= p(\mathbf{y}_{1:T}|S_{1:T}, \boldsymbol{\theta}) p(S_{1:T}|\boldsymbol{\theta}) \\ &= \prod_{t=1}^{T} p(\mathbf{y}_t|S_t, \boldsymbol{\theta}) p(S_t|S_{t-1}, \boldsymbol{\theta}). \end{aligned} \tag{4}$$

And the complete data log-likelihood, denoted as $\mathcal{L}$, is derived as (by ignoring the constant)

$$\begin{aligned}\mathcal{L} &= \log p(S_{0:T}, \mathbf{y}_{1:T}|\boldsymbol{\theta}) \\ &= \sum_{t=1}^{T}\sum_{c=1}^{C}\sum_{i=1}^{m}\gamma_t(i)\Big(y_{c,t}\log\lambda_{ic} - \lambda_{ic}\Big) \\ &\quad + \sum_{i=1}^{m}\gamma_1(i)\log\pi_i \\ &\quad + \sum_{t=2}^{T}\sum_{i=1}^{m}\sum_{j=1}^{m}\xi_t(i,j)\log P_{ij}, \end{aligned} \tag{5}$$

where $\gamma_t(i) = \Pr(S_t = i)$ and $\xi_t(i,j) = \Pr(S_{t-1} = i, S_t = j)$.

The maximum likelihood (ML) inference procedure for the standard finite HMM is given by an efficient estimation procedure known as the EM algorithm [Dempster et al. 1977, McLachlan and Krishnan 2008], which is also referred to as the Baum-Welch algorithm [Baum et al. 1970]. The EM algorithm iteratively and monotonically maximizes (or increases) the log-likelihood function given the incomplete data. In the E-step, a forward-backward procedure is used to recursively estimate the hidden state posterior probability. In the M-step, based on the sufficient state statistics (estimated from the E-step), the re-estimation procedure is used to estimate the unknown parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})$. For self-contained purpose, the details of the EM algorithm is presented in Appendix A.

In our current application, the hidden state trajectory corresponds to the animal's *directional* position in the track, the number of states m corresponds to the number of bins used for representing the virtual environment, and the m-by-C matrix $\boldsymbol{\Lambda}$ corresponds to the place fields of ensemble neurons, with each row representing one neuronal tuning curve with respect to the m-dimensional state space.

2.1 Practical estimation issues

The above-described estimation procedure is based on ML estimation. In practice, the ML estimation might not be desirable while dealing with large-scale problems in the presence of small size. For the current estimation problem, assuming that the spatial environment is divided into m non-overlapping regions that are represented by m discrete states. Given the observation of spike counts from C neurons within T time intervals, the size of unknown parameters is $\dim(\boldsymbol{\theta}) = m^2 + mC + m$. In a typical experimental protocol of a spatial navigation task, we have $T \ll \dim(\boldsymbol{\theta})$. Therefore, for an ensemble of $C = 20 \sim 50$ cells and a reasonable size $m = 60 \sim 200$, the parameter space is very large and estimation might be subject to overfitting. On the other hand, since the EM algorithm only searches for the locally optimal solution that are prone to the local optima problem, the initialization of the parameters are important for obtaining for a good solution.

With these practical concerns in mind, it is important to impose certain constraints or priors onto the HMM. In the probabilistic framework, the ML estimation problem is converted into a *maximum a posteriori* (MAP) problem; and the

likelihood inference is replaced by the Bayesian inference. The Bayesian estimate is optimal, especially in the presence of small sample size in statistical inference [Gelman et al. 2004]. For the HMM, the following three types of Bayesian inference approaches can be considered, with gradually increasing model and computational complexity.

- empirical Bayesian: In this approach, strong structural priors can be imposed onto the HMM, such as the entropic prior [Brand 1999, Brand and Ketnaker 2000]. In this case, the MAP solution is straightforward to resolve a modified optimization problem.
- parametric hierarchical Bayesian: In this approach, the parameters of the HMM are assigned with hierarchical priors. The inference algorithm can be based on either MCMC [Scott 2002, Rydén 2008], or ensemble learning [MacKay 1997], or VB-EM [Beal 2003, Ji et al. 2006, McGrory and Titterington 2009].
- nonparametric hierarchical Bayesian: In this approach, the statistical model is treated as a stochastic process with an infinite capacity; a direct extension of the HMM gives rise to the infinite HMM [Beal et al. 2002, Beal 2003]. Statistical inference is based on either Gibbs sampling [Teh et al. 2006] or beam sampling [van Gael et al. 2008].

In the case of space navigation task for rodent, due to behavior prior or constraint, it is reasonable to impose a sparsity structure on $\boldsymbol{P}$, which is either diagonal or banded diagonal. With this imposed constraint, the size of unknown variables reduces dramatically, decreasing from quadratic $\mathcal{O}(m^2)$ to linear $\mathcal{O}(m)$ order.

Another important issue for using the HMM is to determine m—the size of hidden states. A naive solution is to empirically choose different values of m, and then conduct model selection based on certain statistical criteria. However, this solution is not necessarily effective since the EM algorithm has the local minimum problem and it is dependent on the initialization of the parameters. Alternatively, the natural solution is to learn all unknown parameters $\boldsymbol{\theta} = \{m, \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}\}$ from the observed data. In this paper, for the purpose of reducing computational complexity and gaining empirical insights in the first-round investigation, we fix the model size or the number of the hidden states in the inference procedure. We will revisit the model selection issue in Section 6.

3 Variational Bayesian inference for hidden Markov model

In the literature, the VB inference has been used for HMM in various problem settings [MacKay 1997, Beal 2003, Ji et al. 2006, McGrory and Titterington 2009, Katahira et al. 2010]. The advantage of VB inference lies in its computational efficiency for Bayesian inference. To avoid model overfitting in the ML estimation, instead of maximizing the log-likelihood function $\log p(\mathbf{y}_{1:T}|\boldsymbol{\theta})$, the objective of VB inference is to maxi-

mize the marginal log-likelihood or its lower bound

$$\begin{aligned}
\log p(\mathbf{y}_{1:T}) &= \log \int d\boldsymbol{\pi} \int d\boldsymbol{P} \int d\boldsymbol{\Lambda} \sum_{S_{1:T}} p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) p(\mathbf{y}_{1:T}, S_{1:T} | \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \\
&= \log \int d\boldsymbol{\pi} \int d\boldsymbol{P} \int d\boldsymbol{\Lambda} \sum_{S_{1:T}} q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T}) \frac{p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) p(\mathbf{y}_{1:T}, S_{1:T} | \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})}{q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T})} \\
&\geq \int d\boldsymbol{\pi} \int d\boldsymbol{P} \int d\boldsymbol{\Lambda} \sum_{S_{1:T}} q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T}) \log \frac{p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) p(\mathbf{y}_{1:T}, S_{1:T} | \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})}{q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T})} \\
&= \Big\langle \log p(\mathbf{y}_{1:T}, S_{1:T}, \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \Big\rangle_q + \mathcal{H}_q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T}) \equiv \mathcal{F}(q) \qquad (6)
\end{aligned}$$

where $q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T})$ is called the variational posterior distribution that approximates the joint posterior of the hidden state and parameter $p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T} | \mathbf{y}_{1:T})$. The term $\mathcal{H}_q$ of Eq. (6) represents the entropy of the distribution q, and $\mathcal{F}$ is called the free energy (in light of statistical physics).

By assuming a factorial form of variational posterior distribution

$$\begin{aligned}
q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T}) &= q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) q(S_{1:T}) \\
&\approx p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T} | \mathbf{y}_{1:T}) \qquad (7)
\end{aligned}$$

Eq. (6) can be further simplified as

$$\begin{aligned}
\log p(\mathbf{y}_{1:T}) &\geq \log \int d\boldsymbol{\pi} \int d\boldsymbol{P} \int d\boldsymbol{\Lambda} \\
&\qquad \sum_{S_{1:T}} q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T}) \log \frac{p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) p(\mathbf{y}_{1:T}, S_{1:T} | \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})}{q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T})} \\
&= \log \int d\boldsymbol{\pi} \int d\boldsymbol{P} \int d\boldsymbol{\Lambda} \sum_{S_{1:T}} q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \\
&\qquad \left[\log \frac{p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})}{q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})} + \sum_{S_{1:T}} q(S_{1:T}) \log \frac{p(\mathbf{y}_{1:T}, S_{1:T} | \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})}{q(S_{1:T})} \right] \\
&\equiv \mathcal{F}(q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}), q(S_{1:T})) \qquad (8)
\end{aligned}$$

where for notation simplicity we have made the conditional on $\mathbf{y}_{1:T}$ in the variational posteriors $q(\cdot)$ implicit.

To maximize the free energy $\mathcal{F}(q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}), q(S_{1:T}))$, we optimize alternatingly with respect to its arguments $q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})$ and $q(S_{1:T})$, which will be done in the VB-M and VB-E steps, respectively.

3.1 VB-M step

In the VB-M step, taking functional derivatives of $\mathcal{F}$ with respect to $q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})$ yields

$$\begin{aligned}
\log q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) &\propto \log p(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \big\langle \log p(\mathbf{y}_{1:T}, S_{1:T} | \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \big\rangle_{q(S_{1:T})} \\
&\propto \log p(\boldsymbol{\pi}) + \log p(\boldsymbol{P}) + \log p(\boldsymbol{\Lambda}) + \big\langle \log p(S_1 | \boldsymbol{\pi}) \big\rangle_{q(S_1)} \\
&\quad + \big\langle \log p(S_{2:T} | S_1, \boldsymbol{P}) \big\rangle_{q(S_{1:T})} + \big\langle \log p(\mathbf{y}_{1:T} | S_{1:T}, \boldsymbol{\Lambda}) \big\rangle_{q(S_{1:T})} \qquad (9)
\end{aligned}$$

We further impose a factorial form onto the variational posterior of the parameters

$$q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) = q(\boldsymbol{\pi})q(\boldsymbol{P})q(\boldsymbol{\Lambda}) \tag{10}$$

To derive individual variational posteriors, we assume appropriate conjugate prior in order to get an analytic form of the posterior.

For the initial state probability $\boldsymbol{\pi}$, we assume a conjugate Dirichlet prior (denoted by `Dir`):

$$\begin{aligned} p(\boldsymbol{\pi}) &= \texttt{Dir}\Big(\{\pi_1, \ldots, \pi_m\}|\mathbf{u}^{(\pi)}\Big) \\ &= \frac{\Gamma(u_0^{(\pi)})}{\prod_{i=1}^m \Gamma(u_i^{(\pi)})} \prod_{i=1}^m \pi_i^{u_i^{(\pi)}-1} \end{aligned} \tag{11}$$

where $\mathbf{u}^{(\pi)} = [u_1^{(\pi)}, \ldots, u_m^{(\pi)}]$, $u_i^{(\pi)} \geq 0$, and $u_0^{(\pi)} = \sum_{i=1}^m u_i^{(\pi)}$ denotes the strength of the Dirichlet distribution. From the Bayes rule, it is inferred that the posterior is also a Dirichlet distribution:

$$q(\boldsymbol{\pi}) = \texttt{Dir}\Big(\{\pi_1, \ldots, \pi_m\}|\{w_1^{(\pi)}, \ldots, w_m^{(\pi)}\}\Big) \tag{12}$$

where $w_i^{(\pi)} = u_i^{(\pi)} + q_S(S_1 = i) = u_i^{(\pi)} + \gamma_1(i)$.

Similarly, we can derive the posterior for the transition probability matrix $\boldsymbol{P}$ as the products of posteriors of its row vectors

$$\begin{aligned} q(\boldsymbol{P}) &= \prod_{i=1}^m q(\boldsymbol{P}_i) \\ &= \prod_{i=1}^m \texttt{Dir}\Big(\{P_{i1}, \ldots, P_{im}\}|\{w_{i1}^{(P)}, \ldots, w_{im}^{(P)}\}\Big) \end{aligned} \tag{13}$$

where $w_{ij}^{(P)} = u_{ij}^{(P)} + \sum_{t=2}^T q_S(S_{t-1} = i, S_t = j) = u_{ij}^{(P)} + \sum_{t=2}^T \xi_t(i, j)$.

Given the Poisson likelihood for the rate parameters $\boldsymbol{\Lambda} = \{\lambda_{ic}\}$, we assume a conjugate gamma prior (denoted by `Gam`) for each state i (shared by all cell indices $c = 1, \ldots, C$):

$$\begin{aligned} p(\lambda_{ic}) &= \texttt{Gam}\Big(\alpha_i^{(\lambda)}, \beta_i^{(\lambda)}\Big) \\ &= \frac{(\beta_i^{(\lambda)})^{\alpha_i^{(\lambda)}}}{\Gamma(\alpha_i^{(\lambda)})} \lambda_{ic}^{\alpha_i^{(\lambda)}-1} e^{-\beta_i^{(\lambda)}\lambda_{ic}} \end{aligned} \tag{14}$$

where $\alpha_i^{(\lambda)} > 0$ and $\beta_i^{(\lambda)} > 0$ are the *shape* and *inverse scale* parameters, respectively.[1] The above gamma distribution has a mean $\alpha_i^{(\lambda)}/\beta_i^{(\lambda)}$ and variance $\alpha_i^{(\lambda)}(\beta_i^{(\lambda)})^{-2}$. Correspondingly, the rate parameters follow a gamma posterior

$$\begin{aligned} q(\boldsymbol{\Lambda}) &= \prod_{i=1}^m \prod_{c=1}^C q(\lambda_{ic}) \\ &= \prod_{i=1}^m \prod_{c=1}^C \texttt{Gam}\Big(C\alpha_i^{(\lambda)} + \sum_{t=1}^T y_{c,t}\gamma_t(i), C\beta_i^{(\lambda)} + l_i\Big) \end{aligned} \tag{15}$$

[1] The Jefferey's improper prior corresponds to a limiting case of the gamma distribution, with a shape parameter of 0.5 and inverse scale parameter of 0.

where $l_i = \sum_{t=1}^{T} \gamma_t(i)$.

3.2 VB-E step

In the VB-E step, the variational joint posterior of the hidden state is given by

$$q(S_{1:T}) = \prod_{i=1}^{m} \pi_i^{\gamma_1(i)} \prod_{t=2}^{T} \prod_{i=1}^{m} \prod_{j=1}^{m} P_{ij}^{\xi_t(i,j)} \prod_{t=1}^{T} \prod_{i=1}^{m} \prod_{c=1}^{C} \mathtt{Poi}(y_{c,t}|\lambda_{ic}, S_t = i)^{\gamma_t(i)} \quad (16)$$

Maximizing $\mathcal{F}$ precedes by taking a functional derivative with respect to $q(S_{1:T})$, which yields

$$\begin{aligned} \log q(S_{1:T}) &= \big\langle \log p(S_{1:T}, \mathbf{y}_{1:T}|\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \big\rangle_{q(\boldsymbol{\pi})q(\boldsymbol{P})q(\boldsymbol{\Lambda})} - \log Z(\mathbf{y}_{1:T}) \\ &= \sum_{i=1}^{m} \hat{S}_{1,i} \big\langle \log \pi_i \big\rangle_{q(\boldsymbol{\pi})} + \sum_{t=2}^{T} \sum_{i=1}^{m} \sum_{j=1}^{m} \hat{S}_{t-1,i} \hat{S}_{t-1,j} \big\langle \log P_{ij} \big\rangle_{q(\boldsymbol{P})} \\ &\quad + \sum_{t=1}^{T} \sum_{c=1}^{C} \sum_{i=1}^{m} \hat{S}_{t,i} \big\langle -\lambda_{ic} + y_{c,t} \log \lambda_{ic} \big\rangle_{q(\boldsymbol{\Lambda})} - \log Z(\mathbf{y}_{1:T}) \end{aligned} \quad (17)$$

where $\hat{S}_{t,i} = \mathbb{E}_{q(S_{1:T})}[S_t = i] = q_S(S_t = i|\mathbf{y}_{1:T}) \equiv \gamma_t(i)$ and $\hat{S}_{t-1,i}\hat{S}_{t,j} = \mathbb{E}_{q(S_{1:T})}[S_{t-1} = i, S_t = j] = q_S(S_{t-1} = i, S_t = j|\mathbf{y}_{1:T}) \equiv \xi_t(i,j)$ will be computed from the forward-backward algorithm (Appendix A). The last term of Eq. (17), $\log Z(\mathbf{y}_{1:T})$, is a normalization constant that is independent of the variational posterior. To compute the first term of Eq. (17), we have

$$\begin{aligned} \big\langle \log \pi_i \big\rangle_{q(\boldsymbol{\pi})} &= \int \mathtt{Dir}(\boldsymbol{\pi}|\mathbf{u}^{(\pi)}) \log \pi_i d\boldsymbol{\pi} \\ &= \psi\Big(u_i^{(\pi)}\Big) - \psi\Big(\sum_{i=1}^{m} u_i^{(\pi)}\Big) \end{aligned} \quad (18)$$

where ψ is the *diagamma function*. To compute the second term of Eq. (17), we have

$$\begin{aligned} \big\langle \log P_{ij} \big\rangle_{q(\boldsymbol{P})} &= \int \mathtt{Dir}(P_{ij}|u_{ij}^{(P)}) \log P_{ij} d\boldsymbol{P} \\ &= \psi\Big(u_{ij}^{(P)}\Big) - \psi\Big(\sum_{i=1}^{m} u_{ij}^{(P)}\Big) \end{aligned} \quad (19)$$

To compute the third term of Eq. (17), we have

$$\begin{aligned} \big\langle -\lambda_{ic} + y_{c,t} \log \lambda_{ic} \big\rangle_{q(\boldsymbol{\Lambda})} &= \int \mathtt{Gam}\Big(\lambda_{ic}\Big|C\alpha_i^{(\lambda)} + \sum_t y_{c,t}\gamma_t(i), C\beta_i^{(\lambda)} + l_i\Big) \\ &\qquad \times(-\lambda_{ic} + y_{c,t} \log \lambda_{ic}) d\lambda_{ic} \\ &= -\frac{C\alpha_i^{(\lambda)} + \sum_t y_{c,t}\gamma_t(i)}{C\beta_i^{(\lambda)} + l_i} \\ &\quad + y_{c,t}\psi\Big(C\alpha_i^{(\lambda)} + \sum_t y_{c,t}\gamma_t(i)\Big) \\ &\quad - y_{c,t} \log(C\beta_i^{(\lambda)} + l_i)\Big) \end{aligned} \quad (20)$$

From Eqs. (18) through (20), we update the new initial state probability as

$$\begin{aligned}\tilde{\boldsymbol{\pi}} &= \{\tilde{\pi}_i\} \\ &= \exp\left\langle \log \pi_i \right\rangle_{q(\boldsymbol{\pi})} \\ &= \exp\left(\psi(w_i^{(\pi)}) - \psi(\sum_{i=1}^{m} w_i^{(\pi)})\right),\end{aligned} \tag{21}$$

and update the new state transition probability as [2]

$$\begin{aligned}\tilde{\boldsymbol{P}} &= \{\tilde{P}_{ij}\} \\ &= \exp\left\langle \log P_{ij} \right\rangle_{q(\boldsymbol{P})} \\ &= \exp\left(\psi(w_{ij}^{(P)}) - \psi(\sum_{i=1}^{m} w_{ij}^{(P)})\right),\end{aligned} \tag{22}$$

and update the new emission probability as

$$\begin{aligned}\Pr(\mathbf{y}_t|\{\tilde{\lambda}_{ic}\}, S_t = i) &= \prod_{c=1}^{C} \exp\left(\left\langle -\lambda_{ic} + y_{c,t}\log\lambda_{ic}\right\rangle_{q(\boldsymbol{\Lambda})}\right) \\ &= \prod_{c=1}^{C} \exp\left(-\frac{C\alpha_i^{(\lambda)} + \sum_t y_{c,t}\gamma_t(i)}{C\beta_i^{(\lambda)} + l_i}\right) \\ &\quad \times \exp\left(y_{c,t}\psi\Big(C\alpha_i^{(\lambda)} + \sum_t y_{c,t}\gamma_t(i)\Big) - y_{c,t}\log(C\beta_i^{(\lambda)} + l_i)\Big)\right)\end{aligned} \tag{23}$$

The VB-E step further proceeds with the standard forward-backward algorithm using the new parameter $\tilde{\boldsymbol{\theta}} = (\tilde{\boldsymbol{\pi}}, \tilde{\boldsymbol{P}}, \tilde{\boldsymbol{\Lambda}})$ (computed from Eqs. 31 and 34, Appendix A).

3.3 Computation of the lower bound $\mathcal{F}$

Upon completing every iteration of the VB-E step, we compute the free energy shown in Eq. (8), which is rewritten here:

$$\begin{aligned}\mathcal{F}(q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}), q(S_{1:T})) &= \int q(\boldsymbol{\pi}) \log \frac{p(\boldsymbol{\pi})}{q(\boldsymbol{\pi})} d\boldsymbol{\pi} + \int q(\boldsymbol{P}) \log \frac{p(\boldsymbol{P})}{q(\boldsymbol{P})} d\boldsymbol{P} \\ &\quad + \int q(\boldsymbol{\Lambda}) \log \frac{p(\boldsymbol{\Lambda})}{q(\boldsymbol{\Lambda})} d\boldsymbol{\Lambda} + \log \tilde{Z}(\mathbf{y}_{1:T}) \\ &\leq \log \tilde{Z}(\mathbf{y}_{1:T})\end{aligned} \tag{24}$$

where the inequality holds because the non-negativeness of the Kullback-Leibler (KL) divergence. Note that the first term $\int q(\boldsymbol{\pi}) \log \frac{p(\boldsymbol{\pi})}{q(\boldsymbol{\pi})} d\boldsymbol{\pi}$ of Eq. (24) measures the negative KL divergence between the variational posterior $q(\boldsymbol{\pi}) = \texttt{Dir}(\pi_1, \ldots, \pi_m | u_1, \ldots, u_m)$

[2] Note that the new probabilities are *sub-normalized probabilities* (due to using the geometric mean instead of the standard arithmetic mean), where $\sum_{i=1}^{m} \tilde{\pi}_i \leq 1$ and $\sum_{j=1}^{m} \tilde{P}_{ij} \leq 1$.

and prior Dirichlet distribution $p(\boldsymbol{\pi}) = \texttt{Dir}(\pi_1, \ldots, \pi_m | u'_1, \ldots, u'_m)$ for vector $\boldsymbol{\pi}$ (similarly, for each row of the matrix $\boldsymbol{P}$ in the second term of Eq. 24)

$$KL_{\texttt{Dir}}(q\|p) = \log \frac{\Gamma(u_0)}{\Gamma(u'_0)} + \sum_{i=1}^{m} \log \frac{\Gamma(u'_i)}{\Gamma(u_i)} + \sum_{i=1}^{m} (u_i - u'_i)\Big(\psi(u_i) - \psi(u'_i)\Big) \quad (25)$$

The third term $\int q(\boldsymbol{\Lambda}) \log \frac{p(\boldsymbol{\Lambda})}{q(\boldsymbol{\Lambda})} d\boldsymbol{\Lambda}$ of Eq. (24) measures the negative KL divergence between two gamma distributions $q = \texttt{Gam}(\alpha_1, \beta_1)$ and $p = \texttt{Gam}(\alpha_2, \beta_2)$, which can be computed analytically

$$KL_{\texttt{Gam}}(q\|p) = \log \left(\frac{\Gamma(\alpha_2)\beta_1^{\alpha_1}}{\Gamma(\alpha_1)\beta_2^{\alpha_2}} \right) + (\alpha_1 - \alpha_2)\Big(\psi(\alpha_1) - \log \beta_1\Big) + \alpha_1 \frac{\beta_2 - \beta_1}{\beta_1} \quad (26)$$

Furthermore, the last term $\log \tilde{Z}(\mathbf{y}_{1:T})$ of Eq. (24) is the new normalization constant that can be estimated from the forward-backward algorithm (Eq. 35, Appendix A); it also corresponds to the estimated marginal log-likelihood of the data (Eq. 36, Appendix A).

3.4 Initialization of priors and hyperparameters

The purpose of conjugate priors is to make the VB inference more tractable. However, the hyperparameters of these priors are designed by user, depending on the users belief on the data. The priors can be either very informative or very uninformative. In that sense, the conjugate prior is still quite general. Obviously, a highly structured solution will require a very specific prior for the desirable solution.

In the previous subsection, the hyperparmeters are assumed to be known or set by the user. In our problem, the hyperparameters are set according to the prior knowledge as follows.

- We set $u_1^{(\pi)} = u_2^{(\pi)} = \cdots = u_m^{(\pi)} = 1/m$, which corresponds to a uniform distribution. If the hyperparameter is smaller than $1/m$, it implies that the solution favors a specific initial state, rather than a uniform solution.
- We set $[u_{i1}^{(P)}, u_{i2}^{(P)}, \ldots, u_{im}^{(P)}] = \alpha^{(P)}[1/m, 1/m, \ldots, 1/m]$, where $\alpha^{(P)}$ denotes the concentration parameter. Values of the concentration parameter above 1 prefer variates that are dense, evenly-distributed distributions (i.e. all probabilities returned are similar to each other). Values of the concentration parameter below 1 prefer sparse distributions (i.e., most of the probabilities returned will be close to 0, and the vast majority of the mass will be concentrated in a few probabilities). We set $\alpha^{(P)} = 0.3$.
- We set $\alpha_i^{(\lambda)} = \beta_i^{(\lambda)} = 0.0001$, and the initial mean of the λ_{ic} is set to be the overall mean firing rate of neuron c, i.e. $\frac{1}{T}\sum_{t=1}^{T} y_{c,t}$.

Alternatively, the hyperparameters $\alpha_i^{(\lambda)}$ and $\beta_i^{(\lambda)}$ can be optimized iteratively by maximizing the log-likelihood or marginal log-likelihood (Appendix C). However, no closed-form solution exists for these hyperparameters.

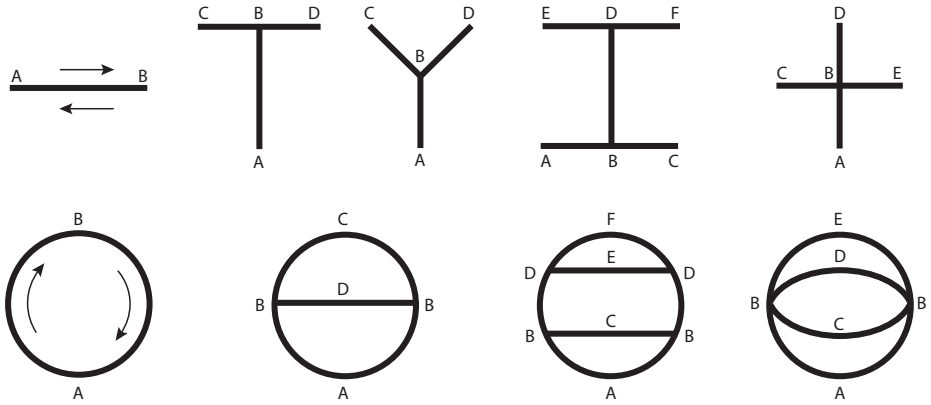

Fig. 1 Examples of experimental space topology explored by animal (For left to right: linear track, T-maze, Y-maze, H-maze, cross maze). The arrow indicates the traveling direction. The first row shows the physical shape of the track, and the second row shows the corresponding (equivalent) topology by considering the bidirectional factor. Note that the T-maze and Y-maze have two bifurcation points (each with 2 possible choices), the H-maze has four bifurcation points (each with 2 possible choices), and the cross-maze has two bifurcation points (each with 3 possible choices).

4 Visualization of spatial topology via force-based algorithm

Spatial topology is a mathematical abstraction of the real environment space. Spatial topology reflects the geometrical structure and spatial relations that are invariant or unaffected by the continuous change of shape or size of figure. For a rodent spatial navigation task in a two-dimensional space, the spatial topology determines the animal's natural behavior. Figure 1 shows a few example experimental spatial topology commonly used in navigation tasks. As seen in Fig. 1, the physical shapes of the experimental tracks (top row) can be converted into the equivalent spatial topology (bottom row) by considering the directional factor; two tracks with physically different shapes (e.g., T-maze vs. Y-maze) share the same spatial topology. The bifurcations of the track increases the complexity of the topology since it introduces multiple pathways connecting one point in the track to another. In representing the space, the spatial environments are often binned and linearized. Note that the linearization strategy is non-unique, and there are multiple ways to discretize and represent the same space.

The inference outcomes of the HMM consist of the estimated state trajectory, the state transition probability matrix, and the tuning curves (place fields) of the population neurons. Particularly, the estimated state trajectory and transition probability matrix reveal important cues about the spatial environment and the animal behavior in that environment. Since the behavior determines the state transition probability, we will use the transition probability matrix to infer the spatial topology of the environment. Specifically, we would like to draw a graph that consists of multiple nodes representing the hidden states, the edges between the graph represents the link between the spatial locations coded by the hidden states, whereas the strengths of the edges reflects the values of transition probability between two states, which not only depends on the spatial topology but also the animal's actual behavior. For instance, for the same spatial topology such

as a linear track, the state transition matrix will be different between a regular back-and-forth navigation without turns and a navigation with frequent turns inside the track. A non-stop navigation between two end points will induce a shifted diagonal-like structure in the state transition matrix, whereas frequent stops and turns inside the track will induce many nonzero off-diagonal elements in the transition matrix.

A direct way to visualize the spatial topology is to draw graphs. A graph displays the geometrical relationship between distinct nodes or different objects via edges. There are various graph-drawing methods, most of which rely on certain distance metrics. In general, a high transition probability implies a short distance between two nodes in the graph. Meanwhile, for the aesthetic reason, it is preferred that all the edges are proportional in length, and there are as few crossing edges as possible. For instance, the classical or nonclassical multidimensional scaling (MDS) methods [Cox and Cox 2001, Borg and Groenen 2005], which are originally used in information visualization for exploring similarities or dissimilarities in data, can be used here for visualizing the relationship between nodes based on a selected distance metric. To do that, we can transform the transition probability matrix into a symmetric distance (or dissimilarity) matrix. However, the choice of transformation is rather ad hoc, and from our practical experiences the visualization of the graph is less satisfactory and the results are more difficult to interpret (results not shown).

Another type of popular graph drawing methods is based on the force-based algorithm [Tollis et al. 1999]. Typically, this type of algorithm is motivated from physics, whereas the nodes are viewed as particles, and the graph is treated as physical (mechanical or electrical) system. At the end of the completing the graph drawing, the total kinetic energy is minimized and the system reaches an equilibrium state. Some publicly available softwares, for instance the Tulip (http://tulip.labri.fr/) and Gephi© (http://gephi.org), provide interfaces to draw aesthetically satisfactory graphs with various levels of user control. The typical force-based algorithms are generally considered to have a $\mathcal{O}(m^3)$ running time, where m is the number of nodes of the graph. It shall be emphasized out that the force-based algorithm is based on iterative optimization, which also has the poor local minimum problem; thus the graph drawing outcome also depends on the initial condition.

In addition to the available public resources, we have also written our own custom MATLAB© (MathWorks, Natick, MA) programs to visualize the spatial topology in either two-dimensional (2D) or three-dimensional (3D) space. The only input for the program is the estimated (with or without thresholding) state transition matrix and the algorithmic convergence criterion (a default value is also set). The pseudocode for the force-based algorithm is given below (**Algorithm 1**).[3]

[3] Online resource: http://en.wikipedia.org/wiki/Force-based_algorithms_(graph_drawing).

Algorithm 1 Pseudocode for the force-based algorithm.

```
Initialize node velocities to (0, 0), initialize node positions randomly.
while non-convergence (i.e., total kinetic energy is greater than desired value) do
  Set total kinetic energy to 0. // running sum of total kinetic energy over all particles
  for each node
      net-force = (0, 0) // running sum of total force on this particular node
      for each other node
          net-force = net-force + Coulomb-repulsion( this node, other node )
      next node
      for each spring connected to this node
          net-force = net-force + Hooke-attraction( this node, spring )
      next spring
  // without damping, it moves forever
      this node.velocity = (this node.velocity + timestep × net-force) × damping
      this node.position = this node.position + timestep × this node.velocity
      total kinetic energy = total kinetic energy + this node.mass × (this node.velocity)^2
  next node
end while
```

Table 1 Summary of all computer simulations.

No.	environment	m	C	T (laps × bin/lap)	remark
1-1	linear track (bidirectional)	62	21	1240 (20×62)	31 bins per direction, without turns
1-2	linear track (bidirectional)	62	21	3720 (20×186)	with turns in both directions
2-1	T-maze	86	21	5070 (15×338)	two bifurcations
2-2	T-maze	86	35	4240 (20×212)	two bifurcations, with turns
3-1	linear track A + T-maze	86	21	8790	A is part of of T, multiple transitions
3-2	linear track A + linear track B	86	21	1075	A and B are gated, one transition

5 Results

5.1 Computer simulations

We have done a variety of computer simulations to verify our analysis. The setup of the simulated experiments is listed in Table 1. In all simulations, we assume that the animal is always in the RUN-mode (i.e., stop periods are excluded), with a 250 ms temporal bin size (about two theta-cycle). For the sake of simulation simplicity, we also assume that the animal runs multiple laps, the running trajectory at each lap is identical (i.e., the overall trajectory is periodic). In addition, the spike activity of ensemble neurons is drawn from a Possion distribution based on the real tuning curves constructed from experimental tracks.

Due to the presence of local maxima, in each experimental condition we run the VB-EM algorithm multiple times, each with different random initializations. We examine and select the results with the free energy criterion. The solution associated with the higher free energy is more likely to be a better solution. However, the free energy criterion alone may not be sufficient, quantitative assessment of the estimated solutions is also necessary.

5.1.1 Quantitative assessment

For computer simulations, we propose two quantitative indices to measure the quality of the estimation results. The first index measures the quality of estimated

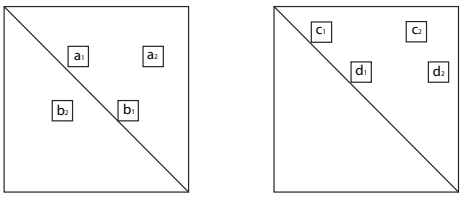

Fig. 2 Cartoon illustration for comparing two matrix-rows that have two dominant off-diagonal elements, where a_1, b_1, c_1, d_1 are the one-column-right-shifted diagonal elements (i.e., column index = row index + 1).

trajectory. It is noted that because of the permutation ambiguity of the state ID, two correct trajectories may exhibit different forms after remapping the state ID. For that reason, we first compute the occupancy time (OT) of each state and then sort these values denoted by a vector **OT**. We then compute the difference between the two vectors

$$D_1 = \frac{\|\mathbf{OT}_{true} - \mathbf{OT}_{est}\|}{T/m}, \tag{27}$$

where $\|\cdot\|$ denotes the L_1 norm of the vector, and the denominator T/m denotes the averaged occupancy time of m states within the total T time bins. The index D_1 is aimed to check the consistency between two state trajectories: when two trajectories are perfectly consistent (upon permutation), $D_1 = 0$. In the presence of the state ID ambiguity, provided that each state has a different occupancy number, after sorting the OT, the consistency of two trajectories can be checked without explicit state remapping.

The second index measures the similarity between the true and estimated state transition matrices. Again, due to permutation ambiguity of the state ID, it is difficult (if not impossible by an exhaustive search) to compare all possible permutations. To illustrate this point, let's consider a simple example shown in Fig. 2. Suppose that we have two matrices, each of them have two rows that have more than one (here, say two) dominant nonzero off-diagonal entries. All entries of the matrix are nonnegative, and each row entries sum to 1. As illustrated in Fig. 2, we denote the row entries as (a_1, a_2) and (b_1, b_2) in the first matrix (say, the true matrix), and denote the row entries as (c_1, c_2) and (d_1, d_2) in the second matrix (say, the estimated matrix), among which a_1, b_1, c_1, d_1 are the one-column-right-shifted diagonal elements. Given the permutation ambiguity, there are two possibilities to compute the matrix row deviation (MRD): one is $MRD_1 = |a_1 - c_1| + |a_2 - c_2| + |b_1 - d_1| + |b_2 - d_2|$, the other is $MRD_2 = |a_1 - d_1| + |a_2 - d_2| + |b_1 - c_1| + |b_2 - c_2|$. Obviously, the one with the smallest MRD value would be a more desirable solution; namely, using the row permutation option associated with the smallest MRD, the estimated matrix will be more similar to the true matrix (at least for the two rows under consideration).

The cartoon example in Fig. 2 is only aimed to illustrate the situation when considering two matrix rows that have two dominant nonzero off-diagonal entries. In a more general setting, when considering to compare n rows in the m-by-m true

matrix (where $n \geq 2$ and $n \ll m$) that have two dominant nonzero off-diagonal entries,[4] there will be $n!$ (the factorial of n) permutation possibilities and we would need to compute a total of $n!$ MRD values. From which, we define the second index as

$$D_2(n) = \min_k \{MRD_k\}, \quad k = 1, \ldots, n! \tag{28}$$

For the sake of computational simplicity, we have ignored the comparison among the remaining $(m - n)$ rows between two matrices. In our case (when only two dominant nonzero off-diagonal elements in each row are considered), it can be shown that $0 \leq D_2(n) \leq n$. In all computer simulations conducted below, we only consider either $n = 2$ or $n = 3$. Since we can only consider a low dimensionality of n, D_2 shall be combined with D_1 as an additional criterion to assess the solution. Alternatively, we can also use the continuity of state trajectory sequences to compare a large chuck of (more than 3) rows between two matrices.

5.1.2 Simulated linear track

In the first simulation scenario, the simulated animal's behavior follows a non-stop back-and-forth navigation (i.e., animal moves from one end of the track to the other end, then returns and the motion repeats). The animal stops nowhere in the middle track (neither at the ends of the track) and makes no turn inside the track. In this case, we would know in advance that the state transition matrix $\boldsymbol{P}$ would have a shifted-diagonal structure. In inference, to impose a linear-track topology preference, in parameter initialization we first set $\boldsymbol{P}$ to have a dominant shifted-diagonal structure; we further add small positive values randomly into the elements of $\boldsymbol{P}$ (to allow other possible state jumps to account for animal's behavior turns inside the track). Each row of $\boldsymbol{P}$ is normalized such that the sum of the entries is 1. In practice, we found that this initialization strategy is very effective and leads to fast algorithmic convergence and good estimation performance.[5]

In the second simulation scenario, the animal still navigates in the same linear track environment, but the animal's behavior is different from the first case in that it now makes a few turns in the middle of the track. For instance, the state sequence in one lap to account for the animal's behavior and moving direction is $[1:15, -15:-1, 1:31, -31:-1, 1:25, -25:-10, 10:31, -31:-1]$ (where the negative sign indicates the reverse direction). The simulated true trajectory in one lap and the decoded trajectory obtained from VB-HMM are shown in Fig. 2. As comparison, two trajectories are very similar, so are the true and estimated transition matrices (Fig. 3). As expected, due to behavioral turns at certain state locations (e.g., state 15, 25), there are more than one nonzero elements in a few rows of the transition matrix, indicating the presence of a shortcut between two non-neighboring states. Furthermore, when comparing with the true tuning curves of 21 cells, it is found that the estimated tuning curves have a faithful resemblance (Fig. 4). It is also

[4] In this paper, we only consider this situation. More generally, if there are more than two (say l) dominant nonzero off-diagonal entries, we have to consider not only row permutation but also column permutation, there will be a total of $n! \times (l-1)!$ permutation possibilities.

[5] Note that this trick attempts to impose a structural prior. In contrast, a completely random $\boldsymbol{P}$ will cause a slow convergence and a poor solution.

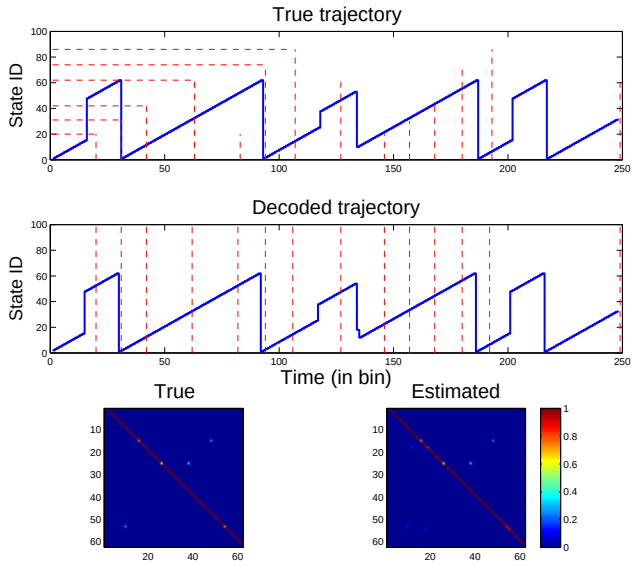

Fig. 3 One illustrated estimation result from the linear track (Simulation 1-2): the true trajectory in one lap (*top*) and the corresponding estimated trajectory (*middle*). State 1-31 represents the left-to-right positions inside the linear track, and state 32-62 represents right-to-left positions inside the track. The color-coded true (*bottom left*) and estimated (*bottom right*) transition matrices are qualitatively and quantitatively similar. Note that the transition matrix has a shifted-diagonal structure. Quantitative indices: $D_1 = 0.1333, D_2(2) = 0.0407, D_2(3) = 0.0479$.

noted that the VB algorithm is capable of decoding state trajectory accurately despite the fact that many cells have multiple-peak place fields.

It shall be pointed out that there are many equivalent solutions (due to the singularity of latent probabilistic model and the ambiguity of state permutation). In other words, even the decoded state sequence trajectory appears different from the true one, but the solution is actually consistent with the true one after remapping state ID (this can be confirmed by visual inspection or quantitative evaluation). Figure 5 shows such an example. In Fig. 5, the quantitative metrics are $D_1 = 0.2, D_2(2) = 0.0292, D_2(3) = 0.0363$, as compared to $D_1 = 0.1333, D_2(2) = 0.0407, D_2(3) = 0.0479$ in Fig. 3.

Notes: At this point, it is worth mentioning several important observations from computer simulations:

– Given a sensible initialization, the VB-HMM algorithm converges very fast, typically within less than 20 iterations. Our algorithm also produces much a better solution than the standard EM-HMM algorithm. Without the imposed prior information (as in the VB), the decoded state trajectory obtained from the EM algorithm is rarely comparable to the true trajectory (result not shown). In addition, the estimated transition matrix from the EM algorithm lacks the sparsity structure (data not shown). This is because without the con-

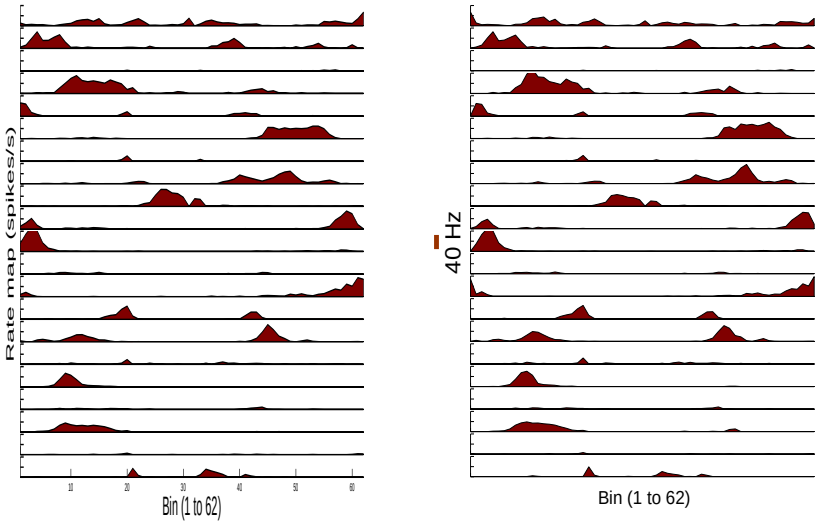

Fig. 4 The simulated (*left*) and estimated (*right*) tuning curves of 21 neurons from the same result shown in Fig. 3 (Simulation 1-2). The full length of the vertical bar marks the firing rate scale of 40 Hz.

straints, the solution space is too large, and the EM algorithm is prone to being stuck in poor local maxima.

- As illustrated in Fig. 3 and Fig. 5, there are non-unique but equally satisfactory solutions for the decoded trajectory (because of permutation ambiguity). Even the decoded trajectory may appear different from the true trajectory at the first sight, the solution can be consistent and valid upon state ID remapping. The proposed quantitative measures D_1 and D_2 provide a hint about the quality of the estimation. However, it shall be noted that the D_1 value also depends on the distribution of the actual state occupancy time. Because of the data dependency, the comparison of the D_1 value only makes sense among the same simulation experiment.
- Typically, a small D_1 value is accompanied by a small D_2 value and a large free energy $\mathcal{F}$. However, the reverse statement is not always true. In other words, sometimes a large free energy may be associated with relatively large D_1 and D_2 values, or sometimes even when D_2 and the negative free energy $-\mathcal{F}$ is small, the D_1 value can be large. To show that, we have conducted 50 independent Monte Carlo simulations for Simulation 1-2, and the statistics of D_1, D_2 and $\mathcal{F}$ are shown in Fig. 6. Note that these results consist of all solutions with different degrees of performance (both failures and successes). In our observations, a "qualitatively good" solution is often accompanied with lower values of D_1 and D_2, in combination of reasonably high value of free energy; a "qualitative bad" solution is often accompanied with a low free energy, a high D_2 value. In this specific Monte Carlo experiment, the conservative failure rate estimate is around 10-14% (5-7 cases).

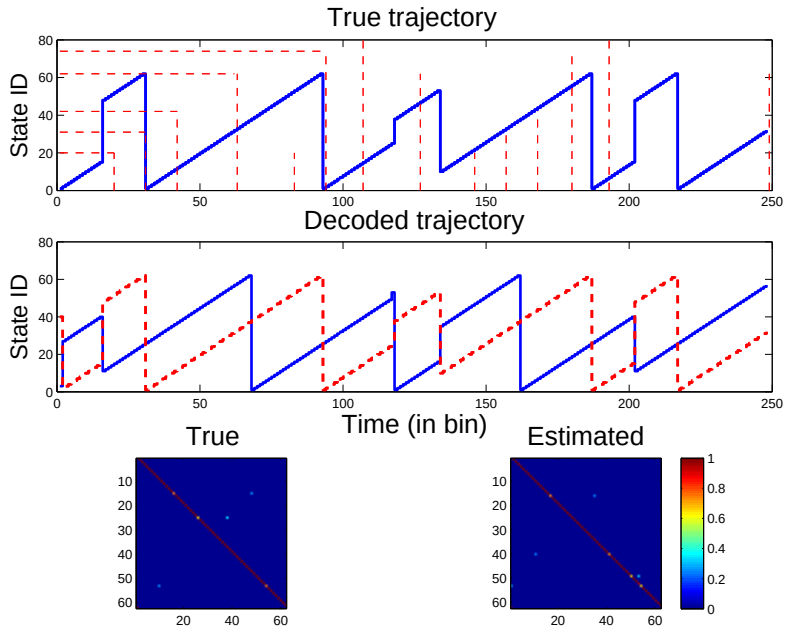

Fig. 5 In comparison with the results illustrated in Fig. 3, another correct estimation result from the trajectory in a linear track (Simulation 1-2). Note that the the raw (top panel) and remapped (second panel, dashed line) state trajectories will become nearly identical upon state ID remapping (using the following ID map: [1:25]→[38:62], [26:62]→[1:37]). Also note that the true (*bottom left*) and estimated (*bottom right*) transition matrices will become nearly identical upon state ID remapping. Quantitative indices: $D_1 = 0.1333, D_2(2) = 0.0560, D_2(3) = 0.093$.

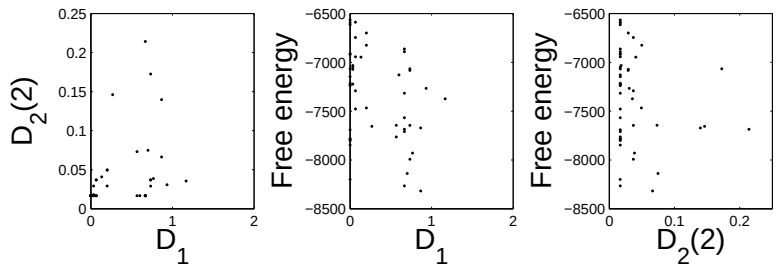

Fig. 6 Scatter plots of statistics of D_1, $D_2(2)$, and free energy F. Statistics are obtained from 50 independent Monte Carlo simulations (Simulation 1-2).

5.1.3 Simulated T-maze

Linear track is the simplest spatial topology among all rat navigation tasks. Next, a slightly more complex spatial topology—T-maze, is considered. In the first simulation scenario, the animal navigates in a T-maze (Fig. 7, leftmost panel). In each lap, the animal makes random exploration to the left or right arms of the maze. In

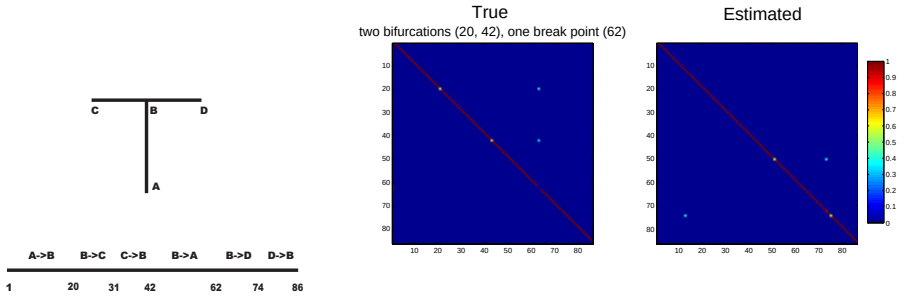

Fig. 7 *Left:* illustration of linearization of a simulated T-maze. Due to the bidirectional factor of the place field, a total of $43 \times 2 = 86$ states represents the 43 bins. Linearized bin assignment: $A \rightarrow B$: 1:20, $B \rightarrow C$: 21:31, $C \rightarrow B$: 32:42, $B \rightarrow A$: 43:62, $B \rightarrow D$: 63:74, $D \rightarrow B$: 75:86. One illustrated result from the simulated T-maze (Simulation 2-1): color-coded true (*middle*) and estimated (*right*) transition probability matrices, and $D_2(2) = 0.015$. The comparison of the true and estimated trajectories are not shown here.

this case, the animal makes no turn in the middle of two arms. Based on the same initialization setup as before, the VB-HMM algorithm is capable of producing very accurate estimation results. One of the estimation results from this simulation is illustrated in Fig. 7. As seen, the estimated state transition probability matrix is very similar to the true one ($D_2(2) = 0.015$). Note that the transition matrix has two bifurcation points at states 20 and 42, and there is also a break point at state 62. In addition, the estimated state sequence trajectory is also consistent with the true one ($D_1 = 0$), and so are the estimated tuning curves (not shown).

In the second simulation scenario, the animal is assumed to navigate in the same T-maze environment. However, the animal makes regular turns inside the two arms of the maze. Using the VB-HMM algorithm, one of the estimation results from this simulation is illustrated in Fig. 8. By inspection, the estimated state transition matrix and state sequence trajectory are also consistent, achieving excellent quantitative metrics: $D_1 = 0, D_2(2) = 0.024, D_2(3) = 0.038$.

5.1.4 Combined environments

We further examine the scenario with two combined spatial environments, which is not uncommon in some rodent navigation protocols. In the first simulation scenario, we consider one linear track A and one T-maze, where the linear track A is part of the T-maze (i.e., one arm of the T-maze). The linear track A is represented by 62 states, and the T-maze is represented by 86 states. Therefore, the combined environment is also represented by 86 states. In each lap, the animal first explores the linear track (state 1-62), and then the animal is exposed to the complete environment (state 1-86). Although the task is more challenging than the first two scenarios, the VB-HMM algorithm still performs quite well. One of the estimated results is illustrated in the left panel of Fig. 9. By inspection, we see the estimated state sequence is consistent with the true one, this is also confirmed by $D_1 = 0.783$ and the statistics of the respective sorted state occupancy time (right panel, Fig. 9). The estimated transition probability matrix has a relatively similar structure as the one shown in Fig. 7 (data not shown, $D_2(2) = 0.698$).

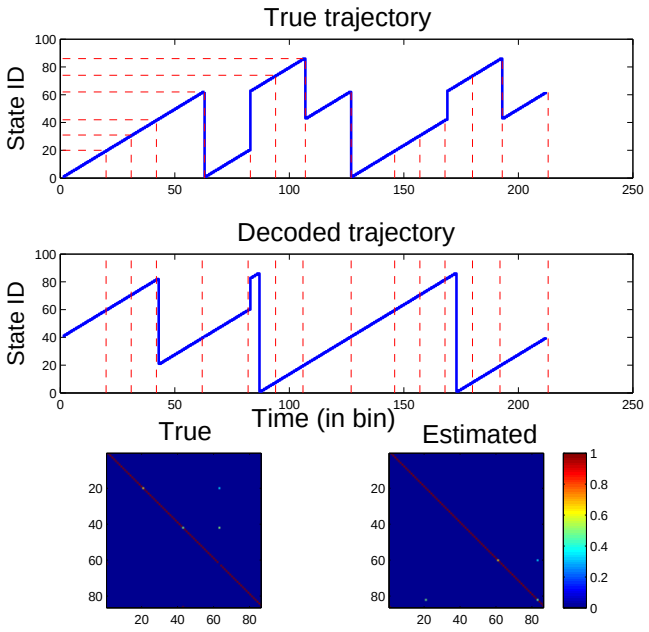

Fig. 8 One illustrated snapshot from the simulated T-maze (Simulation 2-2). *Top*: comparison of the true and estimated trajectories. *Bottom*: comparison of the true (*left*) and estimated transition matrices (*right*). Quantitative indices: $D_1 = 0, D_2(2) = 0.024, D_2(3) = 0.038$.

In the second simulation scenario, we consider the combination of two linear tracks (A and B) without overlapping region between each other. Two linear tracks are represented by 43 states, resulting a total of 86 states with the bidirectional factor. In the first 865 temporal bins, the animal first explores the linear track A (state 1-62), and then the gate between two tracks opens and closes behind once the animal moves to the linear track B (state 63-86) and explores in the remaining 147 temporal bins. Therefore, there is only one single transition chaining two environments. To our little surprise, the VB-HMM algorithm is still capable of recovering the behavior trajectory. One of the estimated results are illustrated in Fig. 10. As seen, even with a sample size as small as 1012 bins, our approach can decode the state trajectory rather reliably ($D_1 = 3.04$) and produce a reasonably good estimate of the state transition matrix ($D_2(2) = 0.313, D_2(3) = 0.445$).

5.1.5 Interpretation of graphs

The topological graph reveals important information about the spatial topology of the environment as well as the animal's behavior. Take a look at the examples of the inferred graphs shown in Fig. 11, Fig. 11a is a graph obtained from Simulation 1-1, Fig. 11b is the inferred graph from the true state transition matrix used in Simulation 1-2, Fig. 11c and Fig. 11d are the inferred graphs from the estimated

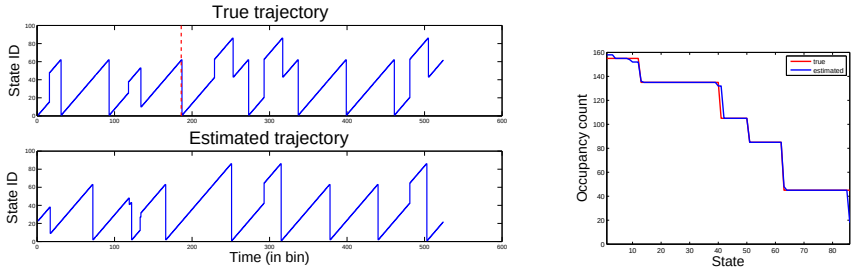

Fig. 9 One illustrated snapshot from two combined environments (Simulation 3-1): the comparison of the snapshots of the true (*left top*) and estimated (*left bottom*) trajectories, with the transition marked by a dashed line. In this case, two environments have overlapping regions. From the true and estimated state trajectories, we can compare the statistics of the state occupancy time (*right*) and obtain $D_1 = 0.783$.

state transition matrix in the same simulation, without and with thresholding, respectively.

We would like to point out a few important facts in interpretation of the graphs:

- The end-to-end navigation behavior (i.e., no turns) in the linear track and T-maze environment will have simpler graphs (shown in Fig. 1, bottom left two graphs). This example is perfectly illustrated in the graph of Fig. 11a, which is inferred from Simulation 1-1.
- Whenever there are navigation turns inside the track, shortcuts will be created in the graph. The number of locations at which the turns occur determines the number of shortcuts. This can be perfectly illustrated in the graph of Fig. 11b: in addition to the "8"-figure topology (solid string in dark color), two shortcuts (weak edges in light color) are created.
- In the T-maze, since there are bifurcation points (e.g., states 20 and 42 in Simulation 2-2) as well as a break point (i.e., discontinuity), the end points of the inferred graph are not connected (left panel of Fig. 12). However, there are two shortcuts due to the existence of behavior turns in the simulation.
- In the combination of two environments (Simulation 3-2), the graph inferred from the true transition matrix (Fig. 13, left panel) consists of two separate yet weakly linked loops, each of them has its own shortcuts. The large loop on the top of the left panel (Fig. 13) represents the state space 1-62, whereas the small loop represents the state space 63-86, and these two loops are linked by a week edge (states 62 and 63).

All of these facts are observed from a "ground truth" graph inferred from a true state transition matrix. In statistical inference, the statistical estimation error in the state transition matrix will inevitably make the graph interpretation more difficult (e.g, Fig. 13, right panel). In practice, we found that thresholding small probabilities before using the force-based algorithm will improve the graph presentation (e.g., Fig. 11c vs. Fig. 11d).

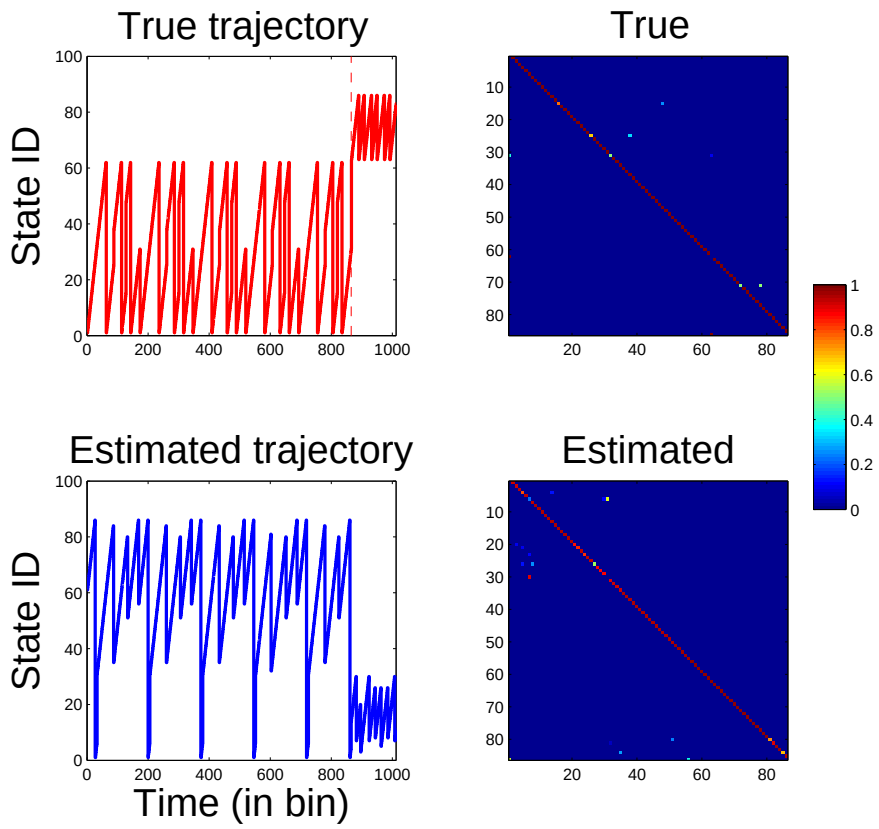

Fig. 10 One illustrated result from two combined environments (Simulation 3-2): the comparison of the snapshots of the true (*top left*) and estimated (*bottom left*) trajectories. Note that two environments have no overlapping region, the one-time transition between A and B ($31 \rightarrow 63$) is marked by a dashed line. Linear track A has state ID 1-31 (forward direction) and 32-62 (reverse direction); linear track B has state ID 63-74 (forward direction) and 75-86 (reverse direction). Quantitative indices: $D_1 = 3.04, D_2(2) = 0.313, D_2(3) = 0.445$.

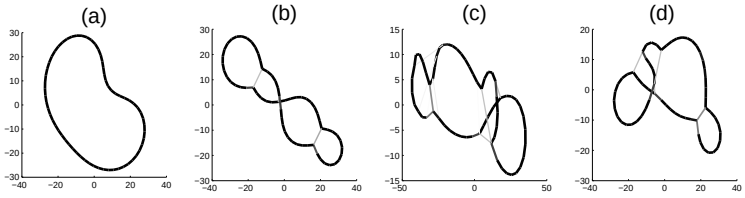

Fig. 11 (a) The graph inferred from the estimated transition matrix from a simulated linear track without behavioral turns (Simulation 1-1, $m = 62$). (b-d): the graphs inferred from a simulated linear track with behavioral turns (Simulation 1-2, $m = 62$): from the true transition matrix (b), the estimated transition matrix (c) and the estimated transition matrix followed by 0.05 thresholding (d).

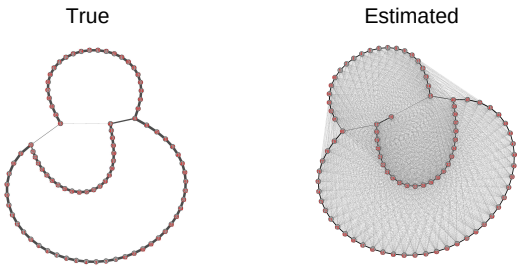

Fig. 12 The graphs inferred from the true estimation matrix (*left*) and the estimated transition matrix without thresholding (*right*) in a simulated T-maze (Simulation 2-2, $m = 86$). The nodes represent the states, and the edges represent the strengths between the nodes. Notice the similar spatial topology between these two graphs.

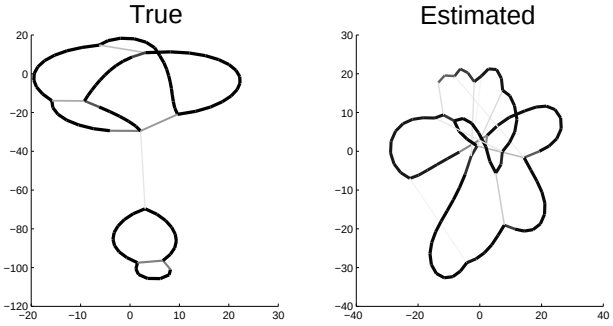

Fig. 13 The graphs inferred from the true transition matrix (*left*) and the estimated transition matrix with 0.01 thresholding (*right*) in Simulation 3-2 ($m = 86$).

5.2 Experimental data

Given the successful estimation results from the extensive computer simulations, we further apply our analysis to two experimental data sets. The statistics of experimental data are summarized in Table 2.

5.2.1 Linear track

In the first experimental protocol, the rat navigated in a 3.1-m linear track environment. To exclude the rat's pause or stop periods inside the track, we apply a velocity filter (15 cm/s) to obtain run-only periods of recording in one session. A total of 30 putative pyramidal cells were simultaneously recorded from the rat hippocampal CA1 area based on multiple tectrodes. Spikes are sorted and binned with a 250-ms window, and the spike count statistics of the ensemble neurons are obtained.

We run the VB-HMM algorithm more than 50 times (each with independent random initialization) and select the result associated with the highest free energy. A range of model size of $m = 60 \sim 80$ has been tested. The estimated trajectory, the transition probability matrix, and the optimized 2D topology from the best

Table 2 Summary of experimental data. All data use a 250-ms bin size. The run-only data are obtained with a 0.15 m/s velocity filter.

environment	selected m	C	T	remark
3.1-m linear track	60∼80	30	1138	∼4.7-min run period extracted from 30-min recording
T-maze	70∼80	39	952	∼4-min run period extracted from 16-min recording

result are illustrated in Fig. 14. The spatial topology is optimized with the force-based algorithm using the estimated transition matrix (upon 0.01 thresholding). In a closer examination of the graph, the basic topology appears to be a closed-loop circle, reflecting the nature of the back-and-forth navigation inside the linear track (Fig. 14, first panel). In addition, there are two or three weak edges within the closed-loop circle, implying there are shortcuts between the states. This is also consistent with the rat's behavior: the rat make turns at two specific locations: one is around 70 cm and other two are around 200 cm and 250 cm.

5.2.2 T-maze

In the second experimental environment, the rat navigated in a T-maze (as illustrated in the left panel of Fig. 2). After linearization, the environment is about 200 cm in length. A total of 39 putative pyramidal cells were simultaneously recorded from the rat hippocampal CA1 area based on multiple tectrodes. Spikes are sorted and binned with a 250-ms window, and the spike count statistics of the ensemble neurons are obtained. Again, a velocity filter is applied to extract about 4-min run periods from a total of 16-min recording.

Similarly, we run the algorithm more than 50 times (each with independent random initialization) and select the best result associated with the highest free energy. A range of model size of $m = 70 \sim 80$ has been tested. The estimated trajectory, estimated transition probability matrix, and the optimized 2D and 3D topologies from the best result are illustrated in Fig. 15. Specifically, the 3D graph is simply another perspective to visualize the topological graph (using the same force-based algorithm). As seen, the spatial topology inferred from the T-maze is more complex than that obtained from the linear track, the presence of twisted loop and weak edges make the inferred graph more difficult to interpret. This result is not too surprising, since in comparison to the simulated T-maze, the animal's behavior is more versatile and the real data length is about 4∼ 5 times shorter; which all make the inference task more challenging.

6 Discussion

6.1 Model selection and local maximum

For the finite m-state HMM, an important issue in statistical inference is to choose the model size m. Various model selection studies have been conducted in the HMM literature [Scott 2002, Cappé et al. 2005, Rydén 2008]. In this paper, we focus on highlighting the methodology of VB inference and uncovering the spatial topology. For this reason, we have either selected the true model size (as in computer simulations) or empirically selected the model size (as in experimental data).

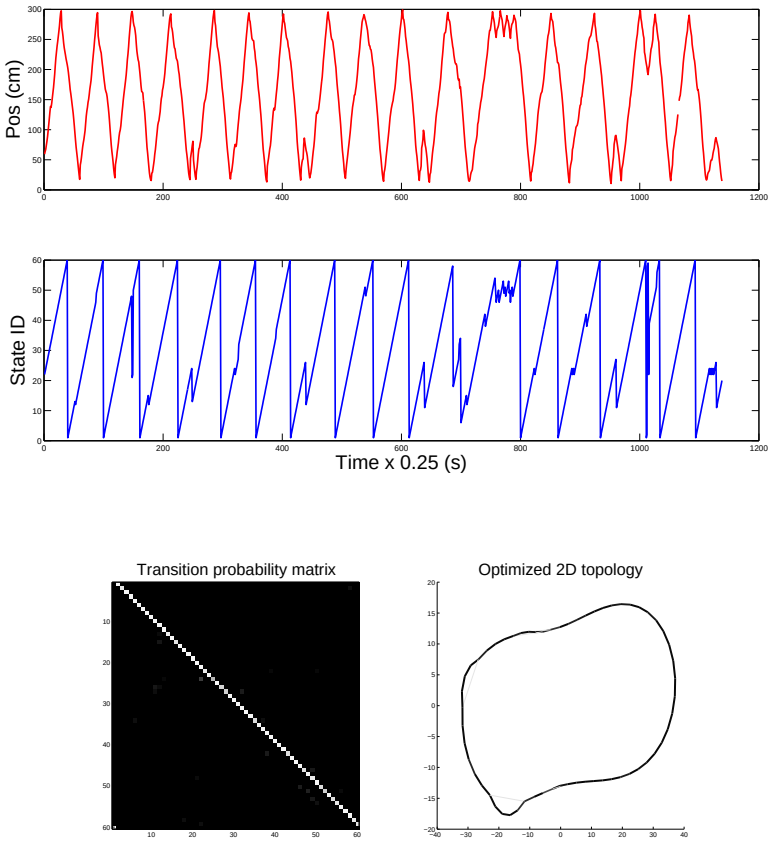

Fig. 14 One illustrated estimation result from the experimental linear track ($m = 60$): True position during the run period (*top panel*) and the decoded state sequence (*middle panel*). The estimated transition probability matrix (*bottom left*) and the inferred 2D graph (*bottom right*) are also shown. Note that there are three weak edges or shortcuts appended to the well-connected closed-loop.

Detailed comparison of results from using various model sizes is beyond the scope of current paper. Despite that, in order to illustrate the important issue of model selection, we use one example to illustrate how different model sizes will affect the estimation results. In the example of Simulation 1-2, the true model size is 62, we have also conducted inference using either a smaller ($m = 50$) or a larger ($m = 80$) model size. Two selected results are shown in Fig. 16. As seen, when the model size is insufficient, mistakes will be found in the inferred results (Fig. 16, left panels); when the model size is too large, redundant states are often found (Fig. 16, right panels).

In the terms of the optimized free energy function, a larger model size is typically accompanied with a greater free energy value. However, the local maximum

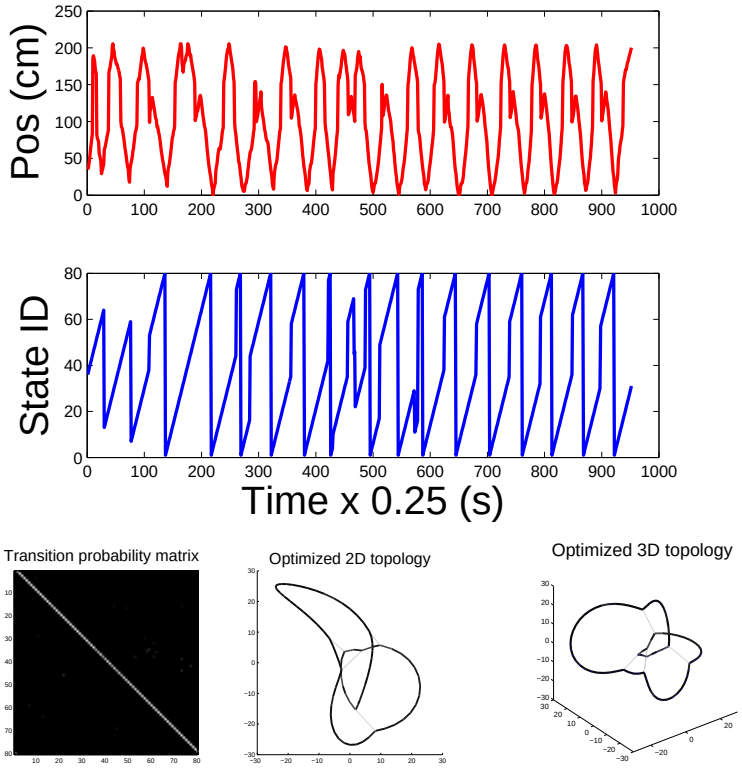

Fig. 15 One illustrated estimation result from the experimental T-maze ($m = 80$): True position during run period (*top panel*) and the decoded state sequence (*middle panel*). The estimated transition probability matrix (*bottom left*) and the inferred 2D (*bottom middle*) and 3D graphs (*bottom right*) are also shown. The 3D graph is simply another perspective to visualize the topological graph.

problem would make direct comparison of different model sizes nontrivial. To illustrate this point, we run Monte Carlo experiments using different model sizes and compare their statistics (Fig. 17). Three important observations are noteworthy from Fig. 17: (i) When the right model size is selected, the greater free energy value is achieved upon convergence; in contrast, models with either too large or too small size will have lower free energy values. (ii) Compared to the small model size ($m = 50$), the large model size ($m = 80$) has a slightly more spread-out free energy distribution, whereas its best performance is slightly better. (iii) In the case of using a large model size, there often appear many redundant states—namely, not all state IDs are used in trajectory decoding. The actually used states are referred to the *effective* states. A commonly observed phenomenon when selecting a large model size is that most of 'good' performance is only obtained when the effective state size is around 62, but not always vice versa (namely, the good performance is not guaranteed when the effective state size is around 62).

In practice, one can use the free energy or the Bayesian deviance information criterion (DIC) as a guiding principle for model selection. Specifically, the DIC is

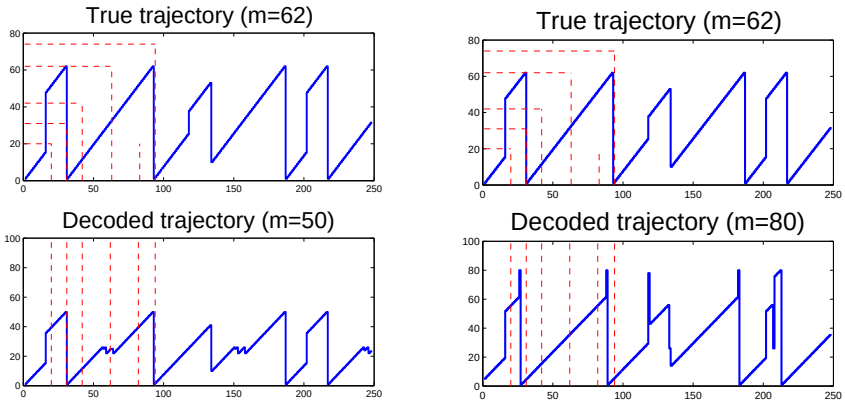

Fig. 16 Illustration of the estimated trajectory where the model size mismatch (Simulation 1-2). Underestimation (*left*): $m = 50,\ D_1 = 8, D_2(2) = 0.087, D_2(3) = 0.093$. Overestimation (*right*): $m = 80,\ D_1 = 4, D_2(2) = 0.017, D_2(3) = 0.182$.

defined as the sum of the expected deviance and the model complexity measure p_D [McGrory and Titterington 2009]:

$$\begin{aligned} DIC &= \mathbb{E}_{p(\boldsymbol{\theta}|\mathbf{y})}\Big[-2\log p(\mathbf{y}|\boldsymbol{\theta})\Big] + p_D \\ &\approx -2\log p(\mathbf{y}|\tilde{\boldsymbol{\theta}}) - 2\int q_{\boldsymbol{\theta}}(\boldsymbol{\theta})\log\frac{q_{\boldsymbol{\theta}}(\boldsymbol{\theta})}{p(\boldsymbol{\theta})}d\boldsymbol{\theta} + 2\log\frac{q_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\theta}})}{p(\tilde{\boldsymbol{\theta}})} \end{aligned} \tag{29}$$

where $\tilde{\boldsymbol{\theta}}$ denotes the posterior mean computed with respect to the variational posterior $q_{\boldsymbol{\theta}}(\boldsymbol{\theta})$, and $p(\mathbf{y}|\tilde{\boldsymbol{\theta}})$ can be computed from the forward-backward algorithm (Appendix A).

We can also consider an alternative HMM. The infinite HMM is a nonparametric Bayesian extension of the HMM with an infinite number of hidden states [Beal et al. 2002]. The key difference in hierarchical Bayesian modeling of the infinite HMM from the finite HMM is to treat the priors in the context of stochastic process. Recall that the prior used for the state transition matrix follows a Dirichlet distribution [van Gael et al. 2008]:

$$\begin{aligned} \boldsymbol{P}_j &\sim \mathtt{Dir}(\alpha\boldsymbol{\beta}) = \frac{\Gamma(\sum_i \alpha\beta_i)}{\prod_i \Gamma(\alpha\beta_i)}\prod_{i=1}^{m}(P_{ij})^{\alpha\beta_i - 1} \\ \boldsymbol{\beta} &\sim \mathtt{Dir}(\gamma/m, \ldots, \gamma/m) \end{aligned}$$

where $\boldsymbol{P}_j$ denotes the j-th row of the transition matrix $\boldsymbol{P}$, and $\boldsymbol{\beta}$ are the shared prior parameters. The infinite-dimensional generalization of the Dirichlet distribution is a Dirichlet process. As $m \to \infty$, the hierarchical prior approaches a hierarchical Dirichlet process (HDP) [Teh et al. 2006]. A HDP is a set of Dirichlet processes (DPs) coupled through a shared random based measure G_0 which is itself drawn from a DP. The concentration parameter $\alpha > 0$ governs the variability of the base measure, with small α implying greater variability. However, learning an infinite HMM would require a large amount of data samples, it may not be very practical in dealing with the experimental data in our current problem.

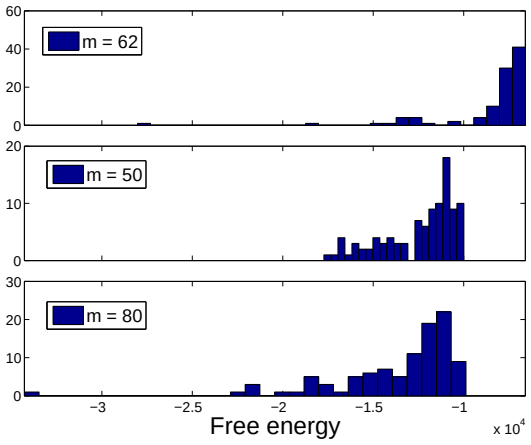

Fig. 17 Illustration of the distribution statistics of the converged free energy value based on independent random initializations. In each setup (from Simulation 1-2), 100 Monte Carlo experiments are conducted. As expected, when the selected model order matches the true model size ($m = 62$), a higher free energy value is typically achieved.

In learning latent probabilistic models, it is well known that the iterative EM and VB-EM algorithms are subject to the local maximum problem during optimization. The value of the local maximum highly depends on the initial conditions of the parameters or priors. Therefore, for every experimental data set, multiple runs of the iterative algorithm is a common practice. Meanwhile, the local maximum problem can be alleviated by using the so-called *deterministic annealing* (DA) procedure (see Appendix B for details). The key idea of the DA is to optimize a modified free energy function using an annealing parameter (in the analogy of the inverse temperature). As the inference process continues, the temperature is lowered and the annealing parameter is increased, it is expected (with higher probability) that the VB-EM algorithm can escape from the local maximum and approach a better solution. As a trade-off, the DA version of the algorithm is computationally slower and the result is also sensitive to the choice of the annealing parameter. Our observation of using DA in our current experiment is that the DA computation is very slow and it is wiser to spend the CPU resource in running the standard algorithm a few more times. Another possible solution is to use MCMC methods for *exact* Bayesian inference (in opposition to *approximate* Bayesian inference in VB). In this case, the VB-EM would be replaced by a Monte Carlo EM algorithm [McLachlan and Krishnan 2008]. The inference principle remains similar: in the E-step, run the forward-backward algorithm, in the Monte Carlo M-step, run the Gibbs sampler for estimating the unknown parameters (using the same conjugate priors). Finally, the posteriors of the parameters would be represented by simulated Monte Carlo samples. However, as we have discussed earlier, the MCMC methods are more computationally expensive and a large memory space is required for storing samples for the parameter $\boldsymbol{\Lambda}$ of size m-by-C.

6.2 Extension with a dummy state

Thus far, we have only considered the spiking activity within the periods of active behavior. In principle, this can also be extended to periods of sleep or quiet wakefulness (although the temporal bin size needs to be adjusted). However, because of the distinct neural mechanisms of hippocampal circuitry between periods of behavior and periods of sleep or quiet awakefulness, it is important to analyze these periods separately. Currently, we use a velocity criterion to segment the run and stop periods in behaving animals. The stop epochs have been excluded in the analysis.

Alternatively, the stop epochs can be treated as an observed indicator variable and included in the analysis. In this case, we use a dummy or NULL state (without loss of generality, the augmented $(m+1)$-st state) to represent the situation in the presence of either non-RUN period or missing data (e.g., no recording is available between two independent episodes or between the change of experimental conditions). Since the $(m+1)$-st state is not hidden (i.e.,, being observable via the velocity filter), the inference of the HMM can be adapted to accommodate this scenario. Basically, the transition probability $P_{i,m+1}$ represents the conditional probability from state i to STOP, and the transition probability $P_{m+1,i}$ represents the conditional probability from STOP to state i, where the i-th state represents the i-th location in the virtual environment. The inference algorithm still remains similar, except for a slight modification of the forward-backward algorithm employed in the VB-E step.

For illustration, we apply the augmented HMM to the experimental data in the linear track. In the experimental linear track example, we include all non-RUN periods into our analysis, which consist of many RUN→STOP and STOP→RUN transitions. As expected, the estimated transition probability matrix has a shifted-diagonal substructure (as before) plus an additional column that reflects the RUN→STOP behavior (Fig. 18, left panel) . The non-sparse patterns in the last column reflects the stop behavior from various spatial locations. Applying the force-based algorithm to the shifted-diagonal substructure of the transition probability matrix (by excluding the last row and the last column) yields the spatial topology shown in the right panel of Fig. 18. Note that the loose ends of the graph are due to systematic stop behavior at the ends of the track. Therefore, the inferred graph reveals not only important cues about the spatial topology, but also important information about the animal's behavior.

6.3 About the Markovian assumption

In this paper, we have assumed that the latent state process, which represents the rat's position combined with the directionality, follows a first-order Markovian process. This assumption is reasonable while using a relatively large temporal bin size (here, 250 ms). In reality, this assumption might not be completely valid. For example, there could be a high-order Markovian dependence in terms of motion, or there could be a non-Markovian or semi-Markovian behavior. Nevertheless, modeling these situations would require a large amount of data for fitting a more complex statistical model, which is beyond the scope of the current paper.

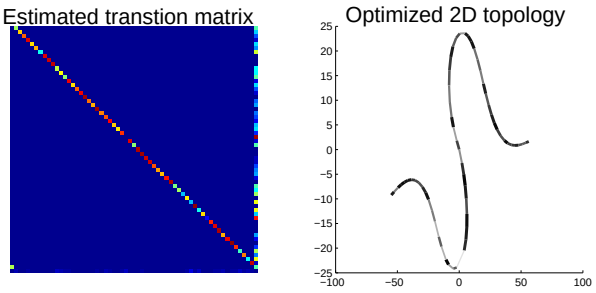

Fig. 18 Illustration of the estimated transition matrix in the presence of dummy state (*left*) and the inferred graph (*right*). The results are obtained from the experimental linear track data (using $m = 60$). Note that in the left panel, the non-sparse pattern of the 61st column implies frequent stop behavior from various spatial locations. Also note that in the right panel, the graph is drawn excluding the dummy state (i.e., based on the 60×60 submatrix of $\boldsymbol{P}$), which gives rise to a spatial topology without the closed loop.

Also note that, based on the decoded state trajectory, one can estimate the high-order transition probability. For instance, the second-order transition probability, represented by a 3D tensor $\boldsymbol{P}^{(2)} = \{P^{(2)}_{ijk}\}$ (where $\sum_j \sum_k P^{(2)}_{ijk} = 1$), reveals information about a 3-bit state sequence $i \to j \to k$. Imaginably, at the bifurcation point (denoted by state j), we will see two (or more) dominant values $P^{(2)}_{ijk}$ and $P^{(2)}_{ijl}$ $(l \neq k)$. These high-order statistics would be even more important when navigating in an open field environment.

6.4 Extension to non-Poissonian firing model

In Eq. (2), we have assumed that all neurons follow a pure Poisson spiking model. However, this assumption can be extended to other non-Poisson firing models, such as the Gamma distribution (which will be associated with a conjugate prior with four hyperparameters). Also, we may introduce individual neuronal firing history or ensemble neuronal firing activity as an observed covariate and characterize the neuronal firing within a generalized linear model (GLM) framework [Truccolo et al. 2005], the regression coefficients of the GLM can be estimated with a VB approach [Chen et al. 2011] in the VB-M step.

6.5 Identifiability

A model is said to be identifiable if it is theoretically possible to learn the true value of this model's underlying parameter after obtaining an infinite number of samples from it, which is also equivalent to saying that different values of the parameter would generate different probability distributions of the observable samples. In our statistical model, the unknown variables $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})$ are estimated by optimizing the free energy (Eq. 24) assuming a factorial form of the posterior distribution (Eq. 10). Due to non-convexity of the objective function, there might be many equivalent solutions in the joint space of $(\boldsymbol{P}, \boldsymbol{\Lambda})$ (permutation). This issue, in

combination with the large dimensionality of $\boldsymbol{\theta}$ and small sample size, makes the task of statistical inference very challenging.

6.6 Assessment criterion

In computer simulations, in addition to visual inspection, we assess the quality of the estimation via two quantitative metrics: D_1 and D_2. However, visual inspection would become difficult when dealing with experimental data associated with complex behavior, or when selecting varying model sizes (since different m values would induce different state reconstruction results). Because of the state permutation ambiguity, it is important to check the consistency between two solutions.

From an information coding perspective, the HMM can be viewed as trying to represent or remap a continuous space $\mathcal{S}$ with a finite discrete alphabet $\mathcal{A}$ using a code book: $\mathcal{S} = f(\mathcal{A})$. The criterion for the consistency is to assure a *one-to-one* mapping between $\mathcal{S}$ and $\mathcal{A}$: (i) Any element in $\mathcal{S}$ is not simultaneously represented by A_i and A_j $(i \neq j)$; (ii) The same A_i does not represent two or more distinct regions in $\mathcal{S}$ (except for neighboring regions, since two neighboring regions can be combined into one by a merging operation). In addition, the binning strategy may be very flexible, A_i and A_j can encode two regions with different amounts of spatial coverage. Although it is easy to state the consistency principle, a practical quantitative evaluation of the estimated result is nontrivial, especially in the absence of ground truth for the experimental data. This issue requires further investigation.

6.7 Computational issue and computer software

Depending on the data size and the initial condition, the convergence speed of the VB-HMM algorithm is fast, typically less than 30 iterations. During the inference process, we can monitor the learning curves of the free energy as well as the estimated parameters, see Fig. 19 for a simulation illustration. The algorithm can handle a large number of neurons with large sample size (in Simulation 3-1, $T = 8790$ corresponds to about 36 minutes with a 250 ms bin size). However, the number of states could become very large when considering a hypothetically complex spatial environment, exploiting the sparse structure of the state-transition matrix would be important in the presence of small sample size.

All software implementations are done in MATLAB©. Custom-written codes on the VB-HMM and the force-based algorithm used in this paper will be made available upon request.

7 Conclusion

In conclusion, we have used the rat hippocampus as a model system to uncover the "spatial topology" represented by the population codes. With the help of graph illustration, we develop a HMM and a VB inference algorithm to achieve this computational goal. Our empirical results from both extensive computer simulations

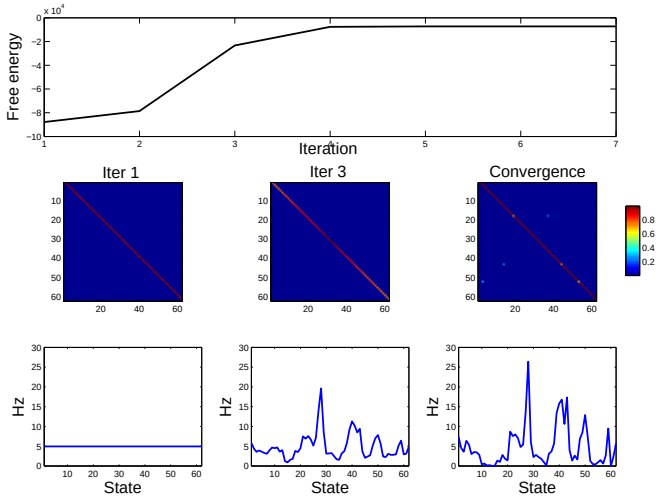

Fig. 19 Illustration of the algorithmic convergence and stability (Simulation 2-2). At different stages (1st and 3rd iterations, and final convergence), the free energy (*top row*), the estimated state transition matrix (*middle row*), and the tuning curve of one neuron (*bottom row*) are shown.

and experimental data have shown a promising direction in uncovering the structural patterns of ensemble spike activity during the periods of active navigation. Since the spatial topology graph is just the proxy of the state-transition matrix, our proposed approach can also be extended to other model systems with the interest of characterizing the behavior-related transition probability [Jones et al. 2007, Kemere et al. 2008].

Our study provides important insights for future direction in exploratory data analysis of population neuronal codes. In addition to further investigation of some technical issues (e.g., selecting optimal temporal window and model size, model extension), we are planning to apply the same methodological analysis to the other rodent data recorded in more complex spatial environments (e.g., H-maze and open field), which will pose more challenges for interpreting the trajectories and graphs. The same exploratory analysis can also be applied to spiking data of ensemble neurons recorded during periods of sleep [Wilson and McNaughton 1994, Louie and Wilson 2001, Lee and Wilson 2002, Ji and Wilson 2007] or during "preplay analysis" without prior exposure of spatial environment [Dragoi and Tonegawa 2011]. Other challenges can also arise due to the complex dynamics and multiple functional representations of the hippocampal place cells, as reported in the literature [Wood et al. 2000, Jackson and Redish 2007]. Incorporating new neurophysiological findings in space representation within the rat hippocampal circuitry, such as the spiking activity from head-direction cells and entorhinal cortical cells [McNaughton et al. 2006], would further enrich the model and pave the way for a deeper understanding of hippocampal neural mechanisms.

Acknowledgements The work was supported in part by the NIH Grant DP1-OD003646 (E.N.B.) and MH061976 (M.A.W.). We thank Stuart Layton for providing the linear track experimental data used in the paper.

Appendix A: EM and Viterbi algorithms

The goal of ML inference is to maximize the log-likelihood function (Eq. 5) based on missing data. In each full iteration, the EM algorithm [Dempster et al. 1977, McLachlan and Krishnan 2008] iteratively maximizes the so-called Q-function

$$\begin{aligned} Q(\boldsymbol{\theta}^{new}|\boldsymbol{\theta}^{old}) &= \mathbb{E}\Big[\log p(\hat{S}_{1:T}, \mathbf{y}_{1:T}|\boldsymbol{\theta})\Big|\boldsymbol{\theta}^{old}\Big] \\ &= \mathbb{E}\Big[\sum_{t=1}^{T}\sum_{c=1}^{C}\sum_{i=1}^{m}\gamma_t(i)\Big(y_{c,t}\log\hat{\lambda}_{ic} - \hat{\lambda}_{ic}\Big) \\ &\quad + \sum_{i=1}^{m}\gamma_1(i)\log\hat{\pi}_i \\ &\quad + \sum_{t=2}^{T}\sum_{i=1}^{m}\sum_{j=1}^{m}\xi_t(i,j)\log\hat{P}_{ij}\Big|\boldsymbol{\theta}^{old}\Big], \end{aligned} \tag{30}$$

and the new $\boldsymbol{\theta}^{new}$ is obtained by maximizing the incomplete data likelihood conditional on the old parameters $\boldsymbol{\theta}^{old}$; and the iterative optimization procedure continues until the algorithm ultimately converges to a local maximum or a stationary point.

E-step: Forward-backward algorithm In the E-step, the major task of the forward-backward procedure is to compute the state conditional marginal probabilities:

$$\begin{aligned} \Pr(S_t = i|\mathbf{y}_{1:T}, \boldsymbol{\theta}) &= \frac{\Pr(\mathbf{y}_{1:T}, S_t = i|\boldsymbol{\theta})}{\Pr(\mathbf{y}_{1:T}|\boldsymbol{\theta})} \\ &= \frac{\Pr(\mathbf{y}_{1:T}, S_t = i|\boldsymbol{\theta})}{\sum_{l=1}^{m}\Pr(\mathbf{y}_{1:T}, S_t = l|\boldsymbol{\theta})} \end{aligned} \tag{31}$$

as well as the state conditional joint probabilities:

$$\begin{aligned} \Pr(S_{t-1} = i, S_t = j|\mathbf{y}_{1:T}, \boldsymbol{\theta}) &= \frac{\Pr(\mathbf{y}_{1:T}, S_{t-1} = i, S_t = j|\boldsymbol{\theta})}{\Pr(\mathbf{y}_{1:T}|\boldsymbol{\theta})} \\ &= \frac{\Pr(\mathbf{y}_{1:T}, S_{t-1} = i, S_t = j|\boldsymbol{\theta})}{\sum_{l=1}^{m}\sum_{n=1}^{m}\Pr(\mathbf{y}_{1:T}, S_{t-1} = l, S_t = n|\boldsymbol{\theta})}. \end{aligned} \tag{32}$$

To make the notation simple, in the derivation below we will let the conditional $\boldsymbol{\theta}$ be implicit in the equation.

To estimate Eq. (7) and Eq. (8), we first factorize the joint probability as

$$\begin{aligned} \Pr(\mathbf{y}_{1:T}, S_t = l) &= \Pr(\mathbf{y}_{1:t}, S_k = l)\Pr(\mathbf{y}_{t+1:T}|\mathbf{y}_{1:t}, S_t = l) \\ &= \Pr(\mathbf{y}_{1:t}, S_t = l)\Pr(\mathbf{y}_{t+1:T}|S_t = l) \\ &\equiv a_t(l)b_t(l) \quad \text{for } l = 1, \ldots, m \end{aligned} \tag{33}$$

where

$$\begin{aligned} a_1(l) &= \pi_l \Pr(\mathbf{y}_1|S_1 = l) \\ a_t(l) &= \Pr(\mathbf{y}_{1:t}, S_t = l) \quad \text{for } t = 2, \ldots, T \\ b_t(l) &= \Pr(\mathbf{y}_{t+1:T}|S_t = l) \quad \text{for } t = 1, \ldots, T-1 \\ b_T(l) &= 1 \end{aligned}$$

and the forward and backward messages $a_t(l)$ and $b_t(l)$ can be computed recursively along the time index t [Rabiner 1989]

$$\begin{aligned} a_t(l) &= \sum_i a_{t-1}(i) P_{il} \Pr(\mathbf{y}_t|S_t = l) \\ b_t(l) &= \sum_i b_{t+1}(i) P_{li} \Pr(\mathbf{y}_{t+1}|S_{t+1} = i), \end{aligned}$$

where P_{il} denotes the transition probability from state i to l.

Given $\{a_t(\cdot), b_t(\cdot)\}$, the state posterior conditional joint probability (Eq. 8) is determined by

$$\Pr(S_{t-1} = i, S_t = j|\mathbf{y}_{1:T}) \propto a_t(i) P_{ij} \Pr(\mathbf{y}_{t+1}|S_{t+1} = j) b_{t+1}(j). \tag{34}$$

In light of Eq. (9), the observed (incomplete) data likelihood is computed as

$$\begin{aligned} p(\mathbf{y}_{1:T}) &= \sum_{l=1}^{m} \Pr(\mathbf{y}_{1:T}, S_t = l) \\ &= \sum_{l=1}^{m} a_t(l) b_t(l) = \sum_{l=1}^{m} a_T(l). \end{aligned} \tag{35}$$

Alternatively, the incomplete data likelihood is given by

$$\begin{aligned} p(\mathbf{y}_{1:T}) &= \prod_{t=1}^{T} p(\mathbf{y}_t|\mathbf{y}_{t-1}) \\ &= \prod_{t=1}^{T} \zeta_t(\mathbf{y}_t) = Z(\mathbf{y}_{1:T}) \end{aligned} \tag{36}$$

where $\zeta_t(\mathbf{y}_t) \equiv p(\mathbf{y}_t|\mathbf{y}_{t-1})$ is a normalization constant; $Z(\mathbf{y}_{1:T})$ is also called the marginal likelihood.

From Eq. (9) and Eq. (11), the state posterior conditional marginal probability (Eq. 7) is determined by the Bayes' rule

$$\begin{aligned} \Pr(S_t = i|\mathbf{y}_{1:T}) &= \frac{\Pr(\mathbf{y}_{1:T}, S_t = i)}{p(\mathbf{y}_{1:T})} \\ &= \frac{a_t(i) b_t(i)}{\sum_{l=1}^{m} a_t(l) b_t(l)} \propto a_t(i) b_t(i). \end{aligned} \tag{37}$$

Equations (10) and (13) are the sufficient statistics computed from the E-step (to be used in the M-step).

In the term of the computational overhead for the m-state HMM, the above-described forward-backward procedure requires an order of computational complexity $\mathcal{O}(m^2 T)$ and memory storage $\mathcal{O}(mT)$.

M-step: Re-estimation In the M-step, we update the unknown parameters by setting the partial derivatives of the Q-function to zeros: $\frac{\partial Q(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0$, from which we may derive either closed-form or iterative solutions.

Let $\xi_t(i,j) = \Pr(S_{t-1} = i, S_t = j|\mathbf{y}_{1:T}, \boldsymbol{\theta})$ and $\gamma_t(i) = \Pr(S_t = i|\mathbf{y}_{1:T}, \boldsymbol{\theta})$ denote, respectively, the state posterior conditional marginal and joint probabilities (which are the sufficient statistics for the complete data log-likelihood). From the E-step, we may obtain

$$\begin{aligned}\gamma_t(i) &= \frac{a_t(i)b_t(i)}{\sum_{l=1}^{m} a_t(l)b_t(l)} \\ &= \sum_j \xi_t(j,i) = \sum_j \xi_{t+1}(i,j). \end{aligned} \tag{38}$$

The transition probability estimates are given by the Baum's re-estimation procedure

$$\begin{aligned}\hat{P}_{ij} &= \frac{\sum_{t=2}^{T} \xi_t(i,j)}{\sum_{t=2}^{T}\sum_{j=1}^{m} \xi_t(i,j)} \\ &= \frac{\sum_{t=2}^{T} \xi_t(i,j)}{\sum_{t=2}^{T} \gamma_t(i)}. \end{aligned} \tag{39}$$

And the rate parameter estimates $\boldsymbol{\Lambda} = \{\lambda_{ic}\}$ are given by solving $\frac{\partial Q}{\partial \lambda_{ic}} = 0$ from Eq. (6)[6]

$$\hat{\lambda}_{ic} = \frac{\sum_{t=1}^{T} y_{c,t}\gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \tag{40}$$

Finally, the convergence of the EM algorithm is monitored by the incremental changes of the log-likelihood as well as the parameters. If the quantity of the absolute change or relative change of the log-likelihood is smaller than a desirable value, the EM algorithm is terminated.

Viterbi algorithm Upon estimating parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda})$, we can run the Viterbi algorithm [Viterbi 1967] for decoding the *most likely* state sequences. The Viterbi algorithm is a dynamical programming method [Bellman 1957] that uses the "Viterbi path" to discover the single most likely explanation for the observations. Specifically, the MAP estimate $\hat{S}_t$ at time t is

$$\hat{S}_t^{\mathrm{MAP}} = \arg \max_{i \in \{1,\ldots,m\}} \gamma_t(i) \quad 1 \le t \le T. \tag{41}$$

The computational overhead of the forward Viterbi algorithm has an overall time complexity $\mathcal{O}(m^2 T)$ and space complexity $\mathcal{O}(mT)$.

[6] To avoid numerical problem we set $\hat{\lambda}_{ic} = 0$ if the denominator is 0 or nearly 0.

Appendix B: Deterministic annealing

For the discrete m-state HMM, there are exponential numbers (i.e., $\mathcal{O}(2^m)$) of local maxima. The local maximum problem is particularly severe when the transition matrix $\boldsymbol{P}$ is sparse (many zero elements) or there are equal state emission probabilities for distinct states. This phenomenon is known as the "singularity" of the objective function [Amari et al. 2003, Watanabe 2009], which is omnipresent in many estimation problems of probabilistic models and artificial neural network models. In order to alleviate the local maximum problem, the so-called *deterministic annealing* (DA) procedure was proposed for several latent probabilistic models, such as the mixture models and HMM [Beal 2003, Katahira et al. 2008].

The key idea of DA-VB is to modify the original free energy function (Eq. 6) by introducing an annealing parameter ρ:

$$\begin{aligned}\mathcal{F}(q) &= \Big\langle \log p(\mathbf{y}_{1:T}, S_{1:T}, \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \Big\rangle_q + \frac{1}{\rho}\mathcal{H}_q(\boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}, S_{1:T}) \\ &= \Big\langle \log p(\mathbf{y}_{1:T}, S_{1:T}, \boldsymbol{\pi}, \boldsymbol{P}, \boldsymbol{\Lambda}) \Big\rangle_{q(\boldsymbol{\theta})q(S_{1:T})} \\ &\quad + \frac{1}{\rho}\mathcal{H}_{q(S_{1:T})}(S_{1:T}) + \frac{1}{\rho}\mathcal{H}_{q(\boldsymbol{\theta})}(\boldsymbol{\theta}) \end{aligned} \tag{42}$$

where $\rho = 1/\mathcal{T}$ can be viewed as an inverse temperature parameter. The annealing procedure gradually lowers the temperature (or increases ρ) during the inference process, hoping to escape from local maxima and ultimately to reach the global maximum with a higher probability.

Consequently, the new state posterior probabilities will be recomputed from the VB-E step:

$$\tilde{\gamma}_t(i) = \frac{\gamma_t(i)^\rho}{\sum_{j=1}^m \gamma_t(j)^\rho} \tag{43}$$

$$\tilde{\xi}_t(i,j) = \frac{\xi_t(i,j)^\rho}{\sum_{l=1}^m \sum_{n=1}^m \xi_t(l,n)^\rho} \tag{44}$$

whereas in the VB-M step, we have the following new update equations (used for Eqs. 12, 13 and 15):

$$w_i^{(\pi)} = \rho\Big(u_i^{(\pi)} + \tilde{\gamma}_1(i) - 1\Big) + 1 \tag{45}$$

$$w_{ij}^{(P)} = \rho\Big(u_{ij}^{(P)} + \sum_{t=2}^T \tilde{\xi}_t(i,j) - 1\Big) + 1 \tag{46}$$

$$q(\lambda_{ic}) = \mathtt{Gam}\Big(C\alpha_i^{(\lambda)} + \sum_{t=1}^T y_{c,t}\tilde{\gamma}_t(i), C\beta_i^{(\lambda)} + \sum_{t=1}^T \tilde{\gamma}_t(i)\Big) \tag{47}$$

Note that when the annealing parameter $\rho = 1$, the standard VB-EM algorithm (Sections 3.1 and 3.2) is recovered.

Appendix C: Optimizing hyperparameters

In light of Eq. (23), taking the derivatives of the logarithm of Eq. (23) with respect to $\alpha_i^{(\lambda)}$ and $\beta_i^{(\lambda)}$ and setting them to zeros yields

$$0 = \sum_{c=1}^{C} -\frac{C + \sum_t y_{c,t}\gamma_t(i)}{C\beta_i^{(\lambda)} + l_i} + Cy_{c,t}\psi'\Big(C\alpha_i^{(\lambda)} + \sum_t y_{c,t}\gamma_t(i)\Big) \tag{48}$$

$$0 = \sum_{c=1}^{C} \frac{C\alpha_i^{(\lambda)} + \sum_t y_{c,t}\gamma_t(i)}{(C\beta_i^{(\lambda)} + l_i)^2} - \frac{y_{c,t}}{C\beta_i^{(\lambda)} + l_i} \tag{49}$$

There is no closed-form solution to these two equations. However, solving these two fixed-point equations using a gradient or Newton-type algorithm within each VB-M step would increase the marginal log-likelihood or free energy.

References

[Amari et al. 2003] Amari, S. Ozeki, T. and Park, H-Y. (2003). Learning and inference in hierarchical models with singularities. *Systems and Computers in Japan, 34*(7), 34–42.

[Baum et al. 1970] Baum, L. E., Petrie, T., Soules, G. and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics, 41*(1), 164–171.

[Beal et al. 2002] Beal, M. J., Ghahramani, Z., & Rasmussen, C. E. (2002). The infinite hidden Markov model. *Advances in Neural Information Processing Systems, 14*. MIT Press.

[Beal 2003] Beal, M. J. (2003). Variational algorithms for approximate Bayesian inference PhD. Thesis, Gatsby Computational Neuroscience Unit, University College London.

[Bellman 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Boston.

[Bishop 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[Borg and Groenen 2005] Borg, I. & Groenen, P. (2005). *Modern Multidimensional Scaling: Theory and Applications*, 2nd edn. Springer-Verlag.

[Brand 1999] Brand, M. (1999). Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Computation, 11*, 1155–1182.

[Brand and Ketnaker 2000] Brand, M. & Ketnaker, V. (2000). Discovery and segmentation of activities in video. *IEEE Trans. Pattern Analysis and Machine Intelligence, 22*(8), 844–851.

[Brown et al. 1998] Brown, E. N., Frank, L. M., Tang, D., Quirk, M. C, & Wilson, M. A. (1998). A statistical paradigm for neural spike train decoding applied to position prediction from ensemble firing patterns of rat hippocampal place cells. *J. Neurosci., 18*, 7411–7425.

[Buzsaki 2006] Buzsaki, G. (2006). *Rhythms of the Brain*. Oxford University Press.

[Cappé et al. 2005] Cappé, O., Moulines, E., & Ryden, T. (2005). *Inference in Hidden Markov Models*, Springer, New York.

[Chen et al. 2009] Chen, Z., Vijayan, S., Barbieri, R., Wilson, M. A., & Brown, E. N. (2009). Discrete- and continuous-time probabilistic models and algorithms for inferring neuronal UP and DOWN states. *Neural Computation, 21*(7), 1797–1862.

[Chen et al. 2011] Chen, Z., Putrino, D., Ghosh, S., Barbieri, R. & Brown, E. N. (2011). Statistical inference for assessing neuronal interactions and functional connectivity with sparse spiking data. *IEEE Trans. Neural Systems and Rehabilitation Engineering, 19*(2), 121–135.

[Cox and Cox 2001] Cox, T. F. & Cox, M. A. A. (2001). *Multidimensional Scaling*. Chapman and Hall.

[Curto and Itskov 2008] Curto, C. & Itskov, V. (2008). Cell groups reveal structure of stimulus space. *PLoS Computational Biology*, 4, e1000205.

[Dabaghian et al. 2008] Dabaghian, Y., Cohn, A. G. & Frank, L. (2008). Topological coding in hippocampus. Online paper. arXiva:q-bio/0702052v1.

[Dabaghian et al. 2011] Dabaghian, Y., Cohn, A. G. & Frank, L. (2011). Topological coding in the hippocampus. In *Computational Modeling and Simulation of Intellect: Current State and Future Prospectives*, Chap. 12, pp. 293–320. IGI Global.

[Darmanjian and Principe 2009] Darmanjian, S. & Principe, J.C. (2009). Spatial-temporal clustering of neural data using linked-mixtures of hidden Markov models. *EURASIP Journal on Advances in Signal Processing*, 2009, Article ID 892461.

[Davidson et al. 2009] Davidson, T. J., Kloosterman, F. & Wilson, M. A. (2009). Hippocampal replay of extended experience. *Neuron, 63*, 497–507.

[Dempster et al. 1977] Dempster, A., Laird, N. and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B, 39*(1), 1–38.

[Diba and Buzsaki 2007] Diba, K. & Buzsaki, G. (2007). Forward and reverse hippocampal place-cell sequences during ripples. *Nature Neuroscience, 10*, 1241–1242.

[Dragoi and Tonegawa 2011] Dragoi, G. & Tonegawa, S. (2011). Preplay of future place cell sequences by hippocampal cellular assemblies. *Nature, 469*, 397–401.

[Frank et al. 2004] Frank, L. M., Stanley, G. B. & Brown, E. N. (2004) Hippocampal plasticity across multiple days of exposure to novel environments. *J. Neurosci., 24*, 7681–7689.

[Foster and Wilson 2006] Foster, D. J. & Wilson, M. A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. *Nature, 440*, 680–683.

[Gelman et al. 2004] Gelman, A., Carlin, J. B., Stern, H. S. & Rubin, D. B. (2004). *Bayesian Data Analysis* (2nd edn.). Chapman & Hall/CRC.

[Gilks et al. 1995] Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. editors. (1995). *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC.

[Harris et al. 2003] Harris, K., Csicsvari, J., Hirase, H., Dragoi, G., & Buzsaki, G. (2003). Organization of cell assemblies in the hippocampus. *Nature, 424*, 552–556.

[Herbst et al. 2008] Herbst, J. A., Gammeter, S., Ferrero, D., Hahnloser, R. H. R. (2008). Spike sorting with hidden Markov models. *J. Neurosci. Methods, 174*, 126–134.

[Jackson and Redish 2007] Jackson, J. & Redish, A. D. (2007). Network dynamics of hippocampal cell-assembliesresemble multiple spatial maps within single tasks. *Hippocampus, 17*, 1209–1229.

[Ji and Wilson 2007] Ji, D. & Wilson, M. A. (2007). Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature Neuroscience, 10*, 100–107.

[Ji et al. 2006] Ji, S., Krishnapuram, B. & Carin, L. (2006). Variational Bayes for continuous hidden Markov models and its application to active learning. *IEEE Trans. Pattern Analysis and Machine Intelligence, 28*(4), 522–532.

[Jones et al. 2007] Jones, L. M., Fontanini, A., Sadacca, B. F., Miller, P., & Katz, D. B. (2007). Natural stimuli evoke dynamic sequences of states in sensory cortical ensembles. *Proc. Natl. Acad. Sci. USA, 104*, 18772–18777.

[Karlsson and Frank 2009] Karlsson, M. P. & Frank, L. M. (2009). Awake replay of remote experiences in the hippocampus. *Nature Neuroscience, 12*, 913–918.

[Katahira et al. 2008] Katahira, K., Watanabe, K. & Okada, M. (2008). Deterministic annealing variant of variational Bayes method. *Journal of Physics: Conference Series, 95*, 012015.

[Katahira et al. 2010] Katahira, K., Nishikawa, J., Okanoya, K., & Okada, M. (2010). Extracting state transition dynamics from multiple spike trains using hidden Markov models with correlated Poisson distribution. *Neural Computation, 22*, 2369–2389.

[Kemere et al. 2008] Kemere, C., Santhanam, G., Yu, B. M., Afshar, A., Ryu, S. I., Meng, T. H. & Shenoy, K. V. (2008). Detecting neural-state transitions using hidden Markov models for motor cortical prostheses. *J. Neurophysiol., 100*(4), 2441–2452.

[Lee and Wilson 2002] Lee, A. K. & Wilson, M. A. (2002). Memory of sequential experience in the hippocampus during slow wave sleep. *Neuron, 36*, 1183–1194.

[Lever et al. 2002] Lever, C., Wills, T., Cacucci, F., Burgess, N. & O'Keefe, J. (2002). Long-term plasticity in hippocampal place-cell representation of environmental geometry. *Nature, 416*, 90–94.

[Louie and Wilson 2001] Louie, K. & Wilson, M. A. (2001). Temporally structured REM sleep replay of awake hippocampal ensemble activity. *Neuron, 29*, 145–156.

[MacKay 1997] Mackay, D. J. C. (1997). Ensemble learning for hidden Markov models. Technical Report, Cavendish Laboratory, Cambridge University, UK.

[MacKay 2003] MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

[McGrory and Titterington 2009] McGrory, C. A. & Titterington, D. M. (2009). Variational Bayesian analysis for hidden Markov models. *Australian & New Zealand Journal of Statistics, 51*(2), 227–244.

[McLachlan and Krishnan 2008] McLachlan, G. J. & Krishnan, T. (2008). *The EM Algorithm and Extensions* (2nd edition). Wiley.

[McNaughton et al. 2006] McNaughton, B. L., Battaglia, F. P. Jensen, O., Moser, E. I. & Moser, M. B. (2006). Path integration and the neural basis of the 'cognitive map'. *Nat. Rev. Neurosci., 7*, 663–678.

[O'Keefe and Nadel 1978] O'Keefe, J. & Nadel, N. (1978). *The Hippocampus as a Cognitive Map*. New York: Oxford University Press.

[Pawitan 2001] Pawitan, Y. (2001). *In All Likelihood: Statistical Modelling and Inference Using Likelihood*. Oxford University Press.

[Rabiner 1989] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77*(2), 257–286.

[Robert 2001] Robert, C. P. (2001). *The Bayesian Choice–A Decision-Theoretic Motivation* (2nd edn.). Springer.

[Rydén 2008] Rydén, T. (2008). EM versus Markov chain Monte Carlo for estimation of hidden Markov models: A computational perspective. *Bayesian Analysis, 3*(4), 659–688.

[Scott 2002] Scott, S. L. (2002). Bayesian methods for hidden Markov models: recursive computing in the 21st century. *Journal of the American Statistical Association, 97*, 337–351.

[Skaggs and McNaughton 1996] Skaggs, W. E. & McNaughton, B. L. (1996). Replay of neuronal firing sequences in rat hippocampus during sleep following spatial experience. *Science, 271*, 1870–1873.

[Teh et al. 2006] Teh, Y. W., Jordan, M. I., Beal, M. J., & Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association, 101*, 1566–1581.

[Tollis et al. 1999] Tollis, I. G., di Battista, G., Eades, P., Tamassia, R. (1999). *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall.

[Truccolo et al. 2005] Truccolo, W., Eden, U. T., Fellow, M., Donoghue, J. P., & Brown, E. N. (2005). A point process framework for relating neural spiking activity to spiking history, neural ensemble and covariate effects. *J. Neurophysiol., 93*, 1074–1089.

[van Gael et al. 2008] van Gael, J., Saatci, Y., Teh, Y. W., & Ghahramani, Z. (2008). Beam sampling for the infinite hidden Markov model. *Proc. 25th Int. Conf. Machine Learing*, Helsinski, Findland.

[Viterbi 1967] Viterbi. J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory, 13*, 260–269.

[Watanabe 2009] Watanabe, S. (2009). *Algebraic Geometry and Statistical Learning Theory*. Cambridge University Press.

[Wills et al. 2005] Wills, T., Lever, C., Cacucci, F., Burgess, N. & O'Keefe, J. (2005). Attractor dynamics in the hippocampal representation of the local environment. *Science, 308*, 873–876.

[Wilson and McNaughton 1993] Wilson, M. A. & McNaughton, B. L. (1993). Dynamics of the hippocampal ensemble code for space. *Science, 261*, 1055–1058.

[Wilson and McNaughton 1994] Wilson, M. A. & McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science, 265*, 676–679.

[Wood et al. 2000] Wood, E. R., Dudchenko, P. A., Robitsek, R. J. & Eichenbaum, H. (2000). Hippocampal neurons encode information about different types of memory episodes occurring in the same location. *Neuron, 27*, 623–633.

[Wu et al. 2011] Wu, W., Chen, Z., Gao, S., & Brown, E. N. (2011). A hierarchical Bayesian approach for learning spatio-temporal decomposition of multichannel EEG. *NeuroImage, 56*(4):1929–1945.

[Zemel et al. 1998] Zemel, R.S., Dayan, P. & Pouget, A. (1998). Probabilistic interpretation of population codes. *Neural Computation, 10*, 403–430.

[Zhang et al. 1998] Zhang, K., Ginzburg, I., McNaughton, B. L., & Sejnowski, T. J. (1998): Interpreting neuronal population activity by reconstruction: Unified framework with application to hippocampal place cells. *Journal of Neurophysiology, 79*, 1017–1044.