vbTPM – software documentation

Martin Lindén, bmelinden@gmail.com, May 20, 2014.

vbTPM is an acronym for variational Bayes for Tethered Particle Motion (TPM), and is a software package for analyzing TPM data. The latest version of this manual and the corresponding software is available as open source via `http://sourceforge.net/projects/vbtpm/`.

If you use vbTPM in your research, please cite the original article in your work:

S. Johnson, J-W van de Meent, Rob Phillips, Chris H Wiggins, and Martin Lindén. Multiple Lac-mediated loops revealed by Bayesian statistics and tethered particle motion (manuscript in preparation, 2014).

Contents

1 Getting started with vbTPM

1.1 Installation

Get the source code, and make sure VB7, HMMcore, and tools are in your Matlab path, for example by running the script `vbTPMstart` from the Matlab command prompt. (To add these paths permanently to your Matlab path, see the Matlab documentation).

Make sure that HMMcore/ contains binaries for your system (`VBforwback.mex[sys]`, `VBwAcount.mex[sys]`, `VBviterbi.mex[sys]`, `VBviterbi_log.mex[sys]`, where `[sys]` is some system-dependent extension). If not, a simple Matlab compilation script can be found in HMMcore/. (See Matlab documentation for how to set up your mex compiler). For large data sets, it is probably faster to compile your own binaries.

1.2 System requirements

Tethered particle motion often produces large data sets of many long trajectories, which makes the HMM analysis computer intensive. As an example, one parameter point in our test data sets, about 90 trajectories averaging 45 min, downsampled to 10 Hz, took about 24 h to go through on two 6 core Intel Xeon E5645 2.40GHz processors. The analysis time increases sharply with the number of states (including spurious ones, like transient sticking events).

vbTPM is implemented in Matlab, tested on R2012a and R2013a, on recent Mac and Ubuntu Linux systems, and requires the signal processing and statistics toolboxes.

Windows users will have to do some scripting to translate the bash-based parallelization scheme (below) to some other script language, but should be fine otherwise.

1.3 A small test problem

A small test problem can be found in example1/, with the actual data in example1/lacdata/. The data set has one calibration (cal) and one production (trj) trajectory for each bead, and contains data from five beads. Using the runinput2.m and runscript2.sh files as described below, this data set takes less than 30 min to analyze on the above machine (running all five trajectories in parallel).

1.4 vbTPM workflow

The workflow of vbTPM, summarized in Fig. 1, is based on runinput files that contain all analysis parameters, including information about where the TPM data files are located, and where various results should be written to. These files can therefore be used as handles to an ongoing analysis and to intermediate results.

The three main tools for handling the analysis, marked in yellow in Fig. 1, are
VB7_batch_run.m, which manages the VB analysis of raw position traces using the simple HMM model,
VB7_batch_manage.m, a tool to collect the analysis results, and also to clean up and reset intermediate result files in case the analysis is interrupted, and finally
VB7_batch_postprocess.m, a graphical tool to aid the manual state classifcation and construct factorial models based on this classification.

More advanced analysis beyond this step, including the EB procedure, requires custom matlab scripting.

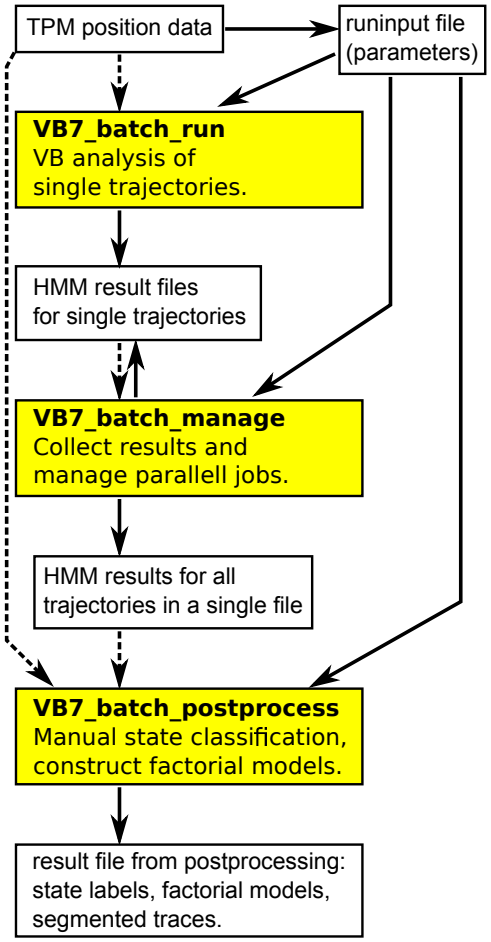

Figure 1: Work flow for TPM analysis using vbTPM. The yellow boxes indicate the three main tools of the vbTPM toolbox and their functions, as described in the text. Solid lines indicate that a file is written by another file, or passed as argument to it. Dashed lines indicate flow of information handled internally by reference to the runinput file.

1.5 Runinput files

Runinput files contain all parameters to run the analysis and access the results. The meaning of the parameters are documented below, and via comments in the runinput files in example1/. `runinput1.m` refers to an already completed analysis (results in example1/HMMresults1/), while `runinput2.m` has not yet run.

The runinput files are parsed by executing them as Matlab scripts, which makes it possible to use Matlab commands to define variables, for example loading filename lists. It also opens the dangerous possibility to interfere with the analysis code in a few instances (so, be careful when introducing temporary variables).

Relative paths specified in runinput files are interpreted relative to the location of the runinput file by vbTPM.

`VB7_getOptions` parses the runinput file and returns a Matlab struct with all runinput parameters as subfields, without interfering with the existing workspace.

1.6 Run basic analysis on single trajectories

To start analyzing the test data set, go to the `example1` folder and type `VB7_batch_run('runinput2')` in the Matlab command prompt. Since the runinput file has `one_at_a_time=true;` this will analyze one trajectory in the data set. Several calls are needed to complete the analysis.

Setting `one_at_a_time=false;` will run analysis of all trajectories in the data set. In our experience however, Matlab tends to hoard memory when several large data sets are analyzed consecutively. To work around that, one can use scripts that starts consecutive Matlab sessions and runs a single trajectory in each.

One example for the bash shell is `runscript1.sh`, which calls `runinput1.m` (there is also a corresponding `runscript2.sh`).

Parallelization

To parallellize, run several instances of the above script at once, with `one_at_a_time=true;` in the runinput file. `VB7_batch_run` keeps track of which trajectories have already been 'checked out', so it is also possible to run on several computers, if the results folder is synced regularly. If the same trajectory is checked out multiple times on different computers, old results are simply overwritten (no harm done if they used the same runinput file).

Manage the analysis

`VB7_batch_manage` is a tool for managing the basic analysis. It can collect the results and write them to a file or return them to the Matlab workspace, count how many trajectories in a data set has been analyzed, and also clean up temporary files from unfinished trajectories, which is useful if an analysis run is interrupted.

By default, collected analysis results are written to a file named `[PIDprefix]_results.mat` (`PIDprefix` is a string set in the runinput file), in the `[target_path]/` folder (see also table 6).

Inspect and classify the results

The GUI for manual state classification is called `VB7_batch_postprocess()`, outlined in Fig. 2. The GUI can be used to inspect the analysis results in detail, and can also start a conversion of the simple HMM models to factorial models for further (manual) analysis. To try it out, use the runinput file `runinput1.m`, which is already analyzed. To access the fitted models directly, use `VB7_batch_manage` with the 'collect' option. The results are returned as cell vectors for calibration and production trajectories, with the same index structure as the filenames in the runinput file.

The results of the state classification are written to `[PIDprefix]_results_analyzed.mat` (in the `[target_path]/` folder) and factorial models are similarly written to `[PIDprefix]_results_analyzed.mat_VB7kin.mat`.

For details on how vbTPM represents the models and contents of results files etc., we refer to section 5.

1.7 Input data format

vbTPM reads individual trajectories as column matrices, one column for x- and one for y-positions, with sampling frequency specified by `fSample` in the runinput file. If the data is not already drift-corrected, vbTPM can do this using a first-order Butterworth filter (`driftcorrection=true;`) with cut-off frequency `fCut`.

Each trajectory is supplied in an individual .mat file, which could either contain a Matlab struct (in which case vbTPM will read the field specified by `calibration_xyfield` or `looping_xyfield`, same for all trajectories), or just the position trace matrix (by setting the xyfield parameters to empty strings).

Trajectories filenames are supplied as string cell vectors `calibration_filename` and `looping_filename` containing in the runinput file. The latter is nested, to allow several production traces per calibration run. Calibration and looping trajectories is analyzed with different model search parameters, and it is usually a good idea to spend more restarts (`restarts_cal, restarts_trj`) and larger initial number of states (`Ninit_cal, Ninit_trj`) on the looping data than on the calibration data. All traces are analyzed individually

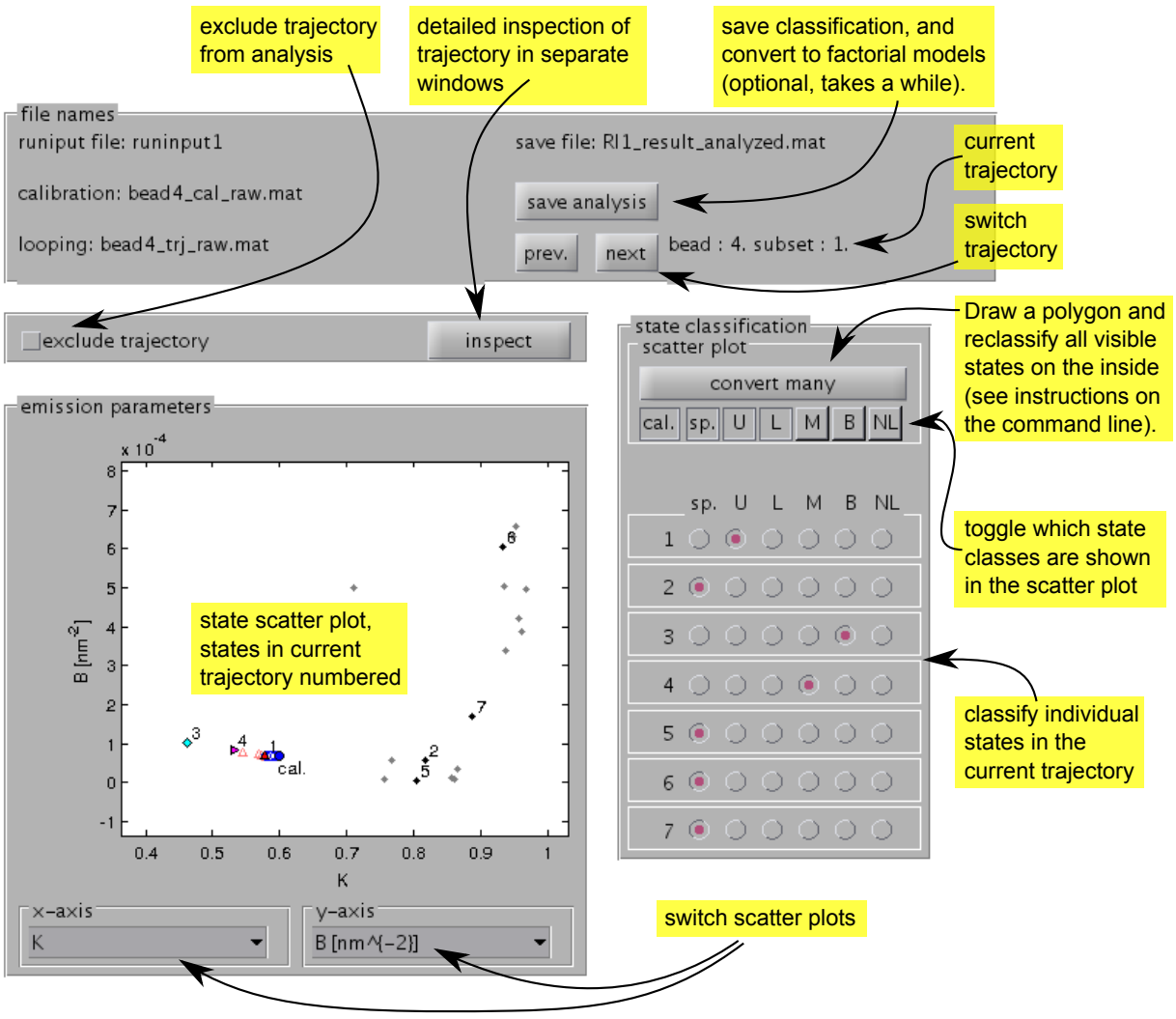

Figure 2: Screenshot of the state classification GUI. There is also an additional window showing the current RMS trace together with the inferred RMS levels and state classifications. The "inspect" button further launches a tool to explore single trajectories and states in more detail (see Fig. 3).

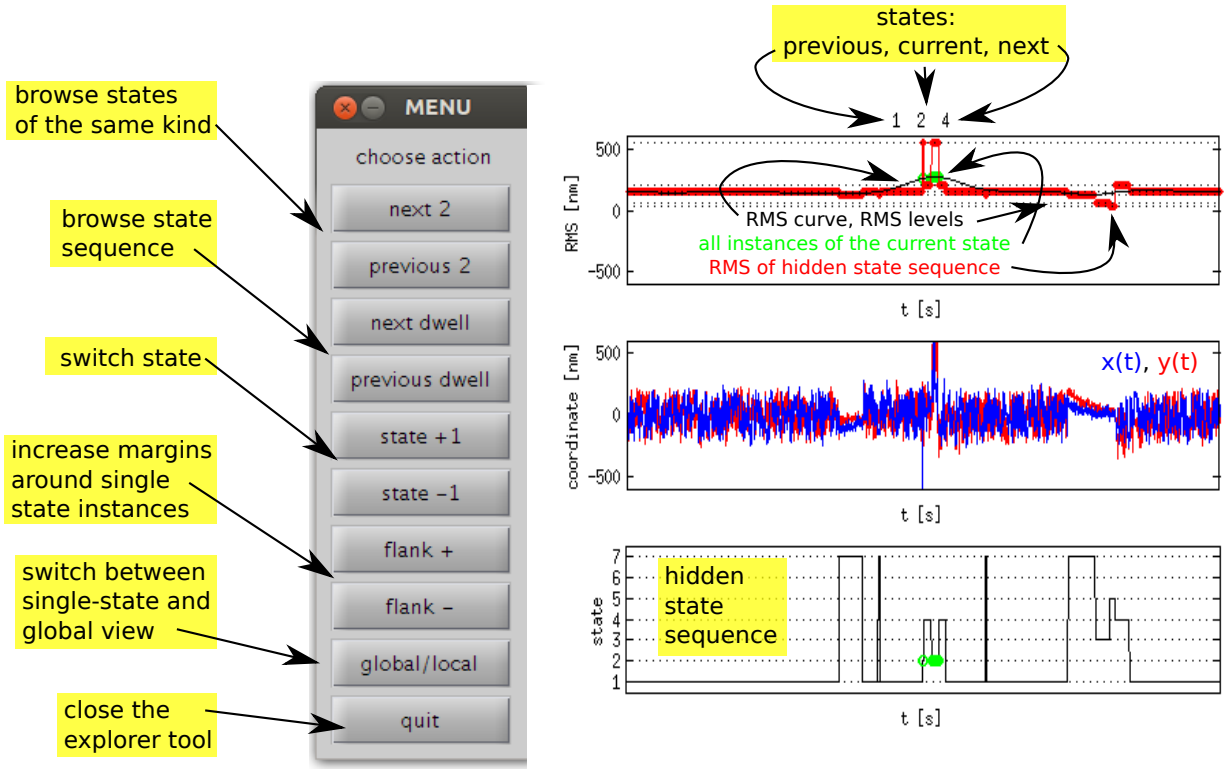

Figure 3: Screenshot of the trajectory explorer tool, launched by the "inspect" button in the state classification GUI.

however, so it is possible to use this structure to organize the data in a different way.

1.8 Other useful scripts

1.8.1 Data and options

VB7_getOptions reads a runinput file and return all variables in a struct.

VB7_preprocess Converts trajectory data to a format that the analysis code uses. Input trajectory should be drift-corrected.

BWdriftcorrect applies driftcorrection to a position trajectory using a Butterworth-filter.

RMSKBgaussfilter computes running averages of RMS and other things, using a Gaussian kernel filter for smoothing.

VB7_getTrjData returns the data for a single trajectory in a runinput file in various formats.

VB7_simTrj is a tool for generating simple synthetic data.

1.8.2 Models

VB7_priorParent A tool to initiate models of various sizes with consistent prior distributions.

VB7_initialGuess_KBregion is a rather complicated function to generate an initial guess for a model struct (e.g., fill out the M and Mc fields) based on analyzing the data.

VB7_GSconversion is a tool to create factorial models, by converting genuine states into spurious ones.

VB7_removeState removes states from a model object.

VB7_findGenuine applies a simple set of rules to determine which states in a given model are genuine and spurious. An analyzed model for the corresponding calibration trace i also needed to provide a baseline.

VB7_inspectStates is a simple tool to navigate in the raw data with the help of a converged model, for example to take a closer look at hard-to-classify states.

1.8.3 VB-EM iterations

VB7_VBEMiter is the computational core of vbTPM, and runs a single VB-EM iteration.

VB7iterator runs VB-EM iteratons of a model until convergence.

VB7_greedySearch runs a model search on a single trace from a given initial guess. Briefly, the search strategy is to systematically remove low-occupancy states until the lower bound F stops increasing.

VB7_analyzeTrace runs several greedy model searches on single traces.

2 Diffusive model for TPM

2.1 Diffusive hidden Markov model

We model the looping dynamics by a discrete Markov process s_t with N states, a transition probability matrix $\mathbf{A}$, and initial state distribution $\boldsymbol{\pi}$,

$$p(s_t|s_{t-1}, \mathbf{A}) = A_{s_{t-1}s_t}, \quad p(s_1) = \pi_{s_t}. \tag{1}$$

This is the standard hidden part of an HMM, and the physics of TPM goes into the emission model, that describes the restricted Brownian motion of the bead. We use a discrete time model of over-damped 2D diffusion in a harmonic potential, that has been suggested as a simplified model for TPM (1, 2),

$$\boldsymbol{x}_t = K_{s_t}\boldsymbol{x}_{t-1} + \boldsymbol{w}_t/(2B_{s_t})^{1/2}, \tag{2}$$

where the index s_t indicate parameters that depend on the hidden state. Thermal noise enters through the uncorrelated Gaussian random vectors $\boldsymbol{w}_t$ with unit variance. The unintuitive parameterization is chosen for computational convenience; K_j and B_j are related to the spring and diffusion constant of the bead, and some insight into their physical meaning can be gained by noting that with a single hidden state, Eq. 2 describes a Gaussian process with zero mean and

$$\begin{aligned} RMS &= \sqrt{\langle \boldsymbol{x}^2 \rangle} = (B(1-K^2))^{-1/2}, \\ \frac{\langle \boldsymbol{x}_{t+m} \cdot \boldsymbol{x}_t \rangle}{\langle \boldsymbol{x}^2 \rangle} &= K^m \equiv e^{-m\Delta t/\tau}, \end{aligned} \tag{3}$$

where Δt is the sampling time, and τ is a bead correlation time. This model thus captures the diffusive character of the bead motion, while still retaining enough simplicity to allow efficient variational algorithms (3, 4).

2.2 Factorial model

The above algorithm is readily extended to treat the factorial model where genuine (s_t) and spurious (c_t) states are separated into two different hidden processes, with $c_t = 1$ indicating a genuine state (bead motion described by genuine parameters K_{s_t}, B_{s_t}), and $c_t > 1$ indicating a spurious state (bead motion described by genuine parameters $\hat{K}_{c_t}, \hat{B}_{c_t}$).

The bead motion/emission parameters are exactly analogous to genuine state parameters in the simple HMM above. For the transitions, we assume that genuine states evolve independently, and likewise for spurious states ($c_t > 1$), but that the transitions from genuine to spurious $c_t = 1 \to c_{t+1} > 1$ depends on the genuine state (see main text (5)). This leads to the following model:

$$p(s_{t+1}, c_{t+1}|s_t, c_t) = p(s_{t+1}|s_t)p(c_{t+1}|s_t, c_t), \tag{4}$$

with $p(s_{t+1}|s_t) = A_{s_t s_{t+1}}$ as earlier, and

$$p(c_{t+1}|s_t, c_t) = \begin{cases} \hat{A}_{s_t c_{t+1}}, & \text{if } c_t = 1, \\ \hat{R}_{c_t c_{t+1}}, & \text{if } c_t > 1, \end{cases} \tag{5}$$

Thus, $\hat{\mathbf{A}}$ is the transition matrix to induce spurious states, and $\hat{\mathbf{R}}$ is the transition matrix to convert between spurious states and to end them.

vbTPM implements a brute force variational treatment of this extended model, where we define new composite hidden states $\hat{s}_t = (s_t, c_t)$ and modify the simple vb algorithm to run this composite model, which mainly involves bookkeeping to update the transition count matrices for $\hat{\mathbf{A}}$, $\hat{\mathbf{R}}$. This has a significant computational cost, since a simple model with $N_{gen.}$ genuine states and $N_{sp.}$ spurious ones gets $N_{gen.} \times (1 + N_{sp.})$ states after conversion. However, since we do not perform exhaustive model search in this

representation and can utilize the simpler model to make good initial guesses, this is not a big problem.

3 The VB-algorithm

We will start by discussing the statistical analysis of the simple HMM in detail, and then discuss generalization to the factorial model.

3.1 Model selection by maximum evidence

Our analysis aims not only to extract parameter values from TPM data, but also to learn the number of hidden states N, corresponding to different DNA-protein conformations. This means that we need to compare models with different number of unknown parameters. We take a Bayesian approach to this problem.

A distinguishing feature of Bayesian data analysis is the treatment of random variables and unknown parameters on an equal footing (6, 7). Hence, given some data $\boldsymbol{x}_{1:T}$ and a set of competing models with different number of states $N = 1, 2, \ldots$ (and *1:T* is a compact way to denote a whole time series), we can use the laws of probability to express our confidence about those models in terms of conditional probabilities,

$$p(N|\boldsymbol{x}_{1:T}) = p(\boldsymbol{x}_{1:T}|N)p(N)/p(\boldsymbol{x}_{1:T}), \qquad (6)$$

where $p(N)$ expresses our beliefs about the different models prior to seeing the data, and $p(\boldsymbol{x}_{1:T})$ is a normalization constant. A Bayesian rule for model selection is therefore to prefer the model that maximizes $p(\boldsymbol{x}_{1:T}|N)$, a quantity known as the evidence. For our more complex model, parameters and hidden states will have to be integrated out,

$$p(\boldsymbol{x}_{1:T}|N) = \int d\theta \sum_{s_{1:T}} p(\boldsymbol{x}_{1:T}, s_{1:T}|\theta)p(\theta|N), \qquad (7)$$

where the first factor in the integrand describes the model, and the second expresses our prior beliefs about the parameters (see below).

The integrand in the evidence, Eq. (7), requires an explicit expression for the probability of a sequence of bead positions and hidden states. This expression can be written down based on the above model, and factorizes in the usual HMM fashion, as

$$\begin{aligned} p(\boldsymbol{x}_{1:T}, s_{1:T}|\theta)p(\theta|N) &= p(\boldsymbol{x}_1)p(s_1|\boldsymbol{\pi}) \\ &\times \prod_{t=2}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, s_t, \boldsymbol{K}, \boldsymbol{B})p(s_t|s_{t-1}, \mathbf{A}) \\ &\times p(\boldsymbol{\pi}|N) \prod_{j=1}^{N} p(K_j, B_j|N)p(A_{j,:}|N), \end{aligned} \qquad (8)$$

where $A_{j,:}$ denote row j of the matrix $\mathbf{A}$. The first right hand side line in Eq. (8) describes the initial state and bead position. We will neglect the factor $p(\boldsymbol{x}_1)$ from now on, but the initial state $p(s_1|\boldsymbol{\pi})$ and transition probabilities $p(s_t|s_{t-1}, \mathbf{A})$ are given by Eq. (1), and the bead motion follows from Eq. (2),

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, s_t, \boldsymbol{K}, \boldsymbol{B}) = \frac{B_{s_t}}{\pi} e^{-B_{s_t}(\boldsymbol{x}_t - K_{s_t}\boldsymbol{x}_{t-1})^2}. \qquad (9)$$

Finally, the last line of Eq. (8) contains prior distributions over parameters conditional on the number of states. We use conjugate priors, parameterized to have minimal impact on the inference results (see SI).

3.2 The variational approximation

An exact computation of the Bayesian evidence is impractical or impossible for most interesting models, and clever approximations are needed. The approximation we use here is variously known as ensemble learning, variational Bayes, or (in statistical physics jargong) mean field theory (4, 7), has previously been applied to biophysical time-series of FRET data (8–10) and *in vivo* single particle tracking (11). The idea is to approximate the log evidence by a lower bound, $\ln p(x|N) \geq F_N$, with

$$F_N = \int d\theta \sum_s q(s)q(\theta) \ln \frac{p(x,s|\theta)p(\theta|N)}{q(s)q(\theta)}, \tag{10}$$

where $q(s)$ and $q(\theta)$ are arbitrary probability distributions over the hidden states and parameters respectively. These are optimized to make the bound as tight as possible for each model, the model that achieves the highest lower bound wins, and the corresponding optimal distributions $q(s)q(\theta)$ can be used for approximate inference about parameter values and hidden states. In particular, optimizing F_N with respect to the variational distributions leads to

$$\ln q(\theta) = -\ln Z_\theta + \ln p(\theta|N) + \langle \ln p(x,s|\theta) \rangle_{q(s)}, \tag{11}$$

$$\ln q(s) = -\ln Z_s + \langle \ln p(x,s|\theta) \rangle_{q(\theta)}, \tag{12}$$

where the Z's are Lagrange multipliers to enforce normalization, and $\langle \cdot \rangle_{q(\cdot)}$ denotes an average over $q(\cdot)$. We solve these equations iteratively until the lower bound converges, repeating the analysis many times with independent initial conditions in order to find a global maximum. The iterative solution approach results in an EM-type variational algorithm, detailed below. We refer to Refs. (3, 11, 12) for details on how to derive variational algorithms for HMMs, and Refs. (4, 7, 12) for more general discussion of variational inference methods.

3.2.1 Parameter distributions

The results of plugging our diffusive HMM into the parameter update equation (11) are as follows. The initial state probability vector, and each row in the transition matrix (denoted $A_{j,:}$), are Dirichlet distributed,

$$q(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{w}^{(\boldsymbol{\pi})}), \tag{13}$$

$$w_j^{(\boldsymbol{\pi})} = \tilde{w}_j^{(\boldsymbol{\pi})} + \langle \delta_{j,s_1} \rangle_{q(s_{1:T})}, \tag{14}$$

$$q(A_{i,:}) = \mathrm{Dir}(A_{i,:}|\boldsymbol{w}^{(\mathbf{A})}), \tag{15}$$

$$w_{ij}^{(\mathbf{A})} = \tilde{w}_{ij}^{(\mathbf{A})} + \sum_{t=2}^{T} \left\langle \delta_{i,s_{t-1}} \delta_{j,s_t} \right\rangle_{q(s_{1:T})}. \tag{16}$$

Here, variables under tilde's ($\tilde{}$) are hyperparameters that parameterize the prior distributions, and can be interpreted as pseudo-observations. The Dirichlet density function is

$$\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{u}) = \frac{\Gamma(u_0)}{\prod_j \Gamma(u_j)} \prod_j \pi_j^{u_j - 1}, \quad u_j > 1, \tag{17}$$

where $u_0 = \sum_j u_j$ is called the strength, and the density is non-zero in the region $0 \leq \pi_j \leq 1$, $\sum_j \pi_j = 1$. Before moving on, we quote some useful expectation values for future reference,

$$\langle \ln \pi_i \rangle_{q(\boldsymbol{\pi})} = \psi(w_i^{(\boldsymbol{\pi})}) - \psi(w_0^{(\boldsymbol{\pi})}), \tag{18}$$

$$\langle \ln A_{ij} \rangle_{q(\mathbf{A})} = \psi(w_{ij}^{(\mathbf{A})}) - \psi(w_{i0}^{(\mathbf{A})}), \tag{19}$$

where $\psi(x)$ is the digamma function, and

$$\langle \pi_i \rangle_{q(\boldsymbol{\pi})} = \frac{w_i^{(\boldsymbol{\pi})}}{w_0^{(\boldsymbol{\pi})}}, \tag{20}$$

$$\mathrm{Var}[\pi_i]_{q(\boldsymbol{\pi})} = \frac{w_i^{(\boldsymbol{\pi})}\big((1 - w_i^{(\boldsymbol{\pi})}\big)}{(w_0^{(\boldsymbol{\pi})})^2\big(1 + w_0^{(\boldsymbol{\pi})}\big)}, \tag{21}$$

$$\langle A_{ij} \rangle_{q(\mathbf{A})} = \frac{w_{ij}^{(\mathbf{A})}}{w_{i0}^{(\mathbf{A})}}, \tag{22}$$

$$\mathrm{Var}[A_{ij}]_{q(\mathbf{A})} = \frac{w_{ij}^{(\mathbf{A})}\big(1 - w_{ij}^{(\mathbf{A})}\big)}{(w_{i0}^{(\mathbf{A})})^2\big(1 + w_{i0}^{(\mathbf{A})}\big)}. \tag{23}$$

The bead motion parameters have the following variational distributions

$$q(K_j, B_j) = \frac{B_j^{n_j}}{W_j} e^{-B_j\left(v_j(K_j - \mu_j)^2 + c_j\right)}, \tag{24}$$

$$W_j = \frac{c^{-(n_j + \frac{1}{2})}\Gamma(n_j + \frac{1}{2})}{\sqrt{v_j/\pi}}, \tag{25}$$

with the range $B_j \geq 0$, $-\infty < K_j < \infty$. Physically, we might rather expect $0 < K_j < 1$, but the extended range for K_j simplifies the calculations a lot, and with enough data, most of the probability mass ends up in the expected interval anyway. The VBM equations are

$$n_j = \tilde{n}_j + M_j, \tag{26}$$

$$c_j = \tilde{c}_j + C_j + \tilde{v}_j\tilde{\mu}_j^2 - \frac{\left(\tilde{v}_j\tilde{\mu}_j + U_j\right)^2}{\tilde{v}_j + V_j}, \tag{27}$$

$$v_j = \tilde{v}_j + V_j, \tag{28}$$

$$\mu_j = \frac{\tilde{v}_j\tilde{\mu}_j + U_j}{\tilde{v}_j + V_j}, \tag{29}$$

$$\tag{30}$$

where hyperparameters describing the prior distribution are again indicated by $\tilde{}$, and the (conjugate) prior distributions are recovered by setting $M_j = C_j = U_j = V_j = 0$. These data-dependent terms are given by

$$M_j = \sum_{t=2}^{T} \langle \delta_{s_t,j} \rangle, \tag{31}$$

$$C_j = \sum_{t=2}^{T} \langle \delta_{s_t,j} \rangle \boldsymbol{x}_t^2, \tag{32}$$

$$V_j = \sum_{t=2}^{T} \langle \delta_{s_t,j} \rangle \boldsymbol{x}_{t-1}^2. \tag{33}$$

$$U_j = \sum_{t=2}^{T} \langle \delta_{s_t,j} \rangle \boldsymbol{x}_t \cdot \boldsymbol{x}_{t-1}. \tag{34}$$

Some useful expectation values for future reference are

$$\langle K_j \rangle_{q(\boldsymbol{B},\boldsymbol{K})} = \mu_j, \tag{35}$$

$$\mathrm{Var}[K_j]_{q(\boldsymbol{B},\boldsymbol{K})} = \frac{c_j}{2v_j(n_j - \frac{1}{2})}. \tag{36}$$

$$\langle B_j \rangle_{q(\boldsymbol{B},\boldsymbol{K})} = \frac{n_j + \frac{1}{2}}{c_j}, \tag{37}$$

$$\mathrm{Var}[B_j]_{q(\boldsymbol{B},\boldsymbol{K})} = \frac{n_j + \frac{1}{2}}{c_j^2}, \tag{38}$$

$$\langle \ln B_j \rangle_{q(\boldsymbol{B},\boldsymbol{K})} = \psi\big(n_j + \frac{1}{2}\big) - \ln c_j, \tag{39}$$

$$\big\langle B_j K_j^2 \big\rangle_{q(\boldsymbol{B},\boldsymbol{K})} = \frac{1}{2v_j} + \mu_j^2 \frac{n_j + \frac{1}{2}}{c_j}, \tag{40}$$

$$\langle B_j K_j \rangle_{q(\boldsymbol{B},\boldsymbol{K})} = \mu_j \frac{n_j + \frac{1}{2}}{c_j}, \tag{41}$$

In addition to being needed during the algorithm, these averages can be used to translate prior knowledge in terms of means and standard deviations of K_j, B_j into prior parameters $\tilde{n}_j, \tilde{c}_j, \tilde{\mu}_j, \tilde{v}_j$

3.2.2 Hidden state distribution

The variational distribution has a simple form,

$$\ln q(s_{1:T}) = -\ln Z + \sum_{t=1}^{T} \ln h_{s_t}(t) + \sum_{t=2}^{T} \ln J_{s_{t-1},s_t}, \tag{42}$$

i.e., an initial state distribution, a point-wise term that depends on the initial conditions and the data, and a transition probability. The point-wise

$$\ln q(s_{1:T}) = -\ln Z + \sum_{t=1}^{T} \ln h_{s_t}(t) + \sum_{t=2}^{T} \ln J_{s_{t-1},s_t}, \tag{43}$$

i.e., an initial state distribution, point-wise terms that depends on the initial conditions and the data, and transition terms. The mathematical form of this expression is the same as encountered in maximum-likelihood optimization of hidden Markov Models, and hence the normalization constant and expectation values needed for the parameter update equations can be computed by the Baum-Welch algorithm (13), which resembles the transfer matrix solution for spin models in statistical physics.

Similarly, and the most likely sequence of hidden states can be computed by the Viterbi algorithm (14).

Specifically, the initial term is given by

$$\ln h_j(1) = \langle p(s_1 = j|\boldsymbol{\pi})\rangle_{q(\boldsymbol{\pi})} = \psi(w_j^{(\boldsymbol{\pi})}) - \psi(w_0^{(\boldsymbol{\pi})}), \tag{44}$$

the point-wise contributions for $t > 1$ are

$$\ln h_j(t) = \psi\big(n_j + \frac{1}{2}\big) - \ln(\pi c_j) - \frac{\boldsymbol{x}_{t-1}^2}{2v_j} - \frac{n_j + \frac{1}{2}}{c_j}\left(\boldsymbol{x}_{t-1}^2\left(\mu_j - \frac{\boldsymbol{x}_t \cdot \boldsymbol{x}_{t-1}}{\boldsymbol{x}_{t-1}^2}\right)^2 + \boldsymbol{x}_t^2 - \frac{\big(\boldsymbol{x}_t \cdot \boldsymbol{x}_{t-1}\big)^2}{\boldsymbol{x}_{t-1}^2}\right), \tag{45}$$

and the transition terms are given by

$$\ln J_{ji} = \psi\big(w_{j,i}^{(\mathbf{A})}\big) - \psi\big(\sum_{k=1}^{N} w_{j,k}^{(\mathbf{A})}\big). \tag{46}$$

3.3 VBEM iterations and model search

The iterative optimization of the variational distributions are done as follows. To start with, an initial guess for the variational parameter distributions are generated. We then alternate between VBE step, in which we construct the hidden state distribution and compute the averages $\langle \delta_{j,s_t} \rangle_{q(s)}$ and $\big\langle \delta_{j,s_t}\delta_{k,s_{t+1}} \big\rangle_{q(s)}$ in a Baum-Welch forward-backward sweep, and a VBM step, in which we use these averages to update the parameter variational distributions, until the lower bound converges.

The variational approach has the additional useful tendency to penalizing overfitting already during the VBEM iterations, by depopulating superfluous states (3, 11, 12). We exploit this property by using a greedy search algorithm to explore the model space. The basic strategy is to start by fitting a model with many states from random initial conditions, and then exploring less complex models by gradually removing the least populated states. This saves computing time by supplying good initial guesses for the

low complexity models (which therefore converge quickly), and by lowering the number of independent restarts, since it is easier to construct a good initial guess for a model with many states.

3.4 The lower bound

The lower bound has an especially simple form just after the VBE step (3, 11, 12), given by the normalization constant $\ln Z$ of the variational hidden state distribution, minus the Kullback-Leibler divergences between the variational and prior parameter distributions,

$$\begin{aligned} F = \ln Z &- \int d\boldsymbol{\pi} q(\boldsymbol{\pi}) \ln \frac{q(\boldsymbol{\pi})}{p(\boldsymbol{\pi})} \\ &- \sum_{j=1}^{N} \left[\int d^N A_{j,:}\; q(A_{j,:}) \ln \frac{q(A_{j,:})}{p_0(A_{j,:})} \right. \\ &\left. + \int dB_j dK_j\; q(B_j, K_j) \ln \frac{q(B_j, K_j)}{p_0(B_j, K_j)} \right]. \end{aligned} \tag{47}$$

The Kullback-Leibler terms can be expressed in terms of the expectation values computed above. For the initial state distribution, we get

$$\begin{aligned} \int d\boldsymbol{\pi} q(\boldsymbol{\pi}) \ln \frac{q(\boldsymbol{\pi})}{p_0(\boldsymbol{\pi})} &= \ln \tilde{w}_0^{(\boldsymbol{\pi})} - \psi(\tilde{w}_0^{(\boldsymbol{\pi})}) - \frac{1}{\tilde{w}_0^{(\boldsymbol{\pi})}} \\ + \sum_{j=1}^{N} \left[\left(w_j^{(\boldsymbol{\pi})} - \tilde{w}_j^{(\boldsymbol{\pi})}\right) \psi(w_j^{(\boldsymbol{\pi})}) - \ln \frac{\Gamma(w_j^{(\boldsymbol{\pi})})}{\Gamma(\tilde{w}_j^{(\boldsymbol{\pi})})} \right]. \end{aligned} \tag{48}$$

To get this simple form, we used that $w_0^{(\boldsymbol{\pi})} = 1 + \tilde{w}_0^{(\boldsymbol{\pi})}$ (since $\sum_j \langle \delta_{j,s_1} \rangle = 1$), and the identities $\Gamma(x+1) = x\Gamma(x)$ and $\psi(x+1) = \psi(x) + \frac{1}{x}$. Furthermore, each row of the transition probability matrix contributes

$$\begin{aligned} &\int d^N A_{j,:}\; q(A_{j,:}) \ln \frac{q(A_{j,:})}{p_0(A_{j,:})} \\ &\quad = \ln \frac{\Gamma(w_{j0}^{(\mathbf{A})})}{\Gamma(\tilde{w}_{j0}^{(\mathbf{A})})} - (w_{j0}^{(\mathbf{A})} - \tilde{w}_{j0}^{(\mathbf{A})}) \psi(w_{j0}^{(\mathbf{A})}) \\ &- \sum_{k=1}^{N} \left[\ln \frac{\Gamma(w_{jk}^{(\mathbf{A})})}{\Gamma(\tilde{w}_{jk}^{(\mathbf{A})})} - \left(w_{jk}^{(\mathbf{A})} - \tilde{w}_{jk}^{(\mathbf{A})}\right) \psi(w_{jk}^{(\mathbf{A})}) \right]. \end{aligned} \tag{49}$$

Finally, the emission parameter of each state contributes

$$\begin{aligned} &\int dB_j \int d^N K_j\; q(B_j, K_j) \ln \frac{q(B_j, K_j)}{p(B_j, K_j)} = \dots \\ &\quad = -\frac{n_j + \frac{1}{2}}{c_j} \left(c_j - \tilde{c}_j - \tilde{v}_j (\mu_j - \tilde{\mu}_j)^2 \right) \\ &+ \frac{1}{2} \ln \frac{v_j}{\tilde{v}_j} + (\tilde{n}_j + \frac{1}{2}) \ln \frac{c_j}{\tilde{c}_j} - \ln \frac{\Gamma\left(n_j + \frac{1}{2}\right)}{\Gamma\left(\tilde{n}_j + \frac{1}{2}\right)} \\ &\quad + (n_j - \tilde{n}_j) \psi\left(n_j + \frac{1}{2}\right) + \frac{\tilde{v}_j}{2 v_j} - \frac{1}{2}. \end{aligned} \tag{50}$$

3.5 Specification of prior distributions

Emission parameters We specify priors for the emission parameters K, B in terms of the the mean values $\langle K_j \rangle$, $\langle B_j \rangle$, the standard deviation of K_j, and the number of pseudo-counts $\tilde{n}_j$, and then solve for the remaining hyper-parameters $\tilde{c}_j, \tilde{\mu}_j, \tilde{v}_j$ using Eqs. (35-38). We take the same hyperparameters for all states, and independent of the number of states N. Priors for spurious states are specified analogously.

Initial state The initial state probability vector has a Dirichlet prior with weights $\tilde{\boldsymbol{w}}^{(\boldsymbol{\pi})}$ (see Eq. (14)). We choose a constant total strength $f^{(\boldsymbol{\pi})}$, i.e.,

$$\tilde{w}_j^{(\boldsymbol{\pi})} = f^{(\boldsymbol{\pi})}/N. \tag{51}$$

Transition matrix The prior for the transition matrix is independent Dirichlet distributions for each row (see Eq. (16)), with pseudocount matrix $\tilde{w}_{ij}^{(\mathbf{A})}$. Following Persson et al. (11), we parameterize this prior in terms of an expected mean dwell time and an overall number of pseudocounts (prior strength) for each hidden state. To make the definition invariant under changes of sampling time, we specify the strength t_A and prior mean dwell time t_D in time units. We then construct a transition *rate* matrix Q with mean dwell time t_D,

$$Q_{ij} = \frac{1}{t_D}\left(-\delta_{ij} + \frac{1-\delta_{ij}}{N-1}\right), \tag{52}$$

and construct the pseudo-counts based on the transition probability propagator per unit time step,

$$\tilde{w}_{ij}^{(\mathbf{A})} = \frac{t_A f_{sample}}{n_{downsample}} e^{\Delta t Q}. \tag{53}$$

This expression uses the downsampled timestep $\Delta t = n_{downsample}/f_{sample}$, where f_{sample} is the sampling frequency (30 Hz in our case), and $n_{downsample}$ is the downsampling factor. Numerical experiments by Persson et al. (11) show that choosing the strength too low compared to the mean dwell time produces a bias towards sparse transition matrices. This is often not desirable, and we use $t_D = 1$ s and $t_A = 5$ s as default parameters, and also throughout our work (5).

Details on how top specify parameters for prior distributions are given in table 1.

Priors for factorial models We take the priors for the spurious states to be as similar to the genuine state prior as possible. Thus, we parameterize the priors for $\hat{K}_j, \hat{B}_j$ in the same way as for the genuine state parameters. For the spurious→spurious transitions $\hat{\mathbf{R}}$, we use the same prior mean dwell time and strength as for the genuine transition matrix $\mathbf{A}$. For the genuine→spurious transition matrix $\hat{\mathbf{A}}$, we use the same strength, but a different (longer) mean life-time `tDc0` for the state $c_t = 1$, since spurious events are presumably rare.

Further details: tables 1, the example runinput files, and `VB7_priorParent`.

3.6 Model search

To find the best number of states, we adopt a greedy search approach with multiple restarts, similar to the one used in vbSPT (11). The basic idea is to exploit the fact that the variational Bayes EM iterations tend to sense overfitting by making superfluous states unpopulated. Each search sweep thus starts by converging the largest model to be considered, and then gradually removes low-occupancy states until the lower bound F does not improve any more. We also attempt to reconverge each size with some extra pseudo-counts added to the transition matrix to avoid getting stuck in non-interconverting models.

Initial conditions for the largest model are generated randomly, partly based on user-specified intervals for the K and B parameters, and partly using the actual data to find high-occupancy values of K, B. For the transition matrices, we initialize using a mean dwell time, similar as for the prior distributions, but with higher total strength. Details are given in the documentation of `VB7_initialGuess_KBregion.m`, and in the example runinput file.

Table 1: Specifying prior parameters in runinput files. See also the documentation of VB7_priorParent, and the example runinput files.

variable	comment	Eq.
`B0`	$\langle B_j \rangle$	(37)
`fB`	$\tilde{n}_j$	(26)
`K0`	$\tilde{\mu}_j$	(29)
`Kstd`	$\sqrt{\mathrm{Var} K_j}$	(36)
`KBscaling`	Possibility to specify an obsolete state-dependence of the prior parameters for K_j, B_j. We recommend the default value 2.	
	`Bc0`, `fBc`, `Kc0`, and `KcStd` are spurious state parameters, with the same interpretation as for genuine states.	
`fPi`	$f^{(\boldsymbol{\pi})}$	(51)
`tD`	Mean prior dwell time t_D. Default 1 s.	(52)
`tA`	$\mathbf{A}$ prior strength t_A. Default 5 s.	(53)
`tDc0`	Mean prior dwell time of the genuine indicator state $c_t = 1$. Default 10 s.	

4 Extensions

4.1 Empirical Bayes update equations

The empirical Bayes update equations optimizes the lower bound with respect to the hyperparameters in the prior distribution. This means optimizing sums of Kullback-Leibler divergence terms.

The initial state probability, and the rows of the transition probability matrix, are both Dirichlet distributed. Thus, for M trajectories with Dirichlet parameters $u_j^{(i)}$, $i = 1, 2, \ldots, M$, and hyperparameters $\tilde{u}_j$ ($u = w^{(\boldsymbol{\pi})}, u^{(\mathbf{A})}$), we need to solve

$$\frac{d}{d\tilde{u}_j} \sum_i \left(\ln \frac{\Gamma(u_0^{(i)})}{\Gamma(\tilde{u}_0)} - (u_0^{(i)} - \tilde{u}_0)\psi(u_0^{(i)}) - \sum_{k=1}^{N} \left[\ln \frac{\Gamma(u_k^{(i)})}{\Gamma(\tilde{u}_k^{(i)})} - \left(u_k^{(i)} - \tilde{u}_k\right)\psi(u_k^{(i)}) \right] \right) = 0, \tag{54}$$

where $u_0^{(i)} = \sum_k u_k^{(i)}$ and similar for $\tilde{u}_0$. This leads to the update equations

$$\psi(\tilde{u}_0) - \psi(\tilde{u}_j) = \frac{1}{M} \sum_i \left(\psi(u_0^{(i)}) - \psi(u_j^{(i)}) \right). \tag{55}$$

A numerical solution turned out to be easier using the variables $\tilde{U}_j = \ln \tilde{u}_j$ (to numerically enforce $\tilde{u}_j > 0$).

For the emission parameters, the update equations are instead derived from minimizing

Eq. (50) summed over M trajectories,

$$f_{KB} = \sum_i \Bigg(-\frac{n^{(i)}+\frac{1}{2}}{c^{(i)}}\Big(c^{(i)} - \tilde{c} - \tilde{v}(\mu^{(i)}-\tilde{\mu})^2\Big) + \frac{1}{2}\ln\frac{v^{(i)}}{\tilde{v}} + (\tilde{n}+\frac{1}{2})\ln\frac{c^{(i)}}{\tilde{c}} - \ln\frac{\Gamma\big(n^{(i)}+\frac{1}{2}\big)}{\Gamma\big(\tilde{n}+\frac{1}{2}\big)} + (n^{(i)}-\tilde{n})\psi\big(n^{(i)}+\frac{1}{2}\big) + \frac{1}{2}\Big(\frac{\tilde{v}}{v^{(i)}}-1\Big)\Bigg). \quad (56)$$

Minimizing with respect to $\tilde{\mu}$ and $\tilde{v}$ leads to

$$\tilde{\mu} = \frac{1}{M}\sum_i \mu^{(i)}, \quad (57)$$

$$\frac{1}{\tilde{v}} = \frac{1}{M}\sum_i \Big(\frac{1}{v^{(i)}} + 2(\tilde{\mu}-\mu^{(i)})^2\Big). \quad (58)$$

The remaining $\tilde{c}$ and $\tilde{n}$ lead to

$$\frac{\tilde{n}+\frac{1}{2}}{\tilde{c}} = \frac{1}{M}\sum_i \frac{n^{(i)}+\frac{1}{2}}{c^{(i)}}, \quad (59)$$

$$\ln\tilde{c} - \psi\big(\tilde{n}+\frac{1}{2}\big) = \frac{1}{M}\sum_i \Big(\ln c^{(i)} - \psi\big(n^{(i)}+\frac{1}{2}\big)\Big), \quad (60)$$

which we solve numerically. This gets easier by defining $\alpha = \frac{\tilde{n}+\frac{1}{2}}{\tilde{c}}$, then solve the second equation for $\tilde{c}$ numerically, and finally compute $\tilde{n} = \alpha\tilde{c} - \frac{1}{2}$.

4.2 Doing empirical Bayes

Our empirical Bayes analysis of multiple models was not implemented as part of our analysis pipeline, but instead ran using tailored scripts. These are not included with vbTPM, but the optimization procedures for the individual prior distributions are included, in `VB7_EBupdate_dirichlet`, and `VB7_EBupdate_KB`.

Table 2: Fields in a model object W.

field	
`W.N`	N, number of (genuine) states.
`W.Nc`	Number of indicator states $\hat{N}$+1. W.Nc=1 means no spurious states, i.e., the simple HMM.
`W.F`	Lower bound F.

5 Notation and symbols

vbSPT stores mathematical objects in Matlab structures that contain parameters for the variational and prior distributions, and various other things. In this section we list some of them.

First, `VB7_batch_manage` with the collect option returns a filename, and cell vectors of structs that contain results of the model search for each trajectory, indexed as the filenames in the run-input file, e.g. `trj{k}{j}` contains the analysis result for `looping_filename{k}{j}` etc.

Most importantly, the Wtrj and Wcal fields are the converged model structs, whose content are detailed below (using W as the generic model name). In addition the columns of the arrays `NFtrj,NFcal` contain $N, \hat{N}, F, iter$ for each model that was converged during the model search, and *iter* is the restart number that produced it. In `NFitrj,NFical`, the best model for each size is listed in the same way, with the last column indicating the rows in `NFtrj,NFcal` where these optimal models can be found.

Further details about the model structs are given in tables 2-5.

Table 3: Representation of variational distributions for genuine states in a model object W.

field	symbol	Eq.
`W.M.wPi(j)`	$w_j^{(\boldsymbol{\pi})}$	
`W.PM.wPi(j)`	$\tilde{w}_j^{(\boldsymbol{\pi})}$	(14)
`W.E.ds_1(j)`	$\langle \delta_{j,s_1} \rangle$	
`W.PM.wA(i,j)`	$\tilde{w}_{ij}^{(\mathbf{A})}$	
`W.M.wA(i,j)`	$w_{ij}^{(\mathbf{A})}$	(16)
`W.E.wA(i,j)`	$\sum_{t=2}^{T} \langle \delta_{i,s_{t-1}} \delta_{j,s_t} \rangle$	
`W.PM.n(j)`	$\tilde{n}_j$	(26)
`W.M.n(j)`	n_j	(26)
`W.E.M(j)`	M_j	(31)
`W.PM.c(j)`	$\tilde{c}_j$	(27)
`W.M.c(j)`	c_j	(27)
`W.E.C(j)`	C_j	(32)
`W.PM.v(j)`	$\tilde{v}_j$	(28)
`W.M.v(j)`	v_j	(28)
`W.E.V(j)`	V_j	(33)
`W.PM.mu(j)`	$\tilde{\mu}_j$	(29)
`W.M.mu(j)`	μ_j	(29)
`W.E.U(j)`	U_j	(34)

Table 4: Representation of variational distributions for indicator states c_t in a model object named W. Fields relating to the emission model (.n, .c, .v, .mu, etc.) have the same meaning as for the genuine states s_t, except that their first element is not used, since $c_t = 1$ indicate a genuine state. Transition counts (.wA, .wR) are similarly analogous to those parameterizing the genuine transition matrix $\mathbf{A}$.

field	symbol
`W.Mc.wPi(j)`	$w_j^{(\hat{\boldsymbol{\pi}})}$
`W.PMc.wPi(j)`	$\tilde{w}_j^{(\hat{\boldsymbol{\pi}})}$
`W.Ec.ds_1(j)`	$\langle \delta_{j,c_1} \rangle$
`W.PMc.wA(j,k)`	$\tilde{w}_{jk}^{(\hat{\mathbf{A}})}$
`W.Mc.wA(j,k)`	$w_{jk}^{(\hat{\mathbf{A}})}$
`W.Ec.wA(j,k)`	$\sum_t \langle \delta_{j,s_t} \delta_{k,c_{t+1}} \delta_{1,c_t} \rangle$
`W.PMc.wR(j,k)`	$\tilde{w}_{jk}^{(\hat{\mathbf{R}})}$
`W.Mc.wR(j,k)`	$w_{jk}^{(\hat{\mathbf{R}})}$
`W.Ec.wR(j,k)`	$\sum_t \langle \delta_{j,c_t} \delta_{k,c_{t+1}} (1 - \delta_{1,c_t}) \rangle$

Table 5: Selected fields that characterize converged models (named W). The fields W.est and W.est2 are constructed by VB7_VBEMiter.m (although W.est2 must be specifically requested), and fields not mentioned here can be looked up there. Averages are w.r.t. variational parameter distributions unless stated otherwise.

field	comment
`W.est.sAverage`	Occupation probability of genuine states s_t, computed by classification, i.e., sAverage(j) proportional to $\sum_t \langle \delta_{j,s_t} \rangle$.
`W.est.cAverage`	Occupation probability of indicator states c_t, by classification.
`W.est.sVisible`	Occupation probability of genuine states s_t, by classification that excludes spurious states, i.e., sAverage(j) proportional to $\sum_t \langle \delta_{j,s_t} \delta_{1,c_t} \rangle$.
`W.est.A`	Mean transition probabilities for genuine states, $\langle \mathbf{A} \rangle_{q(\mathbf{A})}$.
`W.est.dA`	Standard deviation of transition probability matrix $\mathbf{A}$.
`W.est.tD`	Mean dwell times of genuine states in units of time, computed from the elements of $\langle \mathbf{A} \rangle$.
`W.est.lnQss`	Log average transition probabilities, goes into $q(s_{1:T})$.
	Corresponding averages for spurious state distributions are also computed, Ac, dAc, Rc, dRc, tStick, lnQcc, lnQsc, tStick, tUnstick.
`W.est.sKaverage(j)`	$\langle K_j \rangle = \mu_j$.
`W.est.sBaverage(j)`	$\langle B_j \rangle$.
`W.est.sRMS(j)`	$RMS_j = \sqrt{\langle \boldsymbol{x}_t^2 \vert s_t = j \rangle} \approx (\langle B_j \rangle (1 - \langle K_j \rangle^2))^{-\frac{1}{2}}$.
`W.est.sTC(j)`	Approx. correlation time $\tau_j \approx -\Delta t / \log \langle K_j \rangle$, units of time.
`W.est.sKstd(j)`	Standard deviation of K_j.
`W.est.sBstd(j)`	Standard deviation of B_j.
`W.est.cXXX`	Corresponding properties of spurious states are named with $s \to c$.
`W.est2.qt`	State occupancy probability for combined states (s_t, c_t). Use sMap and cMap to extract genuine/spurious occupancies, e.g., $p(s_t = j)$ =sum(W.est2.qt(t,:).*(W.est.sMap==j)).
`W.est2.sMaxP(t)`	Most likely genuine state at time t.
`W.est2.cMaxP(t)`	Most likely indicator state at time t.
`W.est2.sViterbi`	Viterbi path (most likely sequence of states) for s_t.
`W.est2.cViterbi`	Viterbi path (most likely sequence of states) for c_t.
	W.est2 also contains a few other intermediate fields from the VBEM iteration that are mainly good for debugging. This substructure is thus very bulky and somewhat expensive to compute, which is the reason computing it is optional.

Table 6: Output fields from the analysis and state classification GUI `VB7_batch_postprocess`, where `[PIDprefix]` and `[targetpath]` refers to string variable set in the runinput file.

Fields in results file `[PIDprefix]_result.mat`, from `VB7_batch_manage`.	
`cal{k}{b}`	
`trj{k}{b}`	
`trj{k}{b}.Wtrj`	Converged model object for this trajectory.
`cal{k}.Wcal`	Converged model object for this calibration trajectory.
Additional fields in output file from the state classification GUI `VB7_batch_postprocess`, stored by default in `[PIDprefix]_result_analyzed.mat`.	
`hastrj{k}(b)`	=1 if there is trajectory data for `trj{k}{b}`, 0 otherwise.
`includetrj{k}(b)`	=0 if the trajectory was excluded during the GUI session, 1 otherwise. No factorial models are produced from excluded trajectories.
`calGenuine(k)`	State index to the genuine state in calibration trajectory `k`.
`statelabels{s}`	Cell vector of state labels (strings) used to label the states by `VB7_batch_postprocess`.
`trjstates{k}{b}(j)`	Indicates the state class of state `j` in `trj{k}{b}`, with index referring to `statelabels`.
`isSorted`	=1 if the states in the trajectories have been reordered to put spurious states last.
`statelabels_Wxxx`	State labels in different order (obsolete).
Additional output from using the option to create factorial models when saving from the classification GUI (see Fig. 2). This creates a file with the inelegant name `[PIDprefix]_result_analyzed.mat_VB7kin.mat` in the `[targetpath]` folder.	
`trj{k}{b}.Wgs`	Factorial model object with genuine and spurious states separated.
`RMS`	RMS trajectory.
`trj_vit_s`	Segmented state sequence for a trajectory (or calibration trace, if `cal_`) from the simple HMM, using the Viterbi algorithm (or point-wise most likely states. if `_sMaxP_`).
`trj_vit_td`	Segmented state lifetimes corresponding to `trj_vit_s` etc.

References

[1] John F. Beausang, Chiara Zurla, Laura Finzi, Luke Sullivan, and Philip C. Nelson. Elementary simulation of tethered brownian motion. *American Journal of Physics*, 75(6):520–523, 2007. doi: 10.1119/1.2710484.

[2] Moshe Lindner, Guy Nir, Anat Vivante, Ian T. Young, and Yuval Garini. Dynamic analysis of a diffusing particle in a trapping potential. *Phys. Rev. E*, 87 (2), 2013. doi: 10.1103/PhysRevE.87.022716. URL `http://link.aps.org/doi/10.1103/PhysRevE.87.022716`.

[3] D. J. C. MacKay. Ensemble learning for hidden Markov models. accessed Feb 15 2011, 1997. URL `http://www.inference.phy.cam.ac.uk/mackay/abstracts/ensemblePaper.html`.

[4] Christopher Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006. ISBN 9780387310732.

[5] Stephanie Johnson, Jan-Willem van de Meent, Rob Phillips, Chris H. Wiggins, and Martin Lindén. Multiple lac-mediated loops revealed by bayesian statistics and tethered particle motion. 2013. (in preparation).

[6] Sean R Eddy. What is Bayesian statistics? *Nat. Biotech.*, 22(9):1177–1178, 2004. doi: 10.1038/nbt0904-1177.

[7] David MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, Cambridge UK ;;New York, 2003. ISBN 9780521642989.

[8] Jonathan E. Bronson, Jingyi Fei, Jake M. Hofman, Ruben L. Gonzalez Jr., and Chris H. Wiggins. Learning rates and states from biophysical time series: A bayesian approach to model selection and Single-Molecule FRET data. *Biophysical Journal*, 97(12):3196–3205, 2009. doi: 10.1016/j.bpj.2009.09.031.

[9] Jan-Willem van de Meent, Jonathan E. Bronson, Ruben L. Gonzalez Jr., and Chris H. Wiggins. Learning biochemical kinetic models from single-molecule data with hierarchically-coupled hidden markov models. 2012. manuscript in preparation.

[10] Kenji Okamoto and Yasushi Sako. Variational bayes analysis of a photon-based hidden markov model for single-molecule FRET trajectories. *Biophys. J.*, 103(6):1315–1324, 2012. doi: 10.1016/j.bpj.2012.07.047.

[11] Fredrik Persson, Martin Lindén, Cecilia Unoson, and Johan Elf. Extracting intracellular diffusive states and transition rates from single-molecule tracking data. *Nat. Meth.*, 10(3):265–269, 2013. doi: 10.1038/nmeth.2367.

[12] Matthew Beal. *Variational Algorithms for approximate Bayesian inference*. PhD thesis, University of Cambridge, UK, 2003.

[13] L.E. Baum, T. Petrie, G. Soules, , and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41:164–171, 1970.

[14] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.