vbTPM user guide and manual

Martin Lindén, bmelinden@gmail.com, September 11, 2013.

Contents

To-do list

Prior distributions. Maybe move some of it to manuscript SI?
Model search.
Factorial model.
Factorial algorithm.

1 Getting started with vbTPM

1.1 Installation

Get the source code, and make sure VB7, HMM-core, 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 systems. If not, a simple matlab compilation script can be found in HMMcore/.

1.2 Hardware 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, down-sampled 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).

1.3 A small test problem

A small test problem, which runs on a (fast) laptop in about one hour, with examples of data and runinput files, 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.

Runinput files

Runinput files contain all parameters to run the analysis and access the results. The meaning of the parameters are documentet in the help text of `VB7_batch_run.m`, and commented in those files. `runinput1.m` refers to an already completed analysis (results in example1/HMMresults1/), while `runinput2.m` has not yet run.

Run basic analysis

To start analyzing the test data set, type `VB7_batch_run('runinput2')` in the Matab 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.

In our experience, matlab tends to hoard memory if several large data sets are analyzed consecutively. To work around that, one can use scripts that starts several matlab sessions with a single trajectory in each. An example for the bash shell is `runscript1.sh`, which calls `runinput1.m`. To parallellize, run several instances of the script at once. `VB7_batch_run` keeps track of which trajectories have already been 'checked out', so it is also possible to run on several computers, it the results folder is synced regularly (if the same trajectory is checked out multiple times, old results are simply overwritten).

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, 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.

Access the results

The GUI for manual state classification is called `VB7_batch_postprocess()`. The GUI can be used to inspect the analysis results in detail, and can also convert the simple HMM models to factorial models for further 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.

For details on how vbTPM represents the models etc., we refer to section 3.9.

1.4 Other useful scripts

1.4.1 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.

1.4.2 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.

1.4.3 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 $\vec{\pi}$,

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

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),

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

where the index s_t indicate parameters that depend on the hidden state. Thermal noise enters through the uncorrelated Gaussian random vectors $\vec{w}_t$ with unit variance. The unintuitive parametrization 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

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

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

We would like to filter out experimental artifacts such as transient sticking events and tracking errors from the data. To do this, we introduce a second hidden state c_t that works like an indicator variable: $c_t = 1$ indicates genuine TPM, with bead motion described by K_{s_t}, B_{s_t}, but when $c_t > 1$, the bead motion is instead described by parameters $\hat{K}_{c_t}, \hat{B}_{c_t}$, while s_t evolves completely unseen. We also allow the transition probabilities from $c_t = 1$ to $c_t > 1$ to depend on s_t, to allow for the possibility that sticking events happen more easily in looped than unlooped states for example. Thus, the hidden states evolve according to

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

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

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

and initial probabilities

(**?**)

3 The VB-algorithm

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 (5, 6). Hence, given some data $\vec{x}_{1:T}$ 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,

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

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

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

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

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

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 $p(\vec{x}_1)$ from now on, but the initial state $p(s_1|\vec{\pi})$ and transition probabilities $p(s_t|s_{t-1}, \mathbf{A})$ are given by Eq. (1), and the bead motion follows from Eq. (2),

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

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, 6), has previously been applied to biophysical time-series of FRET data (7–9) and *in vivo* single particle tracking (10). The idea is to approximate the log evidence by a lower bound, $\ln p(x|N) \geq F_N$, with

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

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

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

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

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, 10, 11) for details on how to derive variational algorithms for HMMs, and Refs. (4, 6, 11) for more general discussion of variational inference methods.

3.2.1 Parameter distributions

The results of plugging our diffusinve 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,

$$
\begin{aligned}
q(\vec{\pi}) =& \mathrm{Dir}(\vec{\pi}|\vec{w}^{(\vec{\pi})}), & (13)\\
w_j^{(\vec{\pi})} =& \tilde{w}_j^{(\vec{\pi})} + \langle \delta_{j,s_1} \rangle_{q(s_{1:T})}, & (14)\\
q(A_{i,:}) =& \mathrm{Dir}(A_{i,:}|\vec{w}^{(\mathbf{A})}), & (15)\\
w_{ij}^{(\mathbf{A})} =& \tilde{w}_{ij}^{(\mathbf{A})} + \sum_{t=2}^{T} \left\langle \delta_{i,s_{t-1}} \delta_{j,s_t} \right\rangle_{q(s_{1:T})}. & (16)
\end{aligned}
$$

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

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

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,

$$
\begin{aligned}
\langle \ln \pi_i \rangle_{q(\vec{\pi})} =& \psi(w_i^{(\vec{\pi})}) - \psi(w_0^{(\vec{\pi})}), & (18)\\
\langle \ln A_{ij} \rangle_{q(\mathbf{A})} =& \psi(w_{ij}^{(\mathbf{A})}) - \psi(w_{i0}^{(\mathbf{A})}), & (19)
\end{aligned}
$$

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

$$
\begin{aligned}
\langle \pi_i \rangle_{q(\vec{\pi})} =& \frac{w_i^{(\vec{\pi})}}{w_0^{(\vec{\pi})}}, & (20)\\
\mathrm{Var}[\pi_i]_{q(\vec{\pi})} =& \frac{w_i^{(\vec{\pi})}\big((1 - w_i^{(\vec{\pi})}\big)}{(w_0^{(\vec{\pi})})^2\big(1 + w_0^{(\vec{\pi})}\big)}, & (21)\\
\langle A_{ij} \rangle_{q(\mathbf{A})} =& \frac{w_{ij}^{(\mathbf{A})}}{w_{i0}^{(\mathbf{A})}}, & (22)\\
\mathrm{Var}[A_{ij}]_{q(\mathbf{A})} =& \frac{w_{ij}^{(\mathbf{A})}\big(1 - w_{ij}^{(\mathbf{A})}\big)}{(w_{i0}^{(\mathbf{A})})^2\big(1 + w_{i0}^{(\mathbf{A})}\big)}. & (23)
\end{aligned}
$$

The bead motion parameters have the following variational distributions

$$
\begin{aligned}
q(K_j, B_j) =& \frac{B_j^{n_j}}{W_j} e^{-B_j\left(v_j(K_j-\mu_j)^2 + c_j\right)}, & (24)\\
W_j =& \frac{c^{-(n_j+\frac{1}{2})}\Gamma(n_j + \frac{1}{2})}{\sqrt{v_j/\pi}}, & (25)
\end{aligned}
$$

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. The VBM equations are

$$
\begin{aligned}
n_j =& \tilde{n}_j + M_j, & (26)\\
c_j =& \tilde{c}_j + C_j + \tilde{v}_j\tilde{\mu}_j^2 - \frac{\big(\tilde{v}_j\tilde{\mu}_j + U_j\big)^2}{\tilde{v}_j + V_j}, & (27)\\
v_j =& \tilde{v}_j + V_j, & (28)\\
\mu_j =& \frac{\tilde{v}_j\tilde{\mu}_j + U_j}{\tilde{v}_j + V_j}, & (29)\\
& & (30)
\end{aligned}
$$

with

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

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

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

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

Some useful expectation values for future reference are

$$\langle \ln B_j \rangle_{q(\vec{B},\vec{K})} = \psi\big(n_j + \frac{1}{2}\big) - \ln c_j, \tag{35}$$

$$\langle B_j \rangle_{q(\vec{B},\vec{K})} = \frac{n_j + \frac{1}{2}}{c_j}, \tag{36}$$

$$\left\langle B_j K_j^2 \right\rangle_{q(\vec{B},\vec{K})} = \frac{1}{2v_j} + \mu_j^2 \frac{n_j + \frac{1}{2}}{c_j}, \tag{37}$$

$$\langle B_j K_j \rangle_{q(\vec{B},\vec{K})} = \mu_j \frac{n_j + \frac{1}{2}}{c_j}, \tag{38}$$

$$\mathrm{Var}[B_j]_{q(\vec{B},\vec{K})} = \frac{n_j + \frac{1}{2}}{c_j^2}, \tag{39}$$

$$\langle K_j \rangle_{q(\vec{B},\vec{K})} = \mu_j, \tag{40}$$

$$\mathrm{Var}[K_j]_{q(\vec{B},\vec{K})} = \frac{c_j}{2v_j(n_j - \frac{1}{2})}. \tag{41}$$

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 (12), 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 (13).

Specifically, the initial term is given by

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

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

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

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 $\langle\delta_{j,s_t}\delta_{k,s_{t+1}}\rangle_{q(s)}$ 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, 10, 11). 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

has an especially simple form just after the VBE step (3, 10, 11), 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,

$$F = \ln Z - \int d\vec{\pi} q(\vec{\pi}) \ln \frac{q(\vec{\pi})}{p(\vec{\pi})} - \sum_{j=1}^{N} \Bigg[\int d^N A_{j,:}\, q(A_{j,:}) \ln \frac{q(A_{j,:})}{p_0(A_{j,:})} + \int dB_j dK_j\, q(B_j, K_j) \ln \frac{q(B_j, K_j)}{p_0(B_j, K_j)} \Bigg]. \quad (47)$$

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

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

To get this simple form, we used that $w_0^{(\vec{\pi})} = 1 + \tilde{w}_0^{(\vec{\pi})}$ (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

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

Finally, the emission parameter of each state contributes

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

3.5 Two types of states

The above algorithm is readily extended to treat the model where genuine and spurious states are separated into two different hidden processes. We implemented a brute force approach to this problem, where we define new composite hidden states $\hat{s}_t = (s_t, c_t)$ and run the above algorithm on this composite model. 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 significant problem.

3.6 Choice of prior distributions

We would like to choose uninformative prior distributions in order to let the data speak for itself as much as possible. This is unproblematic for the emission parameters K, B, since the amount of data in all states is large enough to overwhelm any prior influence. We use

$$\tilde{\mu}_j = 0.6, \qquad \tilde{n}_j = 1, \quad (51)$$
$$\tilde{v}_j = 5.56 \text{ nm}^2, \qquad \tilde{c}_j = 30000 \text{ nm}^2, \quad (52)$$

which corresponds to

$$\langle K_j \rangle = 0.6, \qquad \langle B_j \rangle = 5 \times 10^{-5} \text{ nm}^{-2}, \quad (53)$$
$$\text{std}(K_j) = 0.3, \quad \text{std}(B_j) = 141.4 \times 10^{-5} \text{ nm}^{-2}. \quad (54)$$

The initial state prior is unproblematic for the opposite reason: the long length of the trajectories makes the initial state relatively unimportant to describe the data. We use a constant prior strength of 5,

$$\tilde{w}_j^{(\vec{\pi})} = 5/N, \quad (55)$$

where N is the number of hidden states.

The transition probabilities needs more care, because the potentially low number of transitions per trajectory makes the prior relatively more influential. Following Persson et al. (10), we parameterize this prior in terms of an expected mean dwell time and an overall number of pseudocounts (prior strength) for each hidden state. In particular, we define a transition *rate* matrix Q with mean dwell time t_D,

$$Q_{ij} = \frac{1}{t_D}\left(-\delta_{ij} + \frac{1 - \delta_{ij}}{N - 1}\right), \quad (56)$$

and then construct the prior based on the transition probability propagator per unit time step,

$$\tilde{w}_{ij}^{(\mathbf{A})} = \frac{t_A f_{sample}}{n_{downsample}} e^{\Delta t Q}. \quad (57)$$

Here, t_A is the prior strength; both t_A and t_D is specified in time units to be invariant under a change of sampling frequency. Further, the timestep is given by $\Delta t = n_{downsample}/f_{sample}$, where f_{sample} is the sampling frequency (30 Hz in our case), and $n_{downsample}$ is the downsampling factor (we use 3).

Numerical experiments by Persson et al. (10) show that choosing the strength too low compared to the mean dwell time produces a bias towards sparse transition matrices. This is not desireable in our case, and we therefore use $t_D = 1$ s, and $t_A = 5$ s throughout this work.

Prior for factorial model: TBA.(?)

3.7 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 $\tilde{u}_j$ ($u = w^{(\vec{\pi})}, u^{(\mathbf{A})}$), we need to solve

$$\frac{d}{d\tilde{u}_j}\sum_i \Bigg(\ln\frac{\Gamma(u_0^{(i)})}{\Gamma(\tilde{u}_0)} - (u_0^{(i)} - \tilde{u}_0)\psi(u_0^{(i)}) - \sum_{k=1}^{N}\left[\ln\frac{\Gamma(u_k^{(i)})}{\Gamma(\tilde{u}_k^{(i)})} - \big(u_k^{(i)} - \tilde{u}_k\big)\psi(u_k^{(i)})\right]\Bigg) = 0, \tag{58}$$

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

$$\psi(\tilde{u}_0) - \psi(\tilde{u}_j) = \frac{1}{M}\sum_i \Big(\psi(u_0^{(i)}) - \psi(u_j^{(i)})\Big). \tag{59}$$

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,

$$f_{KB} = \sum_i \Bigg(-\frac{n^{(i)} + \frac{1}{2}}{c^{(i)}}\Big(c^{(i)} - \tilde{c} - \tilde{v}(\mu^{(i)} - \tilde{\mu})^2\Big) + \frac{1}{2}\ln\frac{v^{(i)}}{\tilde{v}} + (\tilde{n} + \frac{1}{2})\ln\frac{c^{(i)}}{\tilde{c}} - \ln\frac{\Gamma\big(n^{(i)} + \frac{1}{2}\big)}{\Gamma\big(\tilde{n} + \frac{1}{2}\big)} + (n^{(i)} - \tilde{n})\psi\big(n^{(i)} + \frac{1}{2}\big) + \frac{1}{2}\Big(\frac{\tilde{v}}{v^{(i)}} - 1\Big)\Bigg). \tag{60}$$

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

$$\tilde{\mu} = \frac{1}{M}\sum_i \mu^{(i)}, \tag{61}$$

$$\frac{1}{\tilde{v}} = \frac{1}{M}\sum_i \Big(\frac{1}{v^{(i)}} + 2(\tilde{\mu} - \mu^{(i)})^2\Big). \tag{62}$$

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

$$\frac{\tilde{n} + \frac{1}{2}}{\tilde{c}} = \frac{1}{M}\sum_i \frac{n^{(i)} + \frac{1}{2}}{c^{(i)}}, \tag{63}$$

$$\ln\tilde{c} - \psi\big(\tilde{n} + \frac{1}{2}\big) = \frac{1}{M}\sum_i \Big(\ln c^{(i)} - \psi\big(n^{(i)} + \frac{1}{2}\big)\Big), \tag{64}$$

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 $\tilde{n} = \alpha\tilde{c} - \frac{1}{2}$.

3.8 Factorial model

3.9 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

Table 1: Fields in a model object W.

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

that contain results of the model search for each trajectory, indexed as the filenames in the runinput file, e.g. `trj{k}{j}` contains the analysis result for `looping_filename{k}{j}` etc.

Most importantly, teh 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 nodel 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 1-4.

Table 2: Representration of variational distributions for genuine states in a model object named W.

field	symbol	Eq.
`W.M.wPi(j)`	$w_j^{(\vec{\pi})}$	(14)
`W.PM.wPi(j)`	$\tilde{w}_j^{(\vec{\pi})}$	
`W.E.ds_1(j)`	$\langle \delta_{j,s_1} \rangle$	
`W.PM.wA(i,j)`	$\tilde{w}_{ij}^{(\mathbf{A})}$	(16)
`W.M.wA(i,j)`	$w_{ij}^{(\mathbf{A})}$	
`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 3: Representration 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.

field	symbol	Eq.
`W.Mc.wPi(j)`	$w_j^{(\vec{\pi})}$	
`W.PMc.wPi(j)`	$\tilde{w}_j^{(\vec{\pi})}$	(??)
`W.Ec.ds_1(j)`	$\langle \delta_{j,c_1} \rangle$	
`W.PMc.wA(j)`		
`W.Mc.wA(j)`		(??)
`W.Ec.wA(j)`		
`W.PMc.wR(i,j)`		
`W.Mc.wR(i,j)`		(??)
`W.Ec.wR(i,j)`		

Table 4: 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 devition of stransition 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)`	$RMS_j = \sqrt{\langle \vec{x}_t^2 \vert s_t = j \rangle} \approx (\langle B_j \rangle (1 - \langle K_j \rangle^2))^{-\frac{1}{2}}$.
`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)`
`W.est2.cMaxP(t)`
`W.est2.sViterbi`
`W.est2.cViterbi`
	Most likely genuine state at time t.
	Most likely indicator state at time t.
	Viterbi path (most likely sequence of states) for s_t.
	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.

3.10 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`, `VB7_EBupdate_KB`, and `VB7_EBupdate_KB2`.

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

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] L.E. Baum, T. Petrie, G. Soules, , and N. Weiss. A maximization technique occurring in the statistical analysis of prob-

abilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41:164–171, 1970.

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