Supplementary Material: A general recurrent state space framework for modeling neural dynamics during decision-making

David M. Zoltowski[1], Jonathan W. Pillow[1,2], and Scott W. Linderman[3,4]

[1]Princeton Neuroscience Institute, Princeton University, Princeton, NJ
[2]Department of Psychology, Princeton University, Princeton, NJ
[3]Department of Statistics, Stanford University, Palo Alto, CA
[4]Wu Tsai Neurosciences Institute, Stanford University, Palo Alto, CA

A Additional Simulated Experiments

Here we demonstrate fitting models with collapsing boundaries and trial-history effects to simulated data. We also compare the variational Laplace EM inference algorithm with black box variational inference and particle EM approaches.

A.1 Nonlinear collapsing boundaries

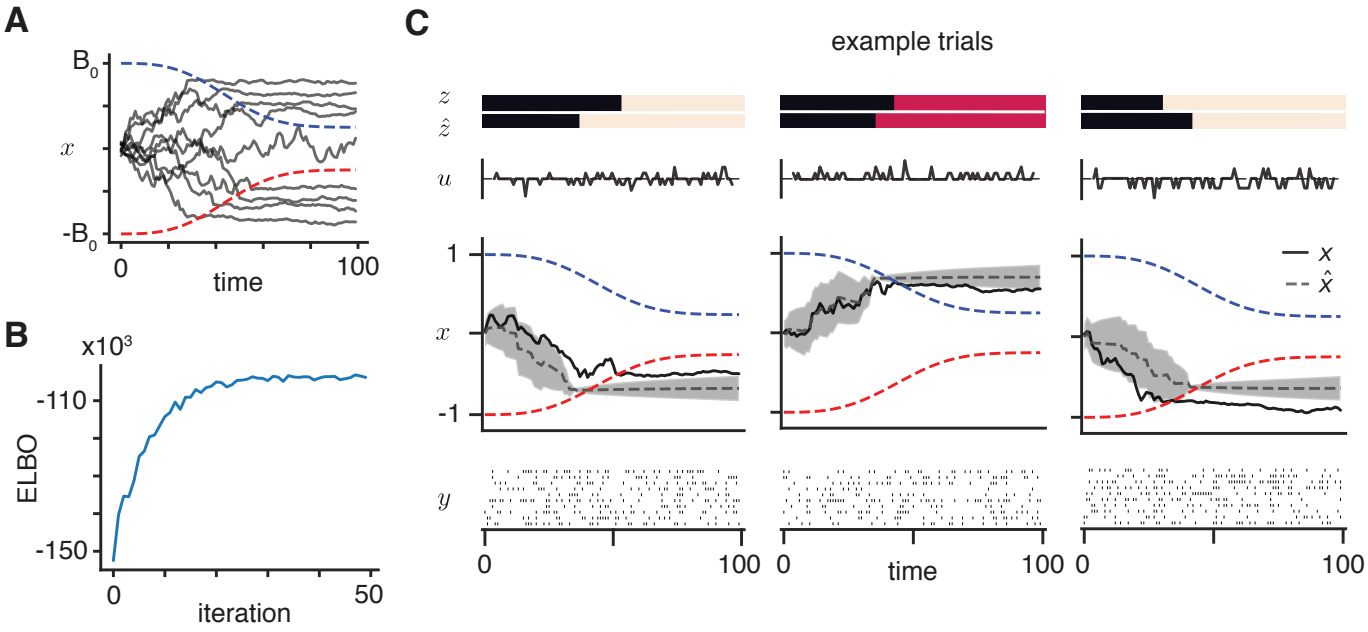

Figure A1. Simulated experiment with nonlinear collapsing boundaries. **A.** Simulated latent trajectories from a one-dimensional accumulation model with collapsing upper and lower boundaries. **B.** ELBO as a function of iteration. **C.** The simulated data and the true and inferred discrete and continuous states for three example trials.

We simulated spike counts from 10 Poisson neurons from a one-dimensional accumulation model with nonlinear collapsing boundaries (Figure A1A). The bin size was $\Delta_t = 0.01$, the trial length was $T = 100$, and the number of trials was 200. The inputs were the difference of two dimensional pulses. The accumulation state parameters were $V_{\text{acc}} = 0.005$ and $\sigma^2_{\text{acc}} = 0.002$.

The boundary parameterization was

$$b_t = b_0 - (1 - e^{-(t/\lambda)^k})(b_0 - b_\infty) \tag{1}$$

where $b_0 = 1.0$ is the initial height of the boundary and $b_\infty = 0.25$ is the final height of the boundary. The upper and lower boundaries were symmetric across zero. We implemented this model by modifying the transition probabilities to depend on the time-varying boundary.

We fit the nonlinear collapsing boundaries model to the simulated data using 50 iterations of the vLEM algorithm (Figure A1B). The inferred continuous and discrete states from the algorithm were similar to the true latent states (Figure A1C).

A.2 Linear collapsing boundaries

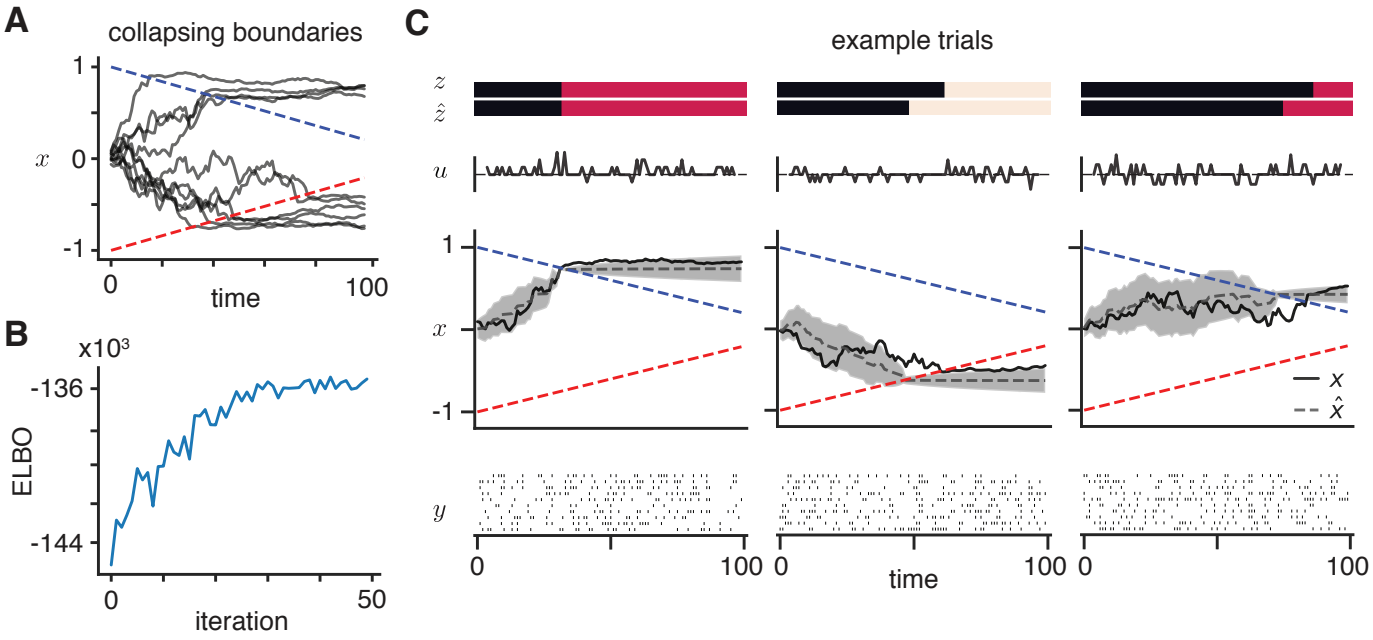

Figure A2. Simulated experiment with linear collapsing boundaries. **A.** Simulated latent trajectories **B.** ELBO as a function of iteration. **C.** The simulated data and the true and inferred discrete and continuous states for three example trials.

We replicated the nonlinear collapsing boundary simulation with a linear collapsing bound, with 10 Poisson neurons, $\Delta_t = 0.01$, $T = 100$, and 200 trials (Figure A2A-C). The inputs were the difference of two dimensional pulses. Here, the accumulation state parameters were $V_{\text{acc}} = 0.01$ and $\sigma^2_{\text{acc}} = 0.001$.

The boundaries started at ± 1 and collapsed towards zero at a rate of 0.008 per time bin, which means that the final boundary values at $T = 100$ were ± 0.2. We implemented this model with the following steps. First, we augmented the input vector with the current time of the trial such that $u_t = [s_t, t]^\top$ where s_t is the current stimulus input. Importantly, we set the second dimension of the input weight parameter $V^{(2)}_{\text{acc}}$ to zero so the time is not input to the continuous

dynamics x. We modified the transitions to depend on the input with the following parameterization

$$p(z_t \mid z_{t-1}, x_{t-1}) \propto \exp\left\{\gamma(R_{z_{t-1}} + r\, x_{t-1} + W u_t)\right\}, \quad W = \begin{bmatrix} 0 & 0 \\ 0 & \beta \\ 0 & \beta \end{bmatrix} \tag{2}$$

where β is a scalar parameter that controls the slope of the boundary. We set the left column of W to zeros so the sensory input does not directly affect the transitions. We set $\beta = 0.008$, which corresponds to the rate of the collapsing boundaries as described above. We note that asymmetric collapsing boundaries can be implemented by having separate β parameters for each dimension. While we fix the slope parameter β, its value could be learned. The parameters $R_{z_{t-1}}$ and r have the same form as in the original 1D accumulator model.

A.3 Trial-history

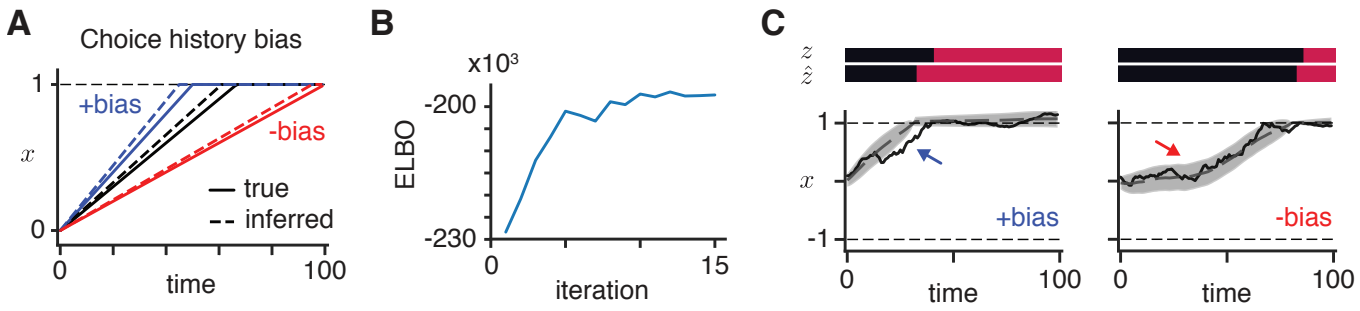

Figure A3. Simulated experiment with trial-history effects. **A.** True and inferred averaged drift rates for positive going trials with and without the choice history bias. The previous choice biases the drift rate upwards (previous choice corresponds to upper boundary) and downwards (previous choice corresponds to lower boundary). **B.** The ELBO as a function of optimization iteration. **C.** True and inferred states for positive going trials with positive (*left*) and negative (*right*) biases.

The modeling framework allows for trial-history effects based on the previous choice, reward, or stimulus. Here we simulated data from a model where the previous trial choice affects the drift rate (Figure A3). We implemented this by including the previous trial choice $c_{\text{prev}} = \{-1, 1\}$ as an additional input covariate. The input at each time point on a given trial was $u_t = [s_t, c_{\text{prev}}]$. In this case, we learn each dimension of the input weights $V_{\text{acc}} \in \mathbb{R}^2$. The element in the second dimension corresponds to the bias in the drift rate. This parameterization enforces a symmetric drift bias, but it is again possible to relax the symmetry.

We simulated spike counts of 5 Poisson neurons from this model with a bin size $\Delta = 0.1$. Each trial had length $T = 100$ and we simulated $N = 200$ trials. In this simulation, the input on each trial was a constant drift of $s_t = 0.015$ for positive going trials and $s_t = -0.015$ for negative going trials. The drift bias was 0.005 and the variance was $\sigma^2_{\text{acc}} = 0.001$. The average drift rate on positive going trials is shown in Figure A3. The bias increased the average drift when the previous choice was $+1$ (blue line) and decreased the average drift when the previous choice was -1.

We fit this model using 15 iterations of the vLEM algorithm (Figure A3B). The inferred drift rates in the fit model were similar to the true drift rates (Figure A3A). Next, the inferred latent states correctly followed the bias shown in the true latent states (Figure A3C).

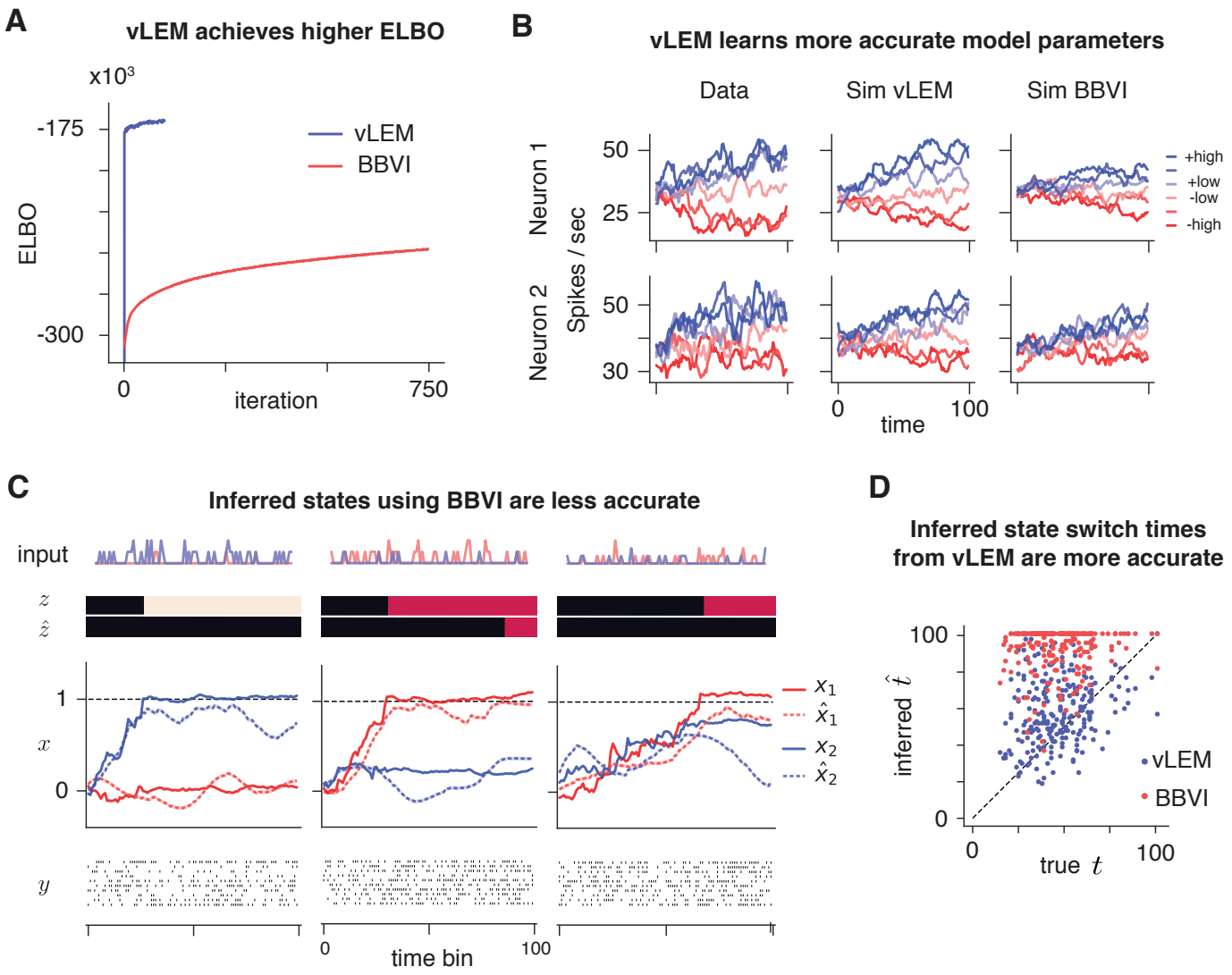

Figure A4. Comparison of vLEM and BBVI for fitting a 2D accumulator model with Poisson neurons. **A**. The ELBO as a function of algorithm iteration (these are the same values as presented in Figure 3). **B.** The true average neural responses across different evidence strengths (line colors) and simulated responses from the fit model using vLEM or BBVI. Here, "+high" is strong stimulus motion towards the"+" direction while "-high" is strong stimulus motion to the opposite "-" direction. **C.** The inferred (dashed lines, $\hat{z}$, and $\hat{x}$) and true (solid lines, z, and x) continuous and discrete states using BBVI for three typical example trials. **D.** The true t and inferred $\hat{t}$ transition times from the accumulation state to the boundary state for vLEM and BBVI for all trials.

A.4 Comparison of vLEM and BBVI

Here we present additional results of the comparison between vLEM and BBVI from the simulated 2D accumulator experiment in Section 5.1. As stated previously, we simulated a 2D race accumulator model and the model using vLEM and BBVI. For BBVI, we used a jointly Gaussian posterior over the continuous latent variables with block-tridiagonal structure in the precision of the covariance matrix and we marginalized the discrete states (Archer et al., 2015; Gao et al., 2016; Linderman et al., 2019). We initialized the models with the same parameters and with the same posterior over the continuous latent variables.

Results of fitting the model with BBVI are shown in Figure A4. First, we found that vLEM achieved substantially higher ELBO values (Figure A4A). Next, the learned model parameters from vLEM generated data that are more similar to the true simulated data than BBVI (Figure A4B). This is shown by the similarity in the PSTHs in the first two columns.

Crucially, vLEM provided more accurate inferences about the latent states (Figure A4C). BBVI had difficulty learning transitions from accumulation to boundary and had qualitatively poorer uncertainty estimates (Figure A4C-D). On many trials BBVI did not infer a switch from accumulation to boundary.

A.5 Comparison of vLEM and particle EM

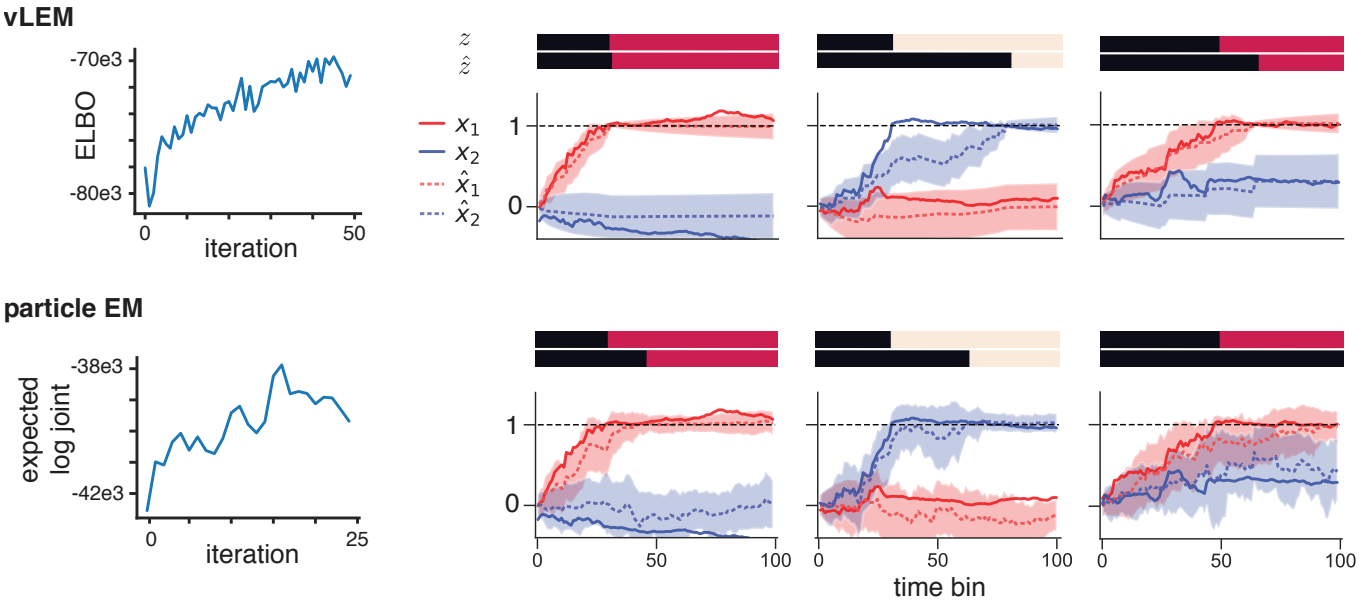

Figure A5. (*top*) ELBO as a function of iteration (*left*) and true and inferred latents on example trials (*right*) when using vLEM to fit the model. (*bottom*) Same as above, except for using particle EM to fit the model and with the expected log joint probability instead of the ELBO.

To test the accuracy of vLEM, we compared vLEM with a particle EM algorithm that used a Rao-Blackwellized particle filter to sample from the marginal posterior over the continuous latent states (see Appendix C). We simulated 100 trials from a 2D accumulator model with 10 neurons and used both vLEM and particle EM to fit the simulated data. (Figure A5). We used $S = 50$ particles in particle EM. This balanced variety in the particles with computational cost, as running the particle EM for 25 iterations took about four hours with our implementation (as opposed to ≈ 10 minutes for 50 iterations of vLEM). The mean squared error between the true and inferred (posterior mean) latent continuous trajectories was smaller for vLEM (0.047 ± 0.0008 for vLEM, 0.064 ± 0.0010 for particle EM). The relatively strong performance of vLEM in this limited data comparison is encouraging. We note that we could improve the particle EM method by optimizing the speed of our implementation, which would allow us to increase the number of particles without incurring a large computational cost. Nonetheless, we consider our current implementation to be a reasonable baseline and are encouraged that vLEM achieves comparable accuracy with lower computational cost.

B Variational Laplace-EM Inference

Here we describe in more detail the variational Laplace-EM inference method. As noted in the main text, we introduce a factorized approximate posterior $q(z)q(x) \approx p(z, x \mid y, \theta)$ over the discrete and continuous latent variables. With those distributions we lower-bound the marginal likelihood with

$$\begin{aligned}\mathcal{L}_q(\theta) &= \mathbb{E}_{q(z)q(x)}[\log p(x, z, y \mid \theta) - \log q(z)q(x)] \\ &= \mathbb{E}_{q(z)q(x)}[\log p(x, z, y \mid \theta)] - \mathbb{E}_{q(z)}[\log q(z)] - \mathbb{E}_{q(x)}[\log q(x)].\end{aligned}$$

To optimize this objective, we alternate between updating 1) $q(z)$, 2) $q(x)$ and 3) θ. The updates to $q(z)$ and θ follow from optimizing the lower bound $\mathcal{L}_q(\theta)$. The update to $q(x)$ is an approximate update and is therefore not guaranteed to increase the value of the lower bound.

B.1 Update discrete state posterior

We update $q(z)$ via the optimal coordinate ascent variational inference update

$$q^\star(z) \propto \exp(\mathbb{E}_{q(x)}[\log p(x, z, y \mid \theta)]). \tag{3}$$

To compute this, we expand the expected log joint probability

$$\begin{aligned}\mathbb{E}_{q(x)}[\log p(x, z, y \mid \theta)] = \mathbb{E}_{q(x)}\Bigg[&\log p(z_1, x_1 \mid \theta) + \sum_{t=2}^{T} \log p(x_t \mid x_{t-1}, z_t, \theta) \\ &+ \sum_{t=1}^{T-1} \log p(z_{t+1} \mid z_t, x_t, \theta) + \sum_{t=1}^{T} \log p(y_t \mid x_t, z_t, \theta)\Bigg] \\ = \phi(z_1, x_1) &+ \sum_{t=2}^{T} \phi(z_t, x_t, x_{t-1}) + \sum_{t=1}^{T-1} \phi(z_t, z_{t+1}, x_t) + \sum_{t=1}^{T} \phi(z_t, x_t, y_t)\end{aligned}$$

where we have introduced the potentials

$$\begin{aligned}\phi(z_1, x_1) &= \mathbb{E}_{q(x)}[\log p(z_1, x_1 \mid \theta)] \\ \phi(z_t, x_t, x_{t-1}) &= \mathbb{E}_{q(x)}[\log p(x_t \mid x_{t-1}, z_t, \theta)] \\ \phi(z_t, z_{t+1}, x_t) &= \mathbb{E}_{q(x)}[\log p(z_{t+1} \mid z_t, x_t, \theta)] \\ \phi(z_t, x_t, y_t) &= \mathbb{E}_{q(x)}[\log p(y_t \mid x_t, z_t, \theta)].\end{aligned}$$

We used samples from $q(x)$ to estimate the expectations in these potentials. We used a default of a single sample in our simulations and applications to data. We note that if the observations are independent of the discrete states when conditioned on the continuous states (i.e. $\log p(y_t \mid x_t, z_t, \theta) = \log p(y_t \mid x_t, \theta)$) then the emission potential $\phi(z_t, x_t, y_t)$ can be disregarded for updating $q(z)$.

We introduce the normalizing constant $Z(\theta)$ of the distribution such that

$$q(z) = \frac{1}{Z(\theta)} \exp\left(\phi(z_1, x_1) + \sum_{t=2}^{T} \phi(z_t, x_t, x_{t-1}) + \sum_{t=1}^{T-1} \phi(z_t, z_{t+1}, x_t) + \sum_{t=1}^{T} \phi(z_t, x_t, y_t)\right). \tag{4}$$

Conditioned on the estimates of the potentials, we have a factor graph equivalent to the factor graph of an HMM. Therefore we compute the unary and pairwise marginals over z and the normalizing constant using the forward-backwards

algorithm. We evaluate the entropy term in the ELBO using the potentials, the unary and pairwise marginals, and the normalizing constant as

$$\begin{aligned}\mathbb{E}_{q(z)}[\log q(z)] &= \mathbb{E}_{q(z)}\Big[-\log Z(\theta) + \phi(z_1, x_1) + \sum_{t=2}^{T} \phi(z_t, x_t, x_{t-1}) + \sum_{t=1}^{T-1} \phi(z_t, z_{t+1}, x_t) + \sum_{t=1}^{T} \phi(z_t, x_t, y_t)\Big] \\ &= -\log Z(\theta) + \mathbb{E}_{q(z)}[\phi(z_1, x_1)] + \sum_{t=2}^{T} \mathbb{E}_{q(z)}[\phi(z_t, x_t, x_{t-1})] + \sum_{t=1}^{T-1} \mathbb{E}_{q(z)}[\phi(z_t, z_{t+1}, x_t)] \\ &\quad + \sum_{t=1}^{T} \mathbb{E}_{q(z)}[\phi(z_t, x_t, y_t)].\end{aligned}$$

B.2 Update continuous state posterior

We update $q(x)$ with a Laplace approximation around the mode of $\mathbb{E}_{q(z)}[\log p(x, z, y \mid \theta)]$ such that

$$\begin{aligned} q^\star(x) &= \mathcal{N}(x^\star, -H^{-1}) \\ x^\star &= \arg\max_x \mathbb{E}_{q(z)}[\log p(x, z, y \mid \theta)] \\ H &= \nabla_x^2 \mathbb{E}_{q(z)}[\log p(x, z, y \mid \theta)]\Big|_{x=x^\star}. \end{aligned}$$

To compute the Hessian we expand the terms in the objective

$$\begin{aligned}\mathcal{L}(x) &= \mathbb{E}_{q(z)}[\log p(x, z, y|\theta)] \\ &= \mathbb{E}_{q(z)}\Big[\log p(z_1 \mid \theta) + \log p(x_1 \mid z_1, \theta) + \sum_{t=2}^{T} \log p(x_t \mid x_{t-1}, z_t, \theta) \\ &\qquad + \sum_{t=1}^{T-1} \log p(z_{t+1} \mid z_t, x_t, \theta) + \sum_{t=1}^{T} \log p(y_t \mid x_t, z_t, \theta)\Big] \\ &= \phi(x_1, z_1) + \sum_{t=2}^{T} \phi(x_t, x_{t-1}, z_t) + \sum_{t=1}^{T-1} \phi(x_t, z_t, z_{t+1}) + \sum_{t=1}^{T} \phi(x_t, y_t, z_t) + \text{const}\end{aligned}$$

where

$$\begin{aligned}\phi(x_1, z_1) &= \mathbb{E}_{q(z)}[\log p(x_1 \mid z_1, \theta)] = \sum_k q(z_1 = k) \log p(x_1 \mid z_1 = k, \theta) \\ \phi(x_t, x_{t-1}, z_t) &= \mathbb{E}_{q(z)}[\log p(x_t \mid x_{t-1}, z_t, \theta)] = \sum_k q(z_t = k) \log p(x_t \mid x_{t-1}, z_t = k, \theta) \\ \phi(x_t, z_t, z_{t+1}) &= \mathbb{E}_{q(z)}[\log p(z_{t+1} \mid z_t, x_t, \theta)] = \sum_k \sum_j q(z_t = k, z_{t+1} = j) \log p(z_{t+1} = j \mid z_t = k, x_t, \theta) \\ \phi(x_t, y_t, z_t) &= \mathbb{E}_{q(z)}[\log p(y_t \mid x_t, z_t, \theta)] = \sum_k q(z_t = k) \log p(y_t \mid x_t, z_t = k, \theta).\end{aligned}$$

The above derivation was written in full generality. If the emission potential does not depend on the discrete state then the emission potential simplifies to $\phi(x_t, y_t, z_t) = \log p(y_t \mid x_t, \theta)$. Also, if there are no recurrent dependencies (as in a standard SLDS) then the transition term $\log p(z_{t+1} \mid z_t, x_t, \theta)$ is equal to $\log p(z_{t+1} \mid z_t, \theta)$ and therefore the transition potential $\phi(x_t, z_t, z_{t+1})$ no longer depends on x_t.

We require the Hessian matrix for the Laplace approximation. This matrix is given by

$$\begin{aligned}\nabla_x^2 \mathcal{L}(x) &= \nabla_x^2 \mathbb{E}_{q(z)}[\log p(x, z, y \mid \theta)] \\ &= \nabla_x^2 \phi(x_1, z_1) + \sum_{t=2}^{T} \nabla_x^2 \phi(x_t, x_{t-1}, z_t) + \sum_{t=1}^{T-1} \nabla_x^2 \phi(x_t, z_t, z_{t+1}) + \sum_{t=1}^{T} \nabla_x^2 \phi(x_t, y_t, z_t)\end{aligned}$$

where

$$\begin{aligned}\nabla_x^2 \phi(x_1, z_1) &= \sum_k q(z_1 = k) \nabla_x^2 \log p(x_1 \mid z_1 = k, \theta) \\ \nabla_x^2 \phi(x_t, x_{t-1}, z_t) &= \sum_k q(z_t = k) \nabla_x^2 \log p(x_t \mid x_{t-1}, z_t = k, \theta) \\ \nabla_x^2 \phi(x_t, z_t, z_{t+1}) &= \sum_k \sum_j q(z_t = k, z_{t+1} = j) \nabla_x^2 \log p(z_{t+1} = j \mid z_t = k, x_t, \theta) \\ \nabla_x^2 \phi(x_t, y_t, z_t) &= \sum_k q(z_t = k) \nabla_x^2 \log p(y_t \mid x_t, z_t = k, \theta).\end{aligned}$$

Therefore, we can compute the Hessian by computing the contributions to the Hessian of the dynamics, emission, and transition potentials.

The Hessian has size $TD \times TD$ for a time series of length T with latent dimensionality D but has a sparse, block tridiagonal structure with blocks of size $D \times D$. The terms in the Hessian from the initial state, transition, and emission potentials only contribute terms to the primary block diagonal. The dynamics potentials contribute terms to both the primary and first off-diagonal blocks. Throughout, we only represent and store the main and lower diagonal blocks of the Hessian. This reduces storage from the full $(TD)^2$ to $(2T-1)D^2$ such that it is linear in T. For linear solves and matrix inversions of the Hessian, we also use algorithms that exploit the block tridiagonal structure.

To find the most likely latent path $x^\star$, we use Newton's method with a backtracking line search. However, we can also use optimization routines that require only gradient information (lBFGS) or require only gradient information and Hessian-vector products (Newton-CG or trust-region Newton-CG).

B.3 Update parameters

We update the model parameters by approximately optimizing the ELBO with respect to the parameters

$$\theta^\star = \arg\max_\theta \mathbb{E}_{q(z)q(x)}[\log p(x, z, y \mid \theta) - \log q(z)q(x)]. \tag{5}$$

Instead of optimizing the expectation under the full distribution of $q(x)$ we optimize

$$\theta^\star = \arg\max_\theta \mathbb{E}_{q(z)}[\log p(\hat{x}, z, y \mid \theta)] \tag{6}$$

where $\hat{x}$ is a sample from $q(x)$ and we have dropped terms that do not depend on θ. Conditioned on $\hat{x}$, the update consists of M-steps on the transition, dynamics, and emission parameters. We use either exact updates (where applicable) or lFBGS to implement the M-steps. Finally, we set the parameters at iteration i via a convex combination of the new parameters $\theta^\star$ and the parameters at the previous iteration

$$\theta_i = (1 - \alpha)\, \theta^\star + \alpha\, \theta_{i-1}. \tag{7}$$

We note that we can also update the parameters using stochastic gradient ascent with samples from $q(x)$.

B.4 Initialization

We can exploit the known structure of the 1D and 2D accumulation-to-bound models to initialize some of the parameters. For the 1D and 2D accumulation-to-bound models we set the emission parameter d to the mean spike counts across trials in the first three time bins. In the 1D model, we set the emission parameter C using the firing rate at the end of trials with strong input to the upper (λ_{UB}) and lower (λ_{LB}) boundaries. Given those values for each neuron and the fact that the boundaries are at ± 1 for this model, we set $C = \frac{1}{2}(\lambda_{\mathsf{UB}} - \lambda_{\mathsf{LB}})$. In the 2D model, for each neuron we initialized the elements of C as the difference between the firing rate at the end of trials with strong net input and the mean rate d. We did this for each dimension of the input and corresponding element in C. For the models and data in this paper, we did not identify procedures to reliably estimate the initial underlying latent dynamics parameters. Therefore we randomly initialized the input weights and dynamics variance and set the initial dynamics matrix to $A_{\mathsf{acc}} = I$.

C Particle EM

We compared vLEM with a particle EM inference method. The first step of this method is to use a Rao-Blackwellized particle filter to obtain S Monte Carlo samples from the marginal posterior over the latent continuous states

$$x^s_{1:T} \sim p(x_{1:T} \mid y_{1:T}, \theta) \tag{8}$$

for $s = 1, ..., S$ and for each time series. The second step is to use the samples from the posterior to estimate the expected log joint probability

$$\mathbb{E}_{p(x,z|y,\theta)}[\log p(x, z, y \mid \theta)] \approx \frac{1}{S}\sum_{s=1}^{S} \mathbb{E}_{p(z|x^s,y,\theta)}[\log p(x^s, z, y \mid \theta)], \quad x^s \sim p(x_{1:T} \mid y_{1:T}, \theta) \tag{9}$$

$$= \hat{\mathbb{E}}_{p(x,z|y,\theta)}[\log p(x, z, y \mid \theta)] \tag{10}$$

where we have dropped the subscripts denoting the entire time series. Finally, we update the parameters by maximizing the sample expectation of the log joint

$$\theta^\star = \arg\max_\theta \hat{\mathbb{E}}_{p(x,z|y,\theta)}[\log p(x, z, y \mid \theta)]. \tag{11}$$

We used the L-BFGS optimizer to maximize this objective.

C.1 Particle filter

Here we describe the Rao-Blackwellized particle filter used in the particle EM algorithm. The posterior over the time series x is

$$p(x_{1:t} \mid y_{1:t}) \propto p(x_{1:t}, y_{1:t}) \tag{12}$$

$$= p(y_t \mid x_{1:t}, y_{1:t-1})\, p(x_t \mid x_{1:t-1}, y_{1:t-1})\, p(x_{1:t-1}|y_{1:t-1}) \tag{13}$$

$$= p(y_t \mid x_t)\, p(x_t \mid x_{1:t-1})\, p(x_{1:t-1} \mid y_{1:t-1}). \tag{14}$$

Let $q(x_{1:t} \mid y_{1:t})$ be the importance sampling density used in the particle filter. If we propose particles according to the prior distribution, $q(x_t \mid x_{1:t-1}, y_{1:t-1}) = p(x_t \mid x_{1:t-1})$, then the particle filter weights are

$$w^s_t \propto \frac{p(x^s_{1:t} \mid y_{1:t})}{q(x^s_{1:t} \mid y_{1:t})} \tag{15}$$

$$= \frac{p(y_t \mid x_t^s)\, p(x_t^s \mid x_{1:t-1}^s)\, p(x_{1:t-1}^s \mid y_{1:t-1})}{q(x_t^s \mid x_{1:t-1}^s, y_{1:t-1})\, q(x_{1:t-1}^s \mid y_{1:t-1})} \tag{16}$$

$$= p(y_t \mid x_t^s)\, w_{t-1}^s. \tag{17}$$

The proposal can be written via a marginalization over z_t

$$p(x_t \mid x_{1:t-1}) = \sum_{z_t} p(x_t \mid z_t, x_{t-1})\, p(z_t \mid x_{1:t-1}). \tag{18}$$

Therefore we can sample from $p(x_t \mid x_{1:t-1})$ with

$$z_t \sim p(z_t \mid x_{1:t-1}) \tag{19}$$

$$x_t \sim p(x_t \mid z_t, x_{t-1}) \tag{20}$$

and ignoring z_t. This procedure requires the 'look-ahead' posterior $p(z_t \mid x_{1:t-1})$ which can be computed using the filtered posterior $p(z_t \mid x_{1:t-1})$ and transition probabilities

$$p(z_t \mid x_{1:t-1}) = \sum_{z_{t-1}} p(z_t \mid z_{t-1}, x_{t-1})\, p(z_{t-1} \mid x_{1:t-1}). \tag{21}$$

We compute and store the filtered posterior $p(z_t \mid x_{1:t})$ for each time point t.

The final Rao-Blackwellized particle filter is given in Algorithm 1.

Algorithm 1 Rao-Blackwellized particle filter for rSLDS

Input: observations y, inputs u, and number of particles S
Initialize particles x_1^s, weights w_1^s, and initial $p(z_1 \mid x_1^s)$ for $s = \{1, ..., S\}$.
for $t = 2$ to T **do**
  for $s = 1$ to S **do**
    Compute look-ahead posterior using transition matrix

$$p(z_t \mid x_{1:t-1}^s) = \sum_{z_{t-1}} p(z_t \mid z_{t-1}, x_{t-1}^s)\, p(z_{t-1} \mid x_{1:t-1}^s)$$

    Sample from look-ahead posterior and dynamics model

$$z_t \sim p(z_t \mid x_{1:t-1}^s)$$
$$x_t^s \sim p(x_t \mid z_t, x_{t-1}^s)$$

    Compute posterior

$$p(z_t \mid x_{1:t}^s) \propto p(x_t^s \mid z_t) \sum_{z_{t-1}} p(z_t \mid z_{t-1}, x_{t-1}^s)\, p(z_{t-1} \mid x_{1:t-1}^s)$$

    Compute likelihood $p(y_t \mid x_t^s)$
    Multiply weight $w_t^s = w_{t-1}^s\, p(y_t \mid x_t^s)$
  end for
  Normalize weights $w_t^s = w_t^s / \sum_t w_t^s$
  Resample particle indices $(I_1, ..., I_S) \sim \mathrm{Mu}(S, (w_t^1, ..., w_t^S))$
  For each index assign $x_{1:t}^s = x_{1:t}^{I_s}$ and $p(z_t \mid x_{1:t}^s) = p(z_t \mid x_{1:T}^{I_s})$
end for
Return $x_{1:T}^s$, $w_{1:t}^s$ for all s

References

Archer, E., Park, I. M., Buesing, L., Cunningham, J., and Paninski, L. (2015). Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*.

Gao, Y., Archer, E. W., Paninski, L., and Cunningham, J. P. (2016). Linear dynamical neural population models through nonlinear embeddings. In *Advances in Neural Information Processing Systems*, pages 163–171.

Linderman, S., Nichols, A., Blei, D., Zimmer, M., and Paninski, L. (2019). Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in c. elegans. *bioRxiv*.