Modeling Latent Neural Dynamics with Gaussian Process Switching Linear Dynamical Systems

Amber Hu
Stanford University
amberhu@stanford.edu

David Zoltowski
Stanford University
dzoltow@stanford.edu

Aditya Nair
Caltech & Howard Hughes Medical Institute
adi.nair@caltech.edu

David Anderson
Caltech & Howard Hughes Medical Institute
wuwei@caltech.edu

Lea Duncker[*]
Columbia University
ld3149@columbia.edu

Scott Linderman[*]
Stanford University
swl1@stanford.edu

Abstract

Understanding how the collective activity of neural populations relates to computation and ultimately behavior is a key goal in neuroscience. To this end, statistical methods which describe high-dimensional neural time series in terms of low-dimensional latent dynamics have played a fundamental role in characterizing neural systems. Yet, what constitutes a successful method involves two opposing criteria: (1) methods should be expressive enough to capture complex nonlinear dynamics, and (2) they should maintain a notion of interpretability often only warranted by simpler linear models. In this paper, we develop an approach that balances these two objectives: the *Gaussian Process Switching Linear Dynamical System* (gpSLDS). Our method builds on previous work modeling the latent state evolution via a stochastic differential equation whose nonlinear dynamics are described by a Gaussian process (GP-SDEs). We propose a novel kernel function which enforces smoothly interpolated locally linear dynamics, and therefore expresses flexible – yet interpretable – dynamics akin to those of recurrent switching linear dynamical systems (rSLDS). Our approach resolves key limitations of the rSLDS such as artifactual oscillations in dynamics near discrete state boundaries, while also providing posterior uncertainty estimates of the dynamics. To fit our models, we leverage a modified learning objective which improves the estimation accuracy of kernel hyperparameters compared to previous GP-SDE fitting approaches. We apply our method to synthetic data and data recorded in two neuroscience experiments and demonstrate favorable performance in comparison to the rSLDS.

1 Introduction

Computations in the brain are thought to be implemented through the dynamical evolution of neural activity. Such computations are typically studied in a controlled experimental setup, where an animal is engaged in a behavioral task with relatively few relevant variables. Consistent with this, empirical neural activity has been reported to exhibit many fewer degrees of freedom than there are neurons in the measured sample during such simple tasks [1]. These observations have driven the use of latent

[*]Corresponding authors.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

variable models to characterize low-dimensional structure in high-dimensional neural population activity [2, 3]. In this setting, neural activity is often modeled in terms of a low-dimensional latent state that evolves with Markovian dynamics [4–14]. It is thought that the latent state evolution is related to the computation of the system, and therefore, insights into how this evolution is shaped through a dynamical system can help us understand the mechanisms underlying computation [15–21].

In practice, choosing an appropriate modeling approach for a given task requires balancing two key criteria. First, statistical models should be expressive enough to capture potentially complex and nonlinear dynamics required to carry out a particular computation. On the other hand, these models should also be interpretable and allow for straightforward post-hoc analyses of dynamics. One model class that strikes this balance is the recurrent switching linear dynamical system (rSLDS) [8]. The rSLDS approximates arbitrary nonlinear dynamics by switching between a finite number of linear dynamical systems. This leads to a powerful and expressive model which maintains the interpretability of linear systems. Because of their flexibility and interpretability, variants of the rSLDS have been used in many neuroscience applications [19, 22–28] and are examples of a general set of models aiming to understand nonlinear dynamics using compact and interpretable components [29, 30].

However, rSLDS models suffer from several limitations. First, while the rSLDS is a probabilistic model, typical use cases do not capture posterior uncertainty over inferred dynamics. This makes it difficult to judge the extent to which particular features of a fitted model should be relied upon when making inferences about their role in neural computation. Second, the rSLDS often suffers from producing oscillatory dynamics in regions of high uncertainty in the latent space, such as boundaries between linear dynamical regimes. This artifactual behavior can significantly impact the interpretability and predictive performance of the rSLDS. Lastly, the rSLDS does not impose smoothness or continuity assumptions on the dynamics due to its discrete switching formulation. Such assumptions are often natural and useful in the context of modeling realistic neural systems.

In this paper, we improve upon the rSLDS by introducing the *Gaussian Process Switching Linear Dynamical System* (gpSLDS). Our method extends prior work on the Gaussian process stochastic differential equation (GP-SDE) model, a continuous-time method that places a Gaussian process (GP) prior on latent dynamics. By developing a novel GP kernel function, we enforce locally linear, interpretable structure in dynamics akin to that of the rSLDS. Our framework addresses the aforementioned modeling limitations of the rSLDS and contributes a new class of priors in the GP-SDE model class. Our paper is organized as follows. Section 2 provides background on GP-SDE and rSLDS models. Section 3 presents our new gpSLDS model and an inference and learning algorithm for fitting these models. In Section 4 we apply the gpSLDS to a synthetic dataset and two datasets from real neuroscience experiments to demonstrate its practical use and competitive performance. We review related work in Section 5 and conclude our paper with a discussion in Section 6.

2 Background

2.1 Gaussian process stochastic differential equation models

Gaussian processes (GPs) define nonparametric distributions over functions. They are a popular choice in machine learning due to their ability to capture nonlinearities and encode reasonable prior assumptions such as smoothness and continuity [31]. Here, we review the GP-SDE, a Bayesian generative model that leverages the expressivity of GPs for inferring latent dynamics [10].

Generative model In a GP-SDE, the evolution of the latent state $\boldsymbol{x} \in \mathbb{R}^K$ is modeled as a continuous-time SDE which underlies observed neural activity $\boldsymbol{y}(t_i) \in \mathbb{R}^D$ at time-points $t_i \in [0, T]$. Mathematically, this is expressed as

$$d\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{x})dt + \boldsymbol{\Sigma}^{\frac{1}{2}} d\boldsymbol{w}, \qquad \mathbb{E}[\boldsymbol{y}(t_i) \mid \boldsymbol{x}] = g\left(\boldsymbol{C}\boldsymbol{x}(t_i) + \boldsymbol{d}\right). \tag{1}$$

The drift function $\boldsymbol{f} : \mathbb{R}^K \to \mathbb{R}^K$ describes the system dynamics, $\boldsymbol{\Sigma}$ is a noise covariance matrix, and $d\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, dt\boldsymbol{I})$ is a Wiener process increment. Parameters $\boldsymbol{C} \in \mathbb{R}^{D\times K}$ and $\boldsymbol{d} \in \mathbb{R}^D$ define an affine mapping from latent to observed space, which is then passed through a pre-specified inverse link function $g(\cdot)$.

A GP prior is used to model each output dimension of the dynamics $\boldsymbol{f}(\cdot)$ independently. More formally, if $\boldsymbol{f}(\cdot) = [f_1(\cdot), \ldots, f_K(\cdot)]^{\mathsf{T}}$, then

$$f_k(\cdot) \overset{\text{iid}}{\sim} \mathcal{GP}(0, \kappa^{\Theta}(\cdot, \cdot)), \quad \text{for } k = 1, \cdots, K, \tag{2}$$

where $\kappa^{\Theta}(\cdot, \cdot)$ is the kernel for the GP with hyperparameters Θ.

Interpretability GP-SDEs and their variants can infer complex nonlinear dynamics with posterior uncertainty estimates in physical systems across a variety of applications [32–34]. However, one limitation of using this method with standard GP kernels, such as the radial basis function (RBF) kernel, is that its expressivity leads to dynamics that are often challenging to interpret. In Duncker et al. [10], this was addressed by conditioning the GP prior of the dynamics $\boldsymbol{f}(\cdot)$ on fixed points $\boldsymbol{f}(\boldsymbol{x}^*) = \mathbf{0}$ and their local Jacobians $J(\boldsymbol{x}^*) = \frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}^*}$, and subsequently learning the fixed-point locations $\boldsymbol{x}^*$ and the locally-linearized dynamics $J(\boldsymbol{x}^*)$ as model parameters. This approach allows for direct estimation of key features of $\boldsymbol{f}(\cdot)$. However, due to its flexibility, it is also prone to finding more fixed points than those included in the prior conditioning, which undermines its overall interpretability.

2.2 Recurrent switching linear dynamical systems

The rSLDS models nonlinear dynamics by switching between different sets of linear dynamics [8]. Accordingly, it retains the simplicity and interpretability of linear dynamical systems while providing much more expressive power. For these reasons, variations of the rSLDS are commonly used to model neural dynamics [19, 22–29].

Generative model The rSLDS is a discrete-time generative model of the following form:

$$\boldsymbol{x}_t \sim \mathcal{N}(\boldsymbol{A}_{s_t} \boldsymbol{x}_{t-1} + \boldsymbol{b}_{s_t}, \boldsymbol{Q}_{s_t}), \qquad \mathbb{E}[\boldsymbol{y}_t \mid \boldsymbol{x}_t] = g(\boldsymbol{C}\boldsymbol{x}_t + \boldsymbol{d}) \tag{3}$$

where dynamics switch between J distinct linear systems with parameters $\{\boldsymbol{A}_j, \boldsymbol{b}_j, \boldsymbol{Q}_j\}_{j=1}^J$. This is controlled by a discrete state variable $s_t \in \{1, \ldots, J\}$, which evolves via transition probabilities modeled by a multiclass logistic regression,

$$p(s_t \mid s_{t-1}, \boldsymbol{x}_{t-1}) \propto \exp(\boldsymbol{w}^{\mathsf{T}} \boldsymbol{x}_{t-1} + r_{s_{t-1}}). \tag{4}$$

The "recurrent" nature of this model comes from the dependence of eq. (4) on latent space locations $\boldsymbol{x}_t$. As such, the rSLDS can be understood as learning a partition of the latent space into J linear dynamical regimes seprated by linear decision boundaries. This serves as important motivation for the parametrization of the gpSLDS, as we describe later.

Interpretability While the rSLDS has been successfully used in many applications to model nonlinear dynamical systems, it suffers from a few practical limitations. First, it often produces unnatural artifacts of modeling nonlinear dynamics with discrete switches between linear systems. For example, it may oscillate between discrete modes with different discontinuous dynamics when a trajectory is near a regime boundary. Next, common fitting techniques for rSLDS models with non-conjugate observations typically treat dynamics as learnable hyperparameters rather than as probabilistic quantities [26], which prevents the model from being able to capture posterior uncertainty over the learned dynamics. Inferring a posterior distribution over dynamics is especially important in many neuroscience applications, where scientists often draw conclusions from discovering key features in latent dynamics, such as fixed points or line attractors.

3 Gaussian process switching linear dynamical systems

To address these limitations of the rSLDS, we propose a new class of models called the *Gaussian Process Switching Linear Dynamical System* (gpSLDS). The gpSLDS combines the modeling advantages of the GP-SDE with the structured flexbility of the rSLDS. We achieve this balance by designing a novel GP kernel function that defines a smooth, locally linear prior on dynamics. While our main focus is on providing an alternative to the rSLDS, the gpSLDS also contributes a new prior which allows for more interpretable learning of dynamics and fixed points than standard priors in the GP-SDE framework (e.g., the RBF kernel). Our implementation of the gpSLDS is available at: `https://github.com/lindermanlab/gpslds`.

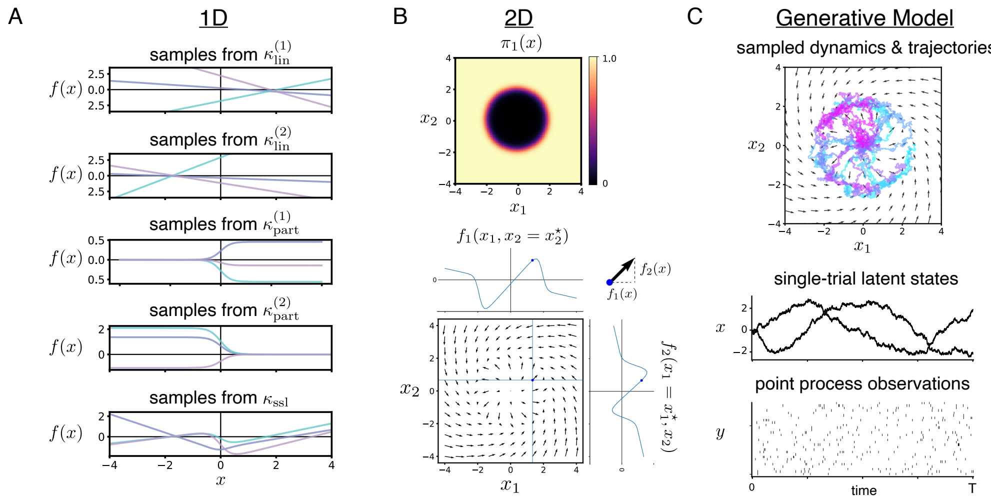

Figure 1: SSL kernel and generative model. **A.** 1D function samples, plotted in different colors, from GPs with five kernels: two linear kernels with different hyperparameters, partition kernels for each of the two regimes, and the SSL kernel. **B.** (*top*) An example $\boldsymbol{\pi}(\boldsymbol{x})$ in 2D and (*bottom*) a sample of dynamics from a SSL kernel in 2D with $\boldsymbol{\pi}(\boldsymbol{x})$ as hyperparameters. The x_1- and x_2- directions of the arrows are given by independent 1D samples of the kernel. **C.** Schematic of the generative model. Simulated trajectories follow the sampled dynamics. Each trajectory is observed via Poisson process or Gaussian observations.

3.1 The smoothly switching linear kernel

The core innovation of our method is a novel GP kernel, which we call the *Smoothly Switching Linear* (SSL) kernel. The SSL kernel specifies a GP prior over dynamics that maintains the switching linear structure of rSLDS models, while allowing dynamics to smoothly interpolate between different linear regimes.

For every pair of locations $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^K$, the SSL kernel with J linear regimes is defined as,

$$\kappa_{\text{ssl}}(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{J} \underbrace{((\boldsymbol{x} - \boldsymbol{c}_j)^\mathsf{T} \boldsymbol{M} (\boldsymbol{x}' - \boldsymbol{c}_j) + \sigma_0^2)}_{\kappa_{\text{lin}}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')} \underbrace{\pi_j(\boldsymbol{x}) \pi_j(\boldsymbol{x}')}_{\kappa_{\text{part}}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')} \tag{5}$$

where $\boldsymbol{c}_j \in \mathbb{R}^K$, $\boldsymbol{M} \in \mathbb{R}^{K \times K}$ is a diagonal positive semi-definite matrix, $\sigma_0^2 \in \mathbb{R}_+$, and $\pi_j(\boldsymbol{x}) \geq 0$ with $\sum_{j=1}^J \pi_j(\boldsymbol{x}) = 1$. To gain an intuitive understanding of the SSL kernel, we will separately analyze each of the two terms in the summands.

The first term, $\kappa_{\text{lin}}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')$, is a standard linear kernel which defines a GP distribution over linear functions [31]. The superscript j denotes that it is the linear prior on the dynamics in regime j. $\boldsymbol{M}$ controls the variance of the function's slope in each input dimension, and $\boldsymbol{c}_j$ is such that the variance of the function achieves its minimum value of σ_0^2 at $\boldsymbol{x} = \boldsymbol{c}_j$. We expand on the relationship between the linear kernel and linear dynamical systems in Appendix A.

The second term is what we define as the *partition kernel*, $\kappa_{\text{part}}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')$, which gives the gpSLDS its switching structure. We interpret $\boldsymbol{\pi}(\boldsymbol{x}) = [\pi_1(\boldsymbol{x}) \quad \dots \quad \pi_J(\boldsymbol{x})]^\mathsf{T}$ as parametrizing a categorical distribution over J linear regimes akin to the discrete switching variables in the rSLDS. We model $\boldsymbol{\pi}(\boldsymbol{x})$ as a multiclass logistic regression with decision boundaries $\{\boldsymbol{w}_j^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}) = 0 \mid j = 1, \dots, J-1\}$, where $\boldsymbol{\phi}(\boldsymbol{x})$ is any feature transformation of $\boldsymbol{x}$. This yields random functions which are locally constant and smoothly interpolate at the decision boundaries. More formally,

$$\pi_j(\boldsymbol{x}) = \frac{\exp(\boldsymbol{w}_j^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}) / \tau)}{1 + \sum_{j=1}^{J-1} \exp(\boldsymbol{w}_j^\mathsf{T} \boldsymbol{\phi}(\boldsymbol{x}) / \tau)}, \quad j = 1, \dots, J \tag{6}$$

where $\boldsymbol{w}_J = 0$. The hyperparameter $\tau \in \mathbb{R}_+$ controls the smoothness of the decision boundaries. As $\tau \to 0$, $\boldsymbol{\pi}(\boldsymbol{x})$ approaches a one-hot vector which produces piecewise constant functions with sharp boundaries, and as $\tau \to \infty$ the boundaries become more uniform. While we focus on the parametrization in eq. (6) for the experiments in this paper, we note that in principle any classification method can be used, such as another GP or a neural network.

The SSL kernel in eq. (5) naturally combines aspects of the linear and partition kernels via sums and products of kernels, which has an intuitive interpretation [35]. The product kernel $\kappa_{\text{lin}}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')\kappa_{\text{part}}^{(j)}(\boldsymbol{x}, \boldsymbol{x}')$ enforces linearity in regions where $\pi_j(\boldsymbol{x})$ is close to 1. Summing over J regimes then enforces linearity in each of the J regimes, leading to a prior on locally linear functions with knots determined by $\boldsymbol{\pi}(\boldsymbol{x})$. We note that our kernel is reminiscent of the one in Pfingsten et al. [36], which uses a GP classifier as a prior for $\boldsymbol{\pi}(\boldsymbol{x})$ and applies their kernel to a GP regression setting. Here, our work differs in that we explicitly enforce linearity in each regime and draw a novel connection to switching models like the rSLDS.

Figure 1A depicts 1D samples from each kernel. Figure 1B shows how a SSL kernel with $J = 2$ linear regimes separated by decision boundary $x_1^2 + x_2^2 = 4$ *(top)* produces a structured 2D flow field consisting of two linear systems, with x_1- and x_2- directions determined by 1D function samples *(bottom)*.

3.2 The gpSLDS generative model

The full generative model for the gpSLDS incorporates the SSL kernel in eq. (5) into a GP-SDE modeling framework. Instead of placing a GP prior with a standard kernel on the system dynamics as in eq. (2), we simply plug in our new SSL kernel so that

$$f_k(\cdot) \overset{\text{iid}}{\sim} \mathcal{GP}(0, \kappa_{\text{ssl}}^{\Theta}(\cdot, \cdot)), \quad \text{for } k = 1, \cdots, K, \tag{7}$$

where the kernel hyperparameters are $\Theta = \{\boldsymbol{M}, \sigma_0^2, \{\boldsymbol{c}_j\}_{j=1}^J, \{\boldsymbol{w}_j\}_{j=1}^{J-1}, \tau\}$. We then sample latent states and observations according to the GP-SDE via eq. (1). A schematic of the full generative model is depicted in fig. 1C.

Incorporating inputs In many modern neuroscience applications, we are often interested in how external inputs to the system, such as experimental stimuli, influence latent states. To this end, we also consider an extension of the model in eq. (1) which incorporates additive inputs of the form,

$$d\boldsymbol{x} = (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{v}(t))dt + \boldsymbol{\Sigma}^{\frac{1}{2}} d\boldsymbol{w}, \tag{8}$$

where $\boldsymbol{v}(t) \in \mathbb{R}^I$ is a time-varying, known input signal and $\boldsymbol{B} \in \mathbb{R}^{K \times I}$ maps inputs linearly to the latent space. The latent path inference and learning approaches presented in the following section can naturally be extended to this setting, with updates for $\boldsymbol{B}$ available in closed form. Further details are provided in Appendix B.3-B.6.

3.3 Latent path inference and parameter learning

For inference and learning in the gpSLDS, we apply and extend a variational expectation-maximization (vEM) algorithm for GP-SDEs from Duncker et al. [10]. In particular, we propose a modification of this algorithm that dramatically improves the learning accuracy of kernel hyperparameters, which are crucial to the interpretability of the gpSLDS. We outline the main ideas of the algorithm here, though full details can be found in Appendix B.

As in Duncker et al. [10], we consider a factorized variational approximation to the posterior,

$$q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{u}) = q(\boldsymbol{x}) \prod_{k=1}^{K} p(f_k \mid \boldsymbol{u}_k, \Theta) q(\boldsymbol{u}_k), \tag{9}$$

where we have augmented the model with sparse inducing points to make inference of $\boldsymbol{f}$ tractable [37]. The inducing points are located at $\{\boldsymbol{z}_m\}_{m=1}^M \subset \mathbb{R}^K$ and take values $(f_k(\boldsymbol{z}_1), \ldots, f_k(\boldsymbol{z}_M))^\top = \boldsymbol{u}_k$. Standard vEM maximizes the evidence lower bound (ELBO) to the marginal log-likelihood $\log p(\boldsymbol{y} \mid \Theta)$ by alternating between updating the variational posterior q and updating model hyperparameters Θ [38]. Using the factorization in eq. (9), we will denote the ELBO as $\mathcal{L}(q(\boldsymbol{x}), q(\boldsymbol{u}), \Theta)$.

For inference of $q(\boldsymbol{x})$, we follow the approach first proposed by Archambeau et al. [39] and extended by Duncker et al. [10]. Computing the ELBO using this approach requires computing variational expectations of the SSL kernel, which we approximate using Gauss-Hermite quadrature as they are not available in closed form. Full derivations of this step are provided in Appendix B.3. For inference of $q(\boldsymbol{u})$, we follow Duncker et al. [10] and choose a Gaussian variational posterior for $q(\boldsymbol{u}_k) = \mathcal{N}(\boldsymbol{u}_k \mid \boldsymbol{m}_u^{k*}, \boldsymbol{S}_u^{k*})$, which admits closed-form updates for the mean $\boldsymbol{m}_u^{k*}$ and covariance $\boldsymbol{S}_u^{k*}$ given $q(\boldsymbol{x})$ and Θ. Duncker et al. [10] perform these updates before updating Θ via gradient ascent on the ELBO in each vEM iteration.

In our setting, this did not work well in practice. The gpSLDS often exhibits strong dependencies between $q(\boldsymbol{u})$ and Θ, which makes standard coordinate-ascent steps in vEM prone to getting stuck in local maxima. These dependencies arise due to the highly structured nature of our GP prior; small changes in the decision boundaries $\{\boldsymbol{w}_j\}_{j=1}^{J-1}$ can lead to large (adverse) changes in the prior, which prevents vEM from escaping suboptimal regions of parameter space. To overcome these difficulties, we propose a different approach for learning Θ: instead of fixing $q(\boldsymbol{u})$ and performing gradient ascent on the ELBO, we optimize Θ by maximizing a partially optimized ELBO,

$$\Theta^* = \arg\max_{\Theta} \left\{ \max_{q(\boldsymbol{u})} \mathcal{L}(q(\boldsymbol{x}), q(\boldsymbol{u}), \Theta) \right\}. \tag{10}$$

Due to the conjugacy of the model, the inner maximization can be performed analytically. This approach circumvents local optima that plague coordinate ascent on the standard ELBO. While other similar approaches exploit model conjugacy for faster vEM convergence in sparse variational GPs [37, 40] and GP-SDEs [41], our approach is the first to our knowledge that leverages this structure specifically for learning the latent dynamics of a GP-SDE model. We empirically demonstrate the superior performance of our learning algorithm in Appendix C.

3.4 Recovering predicted dynamics

It is straightforward to obtain the approximate posterior distribution over $\boldsymbol{f}^* := \boldsymbol{f}(\boldsymbol{x}^*)$ evaluated at any new location $\boldsymbol{x}^*$. Under the assumption that $\boldsymbol{f}^*$ only depends on the inducing points, we can use the approximation $q(\boldsymbol{f}^*) = \prod_{k=1}^{K} \int p(f_k^* \mid \boldsymbol{u}_k, \Theta) q(\boldsymbol{u}_k) \, \mathrm{d}\boldsymbol{u}_k$ which can be computed in closed-form using properties of conditional Gaussian distributions. For a batch of points $\{\boldsymbol{x}_i^*\}_{i=1}^{N}$, this can be computed in $O(NM^2)$ time. The full derivation can be found in Appendix B.4.

This property highlights an appealing feature of the gpSLDS over the rSLDS. The gpSLDS infers a posterior distribution over dynamics at every point in latent space, even in regions of high uncertainty. Meanwhile, as we shall see later, the rSLDS expresses uncertainty by randomly oscillating between different sets of most-likely linear dynamics, which is much harder to interpret.

4 Results

4.1 Synthetic data

We begin by applying the gpSLDS to a synthetic dataset consisting of two linear rotational systems, one clockwise and one-counterclockwise, which combine smoothly at $x_1 = 0$ (fig. 2A). We simulate 30 trials of latent states from an SDE as in eq. (1) and then generate Poisson process observations given these latent states for $D = 50$ output dimensions (i.e. neurons) over $T = 2.5$ seconds (Fig. 2B). To initialize $\boldsymbol{C}$ and $\boldsymbol{d}$, we fit a Poisson LDS [4] to data binned at 20ms with identity dynamics. For the rSLDS, we also bin the data at 20ms. We then fit the gpSLDS and rSLDS models with $J = 2$ linear states using 5 different random initializations for 100 vEM iterations, and choose the fits with the highest ELBOs in each model class.

We find that the gpSLDS accurately recovers the true latent trajectories (fig. 2C) as well as the true rotation dynamics and the decision boundary between them (fig. 2D). We determine this decision boundary by thresholding the learned $\boldsymbol{\pi}(\boldsymbol{x})$ at 0.5. In addition, we can obtain estimates of fixed point locations by computing the posterior probability $\prod_{k=1}^{K} \mathbb{P}_{q(\boldsymbol{f})}(|f_k(\boldsymbol{x})| < \epsilon)$ for a small $\epsilon > 0$; the locations $\boldsymbol{x}$ with high probability are shaded in purple. This reveals that the gpSLDS finds high-probability fixed points which overlap significantly with the true fixed points, denoted by stars. In comparison, both the rSLDS and the GP-SDE with RBF kernel do not learn the correct decision boundary nor the fixed points as accurately (fig. 2E-F). Of particular note, the RBF kernel incorrectly extrapolates and finds a superfluous fixed point outside the region traversed by the true latent states.

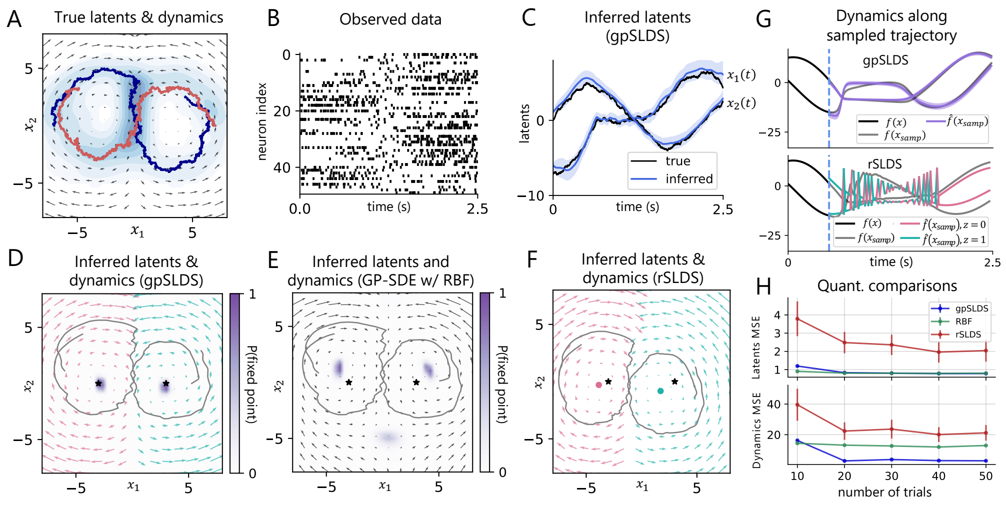

Figure 2: Synthetic data results. **A.** True dynamics and latent states used to generate the dataset. Dynamics are clockwise and counterclockwise linear systems separated by $x_1 = 0$. Two latent trajectories are shown on top of a kernel density estimate of the latent states visited by all 30 trials. **B.** Poisson process observations from an example trial. **C.** True vs. inferred latent states for the gpSLDS and rSLDS, with 95% posterior credible intervals. **D.** Inferred dynamics (pink/green) and two inferred latent trajectories (gray) corresponding to those in Panel A from a gpSLDS fit with 2 linear regimes. The model finds high-probability fixed points (purple) overlapping with true fixed points (stars). **E.** Analogous plot to D for the GP-SDE model with RBF kernel. Note that this model does not provide a partition of the dynamics. **F.** rSLDS inferred latents, dynamics, and fixed points (pink/green dots). **G.** *(top)* Sampled latents and corresponding dynamics from the gpSLDS, with 95% posterior credible intervals. *(bottom)* Same, but for the rSLDS. The pink/green trace represents the most likely dynamics at the sampled latents, colored by discrete switching variable. *H.* MSE between true and inferred latents and dynamics for gpSLDS, GP-SDE with RBF kernel, and rSLDS while varying the number of trials in the dataset. Error bars are ±2SE over 5 random initializations.

Figure 2G illustrates the differences in how the gpSLDS and the rSLDS express uncertainty over dynamics. To the left of the dashed line, we sample latent states starting from $x_0 = (7, 0)$ and plot the corresponding true dynamics. To the right, we simulate latent states $\boldsymbol{x}_{\text{samp}}$ from the fitted model and plot the true dynamics (in gray) and the learned most likely dynamics (in color) at $\boldsymbol{x}_{\text{samp}}$. For a well-fitting model, we would expect the true and learned dynamics to overlap. We see that the gpSLDS produces smooth simulated dynamics that match the true dynamics at $\boldsymbol{x}_{\text{samp}}$ *(top)*. By contrast, the rSLDS expresses uncertainty by oscillating between the two linear dynamical systems, hence producing uninterpretable dynamics at $\boldsymbol{x}_{\text{samp}}$ *(bottom)*. This region of uncertainty overlaps with the $x_1 = 0$ boundary, suggesting that the rSLDS fails to capture the smoothly interpolating dynamics present in the true system.

Next, we perform quantitative comparisons between the three competing methods (fig. 2H). We find that both continuous-time methods consistently outperform the rSLDS on both metrics, suggesting that these methods are likely more suitable for modeling Poisson process data. Moreover, the gpSLDS better recovers dynamics compared to the RBF kernel, illustrating that the correct inductive bias can yield performance gains over a more flexible prior, especially in a data-limited setting.

Lastly, we note that the gpSLDS can achieve more expressive power than the rSLDS by learning *nonlinear* decision boundaries between linear regimes, for instance by incorporating nonlinear features into $\boldsymbol{\phi}(\boldsymbol{x})$ in eq. (6). We demonstrate this feature for a 2D limit cycle in Appendix D.

4.2 Application to hypothalamic neural population recordings during aggression

In this section, we revisit the analyses of Nair et al. [27], which applied dynamical systems models to neural recordings during aggressive behavior in mice. To do this, we reanalyze a dataset which consists of calcium imaging of ventromedial hypothalamus neurons from a mouse interacting with two consecutive intruders. The recording was collected from 104 neurons at 15 Hz over ∼343

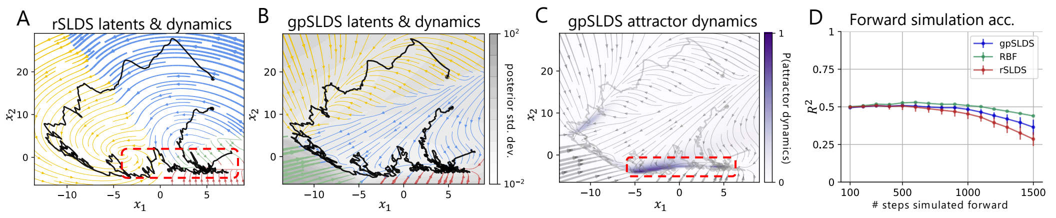

Figure 3: Results on hypothalamic data from Nair et al. [27]. In each of the panels A-C, flow field arrow widths are scaled by the magnitude of dynamics for clarity of visualization. **A.** rSLDS inferred latents and most likely dynamics. The presumed location of the line attractor from [27] is marked with a red box. **B.** gpSLDS inferred latents and most likely dynamics in latent space. Background is colored by posterior standard deviation of dynamics averaged across latent dimensions, which adjusts relative to the presence of data in the latent space. **C.** Posterior probability of slow points in gpSLDS, which validates line-attractor like dynamics, as marked by a red box. **D.** Comparison of in-sample forward simulation R^2 between gpSLDS, rSLDS, and GP-SDE with RBF kernel. To compute this, we choose initial conditions uniformly spaced 100 time bins apart in both trials, simulate latent states k steps forward according to learned dynamics (with k ranging from 100-1500), and evaluate the R^2 between predicted and true observations as in Nassar et al. [23]. Error bars are $\pm$2SE over 5 different initializations.

seconds (i.e. 5140 time bins). Nair et al. [27] found that an rSLDS fit to this data learns dynamics that form an approximate line attractor corresponding to an aggressive internal state (fig. 3A). Here, we supplement this analysis by using the gpSLDS to directly assess model confidence about this finding.

For our experiments, we z-score and then split the data into two trials, one for each distinct intruder interacting with the mouse. Following Nair et al. [27], we choose $J = 4$ linear regimes to compare the gpSLDS and rSLDS. We choose $K = 2$ latent dimensions to aid the visualization of the resulting model fits; we find that even in such low dimensions, both models still recover line attractor-like dynamics. We model the calcium imaging traces as Gaussian emissions on an evenly spaced time grid and initialize $\boldsymbol{C}$ and $\boldsymbol{d}$ using principal component analysis. We fit models with 5 different initializations for 50 vEM iterations and display the runs with highest forward simulation accuracy (as described in the caption of fig. 3).

In fig. 3A-B, we find that both methods infer similar latent trajectories and find plausible flow fields that are parsed in terms of simpler linear components. We further demonstrate the ability of the gpSLDS to more precisely identify the line attractor from Nair et al. [27]. To do this, we use the learned $q(\boldsymbol{f})$ to compute the posterior probability of slow dynamics on a dense (80×80) grid of points in the latent space using the procedure in Section 3.4. The gpSLDS finds a high-probability region of slow points corresponding to the approximate line attractor found in Nair et al. [27] (fig. 3C). This demonstrates a key advantage of the gpSLDS over the rSLDS: by modeling dynamics probabilistically using a structured GP prior, we can validate the finding of a line attractor with further statistical rigor. Finally, we compare the gpSLDS, rSLDS, and GP-SDE with RBF kernel using an in-sample forward simulation metric (fig. 3D). All three methods retain similar predictive power 500 steps into the future. After that, the gpSLDS performs slightly worse than the RBF kernel; however, it gains interpretability by imposing piecewise linear structure while still outperforming the rSLDS.

4.3 Application to lateral intraparietal neural recodings during decision making

In neuroscience, there is considerable interest in understanding how neural dynamics during decision making tasks support the process of making a choice [26, 34, 42–47]. In this section, we use the gpSLDS to infer latent dynamics from spiking activity in the lateral intraparietal (LIP) area of monkeys reporting decisions about the direction of motion of a random moving dots stimulus with varying degrees of motion coherence [42, 43]. The animal indicated its choice of net motion direction (either left or right) via a saccade. On some trials, a 100ms pulse of weak motion, randomly oriented to the left or right, was also presented to the animal. Here, we model the dynamics of 58 neurons recorded across 50 trials consisting of net motion coherence strengths in $\{-.512, -.128, 0.0, .128, .512\}$, where the sign corresponds to the net movement direction of the stimulus. We only consider data 200ms after motion onset, corresponding to the start of decision-related activity.

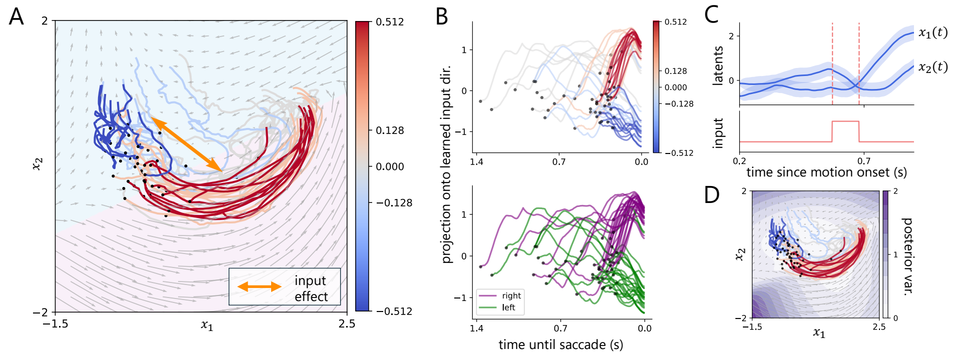

Figure 4: Results on LIP spiking data from a decision-making task in Stine et al. [42]. **A.** gpSLDS inferred latents colored by coherence, inferred dynamics with background colored by most likely linear regime, and the learned input-driven direction depicted by an orange arrow. **B.** Projection of latents onto the 1D input-driven axis from Panel A, colored by coherence *(top)* and choice *(bottom)*. **C.** Inferred latents with 95% credible intervals and corresponding 100ms pulse input for an example trial. **D.** Posterior variance of dynamics produced by the gpSLDS.

To capture potential input-driven effects, we fit a version of the gpSLDS described in eq. (8) with $K = 2$ latent dimensions and $J = 2$ linear regimes over 50 vEM iterations. We encoded the input signal as ± 1 with sign corresponding to pulse direction. In Figure 4A, we find that not only does the gpSLDS capture a distinct visual separation between trials by motion coherence, but it also learns a precise separating decision boundary between the two linear regimes. Our finding is consistent with previous work on a related task, which found that average LIP responses can be represented by a 2D curved manifold [48], though here we take a dynamical systems perspective. Additionally, our model learns an input-driven effect which appears to define a separating axis. To verify this, we project the inferred latent states onto the 1D subspace spanned by the input effect vector. Figure 4B shows that the latent states separate by coherence *(top)* and by choice *(bottom)*, further suggesting that the pulse input relates to meaningful variation in evidence accumulation for this task. Fig. 4C shows an example latent trajectory aligned with pulse input; during the pulse there is a noticeable change in the latent trajectory. Lastly, in Fig. 4D we find that the gpSLDS expresses high confidence in learned dynamics where latent trajectories are present and low confidence in areas further from this region.

5 Related work

There are several related approaches to learning nonlinear latent dynamics in discrete or continuous time. Gaussian process state-space models (GP-SSMs) [49–53] can be considered a discrete-time analogue to GP-SDEs. In a GP-SSM, observations are assumed to be regularly sampled and latent states evolve according to a discretized dynamical system with a GP prior. Wang et al. [49] and Turner et al. [50] learned the dynamics of a GP-SSM using maximum a posteriori estimation. Frigola et al. [51] and Eleftheriadis et al. [52] employed a variational approximation with sparse inducing points to infer the latent states and dynamics in a fully Bayesian fashion. In our work, we use a continuous-time framework that more naturally handles irregularly sampled data, such as point-process observations commonly encountered in neural spiking data. Neural ODEs and SDEs [54, 55] use deep neural networks to parametrize the dynamics of a continuous-time system, and have emerged as prominent tools for analyzing large datasets, including those in neuroscience [33, 56–58]. While these methods can represent flexible function classes, they are likely to overfit to low-data regimes and may be difficult to interpret. In addition, unlike the gpSLDS, neural ODEs and SDEs do not typically quantify uncertainty of the learned dynamics.

In the context of dynamical mixture models, Köhs et al. [59, 60] proposed a continuous-time switching model in a GP-SDE framework. This model assumes a latent Markov jump process over time which controls the system dynamics by switching between different SDEs. The switching process models dependence on time, but not location in latent space. In contrast, the gpSLDS does not explicitly represent a latent switching process and rather models switching probabilities as part of the kernel function. The dependence of the kernel on the location in latent space allows for the gpSLDS to partition the latent space into different linear regimes.

While our work has followed the inference approach of Archambeau et al. [61] and Duncker et al. [10], other methods for latent path inference in nonlinear SDEs have been proposed [32, 41, 62, 63]. Verma et al. [41] parameterized the posterior SDE path using an exponential family-based description. The resulting inference algorithm showed improved convergence of the E-step compared to Archambeau et al. [39]. Course and Nair [32, 62] proposed an amortization strategy that allows the variational update of the latent state to be parallelized over sequence length. In principle, any of these approaches could be applied to inference in the gpSLDS and would be an interesting direction for future work.

6 Discussion

In this paper, we introduced the gpSLDS to infer low-dimensional latent dynamical systems from high-dimensional, noisy observations. By developing a novel kernel for GP-SDE models that defines distributions over smooth locally-linear functions, we were able to relate GP-SDEs to rSLDS models and address key limitations of the rSLDS. Using both simulated and real neural datasets, we demonstrated that the gpSLDS can accurately infer true generative parameters and performs favorably in comparison to rSLDS models and GP-SDEs with other kernel choices. Moreover, our real data examples illustrate the variety of potential practical uses of this method. On calcium imaging traces recorded during aggressive behavior, the gpSLDS reveals dynamics consistent with the hypothesis of a line attractor put forward based on previous rSLDS analyses [27]. On a decision making dataset, the gpSLDS finds latent trajectories and dynamics that clearly separate by motion coherence and choice, providing a dynamical systems view consistent with prior studies [42, 48].

In our experiments, we demonstrated the ability of the gpSLDS to recover ground-truth dynamical systems and key dynamical features using fixed settings of hyperparameters: the latent dimensionality K and the number of regimes J. For simulated data, we set hyperparameters to their true values; for real data, we chose hyperparameters based on prior studies and did not further optimize these values. However, for most real neural datasets, we do not know the true underlying dimensionality or optimal number of regimes. To tune these hyperparameters, we can resort to standard techniques for model comparison in the neural latent variable modeling literature. Two common evaluation metrics are forward simulation accuracy [23, 27] and co-smoothing performance [6, 64, 65].

While these results are promising, we acknowledge a few limitations of the gpSLDS. First, the memory cost scales exponentially with the size of the latent dimension due to using quadrature methods to approximate expectations of the SSL kernel, which are not available in closed form. This computational limitation renders it difficult to fit the gpSLDS with many (i.e. greater than 3) latent dimensions. One potential direction for future work would be to instead use Monte Carlo methods to approximate kernel expectations for models with larger latent dimensionality. In addition, while both the gpSLDS and rSLDS require choosing a discretization timestep for solving dynamical systems, in practice we find that the gpSLDS requires smaller steps for stable model inference. This allows the gpSLDS to more accurately approximate dynamics with continuous-time likelihoods, at the cost of allocating more time bins during inference. Finally, we acknowledge that traditional variational inference approaches – such as those employed by the gpSLDS– tend to underestimate posterior variance due to the KL-divergence-based objective [38]. Carefully assessing biases introduced by our variational approximation to the posterior would be an important topic for future work.

Overall, the gpSLDS provides a general modeling approach for discovering latent dynamics of noisy measurements in an intepretable and fully probabilistic manner. We expect that our model will be a useful addition to the rSLDS and related methods on future analyses of neural data.

Acknowledgments and Disclosure of Funding

This work was supported by grants from the NIH BRAIN Initiative (U19NS113201, R01NS131987, & RF1MH133778) and the NSF/NIH CRCNS Program (R01NS130789), as well as fellowships from the Simons Collaboration on the Global Brain, the Wu Tsai Neurosciences Institute, the Alfred P. Sloan Foundation, and the McKnight Foundation. We thank Barbara Engelhardt, Julia Palacios, and the members of the Linderman Lab for helpful feedback throughout this project. The authors have no competing interests to declare.

References

[1] Peiran Gao and Surya Ganguli. On simplicity and complexity in the brave new world of large-scale neuroscience. *Current Opinion in Neurobiology*, 32:148–155, 2015.

[2] John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, 17(11):1500–1509, 2014.

[3] Shreya Saxena and John P Cunningham. Towards the neural population doctrine. *Current Opinion in Neurobiology*, 55:103–111, 2019.

[4] Anne C Smith and Emery N Brown. Estimating a state-space model from point process observations. *Neural Computation*, 15(5):965–991, 2003.

[5] Liam Paninski, Yashar Ahmadian, Daniel Gil Ferreira, Shinsuke Koyama, Kamiar Rahnama Rad, Michael Vidne, Joshua Vogelstein, and Wei Wu. A new look at state-space models for neural data. *Journal of Computational Neuroscience*, 29:107–126, 2010.

[6] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. *Advances in Neural Information Processing Systems*, 24, 2011.

[7] Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

[8] Scott Linderman, Matthew Johnson, Andrew Miller, Ryan Adams, David Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Artificial Intelligence and Statistics*, pages 914–922. PMLR, 2017.

[9] Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, 15(10):805–815, 2018.

[10] Lea Duncker, Gergo Bohner, Julien Boussard, and Maneesh Sahani. Learning interpretable continuous-time models of latent stochastic dynamical systems. In *International Conference on Machine Learning*, pages 1726–1734. PMLR, 2019.

[11] Marine Schimel, Ta-Chu Kao, Kristopher T Jensen, and Guillaume Hennequin. ilqr-vae: control-based learning of input-driven dynamics with applications to neural data. In *International Conference on Learning Representations*, 2021.

[12] Mikhail Genkin, Owen Hughes, and Tatiana A Engel. Learning non-stationary langevin dynamics from stochastic observations of latent trajectories. *Nature Communications*, 12(1): 5986, 2021.

[13] Christopher Langdon and Tatiana A Engel. Latent circuit inference from heterogeneous neural responses during cognitive tasks. *bioRxiv*, pages 2022–01, 2022.

[14] Matthew Dowling, Yuan Zhao, and Il Memming Park. Large-scale variational gaussian state-space models. *arXiv preprint arXiv:2403.01371*, 2024.

[15] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual Review of Neuroscience*, 43:249–275, 2020.

[16] Lea Duncker and Maneesh Sahani. Dynamics on the manifold: Identifying computational dynamical activity from neural population recordings. *Current Opinion in Neurobiology*, 70: 163–170, 2021.

[17] David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural Computation*, 25(3):626–649, 2013.

[18] Daniel J O'Shea, Lea Duncker, Werapong Goo, Xulu Sun, Saurabh Vyas, Eric M Trautmann, Ilka Diester, Charu Ramakrishnan, Karl Deisseroth, Maneesh Sahani, et al. Direct neural perturbations reveal a dynamical mechanism for robust computation. *bioRxiv*, pages 2022–12, 2022.

[19] Joana Soldado-Magraner, Valerio Mante, and Maneesh Sahani. Inferring context-dependent computations through linear approximations of prefrontal cortex dynamics. *Science Advances*, 10(51):eadl4743, 2024.

[20] Amit Vinograd, Aditya Nair, Joseph Kim, Scott W Linderman, and David J Anderson. Causal evidence of a line attractor encoding an affective state. *Nature*, pages 1–3, 2024.

[21] George Mountoufaris, Aditya Nair, Bin Yang, Dong-Wook Kim, Amit Vinograd, Samuel Kim, Scott W Linderman, and David J Anderson. A line attractor encoding a persistent internal state requires neuropeptide signaling. *Cell*, 187(21):5998–6015, 2024.

[22] Biljana Petreska, Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen Ryu, Krishna V Shenoy, and Maneesh Sahani. Dynamical segmentation of single trials from population neural data. *Advances in Neural Information Processing Systems*, 24, 2011.

[23] Josue Nassar, Scott Linderman, Monica Bugallo, and Il Memming Park. Tree-structured recurrent switching linear dynamical systems for multi-scale modeling. In *International Conference on Learning Representations*, 2018.

[24] Jalil Taghia, Weidong Cai, Srikanth Ryali, John Kochalka, Jonathan Nicholas, Tianwen Chen, and Vinod Menon. Uncovering hidden brain state dynamics that regulate performance and decision-making during cognition. *Nature Communications*, 9(1):2505, 2018.

[25] Antonio C Costa, Tosif Ahamed, and Greg J Stephens. Adaptive, locally linear models of complex dynamics. *Proceedings of the National Academy of Sciences*, 116(5):1501–1510, 2019.

[26] David Zoltowski, Jonathan Pillow, and Scott Linderman. A general recurrent state space framework for modeling neural dynamics during decision-making. In *International Conference on Machine Learning*, pages 11680–11691. PMLR, 2020.

[27] Aditya Nair, Tomomi Karigo, Bin Yang, Surya Ganguli, Mark J Schnitzer, Scott W Linderman, David J Anderson, and Ann Kennedy. An approximate line attractor in the hypothalamus encodes an aggressive state. *Cell*, 186(1):178–193, 2023.

[28] Mengyu Liu, Aditya Nair, Nestor Coria, Scott W Linderman, and David J Anderson. Encoding of female mating dynamics by a hypothalamic line attractor. *Nature*, pages 1–3, 2024.

[29] Jimmy Smith, Scott Linderman, and David Sussillo. Reverse engineering recurrent neural networks with Jacobian switching linear dynamical systems. *Advances in Neural Information Processing Systems*, 34:16700–16713, 2021.

[30] Noga Mudrik, Yenho Chen, Eva Yezerets, Christopher J Rozell, and Adam S Charles. Decomposed linear dynamical systems (dlds) for learning the latent components of neural dynamics. *Journal of Machine Learning Research*, 25(59):1–44, 2024.

[31] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[32] Kevin Course and Prasanth B Nair. State estimation of a physical system with unknown governing equations. *Nature*, 622(7982):261–267, 2023.

[33] Timothy Doyeon Kim, Thomas Zhihao Luo, Tankut Can, Kamesh Krishnamurthy, Jonathan W Pillow, and Carlos D Brody. Flow-field inference from neural data using deep recurrent networks. *bioRxiv*, 2023.

[34] Thomas Zhihao Luo, Timothy Doyeon Kim, Diksha Gupta, Adrian G Bondy, Charles D Kopec, Verity A Elliot, Brian DePasquale, and Carlos D Brody. Transitions in dynamical regime and neural mode underlie perceptual decision-making. *bioRxiv*, pages 2023–10, 2023.

[35] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, 2014.

[36] Tobias Pfingsten, Malte Kuss, and Carl Edward Rasmussen. Nonstationary gaussian process regression using a latent extension of the input space. In *Eighth World Meeting of the International Society for Bayesian Analysis (ISBA 2006)*, 2006.

[37] Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574. PMLR, 2009.

[38] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[39] Cedric Archambeau, Dan Cornford, Manfred Opper, and John Shawe-Taylor. Gaussian process approximations of stochastic differential equations. In *Gaussian Processes in Practice*, pages 1–16. PMLR, 2007.

[40] Vincent Adam, Paul Chang, Mohammad Emtiyaz E Khan, and Arno Solin. Dual parameterization of sparse variational gaussian processes. *Advances in Neural Information Processing Systems*, 34:11474–11486, 2021.

[41] Prakhar Verma, Vincent Adam, and Arno Solin. Variational gaussian process diffusion processes. In *International Conference on Artificial Intelligence and Statistics*, pages 1909–1917. PMLR, 2024.

[42] Gabriel M Stine, Eric M Trautmann, Danique Jeurissen, and Michael N Shadlen. A neural mechanism for terminating decisions. *Neuron*, 111(16):2601–2613, 2023.

[43] Jamie D Roitman and Michael N Shadlen. Response of neurons in the lateral intraparietal area during a combined visual discrimination reaction time task. *Journal of Neuroscience*, 22(21): 9475–9489, 2002.

[44] Jochen Ditterich. Stochastic models of decisions about motion direction: behavior and physiology. *Neural Networks*, 19(8):981–1012, 2006.

[45] Joshua I Gold and Michael N Shadlen. The neural basis of decision making. *Annu. Rev. Neurosci.*, 30:535–574, 2007.

[46] Kenneth W Latimer, Jacob L Yates, Miriam LR Meister, Alexander C Huk, and Jonathan W Pillow. Single-trial spike trains in parietal cortex reveal discrete steps during decision-making. *Science*, 349(6244):184–187, 2015.

[47] Mikhail Genkin, Krishna V Shenoy, Chandramouli Chandrasekaran, and Tatiana A Engel. The dynamics and geometry of choice in premotor cortex. *bioRxiv*, 2023.

[48] Gouki Okazawa, Christina E Hatch, Allan Mancoo, Christian K Machens, and Roozbeh Kiani. Representational geometry of perceptual decisions in the monkey parietal cortex. *Cell*, 184(14): 3748–3761, 2021.

[49] Jack Wang, Aaron Hertzmann, and David J Fleet. Gaussian process dynamical models. *Advances in Neural Information Processing Systems*, 18, 2005.

[50] Ryan Turner, Marc Deisenroth, and Carl Rasmussen. State-space inference and learning with gaussian processes. In *Artificial Intelligence and Statistics*, pages 868–875. JMLR Workshop and Conference Proceedings, 2010.

[51] Roger Frigola, Yutian Chen, and Carl Edward Rasmussen. Variational gaussian process state-space models. *Advances in Neural Information Processing Systems*, 27, 2014.

[52] Stefanos Eleftheriadis, Tom Nicholson, Marc Deisenroth, and James Hensman. Identification of gaussian process state space models. *Advances in Neural Information Processing Systems*, 30, 2017.

[53] Gergo Bohner and Maneesh Sahani. Empirical fixed point bifurcation analysis. *arXiv preprint arXiv:1807.01486*, 2018.

[54] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 31, 2018.

[55] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32, 2019.

[56] Timothy D Kim, Thomas Z Luo, Jonathan W Pillow, and Carlos D Brody. Inferring latent dynamics underlying neural population activity via neural differential equations. In *International Conference on Machine Learning*, pages 5551–5561. PMLR, 2021.

[57] Timothy Doyeon Kim, Tankut Can, and Kamesh Krishnamurthy. Trainability, expressivity and interpretability in gated neural odes. In *International Conference on Machine Learning*, pages 16393–16423, 2023.

[58] Christopher Versteeg, Andrew R Sedler, Jonathan D McCart, and Chethan Pandarinath. Expressive dynamics models with nonlinear injective readouts enable reliable recovery of latent features from neural activity. *arXiv preprint arXiv:2309.06402*, 2023.

[59] Lukas Köhs, Bastian Alt, and Heinz Koeppl. Variational inference for continuous-time switching dynamical systems. *Advances in Neural Information Processing Systems*, 34:20545–20557, 2021.

[60] Lukas Köhs, Bastian Alt, and Heinz Koeppl. Markov chain monte carlo for continuous-time switching dynamical systems. In *International Conference on Machine Learning*, pages 11430–11454. PMLR, 2022.

[61] Cédric Archambeau, Manfred Opper, Yuan Shen, Dan Cornford, and John Shawe-Taylor. Variational inference for diffusion processes. *Advances in Neural Information Processing Systems*, 20, 2007.

[62] Kevin Course and Prasanth Nair. Amortized reparametrization: Efficient and scalable variational inference for latent sdes. *Advances in Neural Information Processing Systems*, 36, 2024.

[63] Botond Cseke, David Schnoerr, Manfred Opper, and Guido Sanguinetti. Expectation propagation for continuous time stochastic processes. *Journal of Physics A: Mathematical and Theoretical*, 49(49):494002, 2016.

[64] Anqi Wu, Stan Pashkovski, Sandeep R Datta, and Jonathan W Pillow. Learning a latent manifold of odor representations from neural responses in piriform cortex. *Advances in Neural Information Processing Systems*, 31, 2018.

[65] Stephen Keeley, Mikio Aoi, Yiyi Yu, Spencer Smith, and Jonathan W Pillow. Identifying signal and noise structure in neural population activity with gaussian process factor models. *Advances in Neural Information Processing Systems*, 33:13795–13805, 2020.

Supplementary Material

Table of Contents

A Relationship between linear kernel and linear dynamical systems

Here, we draw a mathematical connection between functions sampled from a GP with a linear kernel,

$$f(\cdot) \sim \mathcal{GP}(0, \kappa_{\text{lin}}(\cdot,\cdot)), \qquad \kappa_{\text{lin}}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{c})^{\mathsf{T}} \boldsymbol{M} (\boldsymbol{x} - \boldsymbol{c}) + \sigma_0^2 \tag{11}$$

and functions of the form

$$f(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{x} + \beta_0. \tag{12}$$

This connection is useful for understanding the relationship between our model formulation and the typical linear dynamical systems formulation as in the rSLDS. In particular, we will show that the linear kernel in eq. (11) equivalently places a prior on $\boldsymbol{\beta}$ and β_0 in eq. (12), in a similar manner to Bayesian linear regression.

By definition of a GP, for any finite set of input locations $\{\boldsymbol{x}_i\}_{i=1}^N$, we have

$$\begin{bmatrix} f(\boldsymbol{x}_1) & f(\boldsymbol{x}_2) & \dots & f(\boldsymbol{x}_N) \end{bmatrix}^{\mathsf{T}} \sim \mathcal{N}(\boldsymbol{0}, \Phi) \tag{13}$$

where $\Phi_{ij} = \kappa_{\text{lin}}(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Equivalently, for every pair $i, j = 1, \dots, N$,

$$\begin{aligned} \text{Cov}(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)) &= (\boldsymbol{x}_i - \boldsymbol{c})^{\mathsf{T}} \boldsymbol{M} (\boldsymbol{x}_j - \boldsymbol{c}) + \sigma_0^2 & (14) \\ &= \boldsymbol{x}_i^{\mathsf{T}} \boldsymbol{M} \boldsymbol{x}_j - \boldsymbol{c}^{\mathsf{T}} \boldsymbol{M} (\boldsymbol{x}_i + \boldsymbol{x}_j) + \boldsymbol{c}^{\mathsf{T}} \boldsymbol{c} + \sigma_0^2 & (15) \end{aligned}$$

Under eq. (12), treating $\boldsymbol{\beta}$ and β_0 as random, this would become

$$\text{Cov}(\boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{x}_i + \beta_0, \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{x}_j + \beta_0) = \boldsymbol{x}_i^{\mathsf{T}} \text{Cov}(\boldsymbol{\beta}, \boldsymbol{\beta}) \boldsymbol{x}_j + \text{Cov}(\beta_0, \boldsymbol{\beta})(\boldsymbol{x}_i + \boldsymbol{x}_j) + \text{Var}(\beta_0) \tag{16}$$

Equation (15) and eq. (16) are equivalent and eq. (13) is satisfied if and only if

$$\begin{bmatrix} \boldsymbol{\beta} \\ \beta_0 \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{M} & -\boldsymbol{M}\boldsymbol{c} \\ -\boldsymbol{c}^{\mathsf{T}} \boldsymbol{M} & \boldsymbol{c}^{\mathsf{T}} \boldsymbol{c} + \sigma_0^2 \end{bmatrix}\right) \tag{17}$$

This shows that a GP with a linear kernel is equivalent to a Bayesian linear regression model with a prior on coefficients of the form in eq. (17).

B Inference and learning

We present the full inference and learning details for the gpSLDS with additive inputs,

$$d\boldsymbol{x} = (\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{v}(t))dt + \boldsymbol{\Sigma}^{\frac{1}{2}} d\boldsymbol{w}, \qquad \mathbb{E}[\boldsymbol{y}(t_i) \mid \boldsymbol{x}] = g\left(\boldsymbol{C}\boldsymbol{x}(t_i) + \boldsymbol{d}\right). \tag{18}$$

Our approach primarily follows that of Duncker et al. [10]. We use the following notation from Duncker et al. [10] throughout this section:

- $\langle h(\cdot) \rangle_{p(\cdot)}$ denotes the expectation of $h(\cdot)$ with respect to the distribution $p(\cdot)$.
- $\boldsymbol{K}_{xz} \in \mathbb{R}^{N_1 \times N_2}$ denotes the covariance matrix defined by our kernel $\kappa_{\text{ssl}}(\cdot,\cdot)$ for two batches of points $\{\boldsymbol{x}_i\}_{i=1}^{N_1}$ and $\{\boldsymbol{z}_i\}_{i=1}^{N_2}$. Specifically, $[\boldsymbol{K}_{xz}]_{ij} = \kappa_{\text{ssl}}(\boldsymbol{x}_i, \boldsymbol{z}_j)$. If one of these batches has only one point, e.g. $N_1 = 1$, we denote the corresponding covariance vector as $\boldsymbol{k}_{xz} \in \mathbb{R}^{1 \times N_2}$. Note, these quantities depend on Θ through the kernel; we omit writing this dependence for brevity.

B.1 Augmenting the generative model

Following Duncker et al. [10], we first augment our generative model with sparse inducing points $\{\boldsymbol{u}_k\}_{k=1}^K \subset \mathbb{R}^M$ at input locations $\{\boldsymbol{z}_m\}_{m=1}^M \subset \mathbb{R}^D$. Inducing points can be seen as pseudo-observations of $\boldsymbol{f}(\cdot)$ at input locations $\{\boldsymbol{z}_m\}_{m=1}^M$. They allow for tractable inference of $\boldsymbol{f}$ at any new batch of N input locations by reducing computational complexity from $O(N^3)$ to $O(NM^2)$, where typically M is chosen to be much smaller than N [37]. After this augmentation, the joint likelihood of our model becomes

$$p(\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{f}, \boldsymbol{u} \mid \Theta) = p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x} \mid \boldsymbol{f}) \prod_{k=1}^K p(f_k \mid \boldsymbol{u}_k, \Theta) p(\boldsymbol{u}_k \mid \Theta). \tag{19}$$

Treating $\boldsymbol{u}_k$ as pseudo-observations, we assume the following augmented prior:

$$p(\boldsymbol{u}_k \mid \Theta) = \mathcal{N}(\boldsymbol{u}_k \mid \boldsymbol{0}, \boldsymbol{K}_{zz}). \tag{20}$$

Then we can view $f_k(\cdot) \mid \boldsymbol{u}_k$ as a new GP conditioned on $\boldsymbol{u}_k$,

$$f_k(\cdot) \mid \boldsymbol{u}_k \sim \mathcal{GP}(\mu_{f_k|\boldsymbol{u}_k}(\cdot), \kappa_{f_k|\boldsymbol{u}_k}(\cdot,\cdot)), \tag{21}$$

where

$$\begin{aligned}\mu_{f_k|\boldsymbol{u}_k}(\boldsymbol{x}) &= \boldsymbol{k}_{xz}\boldsymbol{K}_{zz}^{-1}\boldsymbol{u}_k\\ \kappa_{f_k|\boldsymbol{u}_k}(\boldsymbol{x},\boldsymbol{x}') &= \kappa_{\text{ssl}}(\boldsymbol{x},\boldsymbol{x}') - \boldsymbol{k}_{xz}\boldsymbol{K}_{zz}^{-1}\boldsymbol{k}_{zx}.\end{aligned}$$

B.2 Variational lower bound

As in Duncker et al. [10], we consider a variational approximation to the posterior of the form

$$q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{u}) = q(\boldsymbol{x}) \prod_{k=1}^{K} p(f_k \mid \boldsymbol{u}_k, \Theta) q(\boldsymbol{u}_k). \tag{22}$$

Using this factorization, we derive the ELBO to the marginal log-likelihood of our model. By Jensen's inequality,

$$\begin{aligned}\log p(\boldsymbol{y} \mid \Theta) &= \log \int p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x} \mid \boldsymbol{f}) p(\boldsymbol{f} \mid \boldsymbol{u}, \Theta) p(\boldsymbol{u} \mid \Theta) d\boldsymbol{x} d\boldsymbol{f} d\boldsymbol{u}\\ &\geq \int q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{u}) \log \frac{p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x} \mid \boldsymbol{f}) p(\boldsymbol{f} \mid \boldsymbol{u}, \Theta) p(\boldsymbol{u} \mid \Theta)}{q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{u})} d\boldsymbol{x} d\boldsymbol{f} d\boldsymbol{u}\\ &= \int q(\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{u}) \log \frac{p(\boldsymbol{y} \mid \boldsymbol{x}) p(\boldsymbol{x} \mid \boldsymbol{f}) \prod_{k=1}^{K} p(\boldsymbol{u}_k \mid \Theta)}{q(\boldsymbol{x}) \prod_{k=1}^{K} q(\boldsymbol{u}_k)} d\boldsymbol{x} d\boldsymbol{f} d\boldsymbol{u}\\ &= \langle \log p(\boldsymbol{y} \mid \boldsymbol{x}) \rangle_{q(\boldsymbol{x})} - \langle \mathrm{KL}[q(\boldsymbol{x}) || p(\boldsymbol{x} \mid \boldsymbol{f})] \rangle_{q(\boldsymbol{f})} - \sum_{k=1}^{K} \mathrm{KL}[q(\boldsymbol{u}_k) || p(\boldsymbol{u}_k \mid \Theta)]\\ &:= \mathcal{L}(q(\boldsymbol{x}), q(\boldsymbol{u}), \Theta),\end{aligned}$$

where

$$q(\boldsymbol{f}) = \prod_{k=1}^{K} \int p(f_k \mid \boldsymbol{u}_k, \Theta) q(\boldsymbol{u}_k) d\boldsymbol{u}_k. \tag{23}$$

B.3 Inference of latent paths

To perform inference over the posterior of latent paths $q(\boldsymbol{x})$, we follow a method first proposed in Archambeau et al. [39] and extended by Duncker et al. [10].

As in Archambeau et al. [39], we choose a posterior distribution $q(\boldsymbol{x})$ characterized by a Markov Gaussian process,

$$q(\boldsymbol{x}) : \{d\boldsymbol{x} = \underbrace{(-\boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{b}(t))}_{:=\boldsymbol{f}_q(\boldsymbol{x})} dt + \boldsymbol{\Sigma}^{\frac{1}{2}} d\boldsymbol{w}, \quad \boldsymbol{x}_0 \sim \mathcal{N}(\boldsymbol{m}(0), \boldsymbol{S}(0))\}. \tag{24}$$

This distribution satisfies posterior marginals $q(\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_t \mid \boldsymbol{m}_t, \boldsymbol{S}_t)$ which satisfy the differential equations

$$\frac{d\boldsymbol{m}(t)}{dt} = -\boldsymbol{A}(t)\boldsymbol{m}(t) + \boldsymbol{b}(t) \tag{25}$$

$$\frac{d\boldsymbol{S}(t)}{dt} = -\boldsymbol{A}(t)\boldsymbol{S}(t) - \boldsymbol{S}(t)\boldsymbol{A}(t)^{\mathsf{T}} + \boldsymbol{\Sigma}. \tag{26}$$

Archambeau et al. [39] maximize the ELBO with respect to $q(\boldsymbol{x})$ subject to the constraints in eq. (25) and eq. (26) using the method of Lagrange multipliers. They show that after applying integration by parts, the Lagrangian becomes

$$\widetilde{\mathcal{L}} = \mathcal{L}(q(\boldsymbol{x}), q(\boldsymbol{u}), \Theta) - \mathcal{C}_1 - \mathcal{C}_2 \tag{27}$$

where

$$\mathcal{C}_1 = \int_0^T \left\{ \boldsymbol{\lambda}(t)^\top (\boldsymbol{A}(t)\boldsymbol{m}(t) - \boldsymbol{b}(t)) - \frac{d\boldsymbol{\lambda}(t)}{dt}^\top \boldsymbol{m}(t) \right\} dt + \boldsymbol{\lambda}(T)^\top \boldsymbol{m}(T) - \boldsymbol{\lambda}(0)^\top \boldsymbol{m}(0) \tag{28}$$

$$\mathcal{C}_2 = \int_0^T \operatorname{Tr}\left[\boldsymbol{\Psi}(t)(\boldsymbol{A}(t)\boldsymbol{S}(t) + \boldsymbol{S}(t)\boldsymbol{A}(t)^\top - \boldsymbol{\Sigma}) - \frac{d\boldsymbol{\Psi}(t)}{dt}\boldsymbol{S}(t) \right] dt \tag{29}$$

$$+ \operatorname{Tr}[\boldsymbol{\Psi}(T)\boldsymbol{S}(T)] - \operatorname{Tr}[\boldsymbol{\Psi}(0)\boldsymbol{S}(0)] \tag{30}$$

As in Archambeau et al. [39], we assume that $\boldsymbol{\lambda}(T) = \mathbf{0}$ and $\boldsymbol{\Psi}(T) = \mathbf{0}$.

To find the stationary points of the Lagrangian, we first take derivatives of $\widetilde{\mathcal{L}}$ with respect to $\boldsymbol{m}(0), \boldsymbol{S}(0), \boldsymbol{m}(t), \boldsymbol{S}(t), \boldsymbol{A}(t), \boldsymbol{b}(t)$ and set them to 0. The derivatives with respect to $\boldsymbol{m}(0)$ and $\boldsymbol{S}(0)$ lead to the updates

$$\boldsymbol{m}(0) = \boldsymbol{\mu}(0) - \boldsymbol{V}(0)\boldsymbol{\lambda}(0), \qquad \boldsymbol{S}(0) = \left(2\boldsymbol{\Psi}(0) + \boldsymbol{V}(0)^{-1}\right)^{-1} \tag{31}$$

where we assume a prior on initial conditions $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0 \mid \boldsymbol{\mu}(0), \boldsymbol{V}(0))$.

The derivatives with respect to $\boldsymbol{m}(t)$ and $\boldsymbol{S}(t)$ lead to the stationary equations

$$\frac{d\boldsymbol{\lambda}(t)}{dt} = \boldsymbol{A}(t)^\top \boldsymbol{\lambda}(t) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{m}(t)} \tag{32}$$

$$\frac{d\boldsymbol{\Psi}(t)}{dt} = \boldsymbol{A}(t)^\top \boldsymbol{\Psi}(t) - \boldsymbol{\Psi}(t)\boldsymbol{A}(t) - \frac{\partial \mathcal{L}}{\partial \boldsymbol{S}(t)} \odot \mathbb{P} \tag{33}$$

with $\mathbb{P}_{ij} = \frac{1}{2}$ for $i \neq j$ and 1 otherwise. The inclusion of $\mathbb{P}$ was proposed by Duncker et al. [10] to adjust for taking derivatives with respect to a symmetric matrix.

To take derivatives with respect to $\boldsymbol{A}(t)$ and $\boldsymbol{b}(t)$, we first extend a result from Appendix A of Archambeau et al. [39] to our affine inputs model. This allows us to rewrite the KL-divergence term between the posterior and prior latent paths in the ELBO as

$$\begin{aligned}
\langle \mathrm{KL}[q(\boldsymbol{x}) || p(\boldsymbol{x} \mid \boldsymbol{f})] \rangle_{q(\boldsymbol{f})} &= \frac{1}{2}\int_0^T \left\langle (\boldsymbol{f} + \boldsymbol{B}\boldsymbol{v}(t) - \boldsymbol{f}_q)^\top (\boldsymbol{f} + \boldsymbol{B}\boldsymbol{v}(t) - \boldsymbol{f}_q) \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} dt \\
&= \frac{1}{2}\int_0^T \left\langle (\boldsymbol{B}\boldsymbol{v}(t) + \Delta\boldsymbol{f})^\top (\boldsymbol{B}\boldsymbol{v}(t) + \Delta\boldsymbol{f}) \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} dt \\
&= \frac{1}{2}\int_0^T \left\langle (\Delta\boldsymbol{f})^\top (\Delta\boldsymbol{f}) \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} dt \\
&\quad + \int_0^T \boldsymbol{v}(t)^\top \boldsymbol{B}^\top \langle \Delta\boldsymbol{f} \rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} dt + \frac{1}{2}\int_0^T \boldsymbol{v}(t)^\top \boldsymbol{B}^\top \boldsymbol{B}\boldsymbol{v}(t) dt
\end{aligned} \tag{34}$$

where $\Delta\boldsymbol{f} := \boldsymbol{f} - \boldsymbol{f}_q$. The integrand of the first term in eq. (34) can be expanded as

$$\begin{aligned}
\left\langle (\Delta\boldsymbol{f})^\top (\Delta\boldsymbol{f}) \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} &= \left\langle \boldsymbol{f}^\top \boldsymbol{f} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} + 2\operatorname{Tr}\left[\boldsymbol{A}(t)^\top \left\langle \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} \boldsymbol{S}(t) \right] \\
&\quad + \operatorname{Tr}\left[\boldsymbol{A}(t)^\top \boldsymbol{A}(t)(\boldsymbol{S}(t) + \boldsymbol{m}(t)\boldsymbol{m}(t)^\top) \right] + 2\boldsymbol{m}(t)^\top \boldsymbol{A}(t)^\top \langle \boldsymbol{f} \rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} \\
&\quad + \boldsymbol{b}(t)^\top \boldsymbol{b}(t) - 2\boldsymbol{b}(t)^\top \langle \boldsymbol{f} \rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} - 2\boldsymbol{b}^\top \boldsymbol{A}(t)\boldsymbol{m}(t)
\end{aligned} \tag{35}$$

where we have used the identity $\left\langle \langle \boldsymbol{f}(\boldsymbol{x}) \rangle_{q(\boldsymbol{f})} (\boldsymbol{x} - \boldsymbol{m})^\top \right\rangle_{q(\boldsymbol{x})} = \left\langle \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} \boldsymbol{S}$, which can be derived from Stein's lemma. Note that computing eq. (35) relies on three quantities,

$$\langle \boldsymbol{f} \rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})}, \left\langle \boldsymbol{f}^\top \boldsymbol{f} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})}, \left\langle \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})}$$

which can be written as terms which depend on expectations of the kernel with respect to $q(\boldsymbol{x})$. We derive these as follows:

$$\begin{aligned}
\langle \boldsymbol{f} \rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} &= \left\langle \boldsymbol{k}_{xz} \boldsymbol{K}_{zz}^{-1} \boldsymbol{u} \right\rangle_{q(\boldsymbol{u}),q(\boldsymbol{x})} \\
&= \langle \boldsymbol{k}_{xz} \rangle_{q(\boldsymbol{x})} \boldsymbol{K}_{zz}^{-1} \boldsymbol{m}_u
\end{aligned}$$

$$
\begin{aligned}
\left\langle \boldsymbol{f}^\mathsf{T} \boldsymbol{f} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} &= \left\langle \sum_{k=1}^{K} \left\langle f_k(\boldsymbol{x})^2 \right\rangle_{q(f_k)} \right\rangle_{q(\boldsymbol{x})} \\
&= \left\langle \sum_{k=1}^{K} \mathrm{Var}_{q(f_k)}[f_k(\boldsymbol{x})] + \langle f_k(\boldsymbol{x}) \rangle_{q(f_k)}^2 \right\rangle_{q(\boldsymbol{x})} \\
&= \sum_{k=1}^{K} \underbrace{\left\langle \mathrm{Var}_{p(f_k|\boldsymbol{u})}[f_k(\boldsymbol{x})] \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{u})}}_{\text{Term 1}} + \underbrace{\left\langle \mathrm{Var}_{q(u)}[\langle f_k(\boldsymbol{x}) \rangle_{p(f_k \mid \boldsymbol{u})}] \right\rangle_{q(\boldsymbol{x})}}_{\text{Term 2}} \\
&\quad + \underbrace{\left\langle \langle f_k(\boldsymbol{x}) \rangle_{p(f_k \mid \boldsymbol{u}),q(\boldsymbol{u})}^2 \right\rangle_{q(\boldsymbol{x})}}_{\text{Term 3}} \\
&= \sum_{k=1}^{K} \underbrace{\left\langle \boldsymbol{k}_{xx} - \boldsymbol{k}_{xz} \boldsymbol{K}_{zz}^{-1} \boldsymbol{k}_{zx} \right\rangle_{q(\boldsymbol{x})}}_{\text{Term 1}} + \underbrace{\left\langle \boldsymbol{k}_{xz} \boldsymbol{K}_{zz}^{-1} \boldsymbol{S}_u^k \boldsymbol{K}_{zz}^{-1} \boldsymbol{k}_{zx} \right\rangle_{q(\boldsymbol{x})}}_{\text{Term 2}} \\
&\quad + \underbrace{\left\langle \boldsymbol{k}_{xz} \boldsymbol{K}_{zz}^{-1} \boldsymbol{m}_u^k (\boldsymbol{m}_u^k)^T \boldsymbol{K}_{zz}^{-1} \boldsymbol{k}_{zx} \right\rangle_{q(\boldsymbol{x})}}_{\text{Term 3}}
\end{aligned}
$$

$$
\begin{aligned}
\left\langle \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} &= \left\langle \boldsymbol{m}_u^\mathsf{T} \boldsymbol{K}_{zz}^{-1} \tfrac{\partial \boldsymbol{k}_{zx}}{d\boldsymbol{x}} \right\rangle_{q(\boldsymbol{x})} \\
&= \boldsymbol{m}_u^\mathsf{T} \boldsymbol{K}_{zz}^{-1} \left\langle \tfrac{\partial \boldsymbol{k}_{zx}}{d\boldsymbol{x}} \right\rangle_{q(\boldsymbol{x})}
\end{aligned}
$$

The above three function expectations can thus be expressed in terms of the following four kernel expectations with respect to $q(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{m}, \boldsymbol{S})$:

$$
\langle \kappa(\boldsymbol{x}, \boldsymbol{x}) \rangle_{q(\boldsymbol{x})}, \quad \langle \kappa(\boldsymbol{x}, \boldsymbol{z}) \rangle_{q(\boldsymbol{x})}, \quad \langle \kappa(\boldsymbol{z}_1, \boldsymbol{x}) \kappa(\boldsymbol{x}, \boldsymbol{z}_2) \rangle_{q(\boldsymbol{x})}, \quad \left\langle \frac{\partial \kappa(\boldsymbol{z}, \boldsymbol{x})}{\partial \boldsymbol{x}} \right\rangle_{q(\boldsymbol{x})}.
$$

For our SSL kernel these kernel expectations are not available in closed form, so in practice we approximate them using Gauss-Hermite quadrature.

Next, differentiating $\widetilde{\mathcal{L}}$ with respect to $\boldsymbol{A}(t)$ and $\boldsymbol{b}(t)$ yields the updates

$$
\boldsymbol{A}(t) = -\left\langle \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \right\rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} + 2\boldsymbol{\Sigma}\boldsymbol{\Psi}(t) \tag{36}
$$

$$
\boldsymbol{b}(t) = \langle \boldsymbol{f}(\boldsymbol{x}) \rangle_{q(\boldsymbol{x}),q(\boldsymbol{f})} + \boldsymbol{A}(t)\boldsymbol{m}(t) + \boldsymbol{B}\boldsymbol{v}(t) - \boldsymbol{\Sigma}\boldsymbol{\lambda}(t) \tag{37}
$$

Note that these updates have one key difference from the previously derived updates in Archambeau et al. [39] and Duncker et al. [10]: the input-dependent term $\boldsymbol{B}\boldsymbol{v}(t)$ in eq. (37). Intuitively, this is because the posterior bias term $\boldsymbol{b}(t)$ is fully time-varying, so it captures changes in the latent states due to input-driven effects in the posterior.

In summary, the inference algorithm for updating $q(\boldsymbol{x})$ is as follows. In each iteration of vEM, we repeat the following forward-backward style algorithm:

1. Solve for $\boldsymbol{m}(t)$, $\boldsymbol{S}(t)$ forward in time starting from $\boldsymbol{m}(0)$, $\boldsymbol{S}(0)$ via eq. (25) and eq. (26).
2. Solve for $\boldsymbol{\lambda}(t), \boldsymbol{\Psi}(t)$ backward in time starting from $\boldsymbol{\lambda}(T), \boldsymbol{\Psi}(T) = \boldsymbol{0}$ via eq. (32) and eq. (33).
3. Update $\boldsymbol{A}(t)$ and $\boldsymbol{b}(t)$ via eq. (36) and eq. (37).

After solving these stationary equations, we update $\boldsymbol{m}(0)$ and $\boldsymbol{S}(0)$ via eq. (31).

Computational details Solving for $\boldsymbol{m}(t)$, $\boldsymbol{S}(t)$, $\boldsymbol{\lambda}(t)$, and $\boldsymbol{\Psi}(t)$ requires integrating continuous-time ODEs. In practice we use Euler integration with a small discretization step Δt relative to the sampling rate of the data, though in principle any ODE solver can be used. We found that the ELBO usually converges within 20 forward-backward iterations.

The ODEs for solving $\boldsymbol{\lambda}(t)$ and $\boldsymbol{\Psi}(t)$ in eq. (32) and eq. (33) depend on evaluating gradients of the ELBO with respect to $\boldsymbol{m}(t)$ and $\boldsymbol{S}(t)$. We use modern autodifferentiation capabilities in JAX to compute these gradients.

B.4 Updating dynamics and hyperparameters with a modified learning objective

As we describe in Section 3.3, our gpSLDS inference algorithm uses a modified objective for hyperparameter learning. In this section, we discuss this objective in detail and present closed-form updates for the inducing points given the hyperparameters. Then, using the inducing points, we will derive the posterior distribution over $\boldsymbol{f}(\cdot)$ at any location in the latent space.

After updating the latent paths $q(\boldsymbol{x})$ as described in Appendix B.3, we update hyperparameters Θ using a partially optimized ELBO. This update can be written as

$$\Theta^* = \arg\max_{\Theta} \left\{ \max_{q(\boldsymbol{u})} \mathcal{L}(q(\boldsymbol{x}), q(\boldsymbol{u}), \Theta) \right\}. \tag{38}$$

Following Duncker et al. [10], we choose the variational posterior

$$q(\boldsymbol{u}_k) = \mathcal{N}(\boldsymbol{u}_k \mid \boldsymbol{m}_u^k, \boldsymbol{S}_u^k). \tag{39}$$

Given $q(\boldsymbol{x})$ and Θ, this leads to the closed-form updates,

$$\boldsymbol{S}_u^{k*} = \boldsymbol{K}_{zz} \left(\boldsymbol{K}_{zz} + \int_0^T \langle \boldsymbol{k}_{zx} \boldsymbol{k}_{xz} \rangle_{q(\boldsymbol{x})} \, dt \right)^{-1} \boldsymbol{K}_{zz} \tag{40}$$

$$\begin{aligned} \boldsymbol{m}_u^* = \boldsymbol{S}_u^{k*} \boldsymbol{K}_{zz}^{-1} \int_0^T & \Big(\langle \boldsymbol{k}_{zx} \rangle_{q(\boldsymbol{x})} \left(-\boldsymbol{A}(t)\boldsymbol{m}(t) + \boldsymbol{b}(t) - \boldsymbol{B}\boldsymbol{v}(t) \right)^\mathsf{T} \\ & - \left\langle \frac{\partial \boldsymbol{k}_{zx}}{\partial \boldsymbol{x}} \right\rangle_{q(\boldsymbol{x})} \boldsymbol{S}(t)\boldsymbol{A}(t)^\mathsf{T} \Big) \, dt \end{aligned} \tag{41}$$

In the above equation, $\boldsymbol{m}_u^* \in \mathbb{R}^{M \times K}$ contains $\boldsymbol{m}_u^{k*} \in \mathbb{R}^M$ in each column. The inside maximization of eq. (38) can be computed analytically using these closed-form updates. Note that $\boldsymbol{m}_u^{k*}$ and $\boldsymbol{S}_u^{k*}$ depend on Θ through the prior kernel covariances $\boldsymbol{K}_{zz}$ and $\boldsymbol{k}_{zx}$. Therefore, eq. (38) can be understood as performing joint optimization of the ELBO with respect to Θ through $\boldsymbol{m}_u^{k*}$ and $\boldsymbol{S}_u^{k*}$, as well as through the rest of the ELBO. In practice, this allows vEM to circumvent dependencies between $q(\boldsymbol{u})$ and Θ, leading to more accurate estimation of both quantities. For our experiments, we use the Adam optimizer to solve eq. (38).

After obtaining Θ^* in each vEM iteration, we explicitly update $q(\boldsymbol{u})$ using eq. (41) and eq. (40) for the next iteration.

Recovering predicted dynamics Given (updated) variational parameters $\boldsymbol{m}_u^k$ and $\boldsymbol{S}_u^k$, it is straightforward to compute the posterior distribution of $\boldsymbol{f}^* := \boldsymbol{f}(\boldsymbol{x}^*)$ at any location $\boldsymbol{x}^*$ in the latent space. Recall the variational approximation from eq. (23). If we apply this approximation to $\boldsymbol{f}^*$, we have

$$q(\boldsymbol{f}^*) = \prod_{k=1}^K q(f_k^*) = \prod_{k=1}^K \int p(f_k^* \mid \boldsymbol{u}_k, \Theta) q(\boldsymbol{u}_k) d\boldsymbol{u}_k. \tag{42}$$

To evaluate this analytically, we use properties of conditional Gaussian distributions. First note that by our augmented prior,

$$p(f_k^* \mid \boldsymbol{u}_k, \Theta) = \mathcal{N}(f_k^* \mid \boldsymbol{k}_{x^*z} \boldsymbol{K}_{zz}^{-1} \boldsymbol{u}_k, \kappa_{\text{ssl}}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}_{x^*z} \boldsymbol{K}_{zz}^{-1} \boldsymbol{k}_{zx^*}). \tag{43}$$

Then, by conjugacy of Gaussian distributions,

$$\begin{aligned} q(f_k^*) &= \int \mathcal{N}(f_k^* \mid \boldsymbol{k}_{x^*z} \boldsymbol{K}_{zz}^{-1} \boldsymbol{u}_k, \kappa_{\text{ssl}}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}_{x^*z} \boldsymbol{K}_{zz}^{-1} \boldsymbol{k}_{zx^*}) \mathcal{N}(\boldsymbol{u}_k \mid \boldsymbol{m}_u^k, \boldsymbol{S}_u^k) d\boldsymbol{u}_k \\ &= \mathcal{N}(f_k^* \mid \boldsymbol{k}_{x^*z} \boldsymbol{K}_{zz}^{-1} \boldsymbol{m}_u^k, \kappa_{\text{ssl}}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}_{x^*z} \boldsymbol{K}_{zz}^{-1} \boldsymbol{k}_{zx^*} + \boldsymbol{k}_{x^*z} \boldsymbol{K}_{zz}^{-1} \boldsymbol{S}_u^k \boldsymbol{K}_{zz}^{-1} \boldsymbol{k}_{zx^*}). \end{aligned} \tag{44}$$

B.5 Learning observation model parameters

For the experiments in this paper, we considered two observation models: Gaussian observations and Poisson process observations.

Gaussian observations We consider the observation model

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t_i} \mathcal{N}(\boldsymbol{y}(t_i) \mid \boldsymbol{C}\boldsymbol{x}(t_i) + \boldsymbol{d}, \boldsymbol{R}). \tag{45}$$

where $\boldsymbol{R} \in \mathbb{R}^D$ is a diagonal covariance matrix. The expected log-likelihood is available in closed form and is given by

$$\langle \log p(\boldsymbol{y} \mid \boldsymbol{x}) \rangle_{q(\boldsymbol{x})} = \sum_{t_i} \left(\log \mathcal{N}(\boldsymbol{y}(t_i) \mid \boldsymbol{C}(t_i) + \boldsymbol{d}, \boldsymbol{R}) - \frac{1}{2} \mathrm{Tr} \left[\boldsymbol{S}(t_i) \boldsymbol{C}^\mathsf{T} \boldsymbol{R}^{-1} \boldsymbol{C} \right] \right) \tag{46}$$

Closed-form updates for $\boldsymbol{C}, \boldsymbol{d}$ and $\boldsymbol{R}$ are also available:

$$\boldsymbol{C}^* = \left(\sum_{t_i} (\boldsymbol{y}(t_i) - \boldsymbol{d}) \boldsymbol{m}(t_i)^\mathsf{T} \right) \left(\sum_{t_i} (\boldsymbol{S}(t_i) + \boldsymbol{m}(t_i)\boldsymbol{m}(t_i)^\mathsf{T}) \right)^{-1} \tag{47}$$

$$\boldsymbol{d}^* = \frac{1}{n_{t_i}} \sum_{t_i} (\boldsymbol{y}(t_i) - \boldsymbol{C}^* \boldsymbol{m}(t_i)) \tag{48}$$

$$R_d^* = \frac{1}{n_{t_i}} \sum_{t_i} (y_d(t_i)^2 - 2 y_d(t_i) \boldsymbol{c}_d^\mathsf{T} \boldsymbol{m}(t_i) + (\boldsymbol{c}_d^\mathsf{T} \boldsymbol{m}(t_i))^2 + \boldsymbol{c}_d^\mathsf{T} \boldsymbol{S}(t_i) \boldsymbol{c}_d) \tag{49}$$

where n_{t_i} is the number of observed time points, R_d^* is the d-th entry of $\boldsymbol{R}$, and $\boldsymbol{c}_d$ is the d-th row of $\boldsymbol{C}$.

Poisson process observations The second observation model we consider is Poisson process observations of the form

$$p(\{t_i\} \mid \boldsymbol{x}) = \mathcal{PP}(g(\boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{d})), \tag{50}$$

where either $g(a) = \exp(a)$ (exponential inverse link) or $g(a) = \log(1 + \exp(a))$ (softplus inverse link). For the exponential inverse link, the expected log-likelihood is available in closed form and is given by

$$\langle \log p(\{t_i\} \mid \boldsymbol{x}) \rangle_{q(\boldsymbol{x})} = -\int_0^T \exp\left(\boldsymbol{C}\boldsymbol{m}(t) + \boldsymbol{d} + \frac{1}{2} \mathrm{diag}(\boldsymbol{C}\boldsymbol{S}(t)\boldsymbol{C}^\mathsf{T}) \right) dt + \sum_{t_i} (\boldsymbol{C}\boldsymbol{m}(t_i) + \boldsymbol{d}) \tag{51}$$

For the softplus inverse link, the expected log-likelihood is not available in closed-form, but can be approximated by Gauss-Hermite quadrature or a second-order Taylor expansion around $\boldsymbol{m}(t)$.

For both Poisson process models, we update $\boldsymbol{C}$ and $\boldsymbol{d}$ using gradient ascent on the expected log-likelihood with the Adam optimizer.

B.6 Learning the input effect matrix

Here we derive a closed-form update for $\boldsymbol{B}$, which linearly maps external inputs to the latent space. The only term in the ELBO which depends on $\boldsymbol{B}$ is $\langle \mathrm{KL}[q(\boldsymbol{x}) || p(\boldsymbol{x} \mid \boldsymbol{f})] \rangle_{q(\boldsymbol{f})}$. We differentiate this term as written in eq. (34) and arrive at the update

$$\boldsymbol{B}^* = -\left(\int_0^T (\langle \boldsymbol{f} \rangle_{q(\boldsymbol{x}), q(\boldsymbol{f})} + \boldsymbol{A}(t)\boldsymbol{m}(t) - \boldsymbol{b}(t)) \boldsymbol{v}(t)^\mathsf{T} dt \right) \left(\int_0^T \boldsymbol{v}(t)\boldsymbol{v}(t)^\mathsf{T} dt \right)^{-1}. \tag{52}$$

Note that the term $\left(\int_0^T \boldsymbol{v}(t)\boldsymbol{v}(t)^\mathsf{T} dt \right)^{-1}$ can be pre-computed since $\boldsymbol{v}(t)$ is known.

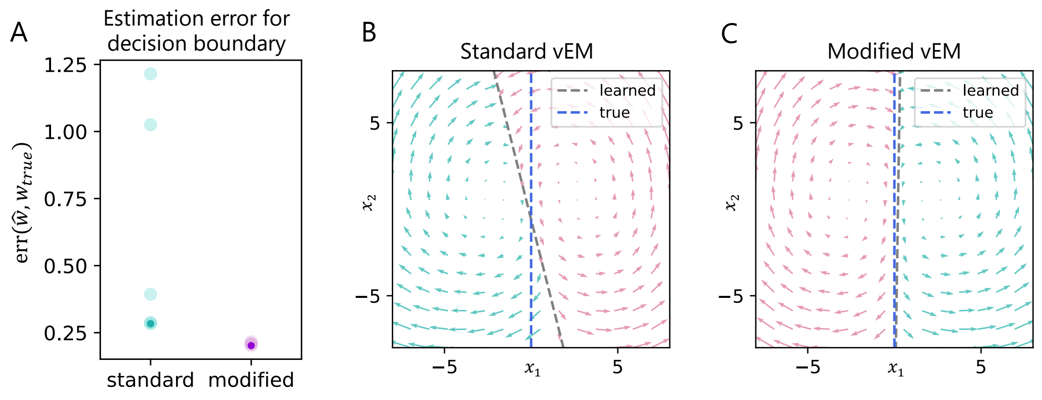

Figure 5: Comparison between the standard vEM approach in Duncker et al. [10] and our modified vEM approach. **A.** Estimation error between the true and learned decision boundaries, computed as described in Appendix C. For each vEM approach, we fit 5 gpSLDS models with different random initializations. The estimation errors are denoted in light blue/purple dots. The runs that we display in the next two panels are denoted by a solid blue/purple dot. **B.** The standard vEM approach fails to learn the true decision boundary of $x_1 = 0$. **C.** The modified vEM approach precisely learns this decision boundary.

C Empirical results for new learning objective

In this section, we empirically compare the standard vEM approach from Duncker et al. [10] to our modified vEM approach in which we learn kernel hyperparameters on a partially optimized ELBO. For this experiment, we use the same synthetic dataset from our main result in Section 4.1. We fit the gpSLDS with 5 different random initializations using both standard vEM and modified vEM. For these fits, we fix the values of $\boldsymbol{C}$ and $\boldsymbol{d}$ throughout learning to ensure that the resulting models are anchored to the same latent subspace (in general, they are not guaranteed to end up in the same subspace due to rotational unidentifiability). Each run was fit with 50 total vEM iterations; each iteration consisted of 15 forward-backward solves to update $q(\boldsymbol{x})$ and 300 Adam gradient steps with a learning rate of 0.01 to update kernel hyperparameters.

To compare the quality of the learned hyperparameters, we quantitatively assess the error between the learned and true decision boundaries. In this simple example with $J = 2$, the decision boundary can be parametrized as $w_0 + w_1 x_1 + w_2 x_2 = 0$ for some $\boldsymbol{w} = (w_0, w_1, w_2)^\mathsf{T}$. The true decision boundary is characterized by $\boldsymbol{w}_{\text{true}} = (0, 1, 0)^\mathsf{T}$. We denote the learned decision boundary as $\hat{\boldsymbol{w}}$. Next, we compute an error metric between the learned and true decision boundaries as follows. We first normalize the learned decision boundary and define $\hat{\boldsymbol{w}}_{\text{norm}} = \frac{\hat{\boldsymbol{w}}}{\|\hat{\boldsymbol{w}}\|_2}$. We do not need to do this for $\boldsymbol{w}_{\text{true}}$ since it is already normalized. Then, we use the error metric

$$\mathrm{err}(\hat{\boldsymbol{w}}, \boldsymbol{w}_{\text{true}}) = \min\left(\|\hat{\boldsymbol{w}}_{\text{norm}} - \boldsymbol{w}_{\text{true}}\|_2, \|\hat{\boldsymbol{w}}_{\text{norm}} + \boldsymbol{w}_{\text{true}}\|_2\right). \tag{53}$$

Including both terms in the minimum is necessary due to unidentifiability of the signs of $\hat{\boldsymbol{w}}$.

Figure 5A compares this error metric across the 5 model fits for each vEM method. It is clear that the modified vEM approach consistently outperforms the standard vEM approach in terms of more accurately estimating the decision boundary. In addition, the error for standard vEM has much higher variance, since the algorithm is prone to getting stuck in local maxima of the ELBO. Figures 5B-C display the learned versus true decision boundaries in the latent space for two selected runs. We select the run from each vEM approach which achieved the lowest decision boundary error metric, as denoted by solid dots in fig. 5A. We find that the model fit with standard vEM learns a decision boundary which noticeably deviates from the true boundary. On the other hand, the model with with modified vEM recovers the true boundary almost perfectly. This illustrates that our modified approach dramatically improves kernel hyperparameter estimation in practice and enables the gpSLDS to be much more interpretable in the latent space.

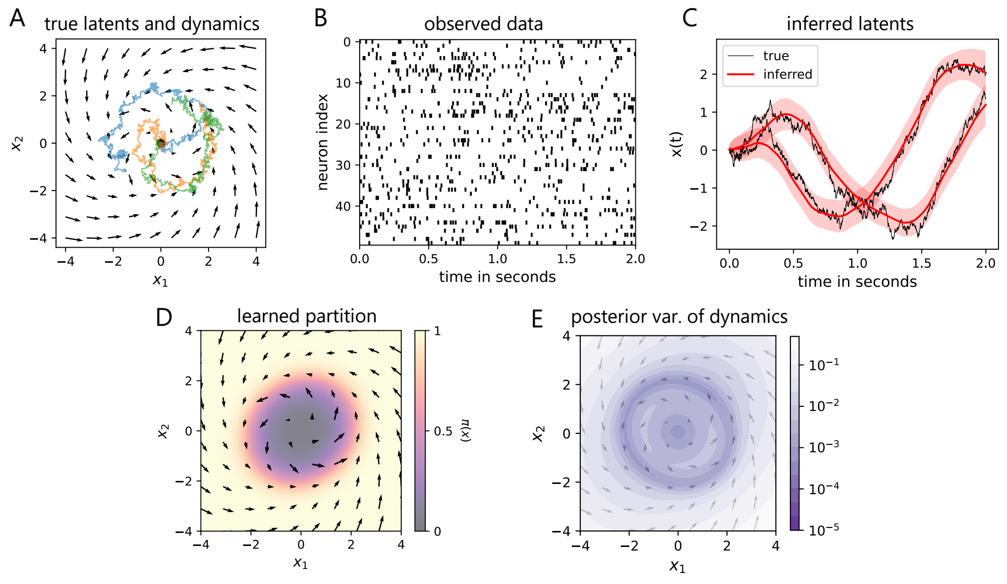

Figure 6: Additional synthetic data results on a 2D limit cycle from a gpSLDS fit with quadratic decision boundaries. **A.** True dynamics and true latent trajectories on 3 example trials used to generate the dataset. Dynamics are an unstable rotation and a stable rotation with fixed points at $(0, 0)$, separated by $x_1^2 + x_2^2 = 4$. **B.** Poisson process observations for an example trial. **C.** gpSLDS inferred latent trajectory with 95% posterior credible intervals for an example trial. **D.** The learned $\boldsymbol{\pi}(\boldsymbol{x})$ accurately recovers the true circular boundary between the two sets of linear dynamics. **E.** The gpSLDS learned posterior variance on dynamics. The posterior variance is low in regions heavily traversed by the true latent paths, and is high in regions with little to no data.

D Additional synthetic data results

To further demonstrate the expressivity of the gpSLDS over the rSLDS, we apply the gpSLDS to a synthetic dataset where the true decision boundary between linear regimes is nonlinear. The rSLDS can only model linear decision boundaries in order for its inference algorithm to remain tractable.

For this example, we generate a synthetic dataset consisting of an unstable linear system and a stable linear system separated by the decision boundary $x_1^2 + x_2^2 = 4$. Both of the linear systems have fixed points at $(0, 0)$. The smooth combination of these linear systems results in a 2D limit cycle (fig. 6A). We simulate 30 trials of Poisson process observations from $D = 50$ neurons over $T = 2$ seconds (fig. 6B). We initialize the observation model parameters $\boldsymbol{C}$ and $\boldsymbol{d}$ using a Poisson LDS with data binned at 20ms. Then, we fit a gpSLDS with $J = 2$ regimes, and with $\boldsymbol{\pi}(\boldsymbol{x})$ modeled using the feature transformation

$$\boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 & \boldsymbol{x}_1^2 & \boldsymbol{x}_2^2 \end{bmatrix}^\mathsf{T}. \tag{54}$$

The results of this experiment are shown in fig. 6C-E. In fig. 6C we find that the gpSLDS successfully recovers the the true latent trajectory with accurate posterior credible intervals for an example trial. Furthermore, fig. 6D demonstrates that by using the quadratic feature transformation in eq. (54), the gpSLDS accurately learns the true flow field and true decision boundary $x_1^2 + x_2^2 = 4$. In addition, the values of $\boldsymbol{\pi}(\boldsymbol{x})$ smoothly transition between 0 and 1 near this boundary, highlighting the ability of our method to learn smooth dynamics if present. Lastly, in fig. 6E we plot the inferred posterior variance of our method. We find that the gpSLDS is more confident in regions of the latent space with more data (e.g. at the decision boundary), and less confident in regions of latent space with little to no data.

E Computing resources

We fit all of our models on a NVIDIA A100 GPU on an internal computing cluster. A breakdown of approximate compute times for the main experiments in this paper includes:

- Synthetic data results in Section 4.1: 1.5 hours per model fit, $\sim$40 hours for the entire experiment.
- Real data results in Section 4.2: 1.5 hours per model fit, $\sim$8 hours for the entire experiment.
- Real data results in Section 4.3: 1 hour per model fit, $\sim$5 hours for the entire experiment.

We note that these estimates do not include the full set of experiments we performed while carrying out this project (such as preliminary or failed experiments).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In our abstract and introduction, we propose a new method for interpretable modeling of neural dynamics. In our paper, we apply our method and demonstrate its interpretability and competitive performance on a synthetic dataset and two real datasets.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations of our work in the discussion section of the paper.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our paper does not contain theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: For our experiments, we provide detailed descriptions and references to datasets, as well as descriptions of the models we fit.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We have made our codebase on GitHub publicly available. We provide both data and code for our main synthetic data result, and we also provide references to the dataset and instructions for fitting models to reproduce the second real data result. We do not provide data or code for the first real data result since that data has not been released to the public.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We provide all details for the experimental setting that are needed to understand the results for all three sets of experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Yes, we report thorough quantitative comparisons with error bars for the experiments in Section 4.1 and 4.2. Throughout the paper, we fit multiple (i.e. at least 5) models with different initializations for our experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide information on computing resources in Appendix E.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed and followed the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work develops a new statistical modeling tool for the analysis of neuroscience data in constrained experiments. We expect that this work will not have broad societal impacts, positive or negative.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper does not involve data or models which have a high risk of misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original papers which analyzed or produced both of the real datasets that we consider in Sections 4.2 and 4.3.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provide our codebase on GitHub which comes with clear documentation.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-1065